SOLENT UNIVERSITY

FACULTY OF BUSINESS LAW AND DIGITAL TECHNOLOGIES

MSc Applied AI and Data Science

Academic Year 2021-2022

Aboh Ibe Kenneth

A MULTI-PURPOSE SPEECH EMOTION RECOGNITION AI TOOLKIT

Supervisor: Dr Jarutas Andritsch

September 2022

This report is submitted in partial fulfilment of the requirements of Solent University for the degree of MSc Artificial Intelligence and Data Science

ACKNOWLEDGEMENTS

To begin, I'd like to thank my wife, who has been nothing but encouraging and supportive throughout this entire master's programme. In addition, I'd like to express my gratitude to Dr. Jarutas Andritsch, my supervisor, for his invaluable assistance throughout this investigation. Finally, I'd like to express my gratitude to all the Lecturers and Professors who have had an impact on my education throughout this master's programme.

ABSTRACT

Emotional intelligence is the capacity to recognize, control, and comprehend one's own emotions as well as those of others. Living in this world entails interacting with a vast array of individuals and coping with recurrent uncertainty. A high emotional quotient (EQ) in the workplace facilitates connection formation, team stress reduction, conflict resolution, and job satisfaction. Through this study, the author intends to conduct research in the interdisciplinary field of Affective Computing, which spans computer science, psychology, and cognitive science. Affective Computing is the study and development of systems and devices that can recognize, interpret, process, and simulate human effects. Speech Emotion Recognition (SER) is one of the most essential components of Affective Computing. This work involves the development of a Multipurpose Toolkit based on SER AI which can understand the users’ sentiments and recognize their emotions using the acoustic features from their speech. Thus, it provides the analytical foundation for enhancing people's emotional well-being.

Table of Content

LIST OF TABLES

LIST OF FIGURES

Acronyms

Acronym	Full-Form
CNN	Convolutional Neural Networks
LSTM	Long Short-Term Memory
SVM	Support Vector Machine
API	Application Program Interface
HCI	Human-Computer Interaction
MFCC	Mel Frequency Cepstrum Coefficient
LPCC	Licensed Professional Clinical Counselors
MEDC	Mel Energy spectrum Dynamic Coefficient
HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
RAVDESS	Ryerson Audio-Visual Database of Emotional Speech and Song
TESS	Toronto Emotional Speech Set
CREMA-D	Crowd-sourced Emotional Multimodal Actors Dataset
SAVEE	Surrey Audio-Visual Expressed Emotion Dataset
AMDF	Average Magnitude Difference Function

CHAPTER 1.

INTRODUCTION AND BACKGROUND

How often has it been the way where you say something and someone understands the tone of sarcasm behind it, or that you are trying to put up a cheerful mirage? It is perhaps one of the most remarkable abilities of us humans that not only can we speak several words, but we can speak them in several ways and we can speak them in different tones and still other humans can almost always completely grasp what we mean and how we feel when we speak.

The pandemic forced many people across the world to isolate themselves. Governments closed down many if not all businesses across all countries for many days. People were requested to not go outside and meet others and therefore contain the spread. This had indeed resulted in many people going without external contact for the longest period and there were many cases of depression (Bucciarelli, Valentia, et al, 2021).

Efforts to help people deal with depression, especially those who did not have a family or were isolated from their family, were made across many nations. One such effort included the set up of a hotline. This hotline would be where a human, an expert to help with the issues of loneliness and depression. And while such efforts did help many people, the problem of helping the neediest and appropriating resources accordingly remains a problem. Who would route the calls and based on what? And most importantly, how does one ascertain they were able to help those who require the most immediate assistance?

While the age of social media and the internet is here, call centres are still around and form a significant redressal method for many customers and consequently companies. Moreover, the experience which customers receive from such call centres can be very crucial as a bad experience can prompt the customer to sever ties with the company and the product and look for alternatives. As such, it becomes quite important for companies to ensure that this process of customer support is as smooth and painless as possible. And while companies would love to give that experience to all customers, giving a preference to agitated customers or having such customers handled by expert care agents can certainly be a preferred strategy for any company. But, in the meantime, that remains a challenge.

Let’s take a lighter case in point where you try to have a conversation with any digital assistant like Siri or Alexa or Google Assistant and you try to be sarcastic, but it ends up taking you quite literally or not understanding you completely. Well, how could companies make the assistants understand sarcasm? Or even so, how could they make their digital assistants more human where you make one request in a rather low tone, and it automatically asks you if you would like to play your favorite song?

Let us take one more light-hearted scenario with a digital assistant again, where, let’s say, you use Alexa for bedtime storytelling for your kids where you would like to filter out certain emotions and would prefer them to be casual and energizing and motivating and inspiring and not depressing or demotivating. Sure, you could filter out those stories that could contain those emotions yourself or some humans at the companies could, but can assistants be automated to achieve the same desired result?

It is no secret that many of those in the television or film industry utilize speech coaches for various purposes, usually to make their speech more suitable to a certain cross-section of people or somehow ‘better’. Such practices are quite expensive, especially so for someone outside of those industries, and feedback practices are available only with the help of the employed speech coaches while there are no two ways in the fact that same utterance can be recognized differently by different humans. Can there be a better way of making speeches ‘better’, if at all that is one’s aim?

AI has been growing phenomenally over the past few years and put very simply, it is nothing but a way for a computer to be able to make decisions on its own like humans based on some relevant training where a computer learns to identify what to look for and how to make those classifications. All of the above scenarios could be solved or made possible with the help of an AI model that could recognize the emotions present in any speech. Such a model, let us call it the Speech Emotion Recognition model or SER model, could help with all the above and much more.

1.1 Research Questions

(1) Can humans converse with machines as naturally as humans converse with other humans?

(2) Can an AI model that focuses on capturing emotions across different scenarios and reporting them or taking actions accordingly be built?

CHAPTER 2.

LITERATURE REVIEW

Humans have, ever since the dawn of AI, back in the early 1950s, visualized being able to converse with machines like they do with other humans. This has led to work on helping machines become able enough to first of all convert human speech into a string of words. And while there has been immense progress in this arena, this still does not, as is evident today, make machines able to understand as naturally as other humans can understand human speech. (Ayadi, Kamel and Karray, 2011).

Speech emotion recognition, however, began as a niche and has now grown into an important component of being able to interact with machines in a humane way, a technical term for which is Human-Computer Interaction. (B. W. Schuller, 2018). However, this is not an easy task for a computer to perform, as there is a lot of variability such as what words are stressed in the speech. This also becomes difficult as the speaking style and rates vary across people. These features are important to consider as they affect the extracted pitch and energy contours, which are used in making predictions across most models. The energy and pitch contours are used, by many researchers, for their well-known capability to carry a large amount of information regarding the user's emotion (Schuller, Rigoll, and Lang, 2003). And since the pitch and energy contours are the preferred features, it becomes a part of the reason why many researchers have primarily focused on monolingual subjects in their studies. Moreover, speech may contain not only one emotion, in fact, often, but it also contains many emotions and the boundaries between some closely related emotions are not well defined and agreed upon. (Ayadi, Kamel and Karray, 2011). Even so, there have been many instances where researchers have worked with not very well-defined, universally used emotional states. For example, (Schuller, Rigol,l and Lang, 2003) noted the same problem of not having a universally acknowledged, well-defined state of emotions and therefore, besides using the 6 most common emotions: anger, disgust, fear, sadness, surprise, and joy, they also used one more classification state which they denoted as neutral to allow their algorithms to classify speech that had none of the 6 emotions. Moreover, even (Nwe, Foo, and De Silva, 2003) note that linguists have a large vocabulary for describing various emotional states with varying levels of granularity and that researchers in the past have used as many as 300 labels while also labelling the above-mentioned 6 emotions as primary of archetypal emotions. Without going into much detail, it should suffice that given the huge amount of literature that follows the 6 emotions model, this dissertation will also be presenting a similar model, with a slightly modified version of that (Schuller, Rigoll, and Lang, 2003) to include an eight emotion state: calm.

Now that we have gotten a gist of various emotional states and how they have been employed in the past literature, the next step would be how machines have identified various features for extraction in the past. Logically, a machine would need to first identify a feature to be able to extract it, and then it can classify the extracted feature. Following the same approach, researchers have been able to identify features such as vocal traction features, prosodic features, and source-based excitation features. In terms of classification of those extracted features, the researchers have employed a plethora of techniques and some of them are Bayesian Networks (BN), Support Vector Machines (SVM), Gaussian Mixture Model (GMM), and Hidden Markov Model (HMM). Out of these, Bayesian Networks (BN) and Support Vector Machines (SVM) are linear classifier methods and Gaussian Mixture Model (GMM) and Hidden Markov Model (HMM) are non-linear classifier methods, which are more effective considering how dynamic speech is. More so, considering that energy contours are extracted as one of the basic features, Mel Energy-spectrum Dynamic Coefficients (MEDC), and Mel-Frequency Cepstrum Coefficient (MFCC) are some of the models that are effective in recognizing emotion on that basis. Moreover, other classifiers or methods such as Principal Component Analysis, Decision Trees, and K-Means are also used to classify emotions sometimes. (Khalil, Jones, Babar, et al, 2019). Interestingly, (Schuller, Rigoll, and Lang, 2003) used a Hamming window function to calculate the contours, and the logarithmic mean energy is used to calculate the values of energy for each frame of 10 ms, whereas the Average Magnitude Difference Function (AMDF) was utilized to calculate the pitch contours. The authors of the paper also noted that by using AMDF, they were able to get faster results in comparison to the autocorrelation function of a frame, and also while it proved robust to noise, it was susceptible to dominant formats. Once the features were obtained, the author’s employed Gaussian mixture models about a global statistics framework of an utterance from the database used, while they also used Hidden Markov Models or HMMs in a second method they introduced which did not rely on statistics but the low-level instantaneously extracted features. The results authors obtained showed an accuracy of 86% compared to the close 80% recognition rate of humans of the same corpus. (Schuller, Rigoll and Lang, 2003)

Besides the difficulty in classification and annotation, another big challenge is data collection. Tons of audio data can be collected from various media platforms but they are quite biased as the entertainers' or performers' emotions are emulated and that does not quite make it an ideal data set to work on. Moreover, there has also been a long debate on how acted utterances can be quite different from natural ones (Ayadi, Kamel, and Karray, 2011) note and some researchers, viz (Williams and Stevens, 1972) found that acted utterances can have emotions that are greatly exaggerated than the real-life counterpart utterances. And even when you get a good enough dataset to work on, you have to painfully go through labelling the vast amount of audio for supervised learning to happen. Even though skilled professionals may be employed to do the labelling, they may have their notions for some emotions and therefore, it becomes inevitable for the labelling to be checked by another team of individuals therefore this process becomes much more nuanced than in the case of image recognition. An example of the above can be noted in (Fayek, Lech, and Cavedon, 2017) where the annotators were free to label independently and the final label that was agreed upon was done with the help of voting by the three annotators and this helped only with 74.6% of the dataset where at least two annotators agreed upon the emotional label given. Another example of this is the research by (Abbaschian, Sierra-Sosa, and Elmaghraby, 2021) used the Toronto Emotional Speech Set or TESS, where two female actors of age 60 and 20 simulated seven emotions, and the labelling of the dataset was carried out by a group of undergrad students and those sentences that had a minimum of 66% confidence regarding the label provided to any particular sentence were the only ones the authors included in their dataset. Moreover, as (Ayadi, Kamel, and Karray, 2011) note, many datasets have low human recognition rates, as low as 65%. Now let us look at some work done in the past that is relevant to the work of this author in greater depth.

Work done by S.Lugovic et al proposes the techniques and applications of Emotion Recognition in Speech. It explains how Affective Computing opens a new area of research in Computer

Science to improve the way how humans and machines interact. Their paper also gives an outline of the areas where emotion recognition could be utilized such as healthcare, psychology, cognitive sciences, civil services, and marketing. They go on to state that emotion recognition could help with the analysis of speeches given by various leaders in a nation's parliament or otherwise, that it could also be sued by NGOs with regards to various social groups they serve, that it could also be utilized in tracking how various companies respond to customers on call centres, analyze how managers respond to various topics and that emotion recognition can also be utilized in interviews. While the focus of the paper is on bridging emotion recognition to social systems, the paper also notes that Artificial Neural Networks (ANN), k-Nearest Neighbor (k-NN), Gaussian mixture model (GMM), and Hidden Markov Model (HMM) are some of the most prominent techniques that can and are being used for emotion recognition. They conclude that if intelligence is the capacity for understanding and the ability to perceive and comprehend meaning, then implementing emotions in speech recognition could increase the intelligence of socio-technical systems that people are part of and the following decades will most certainly provide possibilities to observe progress in this domain. (S. Lugovic, et al, 2016).

Damian Valles and Rezwan Matin provided an audio processing approach based on ensemble learning for SER for children with Autism Spectrum Disorder. They trained the SVM Model, MLP Model, and RNN models using the MFCCs features from the audio samples of various datasets such as RAVDESS, TESS, and CREMA-D. It is noteworthy that the authors introduced the noise of a playground or a street or a shopping mall randomly to all the audio samples available in the above datasets, as the audio samples in them were recorded in a 'noiseless' environment. Of all these models, the MLP model achieved the best precision metrics while SVM showed the least precision metrics. All of these models performed well on the TESS dataset which is only made up of female sounds which suggest that the emotions are relatively easier to recognize for females as compared to males. The authors also noted that the model faced some challenges when it comes to correlating similar emotions such as anger and disgust, sadness and fear. Moreover, they also faced challenges when the utterances to be classified were performed with varying intensities, such as in the case of CREMA-D, where the dataset is captured in 4 levels of varying intensity. Interestingly, it also had issues with catching up the emotions when someone is being sarcastic or upset while smiling which are easily identifiable by humans. (Valles and Matin, 2021).

(Jain, Narayan, Balaji, et al, 2020) have used Support Vector Machine (SVM) as the classifier on the samples obtained from the Linguistic Data Consortium (LDC) and UGA database. The classification was carried out into only 4 emotions with two classification strategies employed concerning SVM, viz One Against All (OAA) and Gender Dependent Classification while MFCC and LPCC extracted features were the ones which were processed and used for classification along with energy and pitch. Based on the experiments conducted by the authors, they noticed that feature extraction using MFCC obtained a higher degree of accuracy, which was 85%, in classification in comparison to feature extraction using LPCC, which was comparably low to 73% (note that the accuracy percentages presented here are averaged). Moreover, regarding datasets, the authors obtained 90% accuracy in the LDC training dataset while they obtained 65% accuracy in the UGA dataset. The difference in this accuracy is thought to be introduced because of the quality of the LDC dataset as the samples in the dataset were recorded in ‘noiseless’ conditions. Concerning the two classification strategies, the Gender Dependent Classification had a higher accuracy on average of 84% compared to OAA’s 73% overall accuracy.

Another work from Surendra Malla et al, while primarily serving as a review paper of the most prominent techniques and models used across literature, also proposed, based on the review conducted, an improved technique in SER which uses the combined approach of Convolutional Neural Networks (CNNs) along with Long-short term memory (LSTMs). They suggested that the combined networks are capable of improving the emotion recognition accuracy of human speech and also noted that the selection of databases based on the availability of relevant and abundant acoustic parameters is crucial to get an optimal level of accuracy. (Malla, et al, 2020).

CHAPTER 3.

METHODOLOGY

3.1. AIM(S) AND OBJECTIVES

Main-Objective

The main goal of this study is to develop the emotion recognition system by training Deep Learning Neural Networks (CNNs / LSTMs) using benchmark datasets for Speech Emotion Recognition that would aid the analytics procedures from the emotions of the people and their talking behaviours by analyzing their speeches from the audio recordings. This system can be leveraged by organizations to analyze their customers' satisfaction with the communication support executives at their workplace. It can also be used for segregation purposes at the intelligence bureau while dealing with relevant lawsuits.

Sub-Objectives

- Perform the ground research of existing work done in the domain and get to know about the most suitable neural network models to be used to deal with the task of Speech Emotion Recognition
- Formulate a normalized dataset of the audio samples of human speech from different emotion categories which cover almost all dominant human emotions
- Train the neural networks using several benchmark datasets containing different emotion categories
- Evaluate the performance of the trained models using various performance metrics and plotting the confusion matrix
- Improve the performance by further hyperparameter tuning or by fine-tuning the models (if required)
- Create an end-to-end AI pipeline to embed the trained models into it which can directly be used to develop any real-world application involving the task of SER.

- Develop a full stack AI web application to embed the above end-to-end pipelines of trained models considering the user-friendliness, security, model integration, and maintenance factors.

3.2 Proposed Artifact

The proposed solution contains the full-fledged system for Emotion Recognition from Human Speeches which involves training and development of artificial intelligence models based on Deep Learning/ Machine Learning algorithms as well as building up the web application which embeds the end-to-end pipelines of trained AI models. The backbone of the solution is constructed by leveraging the novel deep neural network architectures such as Convolutional Neural Networks (CNNs) to ensure the developed classification system is robust, lightweight, yet efficient for emotion recognition from speech.

3.2.1 High Level Design (HLD)

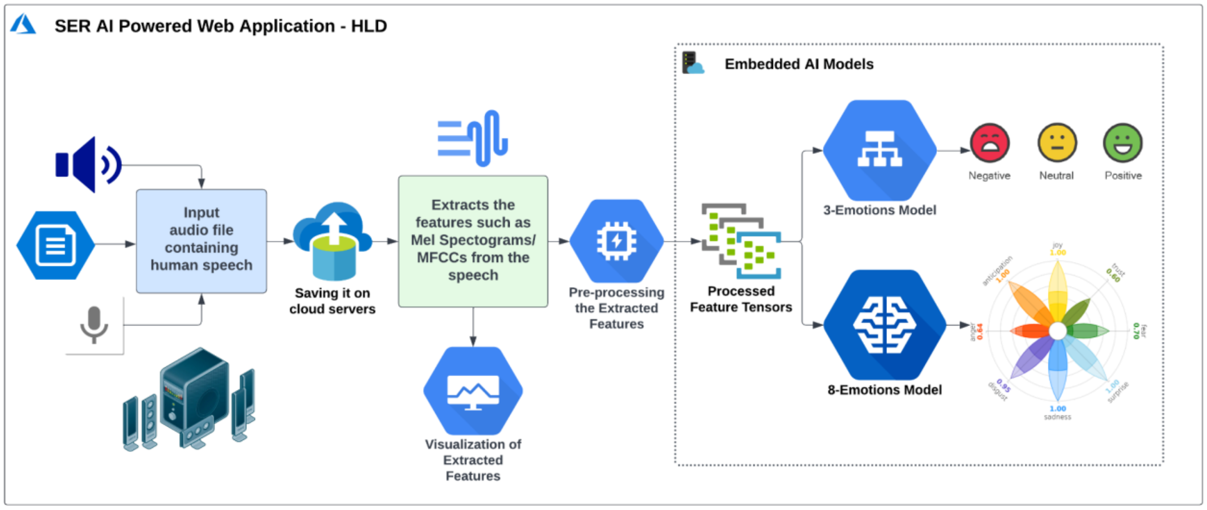

SER AI Powered Web Application - HLD
Input audio file containing human speech
Saving it on cloud servers
Extracts the features such as Mel Spectograms/ MFCCs from the speech
Pre-processing the Extracted Features
Visualization of Extracted Features
Embedded AI Models
Processed Feature Tensors
3-Emotions Model
Negative
Neutral
Positive
8-Emotions Model

figure 1: The above image gives a representation of the proposed model at an abstract level.

3.2.2 Features

As can be seen from the HLD, the proposed artefact consists of a full stack web application that is accessible by any device which supports a web browser such as mobile phones, tablets, laptops, desktops, etc. The entire solution consists of three main modules which are data management, speech processing, and emotion recognition.

The data management module allows users to upload the audio file and store it securely on the cloud server which is further processed by the speech processing module.

The speech processing module would process the audio file uploaded by the user. It will extract the necessary features from the speech such as Mel Spectrograms and MFCCs. It also provides the visualization of the extracted features to the users in the application window.

These extracted features are used by the trained AI models of the emotion recognition module to perform the classification task. These features are pre-processed before feeding into the neural network architecture of the models. Based on the user's preference of the model, the emotions are classified, and their classification scores are demonstrated to the user. The solution consists of two AI models as can be seen in the HLD which are described below:

1. **3-Emotions Model:**

   This model classifies the audio containing human speech into three categories which are Positive, Negative, or Neutral. So, this model gives the overall sentiment of the audio. It can be used to differentiate the satisfied and dissatisfied clients and further aids the organizations to take necessary actions on both sets of clients independently.

2. **8-Emotions Model:**

   Another model type is the 8-Emotions model. This model classifies the audio containing human speech into eight major emotions which are Angry, Calm, Disgust, Fear, Happy, Neutral, Sad, and Surprise. This model provides the classification scores for each of the mentioned emotions for a given input speech. These scores allow businesses to do a deeper analysis of the customers’ experiences with their products and services.

3.3 **Resource Requirements**

Software requirements for this project include open-source frameworks for the development of deep learning models and web applications. For faster training and development of the deep learning models, the Python-based open source numerical computation library Tensorflow is utilized. Another Python package Librosa is utilized for speech processing and audio feature extraction. Librosa uses a variety of signal processing techniques to extract features from audio sources and assists to visualize them. Streamlit APIs are used for building up a web application where all these trained models will be integrated. Streamlit provides real-time interaction of the frontend with the backend and provides support for almost all numerical computation packages such as Tensorflow, Pytorch, Matplotlib, Numpy, Pandas, etc.

Hardware requirements for this project demand system with powerful memory requirements. To train the neural network architectures faster, access to the CUDA-enabled powerful GPUs is required as deep learning models are computing-intensive to train. In case of unavailability of the hardware resources, the tools such as Google Colab can be leveraged which provide all required resources to develop the machine learning or deep learning models.

3.4 Project Plan (Gantt Chart)

The following figure demonstrates the project plan of the whole thesis through a Gantt Chart over the period. The project plan elaborates the subtasks associated with the research and provides the estimated time windows for the same.

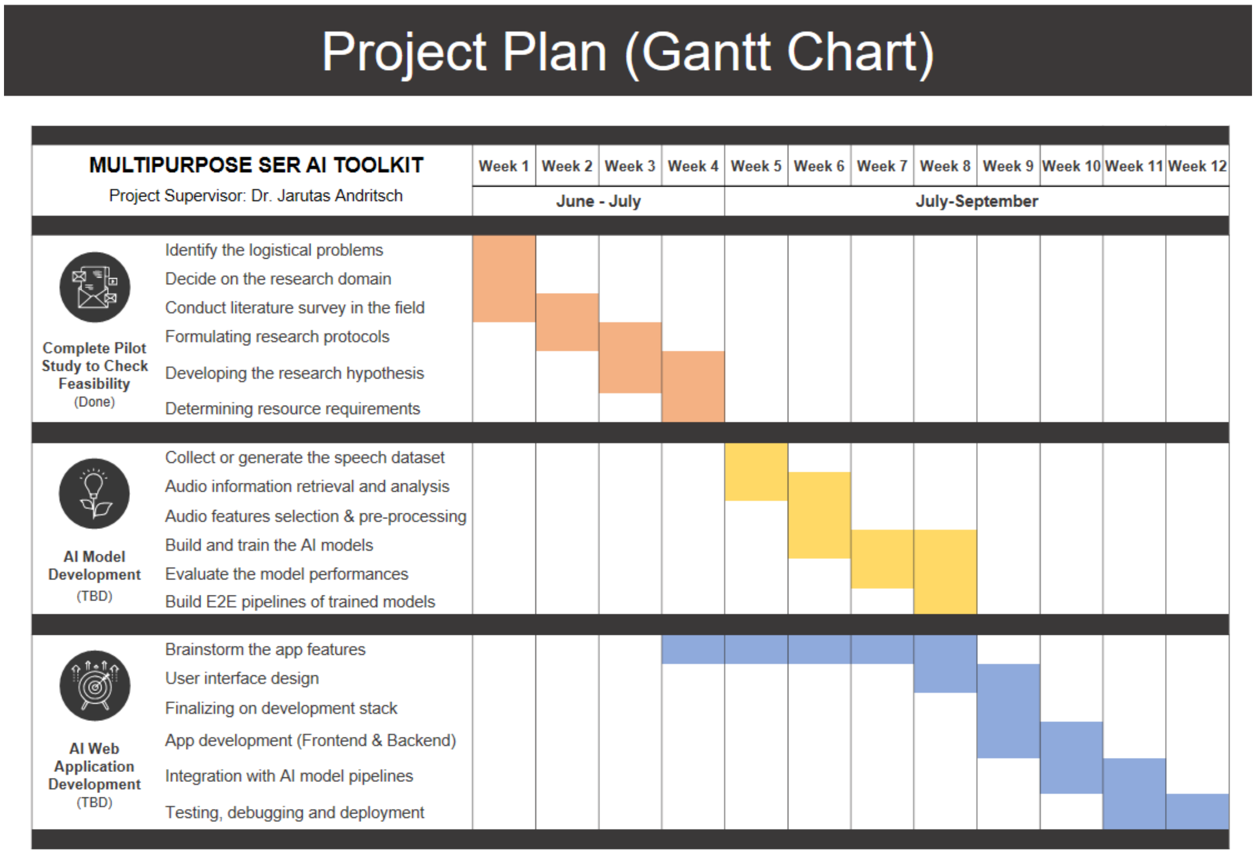


figure 2: Grantt Chart showing project plan

CHAPTER 4: IMPLEMENTATION

4.1 Overview

In this thesis, the transfer learning approach was used for solving the emotion classification task. The pretrained architecture of DenseNet-201 was utilized along with the additional layers on top of it to perform the emotion classification task. The top three benchmark datasets in the domain of Speech Emotional Recognition tasks which are RAVDESS, TESS, and CREMA-D were combined and utilized to train the models. The combination of these datasets ensures the generalization in the train and test sets which helps in efficiently training the models.

4.1.1 Tools and Technology

During the process of the implementation, there were many software developments tools and technology stacks that were important to consider while implementing the product, these are listed below:

- Google Colab as Model Development Environment
- Python with Visual Studio Code for Web Application Development
- Streamlit API Toolkit as a web development framework
- Plotly's Python APIs for Interactive Graph Plotting
- Other important libraries and modules which have been utilized by the program (librosa, tensorflow, numpy, pandas, matplotlib, etc.).

- Google Colab as Model Development Environment

A Google Research tool called Collaboratory, or Colab as shorthand, enables programmers to create and run Python code directly from their browsers. Google Colab is a great utility for developing deep learning applications. It's a dedicated Jupyter notebook with a great free edition that provides costless accessibility to Google's computational resources like Graphical Processing Units (GPUs) and Tensor Processing Units (TPUs) and requires no setup on the host system. Thus, the platform of Google Colab was leveraged for developing speech emotion, recognition-based artificial intelligence models.

- Python in Visual Studio Code

For building the product, the python programming language was used. The reason why python was used for completing this project is that it provided flexibility and allowed me to use different features such as libraries and API that only python can provide. Python is a very powerful language and in its way very simplistic, this has been beneficial as if during the process there was a part of the program which was giving errors, there were many resources online. Based on the research which was undertaken python was mainly used for building similar tools. Some of the benefits of python are that it makes debugging easier as well as easy to learn and implement, compared to languages such as C and Java. Python is allowing its users to have better productivity and speed efficiency when building the solution. The Python IDE is utilized for the solution in Visual Studio Code. The reason why Visual studio code was utilized is that it is lightweight, and most python packages are easily downloadable from the marketplace section in the IDE compared to Pycharm or other IDEs that cover a lot of memory in the system.

- Streamlit API Toolkit as Web application Framework

These trained models are embedded in a web application developed using the Streamlit framework in Python programming language. The first step before implementing the proposed solution was to decide which framework to use for building the solution on a web-based system. The two frameworks which were best suited for this project were Streamlit and Django. Streamlit framework fastens the development speed compared to Django because of the API calls to the Streamlit components. The responsiveness and efficiency with which the website is deployed using Streamlit are admirable. Furthermore, using the Streamlit application for the web application there are not many complexities compared to a desktop program. It is easy to use, from development to deployment, making it suitable for the solution. Overall, the reason behind the decision of choosing the Streamlit Framework is because of the usability and the ease of use, furthermore, reflecting on the users' requirements as users would like to have a simple GUI for navigating.

- Plotly's Python APIs for Interactive Graph Plotting

Plotly or Plotly.js module is the best choice for JavaScript or Python programs that employ charting and graphing techniques. Plotly.js renders visuals using WebGL and D3.js (SVG). The "all-in-one package" Plotly.js includes d3.js and stack.gl modules. It utilizes the JSON Structure. Basic, analytical, scientific, business, statistical, and geographic charts are all supported by plotly.js. Additionally, its open-source GitHub has more than 9000 stars, which is a good sign of the community's expansion. Plotly can be utilized for any data visualization project for multiple reasons which makes the development of various visualization aspects such as charting quite easy for the developers. We utilized this module to build up the visualization charts of the predictions done by the trained models.

4.2 MODEL DEVELOPMENT

For the implementation, 4 datasets in total have been used. They are TESS, RAVDESS, SAVEE, and CREMA-D. A few details of the four datasets are described below:

4.2.1 TESS:

The Toronto Emotional Speech Set or TESS is a dataset containing acted-out audio samples. The samples included are acted out by two females and the primary difference and essentially the motive for anyone using TESS comes from the age difference of the two females, where one of them is aged 20 while the other one is aged 60. Both actors acted out 7 emotions in a total of 200 sentences the 7 emotions being angry, pleasantly surprised, happy, disgusted, sad, fearful, and neutral.

4.2.2 RAVDESS

The Ryerson Audio-Visual Database of Emotional Speech and Song or RAVDESS is perhaps the most widely used database for emotional speech recognition and rightly so, as it covers 8 emotions: happy, angry, sad, disgusted, surprised, calm, fearful, and neutral emotions. Moreover, all these emotions are expressed in two different levels of intensity by all 24 actors who have performed 104 sentences each, which culminates into 2496 clips of richly varied data!

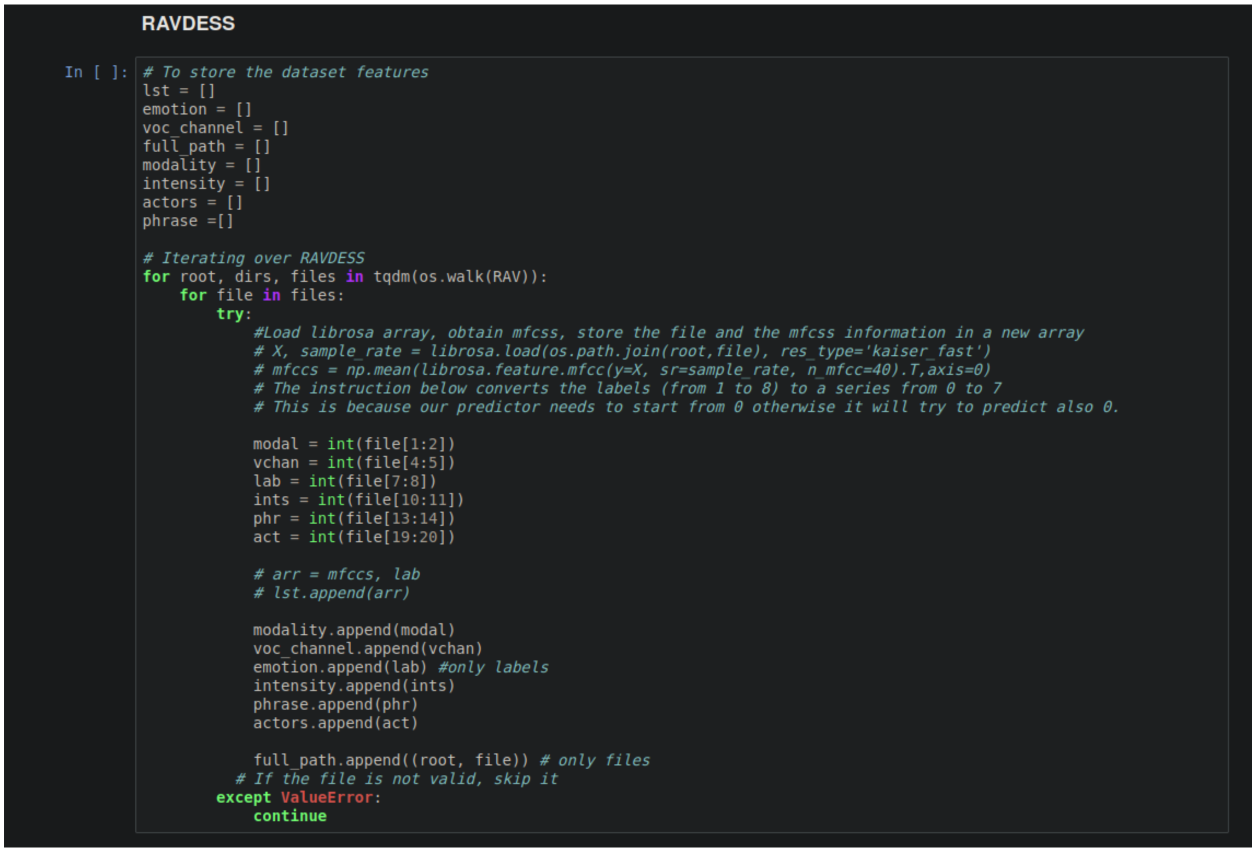
4.2.3 SAVEE

The Surrey Audio-Visual Expressed Emotion dataset, put together by the University of Surrey, consists of samples acted out by 4 male actors. The samples vary in 7 emotions including anger, fear, sadness, happiness, surprise, disgust, and neutral, and contain a total of 480 samples. This dataset primarily contains the British English accent samples. Recorded recently with high-quality equipment, the SAVEE dataset offers comparably very good quality audio samples compared to other datasets.

4.2.4 CREMA-D

CRowd-sourced Emotional Multimodal Actors Dataset or CREMA-D is another widely used dataset as it is also very rich in variation, containing 7442 samples from 91 actors from various ethnic backgrounds! Owing to this, as the name suggests, this dataset is very handy for multimodal emotional detection. The crowd-sourcing was used for labelling the samples, which was carried out by 2443 raters to be exact.

4.3 Dataset Pre-processing

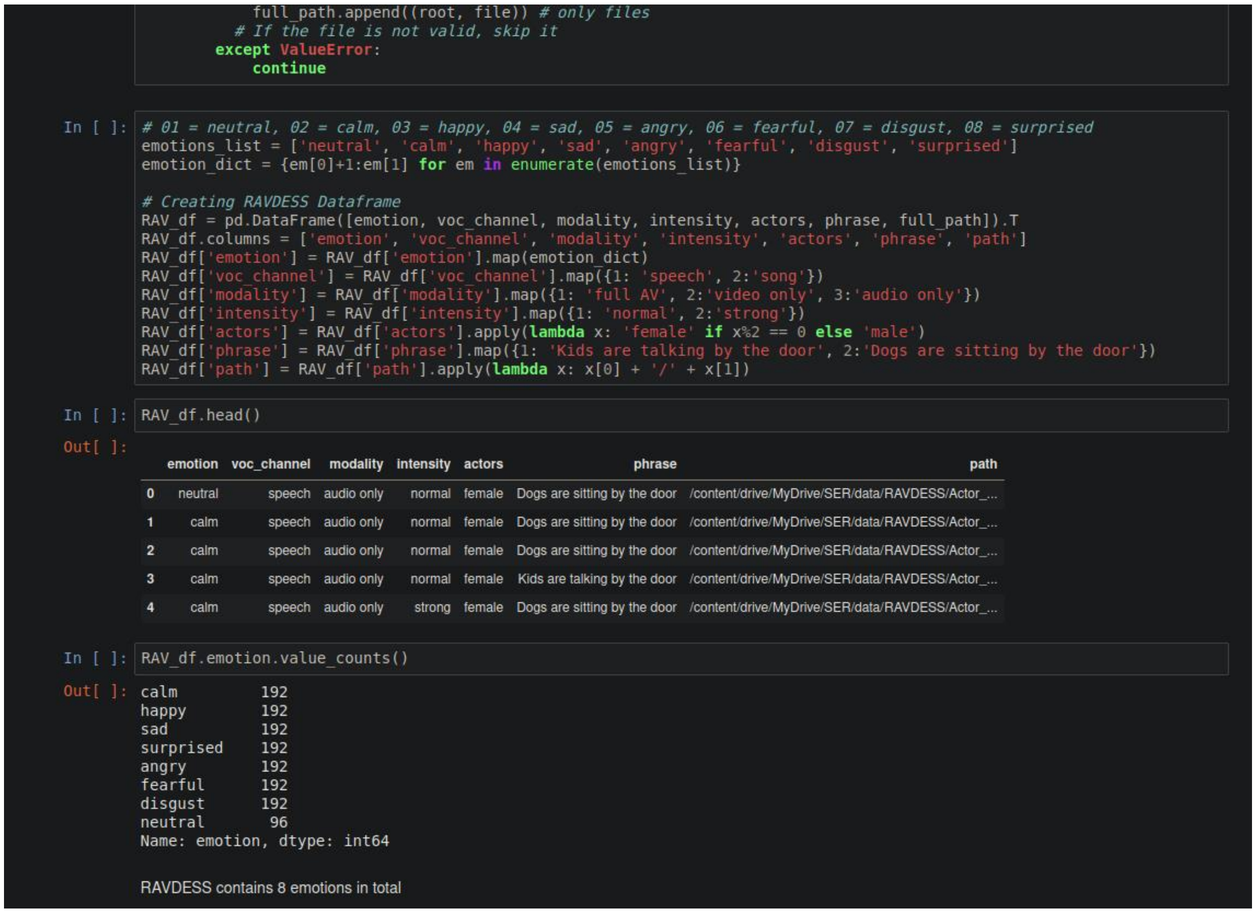


RAVDESS

```
In [ ]: # To store the dataset features
        lst = []
        emotion = []
        voc_channel = []
        full_path = []
        modality = []
        intensity = []
        actors = []
        phrase =[]

        # Iterating over RAVDESS
        for root, dirs, files in tqdm(os.walk(RAV)):
            for file in files:
                try:
                    #Load librosa array, obtain mfcss, store the file and the mfcss information in a new array
                    # X, sample_rate = librosa.load(os.path.join(root,file), res_type='kaiser_fast')
                    # mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T,axis=0)
                    # The instruction below converts the labels (from 1 to 8) to a series from 0 to 7
                    # This is because our predictor needs to start from 0 otherwise it will try to predict also 0.

                    modal = int(file[1:2])
                    vchan = int(file[4:5])
                    lab = int(file[7:8])
                    ints = int(file[10:11])
                    phr = int(file[13:14])
                    act = int(file[19:20])

                    # arr = mfccs, lab
                    # lst.append(arr)

                    modality.append(modal)
                    voc_channel.append(vchan)
                    emotion.append(lab) #only labels
                    intensity.append(ints)
                    phrase.append(phr)
                    actors.append(act)

                    full_path.append((root, file)) # only files
                  # If the file is not valid, skip it
                except ValueError:
                    continue
```

figure 3: screenshot of code iterating over the RAVDESS database to store required parameters

After loading the RAVDESS dataset, the values of modality, the channel of the message, its emotion, intensity in which the message was recorded, the phrase that was recorded, and the actors involved in one list are collected where the values from certain rows in the different audio files for each are appended. It can also be seen that the values are first stored in temporary variables before they are appended to the list. If an invalid file throws a value error, the particular iteration is ended.



```
                full_path.append((root, file)) # only files
              # If the file is not valid, skip it
            except ValueError:
                continue
```

```
In [ ]: # 01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised
        emotions_list = ['neutral', 'calm', 'happy', 'sad', 'angry', 'fearful', 'disgust', 'surprised']
        emotion_dict = {em[0]+1:em[1] for em in enumerate(emotions_list)}

        # Creating RAVDESS Dataframe
        RAV_df = pd.DataFrame([emotion, voc_channel, modality, intensity, actors, phrase, full_path]).T
        RAV_df.columns = ['emotion', 'voc_channel', 'modality', 'intensity', 'actors', 'phrase', 'path']
        RAV_df['emotion'] = RAV_df['emotion'].map(emotion_dict)
        RAV_df['voc_channel'] = RAV_df['voc_channel'].map({1: 'speech', 2:'song'})
        RAV_df['modality'] = RAV_df['modality'].map({1: 'full AV', 2:'video only', 3:'audio only'})
        RAV_df['intensity'] = RAV_df['intensity'].map({1: 'normal', 2:'strong'})
        RAV_df['actors'] = RAV_df['actors'].apply(lambda x: 'female' if x%2 == 0 else 'male')
        RAV_df['phrase'] = RAV_df['phrase'].map({1: 'Kids are talking by the door', 2:'Dogs are sitting by the door'})
        RAV_df['path'] = RAV_df['path'].apply(lambda x: x[0] + '/' + x[1])
```

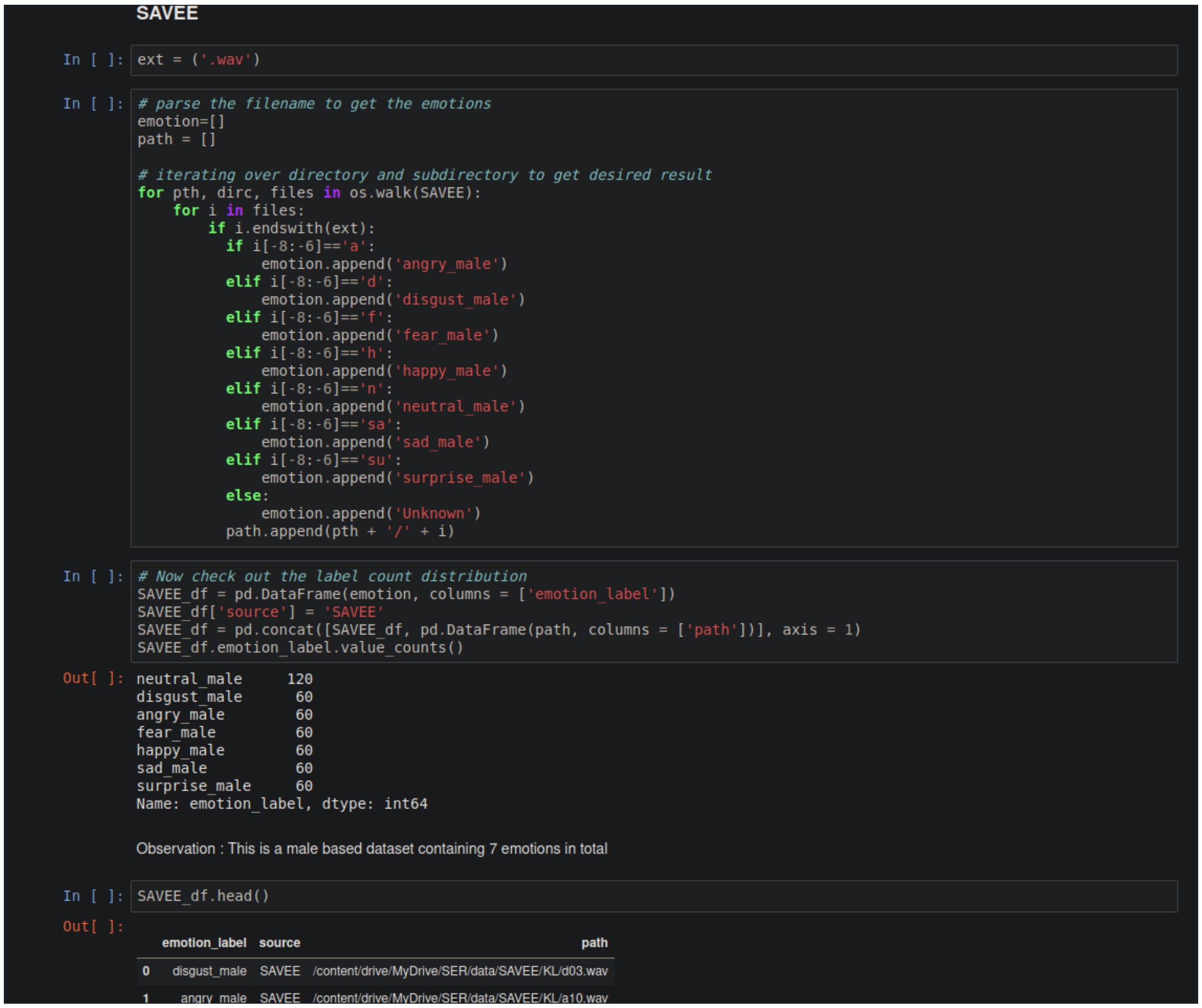
```
In [ ]: RAV_df.head()
Out[ ]:
```

	emotion	voc_channel	modality	intensity	actors	phrase	path
0	neutral	speech	audio only	normal	female	Dogs are sitting by the door	/content/drive/MyDrive/SER/data/RAVDESS/Actor_...
1	calm	speech	audio only	normal	female	Dogs are sitting by the door	/content/drive/MyDrive/SER/data/RAVDESS/Actor_...
2	calm	speech	audio only	normal	female	Dogs are sitting by the door	/content/drive/MyDrive/SER/data/RAVDESS/Actor_...
3	calm	speech	audio only	normal	female	Kids are talking by the door	/content/drive/MyDrive/SER/data/RAVDESS/Actor_...
4	calm	speech	audio only	strong	female	Dogs are sitting by the door	/content/drive/MyDrive/SER/data/RAVDESS/Actor_...

```
In [ ]: RAV_df.emotion.value_counts()
Out[ ]: calm         192
        happy        192
        sad          192
        surprised    192
        angry        192
        fearful      192
        disgust      192
        neutral       96
        Name: emotion, dtype: int64

        RAVDESS contains 8 emotions in total
```

figure 4: Creating a dataframe specific to the RAVDESS database

After collecting the data, a new dataframe for RAVDESS is prepared, consisting of all the collected details like emotion, channel, modality, intensity, and so on. Then, that is split into columns in the newly created dataframe and mapped emotions based on the number they have with the help of the emotion dictionary created just above the dataframe. Similarly, channel and modality are mapped on and so on, and since there are not many options available, rather than creating a dictionary, the values are directly mentioned while mapping them to the newly created dataframe. The obtained results are shown by printing the head of the dataframe created as such.



SAVEE

```
In [ ]: ext = ('.wav')
```

```
In [ ]: # parse the filename to get the emotions
        emotion=[]
        path = []

        # iterating over directory and subdirectory to get desired result
        for pth, dirc, files in os.walk(SAVEE):
            for i in files:
                if i.endswith(ext):
                    if i[-8:-6]=='a':
                        emotion.append('angry_male')
                    elif i[-8:-6]=='d':
                        emotion.append('disgust_male')
                    elif i[-8:-6]=='f':
                        emotion.append('fear_male')
                    elif i[-8:-6]=='h':
                        emotion.append('happy_male')
                    elif i[-8:-6]=='n':
                        emotion.append('neutral_male')
                    elif i[-8:-6]=='sa':
                        emotion.append('sad_male')
                    elif i[-8:-6]=='su':
                        emotion.append('surprise_male')
                    else:
                        emotion.append('Unknown')
                    path.append(pth + '/' + i)
```

```
In [ ]: # Now check out the label count distribution
        SAVEE_df = pd.DataFrame(emotion, columns = ['emotion_label'])
        SAVEE_df['source'] = 'SAVEE'
        SAVEE_df = pd.concat([SAVEE_df, pd.DataFrame(path, columns = ['path'])], axis = 1)
        SAVEE_df.emotion_label.value_counts()

Out[ ]: neutral_male     120
        disgust_male      60
        angry_male        60
        fear_male         60
        happy_male        60
        sad_male          60
        surprise_male     60
        Name: emotion_label, dtype: int64
```

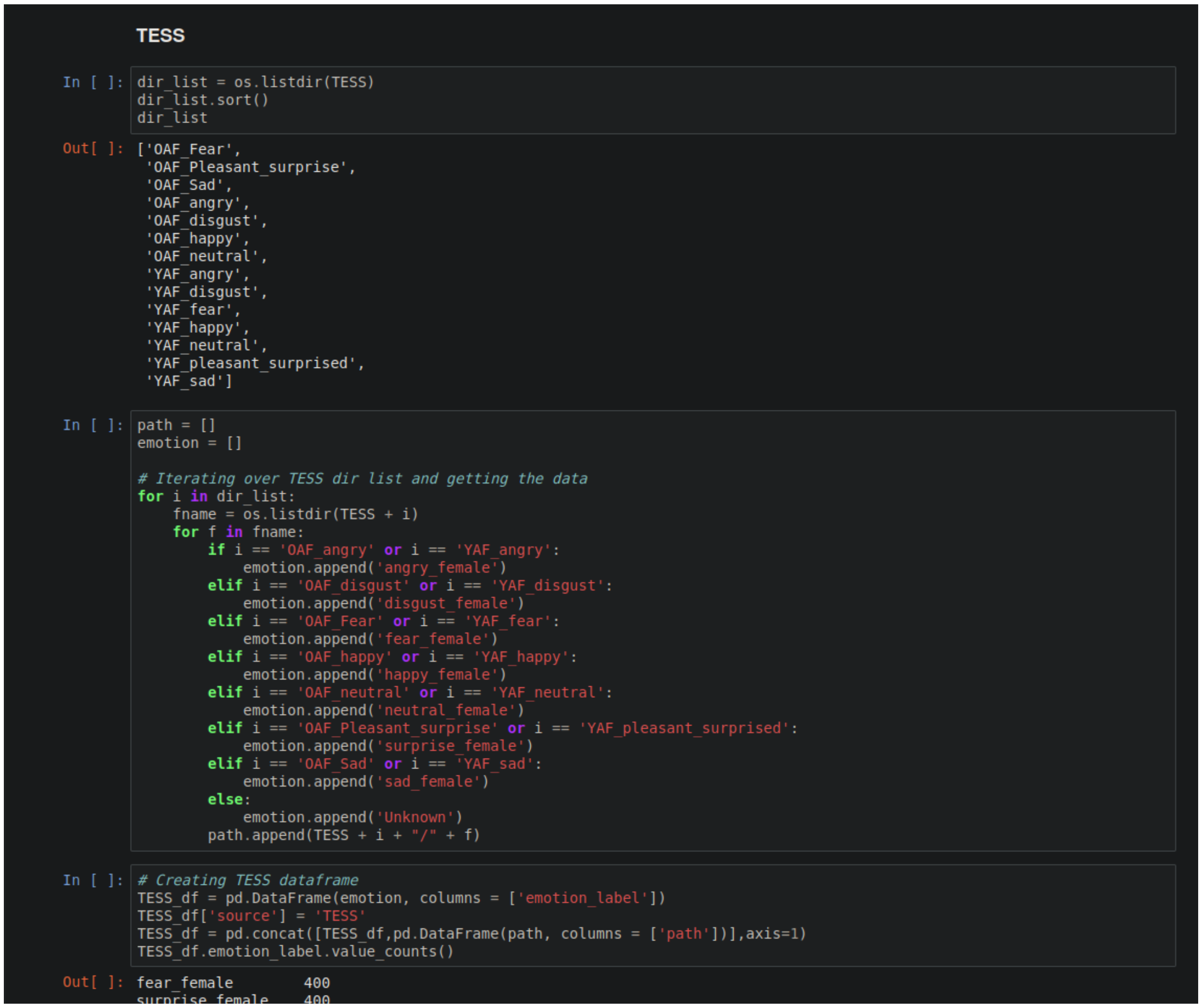
Observation : This is a male based dataset containing 7 emotions in total

```
In [ ]: SAVEE_df.head()
Out[ ]:
```

	emotion_label	source	path
0	disgust_male	SAVEE	/content/drive/MyDrive/SER/data/SAVEE/KL/d03.wav
1	angry_male	SAVEE	/content/drive/MyDrive/SER/data/SAVEE/KL/a10.wav

figure 5: Iterating over the SAVEE dataset for required parameters and adding them to a dataframe.

Similarly, for SAVEE, files and directory are iterated over again but this time, instead of mapping emotions to a number as was the case with RAVDESS, there is a distinct letter at the end of the categorizing text, identifying that leads to the corresponding emotion, which is shown in the first block of code in the above screenshot. After that, a new SAVEE data frame is created with an emotion label in one column, a source in another, and the full path in the last column.

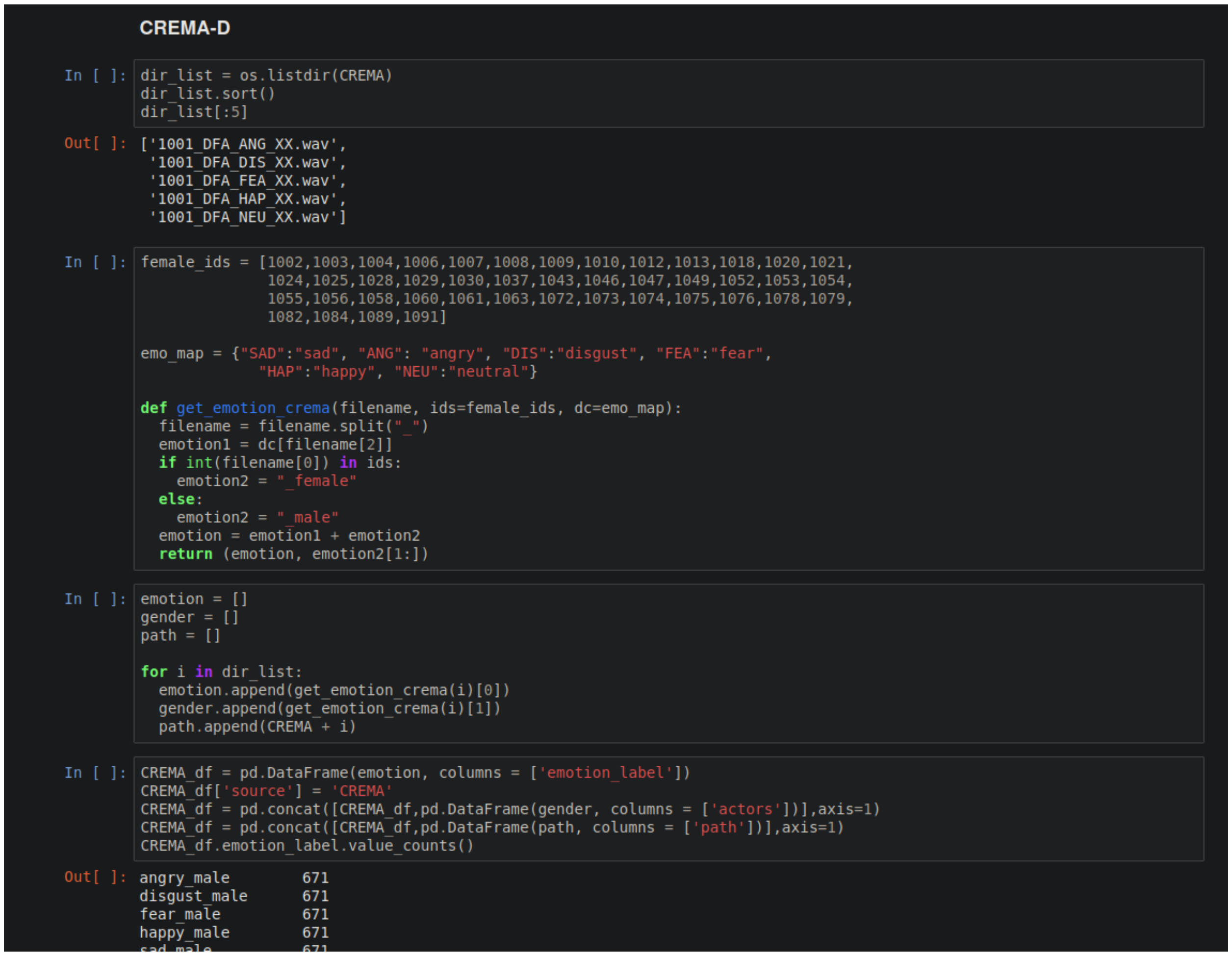


TESS

```
In [ ]: dir_list = os.listdir(TESS)
        dir_list.sort()
        dir_list

Out[ ]: ['OAF_Fear',
         'OAF_Pleasant_surprise',
         'OAF_Sad',
         'OAF_angry',
         'OAF_disgust',
         'OAF_happy',
         'OAF_neutral',
         'YAF_angry',
         'YAF_disgust',
         'YAF_fear',
         'YAF_happy',
         'YAF_neutral',
         'YAF_pleasant_surprised',
         'YAF_sad']

In [ ]: path = []
        emotion = []

        # Iterating over TESS dir list and getting the data
        for i in dir_list:
            fname = os.listdir(TESS + i)
            for f in fname:
                if i == 'OAF_angry' or i == 'YAF_angry':
                    emotion.append('angry_female')
                elif i == 'OAF_disgust' or i == 'YAF_disgust':
                    emotion.append('disgust_female')
                elif i == 'OAF_Fear' or i == 'YAF_fear':
                    emotion.append('fear_female')
                elif i == 'OAF_happy' or i == 'YAF_happy':
                    emotion.append('happy_female')
                elif i == 'OAF_neutral' or i == 'YAF_neutral':
                    emotion.append('neutral_female')
                elif i == 'OAF_Pleasant_surprise' or i == 'YAF_pleasant_surprised':
                    emotion.append('surprise_female')
                elif i == 'OAF_Sad' or i == 'YAF_sad':
                    emotion.append('sad_female')
                else:
                    emotion.append('Unknown')
                path.append(TESS + i + "/" + f)

In [ ]: # Creating TESS dataframe
        TESS_df = pd.DataFrame(emotion, columns = ['emotion_label'])
        TESS_df['source'] = 'TESS'
        TESS_df = pd.concat([TESS_df,pd.DataFrame(path, columns = ['path'])],axis=1)
        TESS_df.emotion_label.value_counts()

Out[ ]: fear_female        400
```

figure 6: Iterating over the TESS dataset for required parameters and adding them to a dataframe.

TESS has a lot of directories each containing a particular emotion and is categorized further by whether the audio sample is generated by an old actor or a young actor. After that, the classification of emotions is carried out in a fashion quite similar to that done for SAVEE, where instead of a categorizing letter, there is a complete string of letters, based on which the emotions are being appended as shown. Note that, an unknown emotion is also appended in case no well-defined string of letters is found. After that, again in a fashion similar to that carried out in SAVEE, the author creates a dataframe with columns of emotion labels, mentioning the source, and finally giving the complete path to the audio sample in the last column.

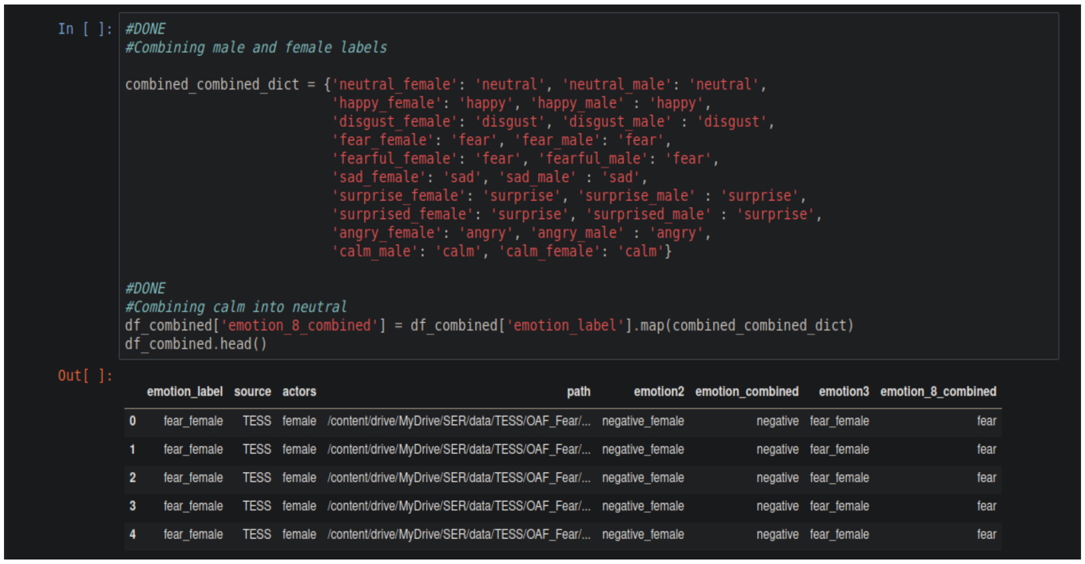


CREMA-D

```
In [ ]: dir_list = os.listdir(CREMA)
        dir_list.sort()
        dir_list[:5]

Out[ ]: ['1001_DFA_ANG_XX.wav',
         '1001_DFA_DIS_XX.wav',
         '1001_DFA_FEA_XX.wav',
         '1001_DFA_HAP_XX.wav',
         '1001_DFA_NEU_XX.wav']

In [ ]: female_ids = [1002,1003,1004,1006,1007,1008,1009,1010,1012,1013,1018,1020,1021,
                      1024,1025,1028,1029,1030,1037,1043,1046,1047,1049,1052,1053,1054,
                      1055,1056,1058,1060,1061,1063,1072,1073,1074,1075,1076,1078,1079,
                      1082,1084,1089,1091]

        emo_map = {"SAD":"sad", "ANG": "angry", "DIS":"disgust", "FEA":"fear",
                   "HAP":"happy", "NEU":"neutral"}

        def get_emotion_crema(filename, ids=female_ids, dc=emo_map):
          filename = filename.split("_")
          emotion1 = dc[filename[2]]
          if int(filename[0]) in ids:
            emotion2 = "_female"
          else:
            emotion2 = "_male"
          emotion = emotion1 + emotion2
          return (emotion, emotion2[1:])

In [ ]: emotion = []
        gender = []
        path = []

        for i in dir_list:
          emotion.append(get_emotion_crema(i)[0])
          gender.append(get_emotion_crema(i)[1])
          path.append(CREMA + i)

In [ ]: CREMA_df = pd.DataFrame(emotion, columns = ['emotion_label'])
        CREMA_df['source'] = 'CREMA'
        CREMA_df = pd.concat([CREMA_df,pd.DataFrame(gender, columns = ['actors'])],axis=1)
        CREMA_df = pd.concat([CREMA_df,pd.DataFrame(path, columns = ['path'])],axis=1)
        CREMA_df.emotion_label.value_counts()

Out[ ]: angry_male        671
        disgust_male      671
        fear_male         671
        happy_male        671
```

figure 7: Iterating over the CREMA-D dataset for required parameters and adding them to a database.

For CREMA-D, there is mapping carried out based on ids given to different male and female actors, of which the female ids are listed here in the cell output, the purpose of which will be clear soon, as well as based on a string of letters representing a particular emotion. The mapping of the string of 3 letters here is carried out to the previously used emotion keywords in the emo map to keep uniformity. Then, get_emotion_crema function is used to get not only emotion but also whether the actor in the audio sample was male or female by checking if the parsed id while calling the function is contained in the list female_ids created above. The emotion is directly picked up from the filename as is visible. After these details are extracted and appended to the frame, the path is also appended and the process of creating a dataframe is begun, similar as already shared above. The first column would contain the emotion label obtained, the second would contain the source, which would be CREMA-D here, and the final column would give the full path to the audio sample.

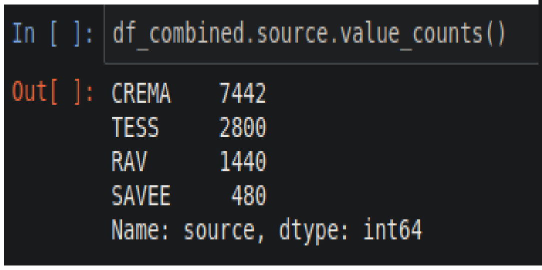


```
In [ ]: #DONE
        #Combining male and female labels

        combined_combined_dict = {'neutral_female': 'neutral', 'neutral_male': 'neutral',
                                  'happy_female': 'happy', 'happy_male' : 'happy',
                                  'disgust_female': 'disgust', 'disgust_male' : 'disgust',
                                  'fear_female': 'fear', 'fear_male': 'fear',
                                  'fearful_female': 'fear', 'fearful_male': 'fear',
                                  'sad_female': 'sad', 'sad_male' : 'sad',
                                  'surprise_female': 'surprise', 'surprise_male' : 'surprise',
                                  'surprised_female': 'surprise', 'surprised_male' : 'surprise',
                                  'angry_female': 'angry', 'angry_male' : 'angry',
                                  'calm_male': 'calm', 'calm_female': 'calm'}

        #DONE
        #Combining calm into neutral
        df_combined['emotion_8_combined'] = df_combined['emotion_label'].map(combined_combined_dict)
        df_combined.head()
Out[ ]:
```

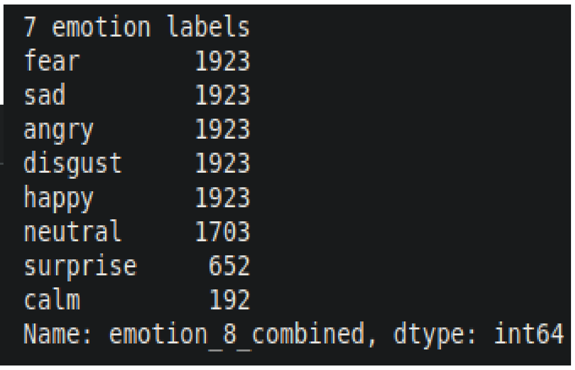
	emotion_label	source	actors	path	emotion2	emotion_combined	emotion3	emotion_8_combined
0	fear_female	TESS	female	/content/drive/MyDrive/SER/data/TESS/OAF_Fear/...	negative_female	negative	fear_female	fear
1	fear_female	TESS	female	/content/drive/MyDrive/SER/data/TESS/OAF_Fear/...	negative_female	negative	fear_female	fear
2	fear_female	TESS	female	/content/drive/MyDrive/SER/data/TESS/OAF_Fear/...	negative_female	negative	fear_female	fear
3	fear_female	TESS	female	/content/drive/MyDrive/SER/data/TESS/OAF_Fear/...	negative_female	negative	fear_female	fear
4	fear_female	TESS	female	/content/drive/MyDrive/SER/data/TESS/OAF_Fear/...	negative_female	negative	fear_female	fear

*figure 8: Combining previously made dataframes into one***.**

These all datasets were combined in the final dataframe after processing each of them individually. To combine them, the respective male and female labels were combined first and those labels were mapped to the corresponding emotion class.

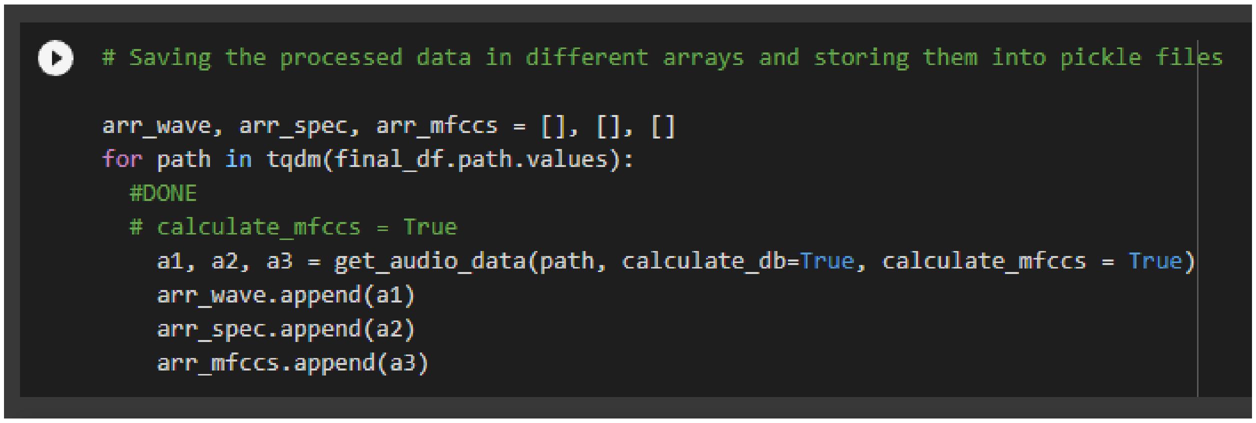


```
In [ ]: df_combined.source.value_counts()

Out[ ]: CREMA    7442
        TESS     2800
        RAV      1440
        SAVEE     480
        Name: source, dtype: int64
```



```
7 emotion labels
fear        1923
sad         1923
angry       1923
disgust     1923
happy       1923
neutral     1703
surprise     652
calm         192
Name: emotion_8_combined, dtype: int64
```

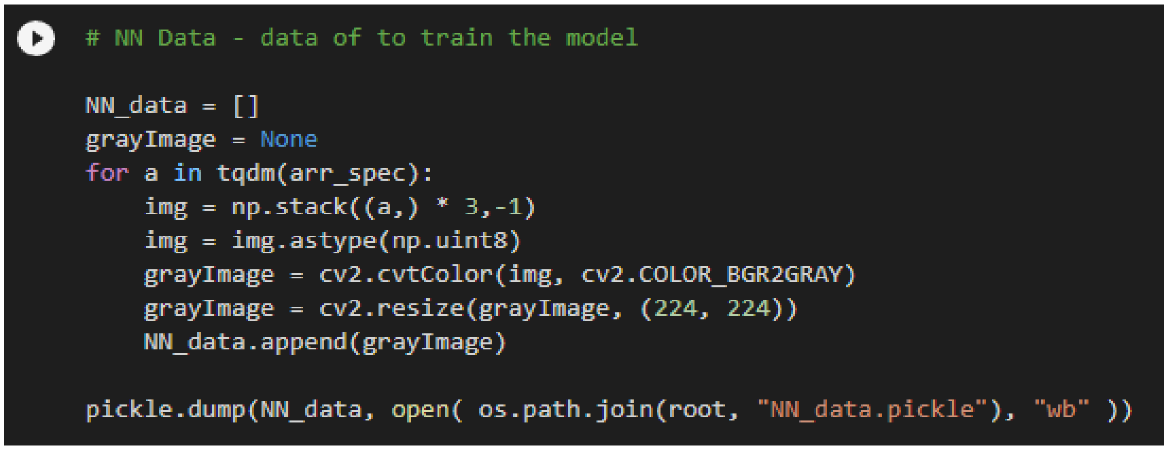
After combining, this is the proportion of the entries from the respective dataset and the respective emotion classes in the final dataframe.



```
# Saving the processed data in different arrays and storing them into pickle files

arr_wave, arr_spec, arr_mfccs = [], [], []
for path in tqdm(final_df.path.values):
  #DONE
  # calculate_mfccs = True
    a1, a2, a3 = get_audio_data(path, calculate_db=True, calculate_mfccs = True)
    arr_wave.append(a1)
    arr_spec.append(a2)
    arr_mfccs.append(a3)
```

Figure 9: Code Snippet obtaining Mel Spectrograms and MFCC.

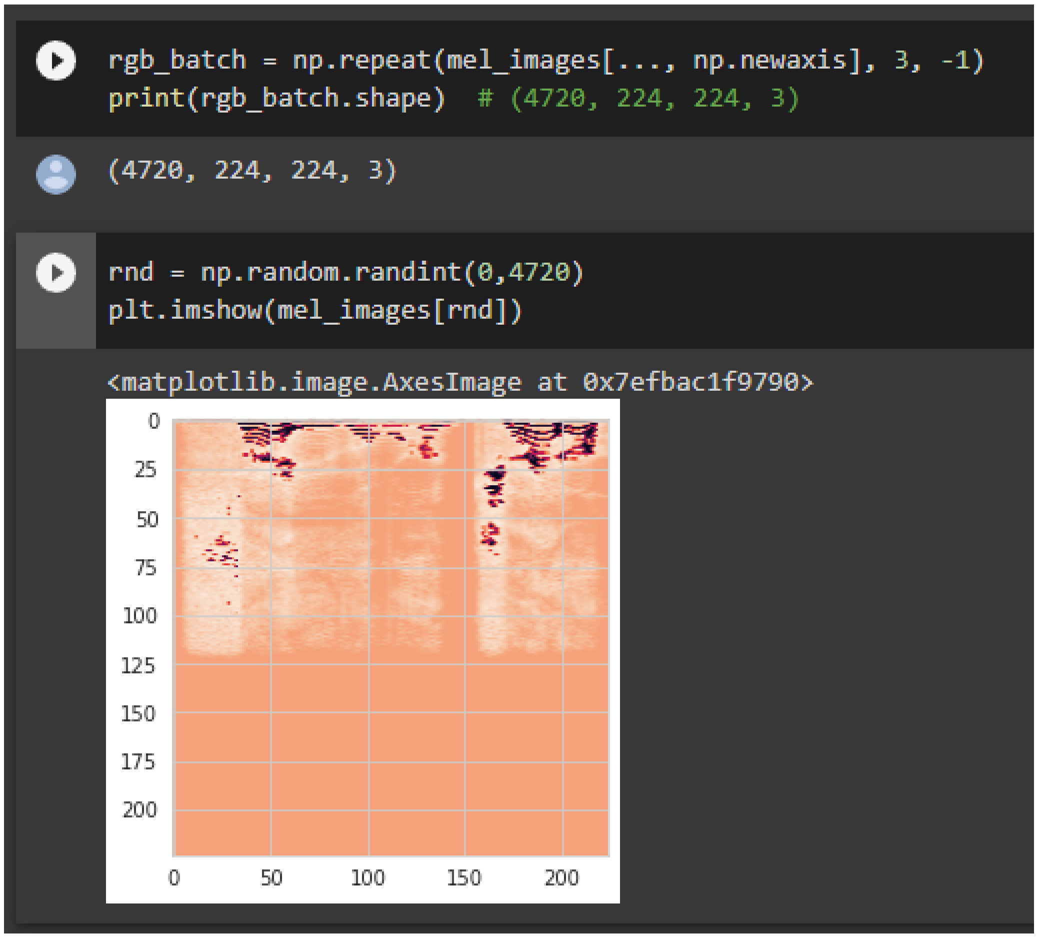
Once the final data frame is ready, the audio features such as Mel Spectrograms and MFCCs are extracted from the audio files of the final dataset. To fetch these features, the utility function ***get_audio_data*** is used which leverages a Python module named Librosa. These features are stored in different arrays which are processed further.



```
# NN Data - data of to train the model

NN_data = []
grayImage = None
for a in tqdm(arr_spec):
    img = np.stack((a,) * 3,-1)
    img = img.astype(np.uint8)
    grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    grayImage = cv2.resize(grayImage, (224, 224))
    NN_data.append(grayImage)

pickle.dump(NN_data, open( os.path.join(root, "NN_data.pickle"), "wb" ))
```

Figure 10: Transforming spectrogram images to grey and storing them in a batch.

Finally, the dataset to train the model is prepared by converting the spectrogram images into grayscale with dimensions of (224, 224) using the OpenCV library. The final grayscale images are appended in an array and are stored as a pickle file to use in batches while training the model.



```
rgb_batch = np.repeat(mel_images[..., np.newaxis], 3, -1)
print(rgb_batch.shape)  # (4720, 224, 224, 3)
```

```
(4720, 224, 224, 3)
```

```
rnd = np.random.randint(0,4720)
plt.imshow(mel_images[rnd])
```

```
<matplotlib.image.AxesImage at 0x7efbac1f9790>
```

figure 11: Random processed Mel Spectrogram

So, the final shape of the dataset becomes (4720, 224, 224, 3). The Mel spectrogram image of a random entry from the dataset is visualized in the figure above.

CHAPTER 5.

MODEL ARCHITECTURE

DenseNet-201 (Huang, Liu, Van Der Mateen, 2017)

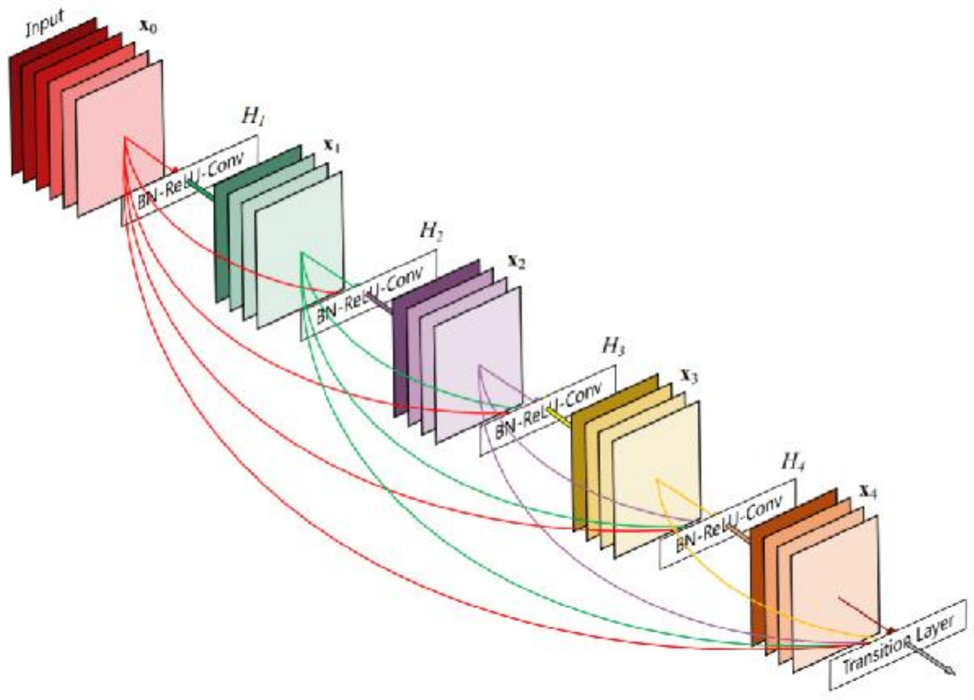
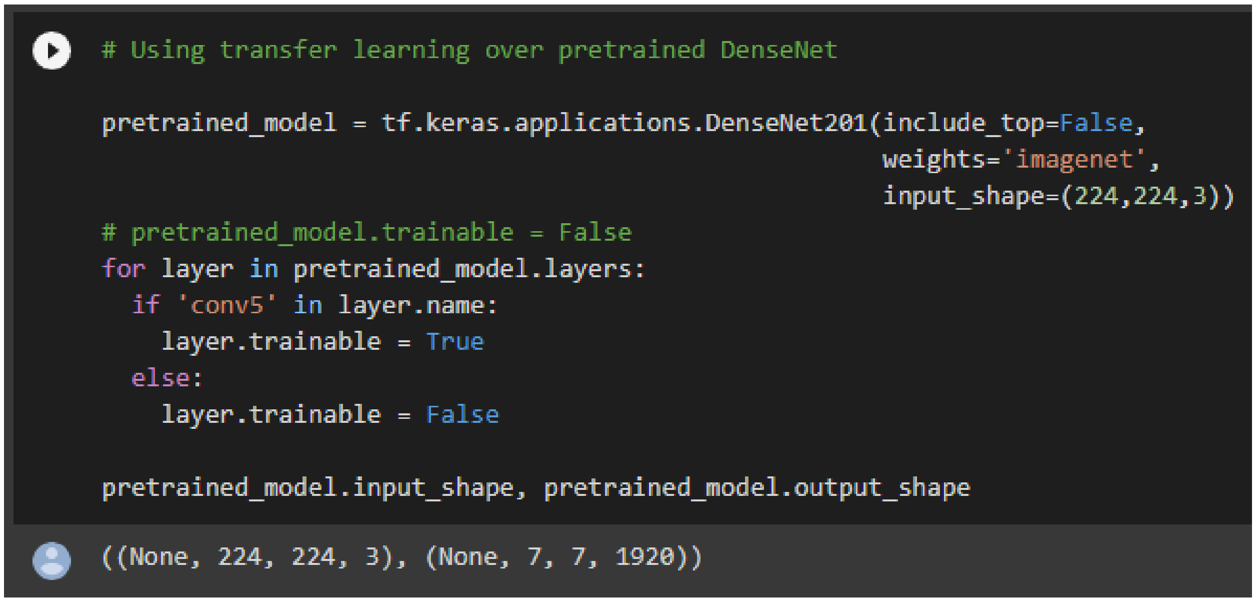


figure 12: Pictorial representation of DenseNet architecture with a 5-layer dense block and a growth rate of k=4.

Gao Huang et al proposed the Dense Convolutional Network, a novel convolutional neural network architecture also known as DenseNet for shorthand in their paper named “Densely Connected Convolutional Networks”. As can be seen from the figure above, every two consecutive layers with an equal feature map size in this architecture have direct links among them. They demonstrated that DenseNet scales up to thousands of layers without experiencing any scalability or optimization challenges. Thus, DenseNet-201 is a convolutional neural network architecture which is targeted to solve the tasks of image classification in a more scalable and optimized manner.



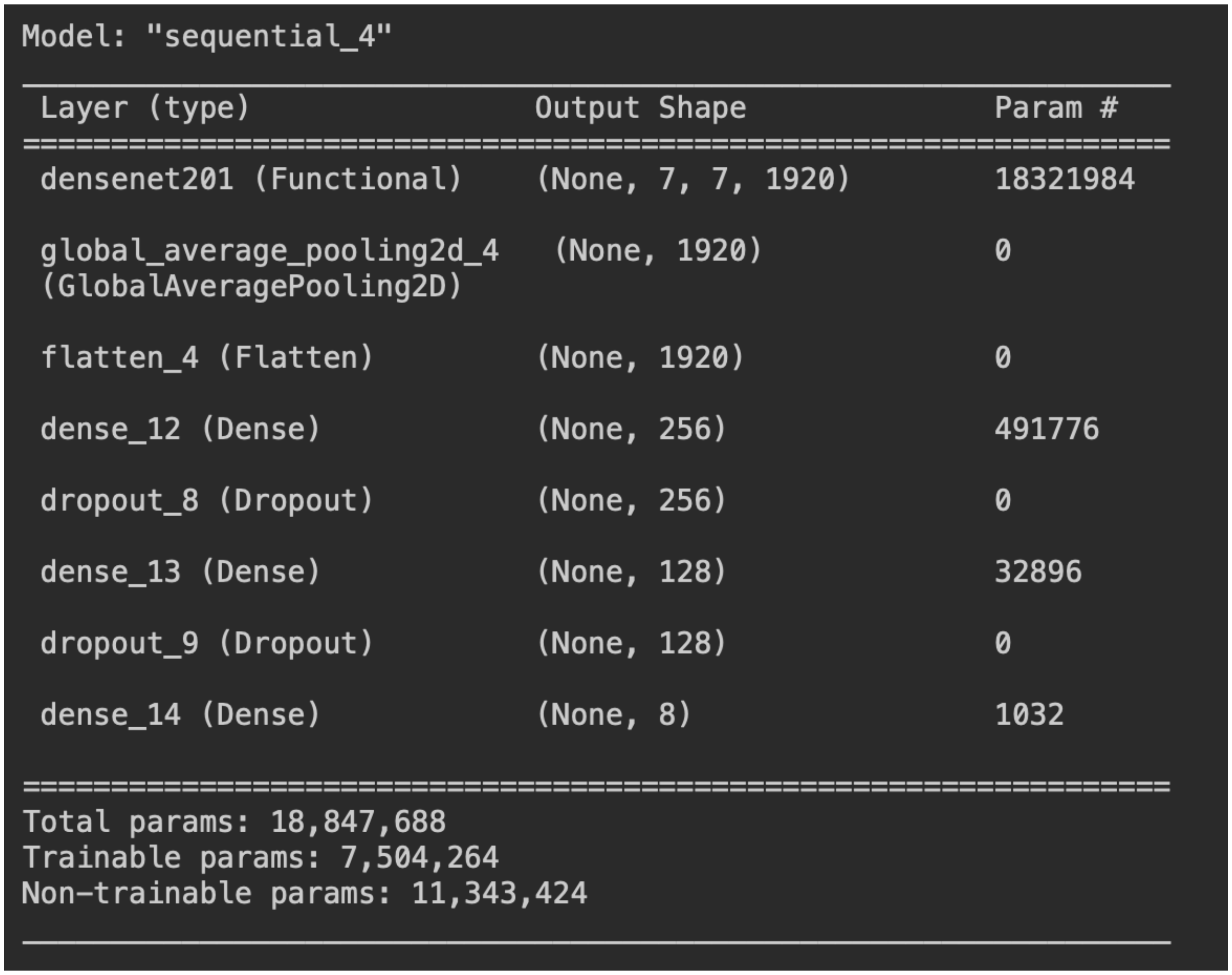
```
# Using transfer learning over pretrained DenseNet

pretrained_model = tf.keras.applications.DenseNet201(include_top=False,
                                                      weights='imagenet',
                                                      input_shape=(224,224,3))

# pretrained_model.trainable = False
for layer in pretrained_model.layers:
  if 'conv5' in layer.name:
    layer.trainable = True
  else:
    layer.trainable = False

pretrained_model.input_shape, pretrained_model.output_shape
```

```
((None, 224, 224, 3), (None, 7, 7, 1920))
```

figure 13: Using transfer learning over pre-trained DenseNet

As can be seen in the snippet above, the pre-trained DenseNet is set to accept the input image of dimension (224, 224, 3) which is the same as of entries in our dataset. The additional layers were added on top of the DenseNet-201 as can be seen in the architecture figures of each model mentioned below.

5.1 8 Emotion Model CNN



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet201 (Functional)    (None, 7, 7, 1920)        18321984

 global_average_pooling2d_4   (None, 1920)             0
 (GlobalAveragePooling2D)

 flatten_4 (Flatten)         (None, 1920)              0

 dense_12 (Dense)            (None, 256)               491776

 dropout_8 (Dropout)         (None, 256)               0

 dense_13 (Dense)            (None, 128)               32896

 dropout_9 (Dropout)         (None, 128)               0

 dense_14 (Dense)            (None, 8)                 1032

=================================================================
Total params: 18,847,688
Trainable params: 7,504,264
Non-trainable params: 11,343,424
_________________________________________________________________
```

figure 14: 8 emotion model confusion matrix

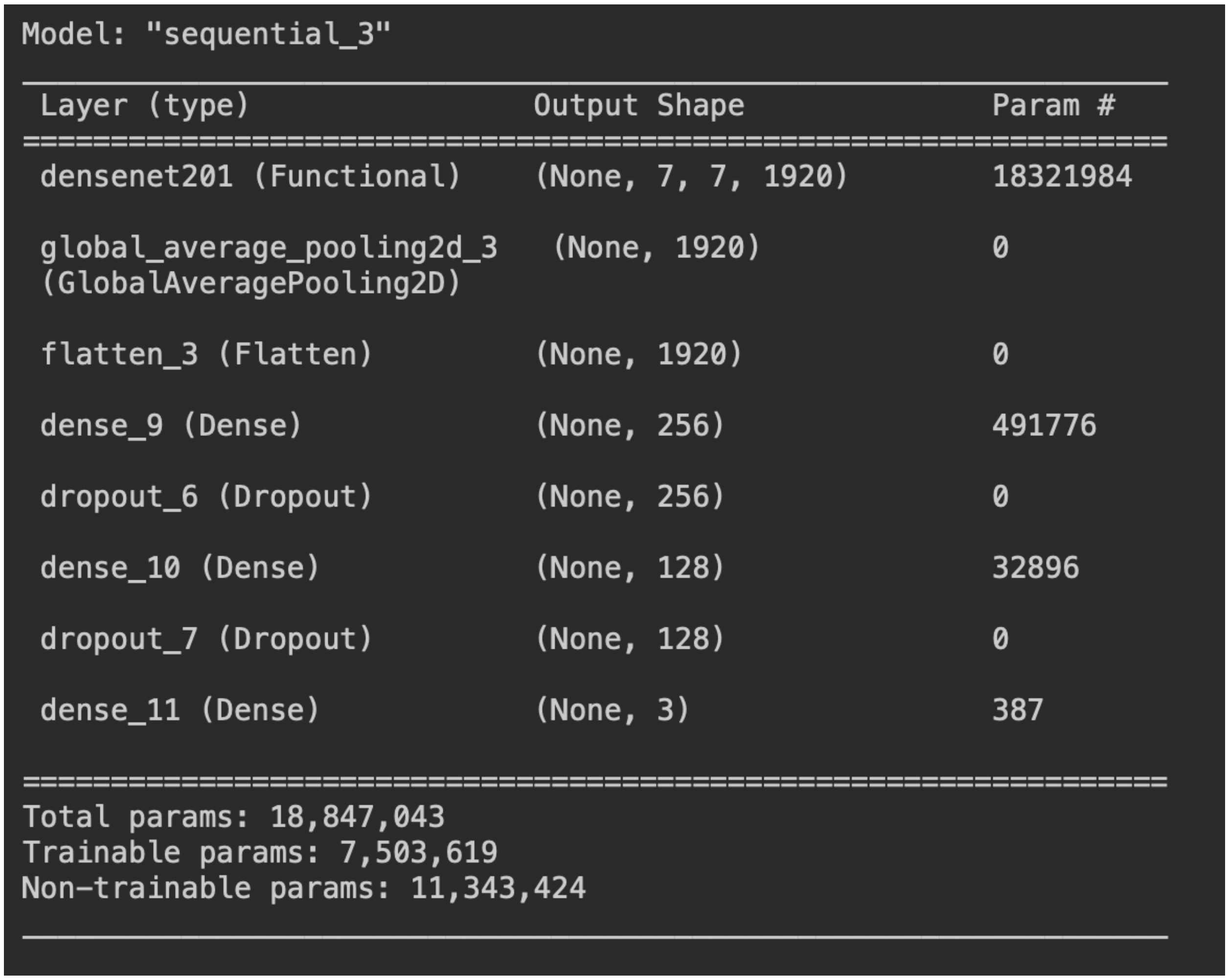
As can be seen in the figure above, the architecture of the model contains the additional layers on top of densenet201. The architecture is explained below:

- The first chunk is built up using Dense Net 201 which is a pre-trained CNN model which has the output shape of (7, 7, 1920).
- This output layer with the above shape is followed by the Global Average Pooling 2D layer with the same 1920 shape.

- The output of the Global Average Pooling 2D layer is flattened using the Flatten layer with the same dimensions.
- The flattened output is then passed on to the fully connected neural network which is made up of various layers of Dense + Dropout.
- The first two Dense + Dropout blocks have the neuron count as 256 and 128 respectively.
- The last Dense layer on the network acts as a classification head. Since the goal of the model is to classify 8 emotions of the given input speech, this classification head has 8 nodes as its output.

The total parameters of the model are 18, 847, 688 out of which the non-trainable parameters are 11, 343, 424 and trainable parameters are only 7, 504, 264.

5.1.1 3 Emotion Model CNN

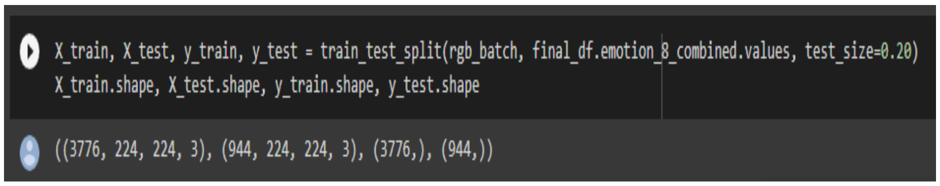


```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet201 (Functional)    (None, 7, 7, 1920)        18321984

 global_average_pooling2d_3   (None, 1920)             0
 (GlobalAveragePooling2D)

 flatten_3 (Flatten)         (None, 1920)              0

 dense_9 (Dense)             (None, 256)               491776

 dropout_6 (Dropout)         (None, 256)               0

 dense_10 (Dense)            (None, 128)               32896

 dropout_7 (Dropout)         (None, 128)               0

 dense_11 (Dense)            (None, 3)                 387

=================================================================
Total params: 18,847,043
Trainable params: 7,503,619
Non-trainable params: 11,343,424
_________________________________________________________________
```

figure 15: 3 emotion model confusion matrix

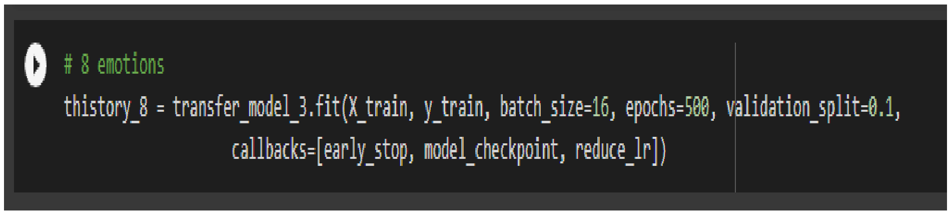
Except for the classification head, the 3-Emotion Model followed the same architecture as the 8-Emotion Model. The last layer had only differences in the classification head with the number of classes being 3 i.e., positive, negative and neutral.

5.2 Model Hyper Parameters:



```
X_train, X_test, y_train, y_test = train_test_split(rgb_batch, final_df.emotion_8_combined.values, test_size=0.20)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((3776, 224, 224, 3), (944, 224, 224, 3), (3776,), (944,))
```



```
# 8 emotions
thistory_8 = transfer_model_3.fit(X_train, y_train, batch_size=16, epochs=500, validation_split=0.1,
                      callbacks=[early_stop, model_checkpoint, reduce_lr])
```

Using an 80:20 split, the dataset was divided into train and test sets. The model was trained using 16 batches, 500 epochs, and a validation split of 0.1. The additional callbacks were added for efficient training which was early stopping for interrupting training while training is getting stagnant, model checkpoint to save the best weights and reduce or to control the learning rates.

5.2.1 Model Performance

8 Emotion Model CNN

Performance Metrics:

	precision	recall	f1-score	support
angry	0.95	0.89	0.92	125
calm	0.61	0.81	0.70	37
disgust	0.85	0.85	0.85	130
fear	0.83	0.81	0.82	138
happy	0.83	0.80	0.82	130
neutral	0.89	0.89	0.89	131
sad	0.84	0.84	0.84	121
surprise	0.85	0.88	0.87	132
accuracy			0.85	944
macro avg	0.83	0.85	0.84	944
weighted avg	0.85	0.85	0.85	944

Table 2: 8 Emotion Model CNN Performance Metrics

```
Accuracy: 0.8496
```

As we can see above, the 8-emotion model, layers of which were described and shown in the previous section, has high levels of accuracy of almost ~85%, which is lower than its 3-emotion model counterpart, but still considerably accurate nonetheless. We can also see that emotion detection for anger got the highest level of precision of 95%, closely followed by neutral emotion detection with 89%, while emotion detection of surprise, disgust and sadness came in third but still has a precision level close to 85%. While fear and happiness were also detected fairly precisely with 83%, it is evident that the calm emotion was the toughest to detect with a considerably lower 61% precision. And given that it was only the calm emotion that had a comparatively poor precision level, still a good weighted average precision and accuracy almost ~85% has been achieved.

Confusion Matrix:

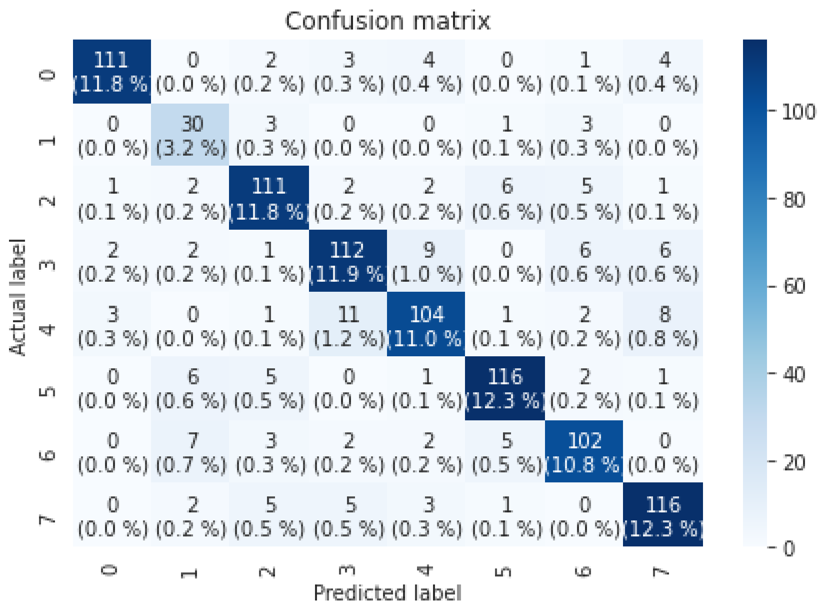


figure 16: 8 emotion model CNN confusion matrix

The labels 0, 1, 2,...,7 represent neutral, calm, happy, sad, angry, fearful, disgust, and surprised emotions respectively.

From the confusion matrix for the 8 emotion model, it is evident that the diagonal is quite well populated, which is a good sign that predicted and actual labels match for a large number of cases, to be exact, 11.8% (correctly predicted neutral emotions) + 3.2% (correctly predicted calm emotions) + 11.8% (correctly predicted happy emotions) + 11.9% (correctly predicted sad emotions) + 11.0% (correctly predicted angry emotions) + 12.3% (correctly predicted fearful emotions) + 10.8% (correctly predicted disgust emotions) + 12.3% (correctly predicted surprised emotions)= 85.1%. Moreover, it is also evident that the least false positives are obtained for calm and surprised emotion detection, whereas for disgust and sad emotion detection, the false positives are highest.

5.2.2 3 Emotion Model CNN

<u>Performance Metrics</u>:

	precision	recall	f1-score	support
negative	0.95	0.94	0.94	525
neutral	0.94	0.91	0.93	157
positive	0.91	0.95	0.93	262
accuracy			0.94	944
macro avg	0.93	0.93	0.93	944
weighted avg	0.94	0.94	0.94	944

Table 1: 3 Emotion Model CNN Performance Metrics

```
Accuracy: 0.9364
```

It is evident from the above figure, the 3-emotion model, layers of which were described and shown in the previous section, has considerably high levels of accuracy of almost ~94%, with negative emotion detection having the highest level of precision, closely followed by neutral emotion detection while positive emotion detection comes in third but still has a precision level of over 90%.

<u>Confusion Matrix</u>:

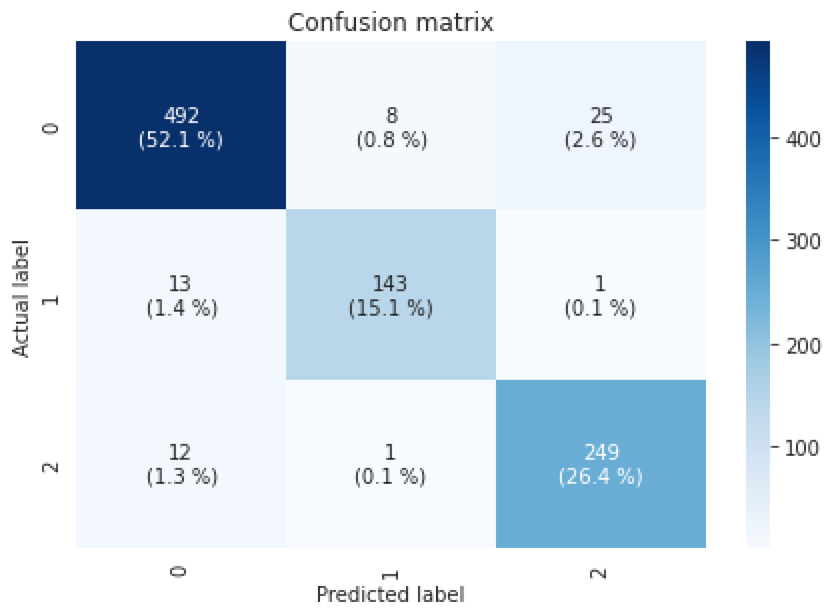
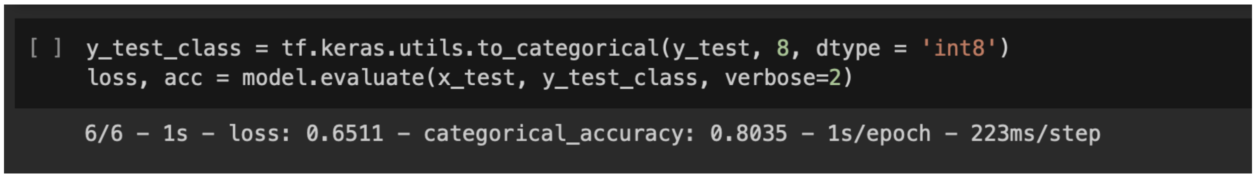


figure 17: 3 emotion model CNN confusion matrix

The labels 0,1, and 2 represent negative, neutral and positive emotions respectively

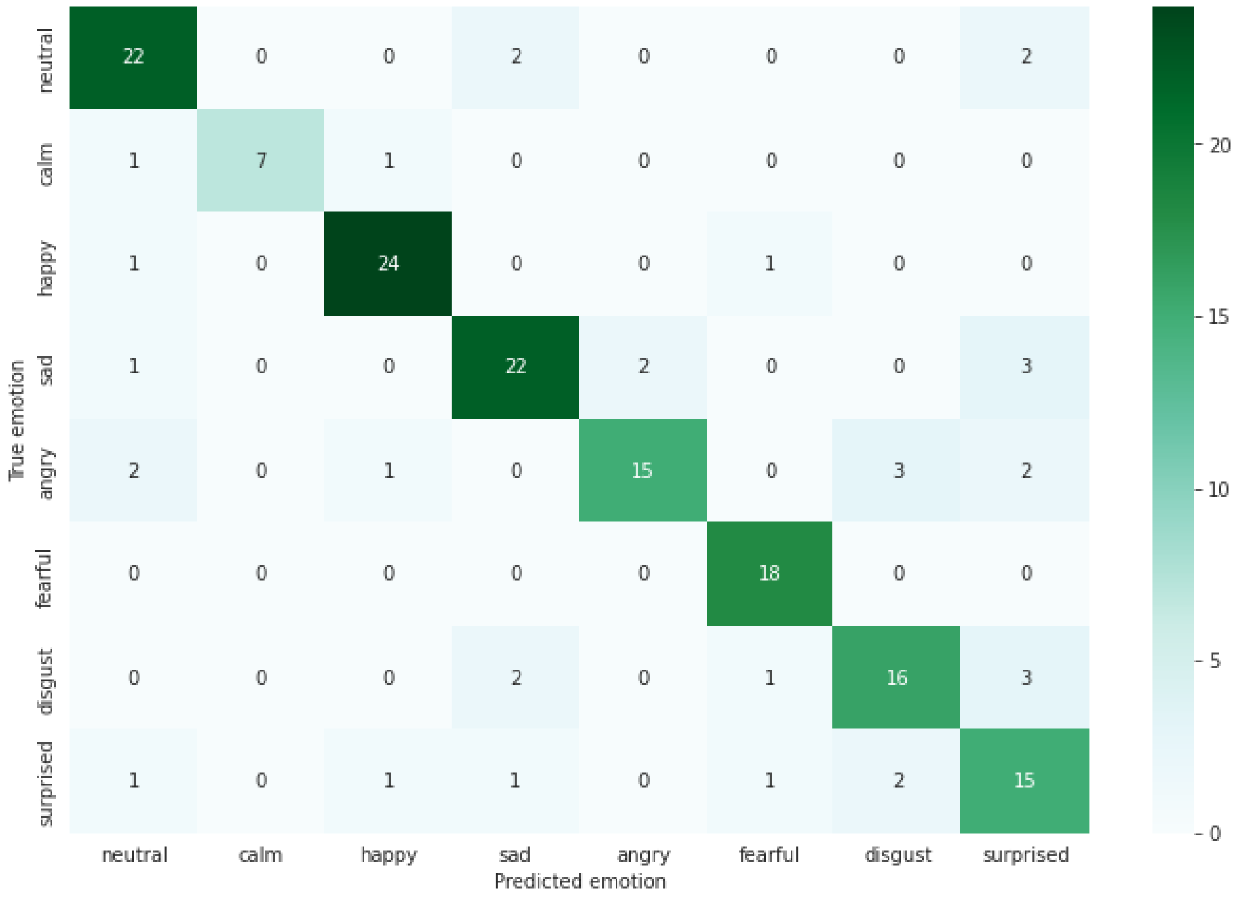
From the confusion matrix for the 3-emotion model, it is evident that the predicted and actual labels match for 52.1% (correctly predicted negative emotions) + 15.1% (correctly predicted neutral emotions) + 26.4% (correctly predicted positive emotions) = 83.6%. Moreover, it is also apparent that the least false positives are for neutral emotion detection, whereas for positive and negative emotion detection, the false positives are almost the same.

5.2.3 **8 Emotions Model LSTM:**



```
[ ] y_test_class = tf.keras.utils.to_categorical(y_test, 8, dtype = 'int8')
    loss, acc = model.evaluate(x_test, y_test_class, verbose=2)

    6/6 - 1s - loss: 0.6511 - categorical_accuracy: 0.8035 - 1s/epoch - 223ms/step
```

<u>Confusion Matrix</u>:

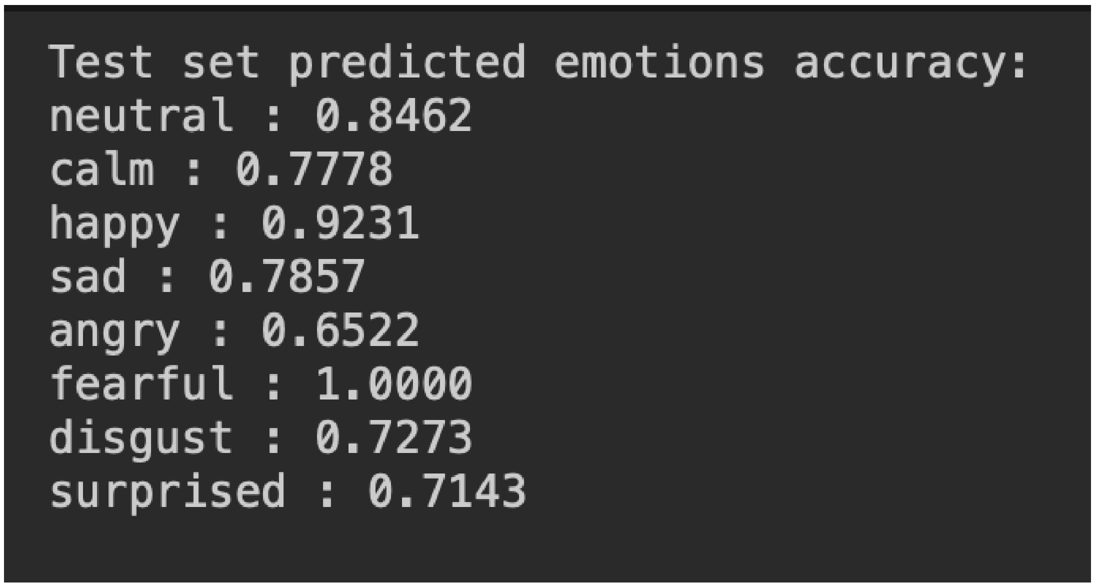


True emotion \ Predicted emotion	neutral	calm	happy	sad	angry	fearful	disgust	surprised
neutral	22	0	0	2	0	0	0	2
calm	1	7	1	0	0	0	0	0
happy	1	0	24	0	0	1	0	0
sad	1	0	0	22	2	0	0	3
angry	2	0	1	0	15	0	3	2
fearful	0	0	0	0	0	18	0	0
disgust	0	0	0	2	0	1	16	3
surprised	1	0	1	1	0	1	2	15

figure 18: 8 emotion model LTSM confusion matrix

From the confusion matrix for the 8 emotion LSTM model, it is apparent that the diagonal is quite well populated, which is a good sign that predicted and actual labels match for a large number of cases, to be exact, 22/26% (correctly predicted neutral emotions) + 7/9% (correctly predicted calm emotions) + 24/26% (correctly predicted happy emotions) + 22/28% (correctly predicted sad emotions) + 15/23% (correctly predicted angry emotions) + 18/18% (correctly predicted fearful emotions) + 16/22% (correctly predicted disgust emotions) + 15/21 (correctly predicted surprised emotions)= 80.33% on average for all the 8 emotions. Moreover, it is also evident that the least false positives are obtained for fearful emotion detection, whereas for surprised and angry emotion detection, the false positives are highest and almost the same.

5.3 Evaluation on test set:



```
Test set predicted emotions accuracy:
neutral : 0.8462
calm : 0.7778
happy : 0.9231
sad : 0.7857
angry : 0.6522
fearful : 1.0000
disgust : 0.7273
surprised : 0.7143
```

Now, as described, the above image gives the prediction accuracy on the test set, in contrast with the results above obtained on the training set and from this image, it is evident that the best-predicted emotion is fear, which is in contrast to the angry emotion being predicted most precisely in the training dataset above. In fact, in this test set, calm emotion is detected better than disgust, which had 83% precision in the training data set and has almost 73% accuracy here, and it is also detected better than surprise, which had almost 85% precision in the training set. It is also evident that angry emotion is the worst accurately predicted emotion, compared to the accuracy rates of other emotions. We have happy and fearful emotions predicted with an accuracy of over 90% while sad and calm emotion prediction lies close to 80%.

5.4 Web Application Development

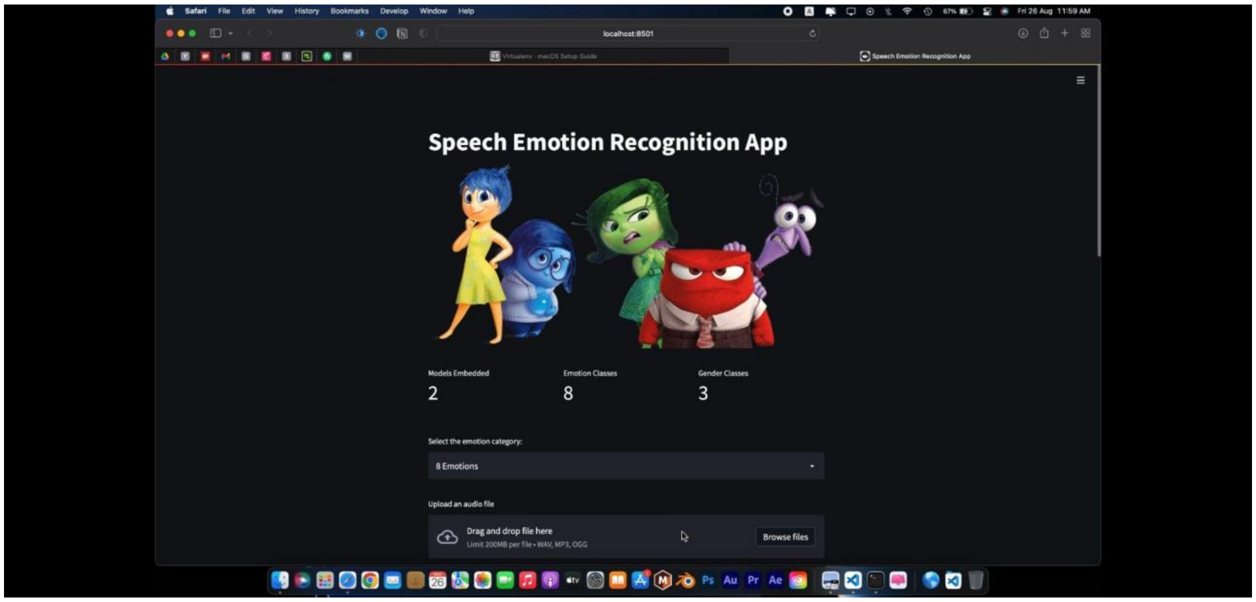
- Stack Selection

As discussed in the subsection above, Streamlit API was selected as the web development backbone. It offers great flexibility to develop Machine Learning or Data Science related applications and also provides the facility to host the app on their own or external cloud servers.

- Features and Workflow:

The application consists of the main features required to assess the feasibility of the Speech Emotion Recognition system in real life.

The workflow of the system is given as below:


Speech Emotion Recognition App

Models Embedded 2
Emotion Classes 8
Gender Classes 3

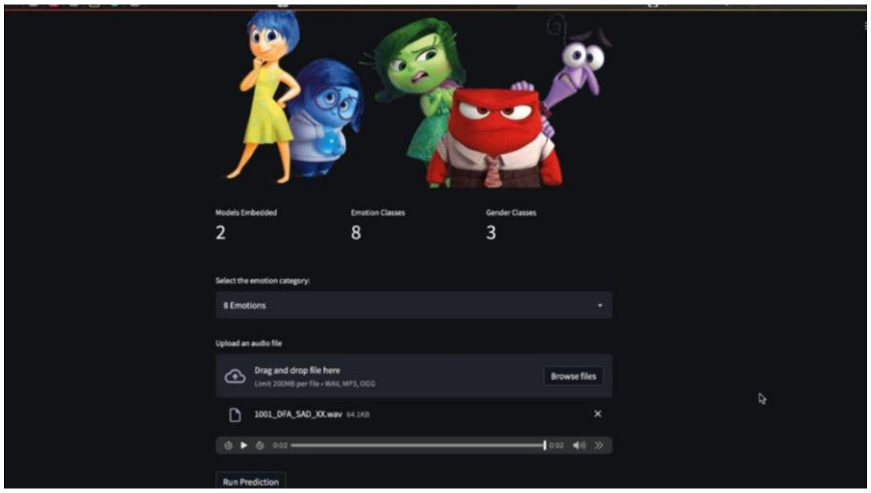
Select the emotion category: 8 Emotions

Upload an audio file

Drag and drop file here
Limit 200MB per file • WAV, MP3, OGG

figure 19: Home page of the web application

- The user selects the model to classify the emotions underlying the human audio speech which is uploaded according to the model selection users see the output of emotion classification.


Models Embedded
2
Emotion Classes
8
Gender Classes
3

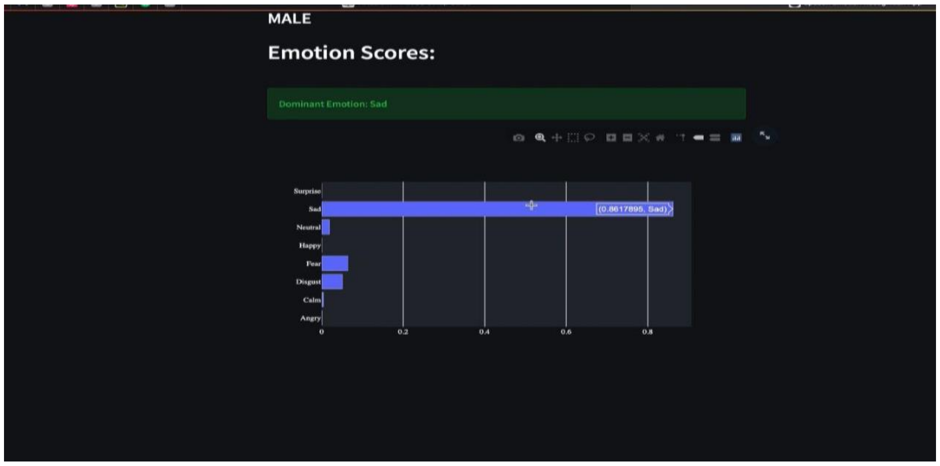
Select the emotion category:
8 Emotions

Upload an audio file

1001_DFA_SAD_XX.wav 64.1KB

figure 20: Application after an audio file has been uploaded and a model has been selected.

- The user uploads the audio file consisting of the human speech.


MALE

Emotion Scores:

Dominant Emotion: Sad

figure 21: Model outcome displayed on the web app

- If the user selects the 8 emotions model, the output says the emotion classification of 8 emotions classes from the uploaded audio clip consisting of the emotions as can be seen in the figure. Same way if the user selects the 3 emotions model, the output says the positive, negative and neutral emotions with percentage scores for the prediction.

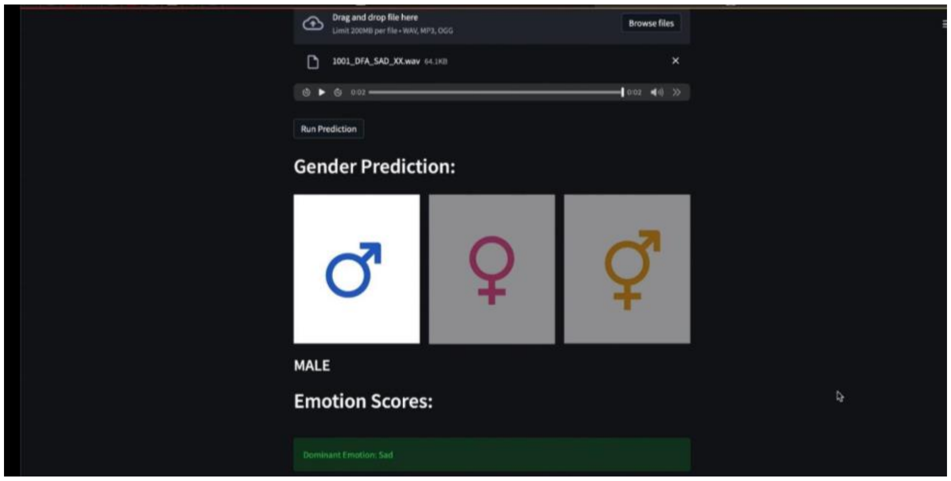


figure 22: Gender prediction of the model is shown as depicted above.

As an additional feature, the gender prediction model has been added to the application. The pretrained VGG model was utilized for dealing with the task. This classifies the gender into 3 categories: Male, Female, and Neutral, based on the percentage values of the predictions.

The source code of the application is attached in the appendix section.

CHAPTER 6. DISCUSSIONS

From the Gantt Chart mentioned in the above section, it is evident that once the pilot study was finished, the tasks that followed were:

1. AI Model Development
    a. Collecting or generating a speech dataset
    b. Analyzing audio information retrieved as such
    c. Selecting appropriate audio features and pre-processing
    d. Build and train models
    e. Evaluate model performances
    f. Build E2E pipelines of trained models.
2. Web Application Development
    a. Brainstorm the app features
    b. User interface design
    c. Finalizing on development stack
    d. App development, including frontend and backend
    e. Integration with AI model pipelines
    f. Testing, debugging and deployment

From the AI Model development, while collecting appropriate speech datasets did not take much time thanks to the work of many researchers across the globe, analyzing audio information and selecting appropriate audio features also was guided by the work of other researchers who were working or have worked in this field, as is described in detail in the literature review of this dissertation. The work got much more interesting and complex when it came to developing the models, so while the Gantt chart showed a reasonable amount of times across these 4 tasks, it is noteworthy that the first three, speech dataset and audio feature analysis, did not take as much time as was appropriated but the remainder of that time was utilized in building and training the models. More time was appropriated to testing and improving models iteration by iteration to get better results. The end-to-end pipeline development, while taking considerable time, was not much complicated and was relatively easily done compared to training the models.

In regards to Web Development, the integration of models into the workflow and further testing and debugging of the same was the technically taxing part and required considerable effort and time when compared to the initially appropriated time for it. However, what remains true to the Gantt chart is the time taken to think upon what app features could be most useful in this use case scenario, as that involved multiple tweaks along the way beyond the time appropriated at the beginning of the development, which helped build a base version, so to say, of features to be included. As a result of features to be included, it also influenced the choice of development stack. Given the time constraints, the other parts of the Web App development were rushed a bit while maintaining the best possible practices.

The thesis, in particular the 8 emotion LSTM model, obtained slightly different prediction accuracy on the test set, in contrast with the results obtained on the training set. The best-predicted emotion is fear, which is in contrast to the angry emotion being predicted most precisely in the training dataset above. In fact, in the test set, calm emotion is detected better than disgust, which had 83% precision in the training data set and almost 73% accuracy in the test set, and it is also detected better than surprise, which had almost 85% precision in the training set. It was also evident that angry emotion is the worst accurately predicted emotion in the test set, compared to the accuracy rates of other emotions. We have happy and fearful emotions predicted with an accuracy of over 90% while sad and calm emotion prediction lies close to 80%. For the 3 emotion model, the model has considerably high levels of accuracy of almost ~94%, with negative emotion detection having the highest level of precision, closely followed by neutral emotion detection while positive emotion detection comes in third but still has a precision level of over 90%.

Now let us consider how such work can affect society. The societal impact reaches out to a variety of case scenarios. Now, there can be a hotline where the elderly and others who do not have a family of their own can reach out in case they ever get isolated and an SER model can optimize for people who need immediate attention and can therefore solve the problem of helping those who need it the most. Similarly, in the case of call centres, the usage can be optimized by a company by employing such a model to determine how agitated a customer is and then redirect such a customer to the best customer care agent who has experience in dealing with agitated customers to ensure the customers stick to their company based on the great support offered with the help of an SER model. Digital assistants like Siri or Alexa or Google Assistant can now be more humane than ever, understand sarcasm, and make suggestions based on determining the emotional state of their user, such as if they sound sad, the assistants could recommend playing their favourite song. More so, these assistants can now become better and automated at classifying content and stories and so much more media that is not suitable for children based on recognizing emotions present in those stories or media. Anyone can finally afford a speech assistant now! A speech assistant that stays with you throughout your day helps you whenever and wherever you need it and is not expensive can drastically affect how the people in the film and television industry, and even people from other industries, employ such services and practices. All these are a fraction of the things possible and the societal impact such a toolkit can have goes much beyond these examples. Therefore, the SER toolkit offers immense potential with great societal impact across various industries and domains.

CHAPTER 7. LIMITATIONS

Human-human interactions are rarely a useful basis for creating user interfaces that work well. Although the spoken human language is useful for human-human communication, it frequently has significant drawbacks when used for human-computer interaction (HCI). Speech or audio data is sluggish for informational context presentation. It is also fleeting and therefore as a consequence, it is challenging to evaluate or modify, and it severely hinders important cognitive processes.

People get to know one another better by ongoing connections and give significance to differences from their previous experiences. Bonding and trustworthiness are developed over time via regular encounters with similar feelings and experiences, empathetic reactions, and helpful support. Accurate modelling or detection of emotional responses is typically impossible since human expression of emotions is so variable among different personalities, subtle in the sense of blending rage, annoyance, restlessness, and more. After all this, the emotional expression of humans is placed which means it is contextually impacted in countless ways. Thus, despite having such a wide range of speech-recognition technologies, they all share some similar drawbacks depending on the individual's way of representing their emotions, their experiences and their perspective of looking towards different things.

While the range of human emotions is vast, this dissertation could only consider the 8 emotions to train and develop the models. This is caused due to the limitations of the available datasets, which consider only a limited range of emotions, usually 8. And while sarcasm forms a vital part of human-human interaction, there's not enough data for models to be able to detect sarcasm or irony. The available benchmark datasets also lack dialects. Moreover, out of the four datasets the author has considered, only one dataset considered the age factor.

Consequently, the dataset was a little bit biased towards these important features in terms of speech emotion recognition. Therefore, the author has tried to combine the different benchmark datasets to make the trained models more generalized.

Despite Streamlit offering enormous advantages, such as hassle-free routing of web pages, frontend development with literally no HTML or CSS or Javascript, super easy to adapt for development and swift development to the deployment process, it still has some limitations.

- Streamlit is quite new in the market so it's difficult to find the issue resolution through online forums, tutorials or blog posts.
- The streamlit is not scalable after a certain limit due to resource constraints. The codebase needs to get shifted to other hosts to make it extremely scalable.
- The modifications in the frontend aspects of the streamlit app are not very convenient. Though the upcoming versions of Streamlit are planning to offer ease in frontend customization.

But since the scope of this thesis was just to check the feasibility of the underlying artificial intelligence technology in the context of Speech Emotion Recognition, the streamlit platform served as a perfect match for our purpose.

CHAPTER 8. CONCLUSION AND FUTURE WORK

As a part of this dissertation, the author has established that Human-Computer Interaction with the computer being able to understand and recognize a range of emotions is possible and brings it therefore quite close to a natural Human-Human Interaction.

This dissertation worked not only on one dataset but multiple datasets, four to be exact, to include a wider range of audio samples and each data frame built for an individual dataset was combined later into a universal dataframe.

To better optimize and make the learning efficient, the author has shown how the transfer learning approach is used which involves using DenseNet 201 architecture as the base model.

The thesis, in particular the 8 emotion LSTM model, obtained slightly different prediction accuracy on the test set, in contrast with the results obtained on the training set. The best-predicted emotion is fear, which is in contrast to the angry emotion being predicted most precisely in the training dataset above. In fact, in the test set, calm emotion is detected better than disgust, which had 83% precision in the training data set and almost 73% accuracy in the test set, and it is also detected better than surprise, which had almost 85% precision in the training set. It was also evident that angry emotion is the worst accurately predicted emotion in the test set, compared to the accuracy rates of other emotions. We have happy and fearful emotions predicted with an accuracy of over 90% while sad and calm emotion prediction lies close to 80%. For the 3 emotion model, the model has considerably high levels of accuracy of almost ~94%, with negative emotion detection having the highest level of precision, closely followed by neutral emotion detection while positive emotion detection comes in third but still has a precision level of over 90%.

Future Work:

- The Speech Recognition Model has a lot to improve upon, specifically in terms of diversity in audio samples over the following features:
  - Age,

    - Dialects
    - Wider emotional range
- We also need better datasets to be able to train models to understand sarcasm in one’s tone.

Reference List / Bibliography:

1. Abbaschian, B.J., Sierra-Sosa, D. and Elmaghraby, A., 2021. Deep learning techniques for speech emotion recognition, from databases to models. *Sensors*, *21*(4), p.1249.
2. Ayadi, El., Moataz, Mohamed S. Kamel, and Fakhri Karray., 2011. "Survey on speech emotion recognition: Features, classification schemes, and databases." *Pattern recognition* 44.3: 572-587.
3. Bucciarelli, Valentina, et al. "Depression pandemic and cardiovascular risk in the COVID-19 era and long COVID syndrome: gender makes a difference." *Trends in Cardiovascular Medicine* (2021).
4. Cao H, Cooper DG, Keutmann MK, Gur RC, Nenkova A, Verma R., 2014 CREMA-D: Crowd-sourced Emotional Multimodal Actors Dataset. IEEE Trans Affect Comput;5(4):377-390. doi: 10.1109/TAFFC.2014.2336244. PMID: 25653738; PMCID: PMC4313618.
5. Fayek, H.M., Lech, M. and Cavedon, L., 2017. Evaluating deep learning architectures for speech emotion recognition. *Neural Networks*, *92*, pp.60-68.
6. Haq, S., and Jackson, P.J.B., 2010. "Multimodal Emotion Recognition", In W. Wang (ed),

   Machine Audition: Principles, Algorithms and Systems, IGI Global Press,

   ISBN 978-1615209194, DOI 10.4018/978-1-61520-919-4, chapter 17, pp. 398-423.
7. Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K.Q., 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).
8. Jain, M., Narayan, S., Balaji, P., Bhowmick, A. and Muthu, R.K., 2020. Speech emotion recognition using Support Vector Machine. *arXiv preprint arXiv:2002.07590*.

9. Kaggle - Crowd Sourced Emotional Multimodal Actors Dataset (CREMA-D) https://www.kaggle.com/datasets/ejlok1/cremad. 2022

10. Khalil, Ruhul Amin, et al. "Speech emotion recognition using deep learning techniques: A review." *IEEE Access* 7 (2019): 117327-117345.

11. Livingstone SR, Russo FA (2018) The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. PLoS ONE 13(5): e0196391. https://doi.org/10.1371/journal.pone.0196391.

12. Lugović, S., Dunđer I. and Horvat, M. 2016 "Techniques and applications of emotion recognition in speech," 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 1278-1283, DOI: 10.1109/MIPRO.2016.7522336

13. Malla, S., A. Alsadoon and S. K. Bajaj, "A DFC taxonomy of Speech emotion recognition based on the convolutional neural network from the speech signal," 2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA), 2020, pp. 1-10, III: 10.1109/CITISIA50690.2020.9371841

14. Nwe, T.L., Foo, S.W. and De Silva, L.C., 2003. Speech emotion recognition using hidden Markov models. *Speech communication*, *41*(4), pp.603-623.

15. Pichora-Fuller, M. Kathleen; Dupuis, Kate, 2020, "Toronto emotional speech set (TESS)", https://doi.org/10.5683/SP2/E8H2MF, Borealis, V1

16. Schuller, B. W. ‘‘Speech emotion recognition: Two decades in a nutshell, benchmarks, and ongoing trends,’’ Commun. ACM, vol. 61, no. 5, pp. 90–99, 2018.

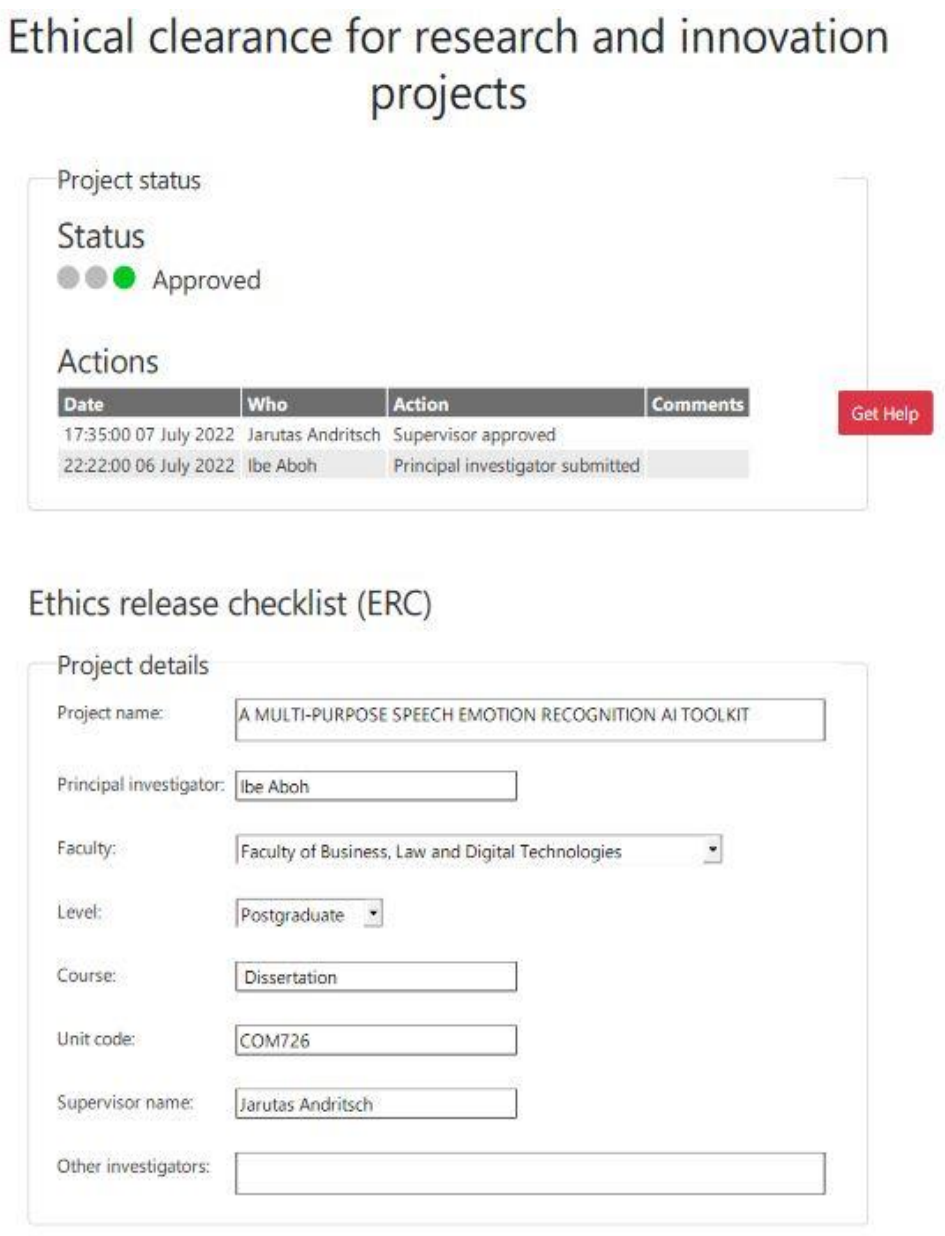
17. Schuller, B., Rigoll, G. and Lang, M., 2004, May. Speech emotion recognition combines acoustic features and linguistic information in a hybrid support vector machine-belief network architecture. In *2004 IEEE international conference on acoustics, speech, and signal processing* (Vol. 1, pp. I-577). IEEE.

18. Valles, D. and R. Matin., 2021, "An Audio Processing Approach using Ensemble Learning for Speech-Emotion Recognition for Children with ASD," 2021 IEEE World AI IoT Congress (AIIoT), pp. 0055-0061, DOI: 10.1109/AIIoT52608.2021.9454174.

19. Williams, C.E. and Stevens, K.N., 1972. Emotions and speech: Some acoustical correlates. *The journal of the acoustical society of America*, *52*(4B), pp.1238-1250.

APPENDICES

Appendix A: Ethical Approval



Ethical clearance for research and innovation projects

Project status

Status

● ● ● Approved

Actions

Date	Who	Action	Comments
17:35:00 07 July 2022	Jarutas Andritsch	Supervisor approved	
22:22:00 06 July 2022	Ibe Aboh	Principal investigator submitted	

Ethics release checklist (ERC)

Project details

Project name:	A MULTI-PURPOSE SPEECH EMOTION RECOGNITION AI TOOLKIT
Principal investigator:	Ibe Aboh
Faculty:	Faculty of Business, Law and Digital Technologies
Level:	Postgraduate
Course:	Dissertation
Unit code:	COM726
Supervisor name:	Jarutas Andritsch
Other investigators:	

Appendix B: **Useful Definitions**

Transfer Learning:

Transfer Learning is a renowned method in Deep Learning in which the model trained and developed for the purpose to deal with some specific tasks is utilized as a base for another model which is targeting a different task. It saves computational overheads and resource requirements while training the vast models.

Average Magnitude Difference Function (AMDF):

The average Magnitude Difference Function is especially useful in scenarios where there is low or no background noise to extract the period of a periodical signal. Its usefulness is especially highlighted in terms of its high accuracy and low computational complexity.

DenseNet:

DenseNet is also known as Dense Convolutional Network is a CNN-based architecture where each layer is densely connected with all previous layers. Thus all layers get an overall combined knowledge from the previous layers. It is targeted to deal with image classification tasks.

WebGL:

WebGL which stands for Web Graphics Library is a Javascript API which is used specifically for the purpose to render interactive and high-performance graphics, inclusive of both 2D and 3D, throughout any web browser without utilizing any third-party plugins.

Appendix **C: Model Development Code**

1. The Organization of the Notebook

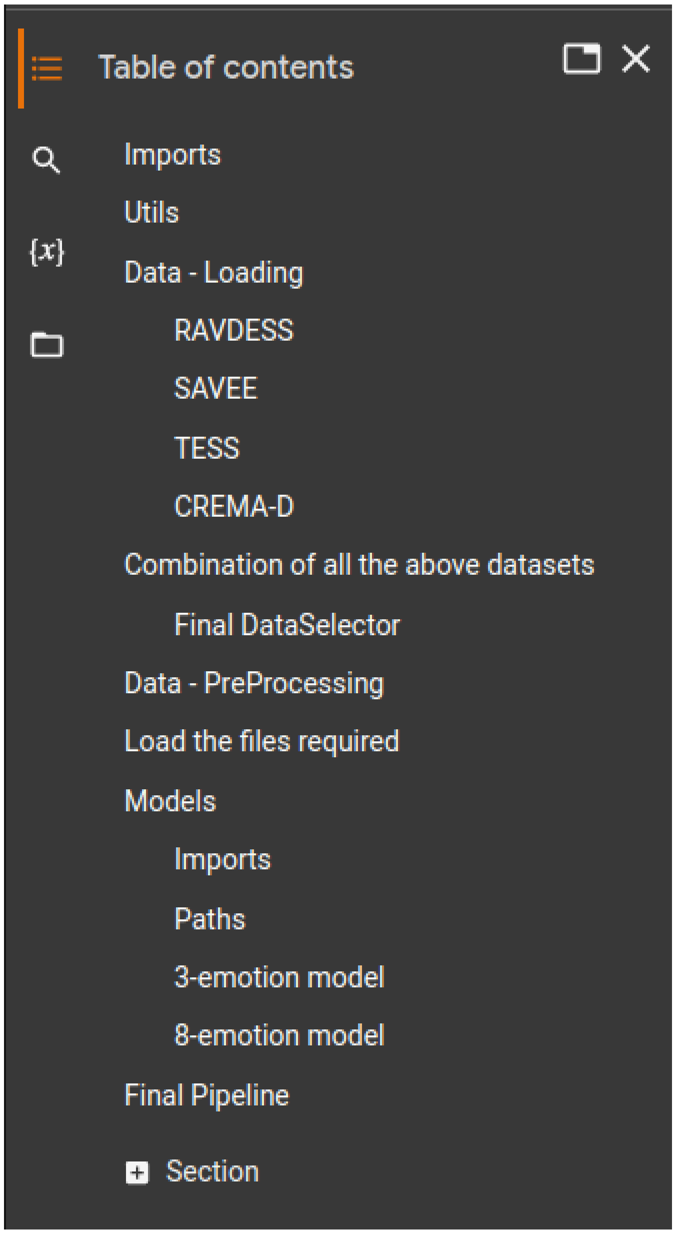
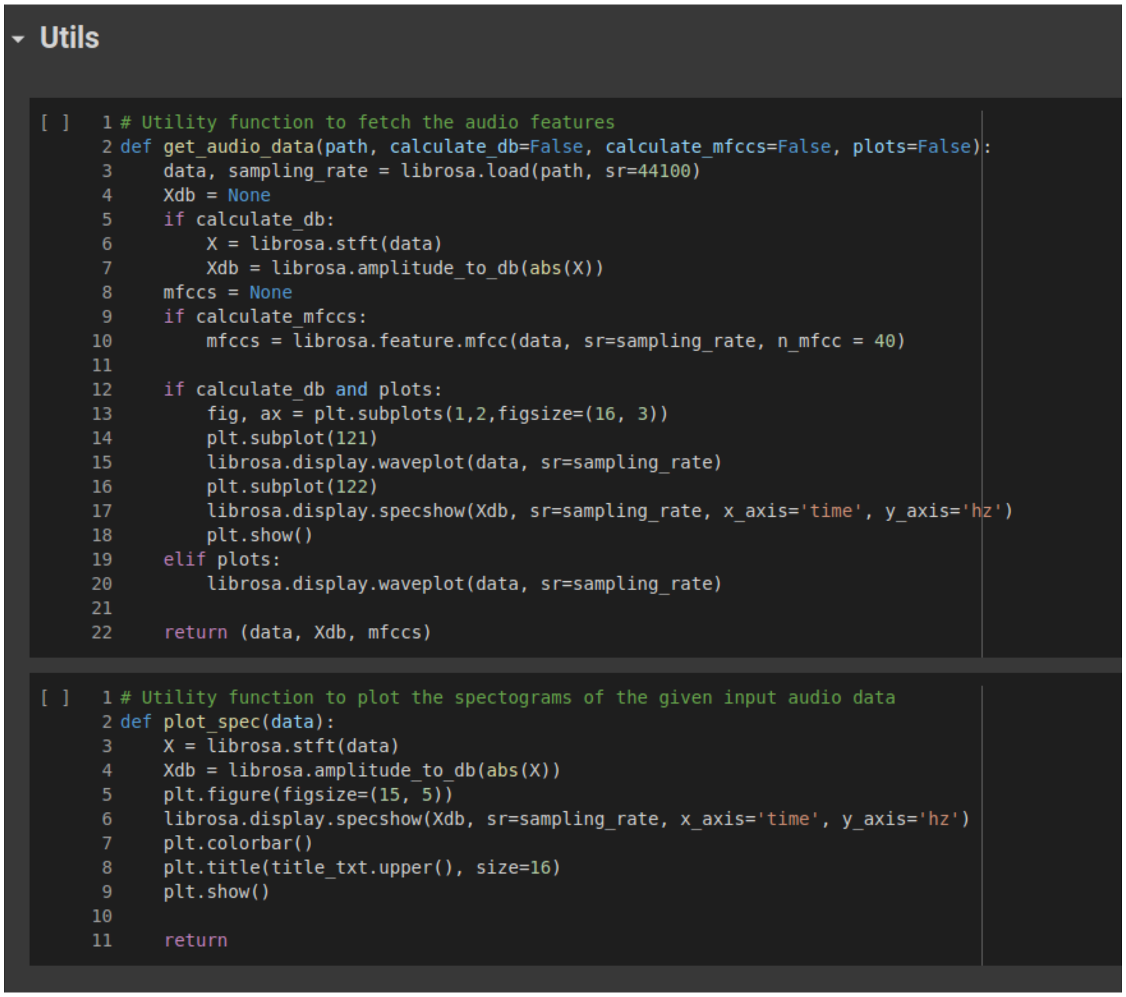


Table of contents

Imports
Utils
Data - Loading
    RAVDESS
    SAVEE
    TESS
    CREMA-D
Combination of all the above datasets
    Final DataSelector
Data - PreProcessing
Load the files required
Models
    Imports
    Paths
    3-emotion model
    8-emotion model
Final Pipeline

Section

2. Utility functions that were used for the model training

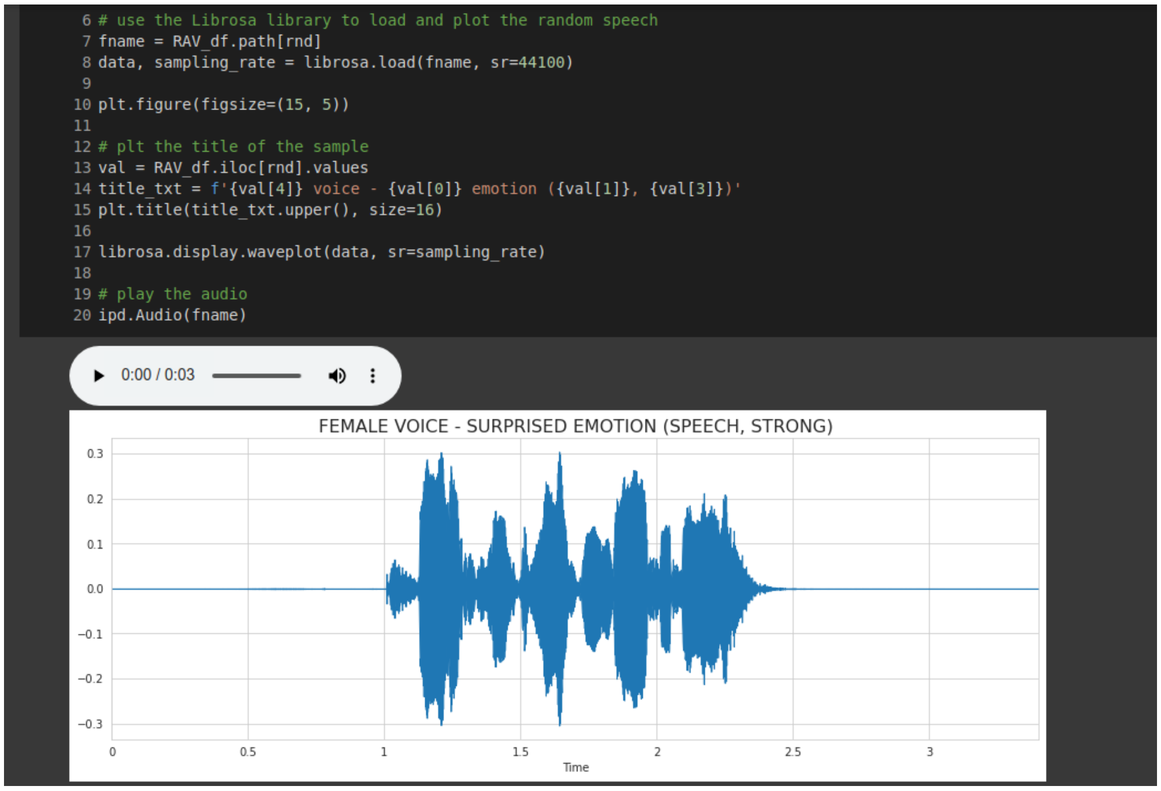


Utils

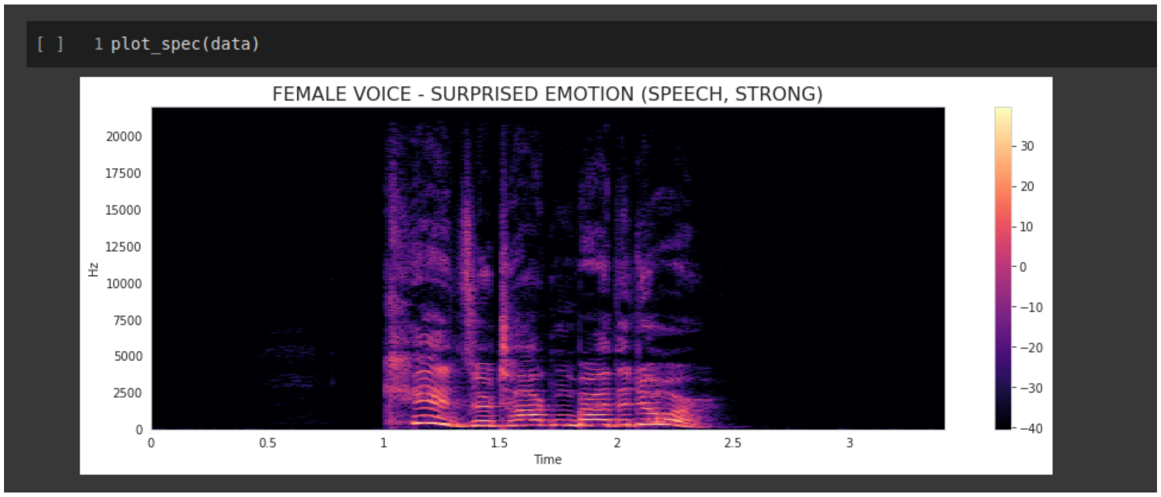
```
[ ]
# Utility function to fetch the audio features
def get_audio_data(path, calculate_db=False, calculate_mfccs=False, plots=False):
    data, sampling_rate = librosa.load(path, sr=44100)
    Xdb = None
    if calculate_db:
        X = librosa.stft(data)
        Xdb = librosa.amplitude_to_db(abs(X))
    mfccs = None
    if calculate_mfccs:
        mfccs = librosa.feature.mfcc(data, sr=sampling_rate, n_mfcc = 40)

    if calculate_db and plots:
        fig, ax = plt.subplots(1,2,figsize=(16, 3))
        plt.subplot(121)
        librosa.display.waveplot(data, sr=sampling_rate)
        plt.subplot(122)
        librosa.display.specshow(Xdb, sr=sampling_rate, x_axis='time', y_axis='hz')
        plt.show()
    elif plots:
        librosa.display.waveplot(data, sr=sampling_rate)

    return (data, Xdb, mfccs)
```

```
[ ]
# Utility function to plot the spectograms of the given input audio data
def plot_spec(data):
    X = librosa.stft(data)
    Xdb = librosa.amplitude_to_db(abs(X))
    plt.figure(figsize=(15, 5))
    librosa.display.specshow(Xdb, sr=sampling_rate, x_axis='time', y_axis='hz')
    plt.colorbar()
    plt.title(title_txt.upper(), size=16)
    plt.show()

    return
```

3. Visualization of a random speech entry

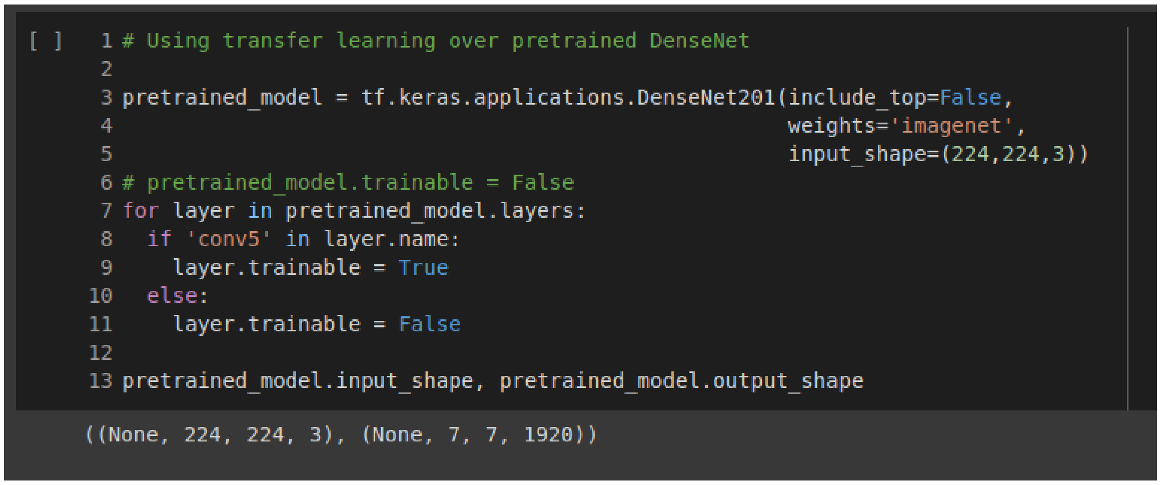


```
# use the Librosa library to load and plot the random speech
fname = RAV_df.path[rnd]
data, sampling_rate = librosa.load(fname, sr=44100)

plt.figure(figsize=(15, 5))

# plt the title of the sample
val = RAV_df.iloc[rnd].values
title_txt = f'{val[4]} voice - {val[0]} emotion ({val[1]}, {val[3]})'
plt.title(title_txt.upper(), size=16)

librosa.display.waveplot(data, sr=sampling_rate)

# play the audio
ipd.Audio(fname)
```



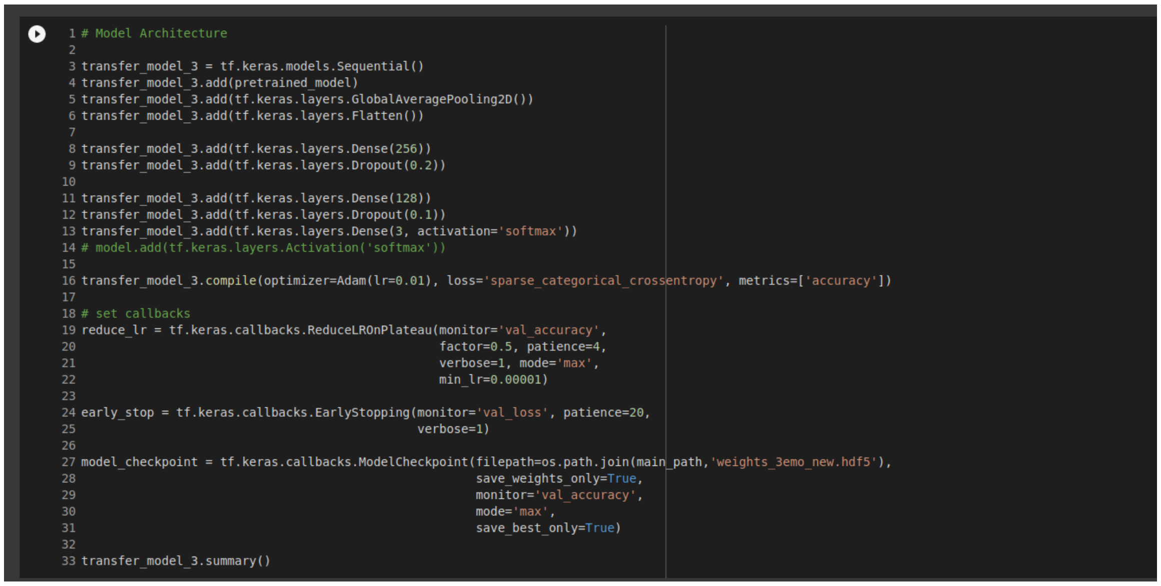
```
[ ]  plot_spec(data)
```

4. Model Architecture



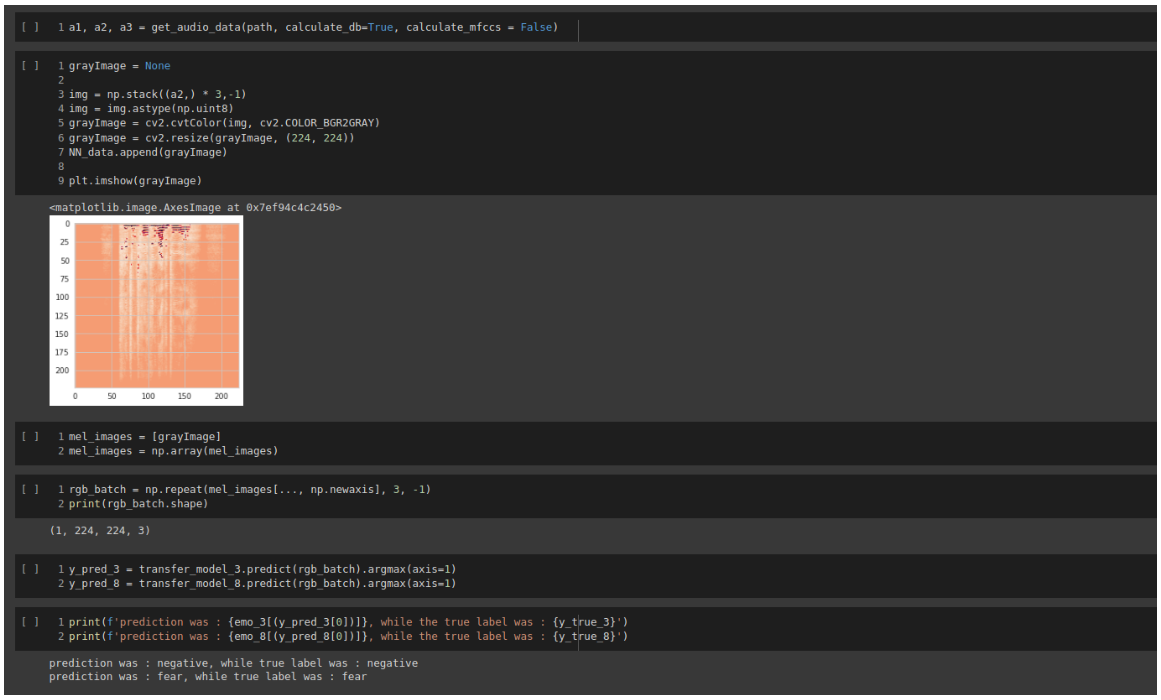
```
[ ]
# Using transfer learning over pretrained DenseNet

pretrained_model = tf.keras.applications.DenseNet201(include_top=False,
                                                     weights='imagenet',
                                                     input_shape=(224,224,3))

# pretrained_model.trainable = False
for layer in pretrained_model.layers:
  if 'conv5' in layer.name:
    layer.trainable = True
  else:
    layer.trainable = False

pretrained_model.input_shape, pretrained_model.output_shape
```

```
((None, 224, 224, 3), (None, 7, 7, 1920))
```



```
# Model Architecture

transfer_model_3 = tf.keras.models.Sequential()
transfer_model_3.add(pretrained_model)
transfer_model_3.add(tf.keras.layers.GlobalAveragePooling2D())
transfer_model_3.add(tf.keras.layers.Flatten())

transfer_model_3.add(tf.keras.layers.Dense(256))
transfer_model_3.add(tf.keras.layers.Dropout(0.2))

transfer_model_3.add(tf.keras.layers.Dense(128))
transfer_model_3.add(tf.keras.layers.Dropout(0.1))
transfer_model_3.add(tf.keras.layers.Dense(3, activation='softmax'))
# model.add(tf.keras.layers.Activation('softmax'))

transfer_model_3.compile(optimizer=Adam(lr=0.01), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# set callbacks
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy',
                                                 factor=0.5, patience=4,
                                                 verbose=1, mode='max',
                                                 min_lr=0.00001)

early_stop = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=20,
                                              verbose=1)

model_checkpoint = tf.keras.callbacks.ModelCheckpoint(filepath=os.path.join(main_path,'weights_3emo_new.hdf5'),
                                                      save_weights_only=True,
                                                      monitor='val_accuracy',
                                                      mode='max',
                                                      save_best_only=True)

transfer_model_3.summary()
```

5. Sample Test Prediction



```
[ ]  a1, a2, a3 = get_audio_data(path, calculate_db=True, calculate_mfccs = False)
```

```
[ ]  grayImage = None

     img = np.stack((a2,) * 3,-1)
     img = img.astype(np.uint8)
     grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
     grayImage = cv2.resize(grayImage, (224, 224))
     NN_data.append(grayImage)

     plt.imshow(grayImage)
```

```
<matplotlib.image.AxesImage at 0x7ef94c4c2450>
```

```
[ ]  mel_images = [grayImage]
     mel_images = np.array(mel_images)
```

```
[ ]  rgb_batch = np.repeat(mel_images[..., np.newaxis], 3, -1)
     print(rgb_batch.shape)
```

```
(1, 224, 224, 3)
```

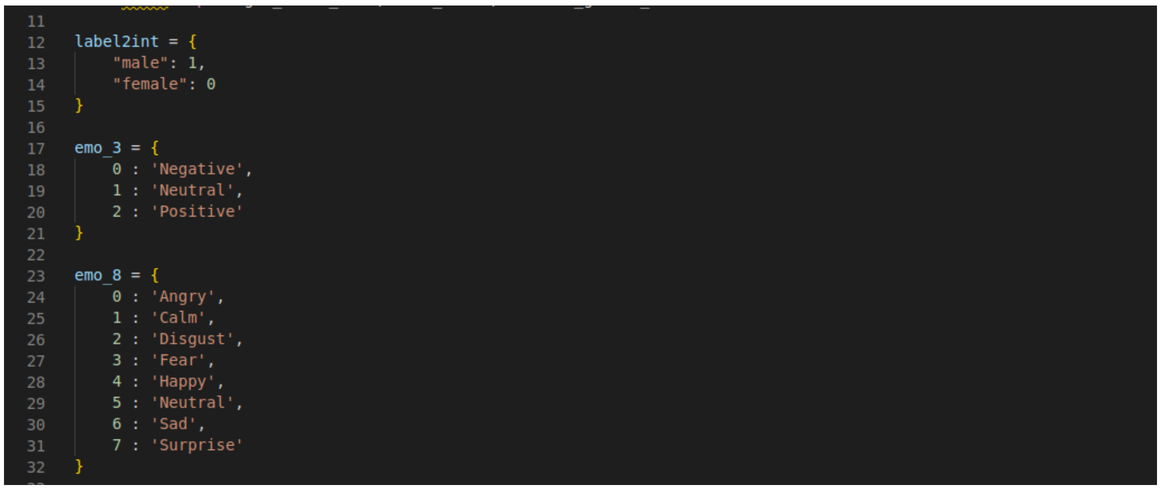
```
[ ]  y_pred_3 = transfer_model_3.predict(rgb_batch).argmax(axis=1)
     y_pred_8 = transfer_model_8.predict(rgb_batch).argmax(axis=1)
```

```
[ ]  print(f'prediction was : {emo_3[(y_pred_3[0])]}, while the true label was : {y_true_3}')
     print(f'prediction was : {emo_8[(y_pred_8[0])]}, while the true label was : {y_true_8}')
```

```
prediction was : negative, while true label was : negative
prediction was : fear, while true label was : fear
```

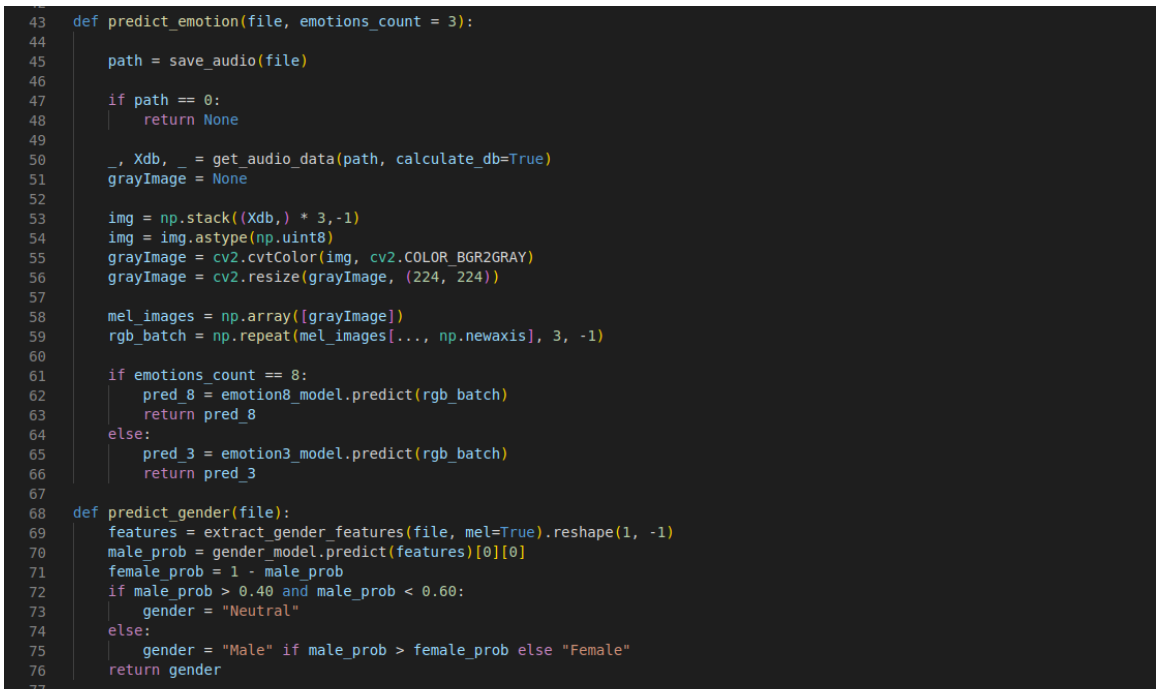
Appendix D: Web Application Code

1. Label Encoding for Emotions and Gender Classes



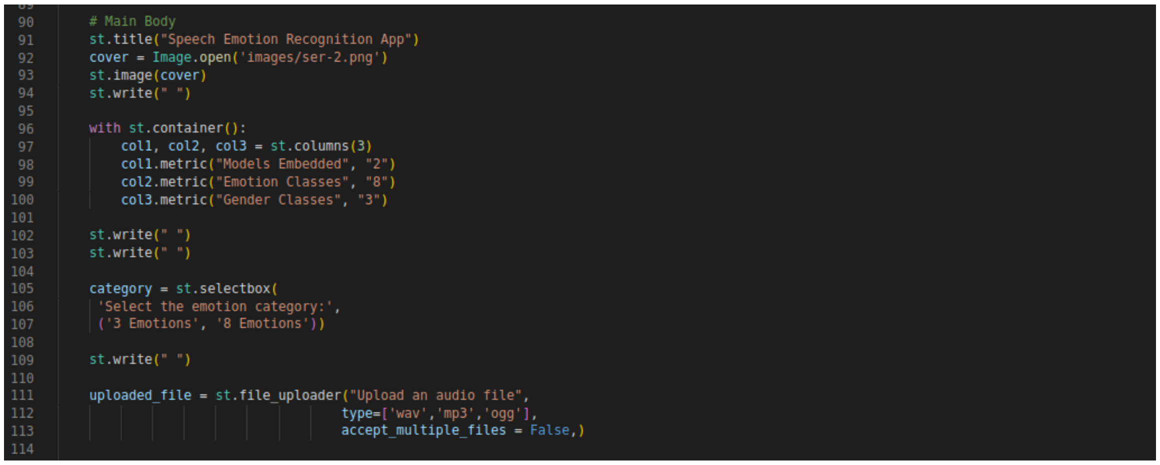
```

label2int = {
    "male": 1,
    "female": 0
}

emo_3 = {
    0 : 'Negative',
    1 : 'Neutral',
    2 : 'Positive'
}

emo_8 = {
    0 : 'Angry',
    1 : 'Calm',
    2 : 'Disgust',
    3 : 'Fear',
    4 : 'Happy',
    5 : 'Neutral',
    6 : 'Sad',
    7 : 'Surprise'
}
```

2. Utility functions to make predictions



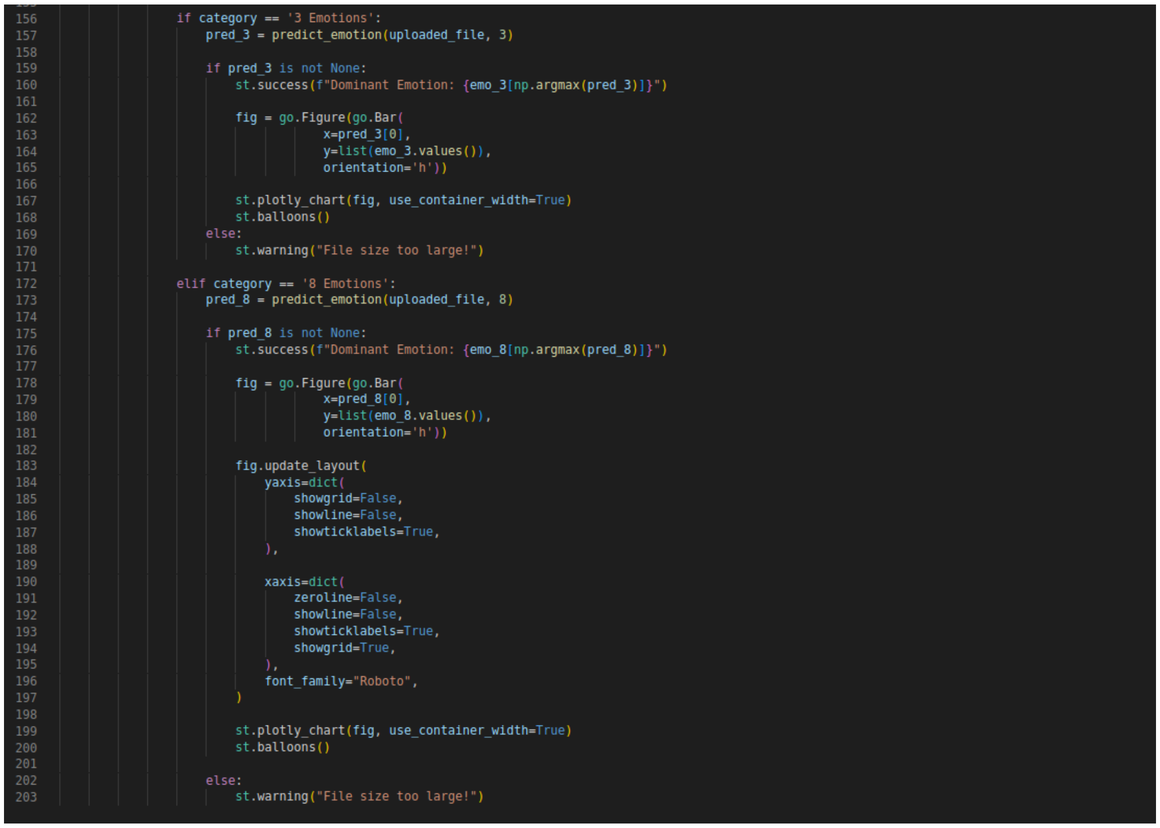
```
def predict_emotion(file, emotions_count = 3):

    path = save_audio(file)

    if path == 0:
        return None

    _, Xdb, _ = get_audio_data(path, calculate_db=True)
    grayImage = None

    img = np.stack((Xdb,) * 3,-1)
    img = img.astype(np.uint8)
    grayImage = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    grayImage = cv2.resize(grayImage, (224, 224))

    mel_images = np.array([grayImage])
    rgb_batch = np.repeat(mel_images[..., np.newaxis], 3, -1)

    if emotions_count == 8:
        pred_8 = emotion8_model.predict(rgb_batch)
        return pred_8
    else:
        pred_3 = emotion3_model.predict(rgb_batch)
        return pred_3

def predict_gender(file):
    features = extract_gender_features(file, mel=True).reshape(1, -1)
    male_prob = gender_model.predict(features)[0][0]
    female_prob = 1 - male_prob
    if male_prob > 0.40 and male_prob < 0.60:
        gender = "Neutral"
    else:
        gender = "Male" if male_prob > female_prob else "Female"
    return gender
```

3. Front-end Setup Using Streamlit



```python
# Main Body
st.title("Speech Emotion Recognition App")
cover = Image.open('images/ser-2.png')
st.image(cover)
st.write(" ")

with st.container():
    col1, col2, col3 = st.columns(3)
    col1.metric("Models Embedded", "2")
    col2.metric("Emotion Classes", "8")
    col3.metric("Gender Classes", "3")

st.write(" ")
st.write(" ")

category = st.selectbox(
 'Select the emotion category:',
 ('3 Emotions', '8 Emotions'))

st.write(" ")

uploaded_file = st.file_uploader("Upload an audio file",
                                type=['wav','mp3','ogg'],
                                accept_multiple_files = False,)
```

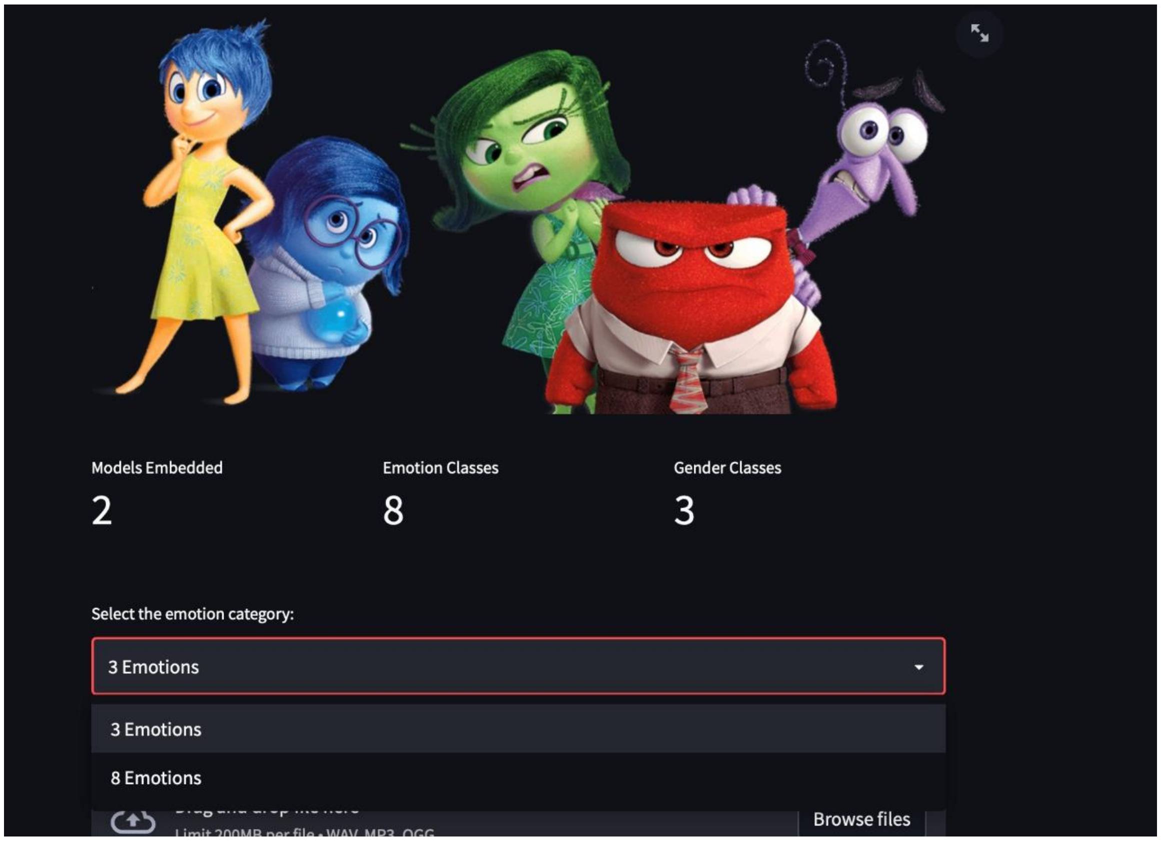
4. Emotion Classification based on the selected model



```python
                if category == '3 Emotions':
                    pred_3 = predict_emotion(uploaded_file, 3)

                    if pred_3 is not None:
                        st.success(f"Dominant Emotion: {emo_3[np.argmax(pred_3)]}")

                        fig = go.Figure(go.Bar(
                                    x=pred_3[0],
                                    y=list(emo_3.values()),
                                    orientation='h'))

                        st.plotly_chart(fig, use_container_width=True)
                        st.balloons()
                    else:
                        st.warning("File size too large!")

                elif category == '8 Emotions':
                    pred_8 = predict_emotion(uploaded_file, 8)

                    if pred_8 is not None:
                        st.success(f"Dominant Emotion: {emo_8[np.argmax(pred_8)]}")

                        fig = go.Figure(go.Bar(
                                    x=pred_8[0],
                                    y=list(emo_8.values()),
                                    orientation='h'))

                        fig.update_layout(
                            yaxis=dict(
                                showgrid=False,
                                showline=False,
                                showticklabels=True,
                            ),

                            xaxis=dict(
                                zeroline=False,
                                showline=False,
                                showticklabels=True,
                                showgrid=True,
                            ),
                            font_family="Roboto",
                        )

                        st.plotly_chart(fig, use_container_width=True)
                        st.balloons()

                    else:
                        st.warning("File size too large!")
```

Appendix E: Web Application Screenshots

1. Models Selection: 3 Emotions and 8 Emotions



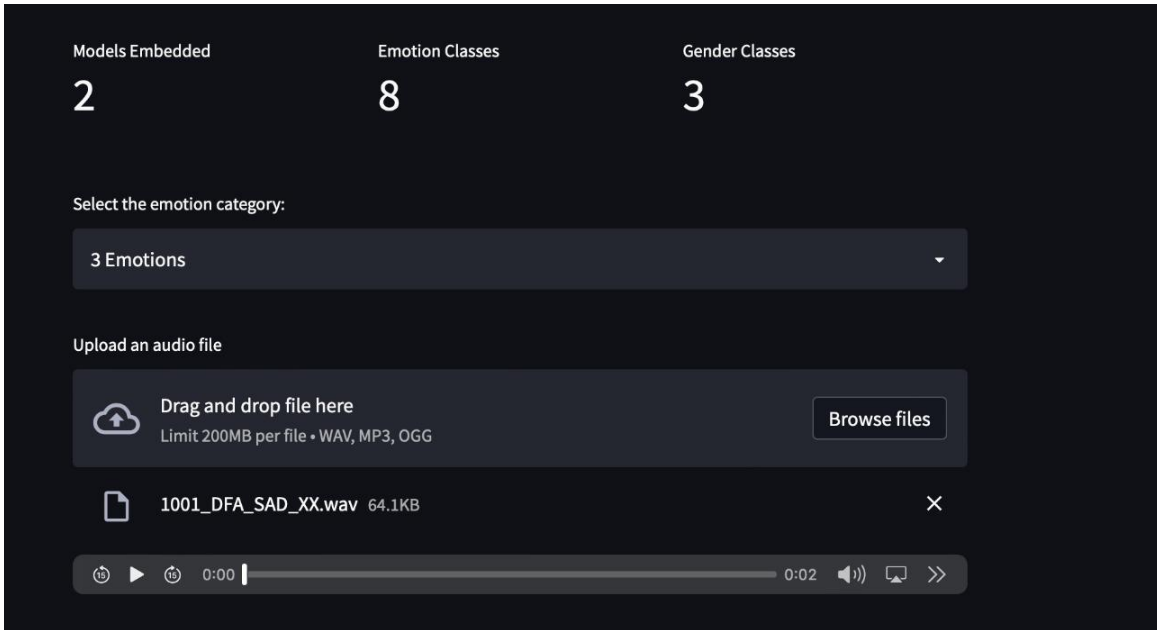
Models Embedded 2

Emotion Classes 8

Gender Classes 3

Select the emotion category: 3 Emotions

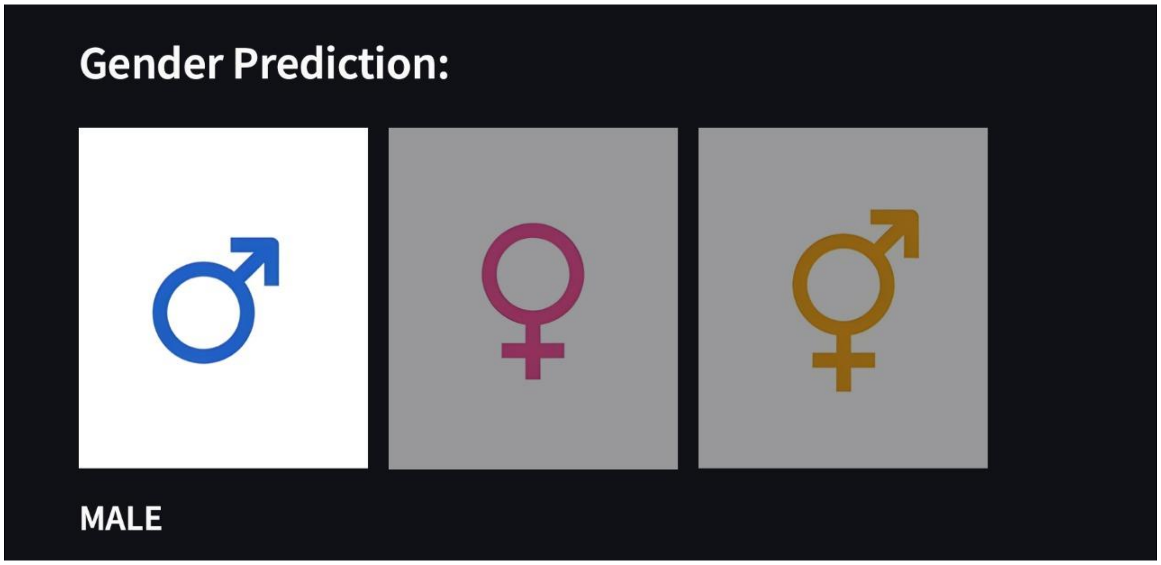
2. File Uploader with Play Audio Feature


Models Embedded
2
Emotion Classes
8
Gender Classes
3

Select the emotion category:
3 Emotions

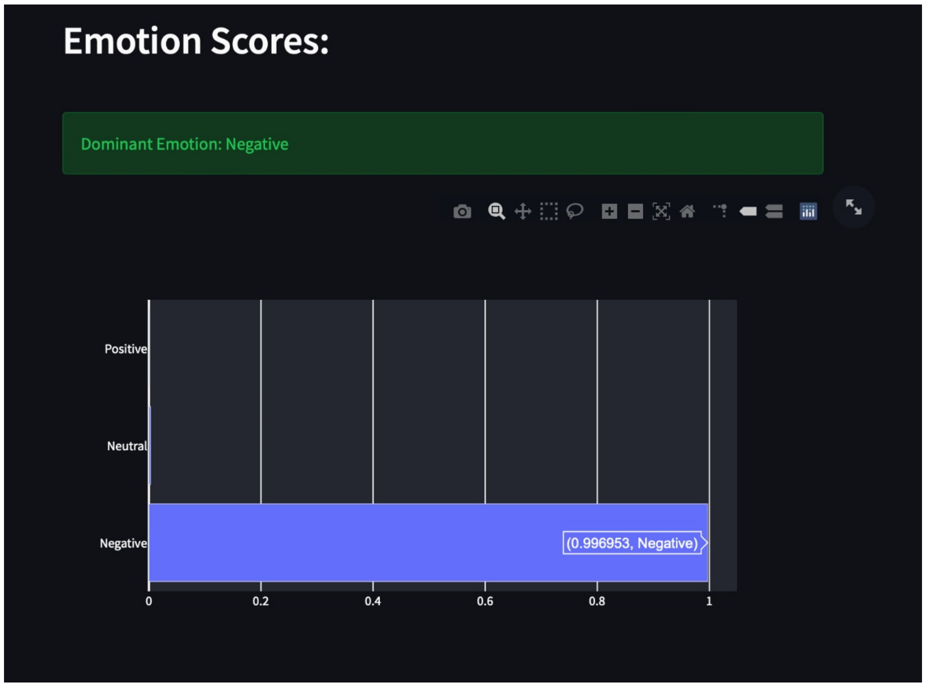
Upload an audio file
Drag and drop file here
Limit 200MB per file • WAV, MP3, OGG
Browse files
1001_DFA_SAD_XX.wav 64.1KB

3. Gender Prediction


Gender Prediction:

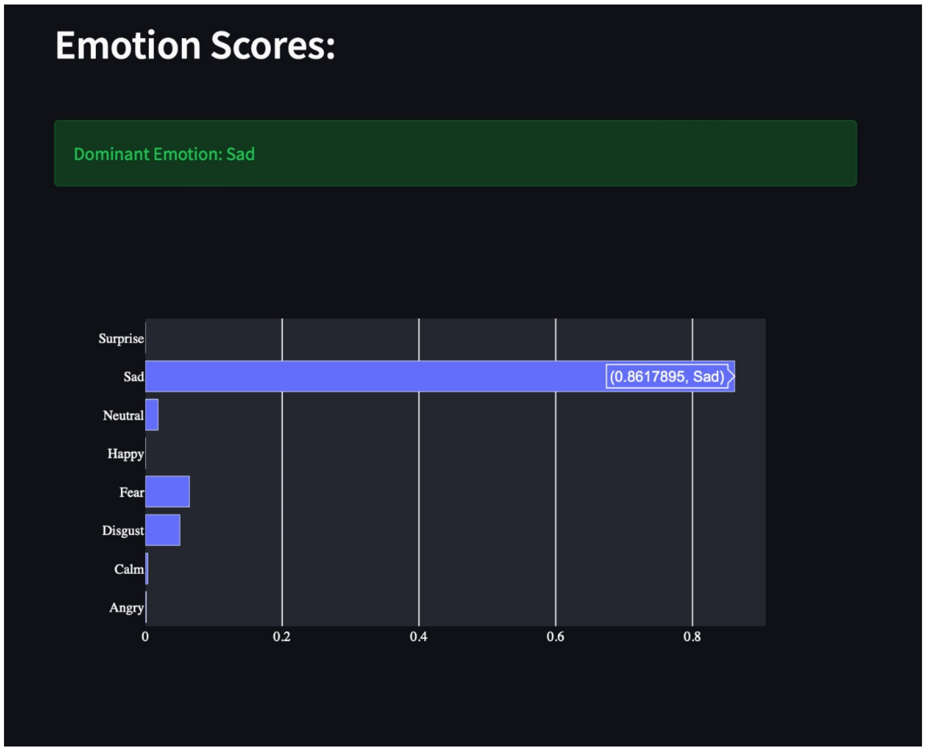
MALE

4. 3 Emotions Model Prediction Visualization


Emotion Scores:

Dominant Emotion: Negative

5. 8 Emotions Model Prediction Visualization


Emotion Scores:

Dominant Emotion: Sad