ABBREVIATIONS

CNN	*Convolution Neural Network*
ReLU	*Rectified Linear Activation Function*
SER	*Speech Emotion Recognition*
DFT	*Discrete Fourier Transform*
DCT	*Discrete Cosine Transform*
MFCC	*Mel Frequency Cepstral Co-efficients*
MS	*Mel Spectrogram*
RAVDESS	*Ryerson Audio Visual Emotion Dataset*

ABSTRACT

Emotions are deep-rooted part of human life and communication. Humans are well trained in reading and recognition of emotions that make them compassionate and animated. Machines although intelligent are not as receptive to human emotions that makes them monotonic and inanimate. For better understanding and proper communication computers must be capable of perceiving sentiments and feelings. Emotion analysis has uses in psychiatric analysis, improving quality of service in consumer oriented services.

Keywords: *Convolutional neural network, Mel-spectral cepstral Co-efficients,Mel spectrogram, Chroma, RAVDESS dataset, speech processing, Chroma, Emotion Recognition.*

TABLE OF CONTENTS

CHAPTER 1
SUMMARY OF THE BASE PAPER

Title	***Convolution Neural Network-based Speech Emotion Recognition***
Journal Name	International Research Journal of Engineering and Technology.
Publisher	Abdul Ajij Ansari, Ayush Kumar Singh, Ashutosh Singh
Year	June 2020

SUMMARY

1 The authors of this foundational research suggested a Convolutional Neural Network(CNN) model for learning how to recognize speech emotion and the accuracy rate of the model

2 We used SciKit Learn (an open-source library written in Python) as the programming framework to implement our CNN models.

3 The model in this base paper uses RAVDESS dataset and its accuracy other measures were presented to assist future research in the selection of features.

4 Human-machine interaction was based on speech recognition where the literal meaning of the spoken was recognized, where it is important to emphasize emotions as well.

5 Using deep learning methods, the suggested system in the base study in many cases the system predicted correctly.

CHAPTER 2

DEEP LEARNING AND CNN

DEEP LEARNING

The various deep learning models train a computer to visualize and make decisions like a human-based on input parameters. The network type is similar to that of a Human Nervous System where each node performs similar to a neuron in a large network. Thus they can be said to be part of an artificial neural network. Characteristics like edges detection from the initial layers, and successive layers work with characteristics from prior layers in a more philosophical representation. Images, sounds, sensor data, and other data are those digital form patterns that Deep Learning recognizes. For prediction, a training and a test set were arranged by splitting the RAVDESS dataset. As our prediction obtains an optimum node such that the predicted node provides the satisfactory output.

CONVOLUTION NEURAL NETWORKS (CNN):

Introduction:

Convolutional neural network is a artificial network of feed forwarding type. Single neurons give a response to the stimuli at a prohibited area of the region known as the receptive area. Convolutional Neural Networks is a highly capable of classification if fed with signals such as frequency and amplitude changes. As they are multilevel networks, the first layer is fed with input signals. Second layer is fed with features to detect.

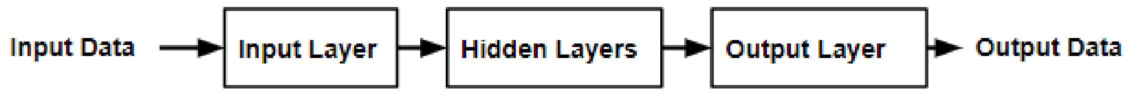


Fig. 2.1 Basic Blocks of CNN

The convolutional neural network matches the parts of the signal instead of considering the whole signal of data as it becomes difficult for a computer to identify the signal when the set of data is considered as a whole.

Convolutional Layer:

CNN model's one significant role is to modify given input using linked neurons that are grouped from previous layers. A Dot product is conducted in between the neurons of that area which are present in the input layer and the weights which are connected locally in the output layer A convolution layer is a fundamental component of the CNN architecture that performs feature extraction. This process consists of linear and nonlinear operations, i.e., convolution operation and activation function.

Convolution:

The act of convolving signals with a bunch of filters, a bunch of features that create a stack of the filtered signal is called a convolutional layer. This is a layer because it operates based on the stack that is in convolution with one signal becomes a stack of filtered signals. We get a many filtered signals because of the usage of filters.

The second layer is called pooling which is how a signal stack can be compressed. It is done by using a small pixel window which might be a 2 by 2 window or 3 by 3. On considering a 2 by 2 window pixel and passing it in strides across the filtered signals, from each window the maximum value is considered. This passed through the whole signal. In the end, it is found that by considering only the maximum values the size of the filtered signal is reduced.

The third layer is normalization, in this, if a pixel value is negative then the negative values are replaced with zeros. This is done to all the filtered signals. This becomes another type of layer which is known as a rectified linear unit (RELU), a stack of signals becomes a stack with no negative values. The layers are stacked up so that one output will become the input for the next. The final layer is the fully connected layer.

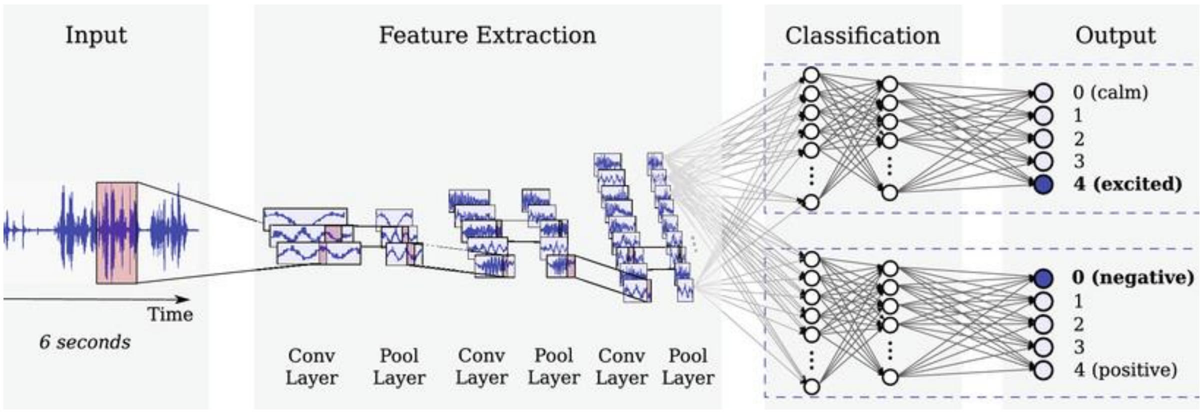


Fig. 2.2 CNN Architecture for Speech Emotion Recognition

Activation Layer(ReLU):

During a forward pass through CNN, each filter is slid over the spatial dimensions of the inputvolume to create an activation map. When a neuron decides to transmit information, it generates a numerical value. ReLU prevents exponential growth in the processing power required to operate the neural network.

$$ReLU(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases}$$

Filters:

Filters are the most important Convolutional Neural Network parameters that are responsible todeliver the significant activation to spatially located input data of patterns, which means theywill be active only if that required pattern is available in the data(training input data). As the depth of the CNN rises, the filters will be able to identify and locate the non-linear combinationof features also.

HYPER PARAMETERS

These determine the output volume of a convolutional layer's spatial arrangement and size. Thefollowing are some of the most essential hyperparameters:

Padding:

Sometimes input and filter does not match. In such cases we have two rectification options:

1. Pad with zeroes (Zero Padding)
2. Drop part of the input where the filter did not fit (valid padding)

Stride:

The filter's sliding speed is specified by a stride. The depth of output volume and the stride value are inversely proportional. The number of pixels shifted across the input matrix is known as the stride value.

Pooling Layer:

It reduces the number of parameters when the input is too large. Thus it helps in preventing the overfitting of training data by gradually lowering the data representation spatial size.

Max Pooling: It is the operation that selects the highest element in from the region of the feature map.

Average Pooling: It is the sum of elements divided by the total number of elements present in the pooling region.

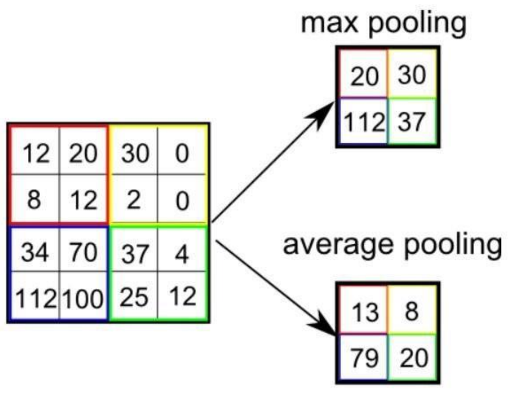


Fig. 2.3 Pooling

Flattening Layer:

This layer takes pooled feature layer’s map and converts it into a one-dimensional vector. This vector is then fed to the fully connected layer.

Fully Connected Layer:

This Fully Connected layer has an output volume of [1 x 1 x M] this is the network's output layer. The ‘M’ denotes the number of classes to be calculated for output. Normal network layers of neural settings, as well as those hyper parameters, are present in a completely linked layer.

Softmax Layer:

This is a layer that is mainly used to normalize the neural network's output between zero and one. It represent network's output as a probability distribution. This is usally appplied as the very last layer in the neural network.

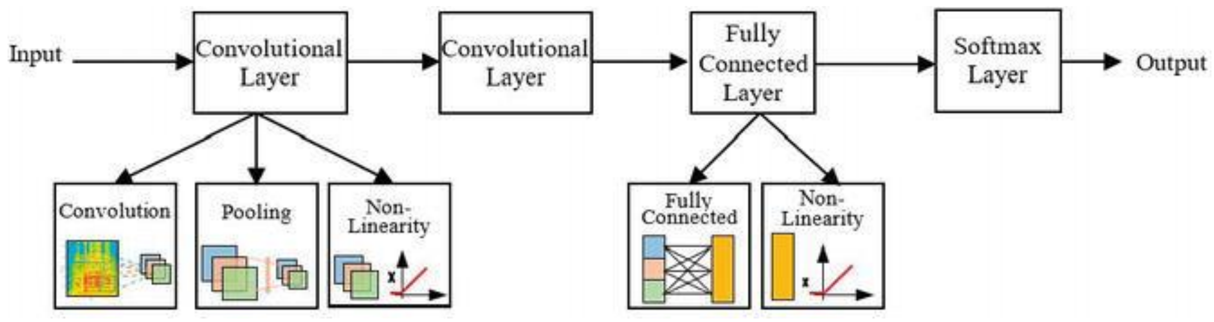
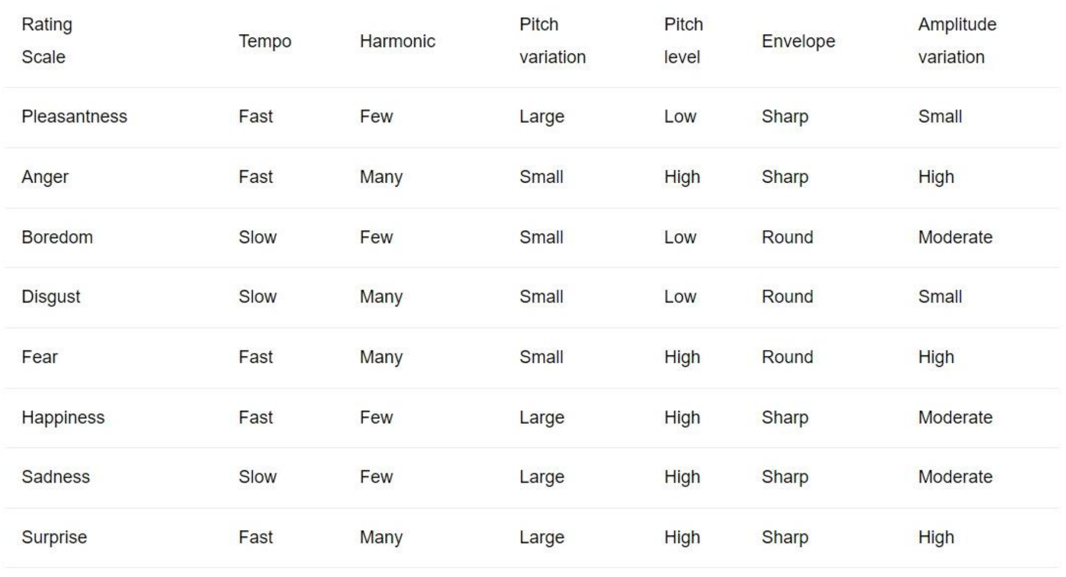


Fig. 2.4 Fully connected and Softmax Layer



Rating Scale	Tempo	Harmonic	Pitch variation	Pitch level	Envelope	Amplitude variation
Pleasantness	Fast	Few	Large	Low	Sharp	Small
Anger	Fast	Many	Small	High	Sharp	High
Boredom	Slow	Few	Small	Low	Round	Moderate
Disgust	Slow	Many	Small	Low	Round	Small
Fear	Fast	Many	Small	High	Round	High
Happiness	Fast	Few	Large	High	Sharp	Moderate
Sadness	Slow	Few	Large	High	Sharp	Moderate
Surprise	Fast	Many	Large	High	Sharp	High

Fig 2.5 Emotions and their audio characteristics

CHAPTER 3
DATASET

RAVDESS: Ryerson Audio-Visual Dataase of Emotional Speech and Song

RAVDESS contains 1440 files: 60 trials per actor x 24 actors = 1440. It contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech emotions includes calm, happy, sad, angry, fearful, surprise, and disgust expressions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

Filenaming Conventions

- Each of the 1440 files has a unique filename.
- The filename consists of a 7-part numerical identifier (e.g., 03-01-05-01-02-01-01.wav). These identifiers define the stimulus characteristics

Filename Identifiers

- Modality (01 = full-AV, 02 = video-only, 03 = audio-only).
- Vocal channel (01 = speech, 02 = song).
- Emotion (01 = neutral, 02 = calm, 03 = happy, 04 = sad, 05 = angry, 06 = fearful, 07 = disgust, 08 = surprised).
- Emotional intensity (01 = normal, 02 = strong). NOTE: There is no strong intensity for the 'neutral' emotion.
- Statement (01 = "Kids are talking by the door", 02 = "Dogs are sitting by the door").
- Repetition (01 = 1st repetition, 02 = 2nd repetition).
- Actor (01 to 24. Odd numbered actors are male, even numbered actors are female).

Example

1. Audio-only (03)
2. Speech (01)
3. Angry (05)
4. Normal intensity (01)
5. Statement "dogs" (02)
6. 1st Repetition (01)
7. 1st Actor (01)
8. Male, as the actor ID number is odd.

CHAPTER 4
PREPROCESSING AND FEATURE EXTRACTION

We have used three types of preprocessing in this project is

1. Mel-frequency Cepstral Coefficient (MFCC)
2. Mel spectrogram
3. Chroma

MEL SPECTROGRAM

The Mel spectrogram is used to provide our models with sound information similar to what a human would perceive. Input signal is passed through filter a banks to obtain the Mel spectrogram. The various steps involved in this feature extraction are seen below

Pre-emphasis

Higher frequencies are emphasized when the audio is passed through this step. Its purpose is to balance the spectrum as voiced sounds have a roll-off region in the high frequencies.

Windowing

The audio signal has many phones this needs to be broke into many segments for efficient processing. Hamming or hanning windows are preferred for breaking the signals into segments since this prevents noise in high-frequency domain which is caused when there is a sudden fall in amplitude that arises when the signal is directly chopped off.

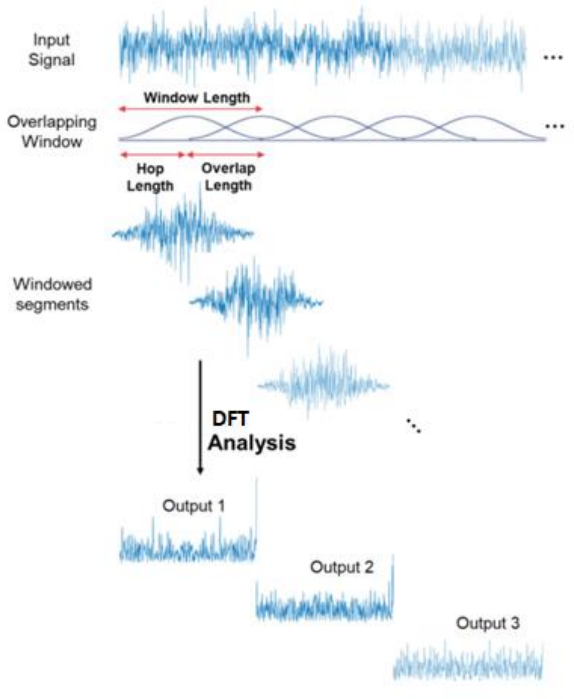


Fig 4.1 Windowing of an input signal

Discrete Fourier Transform (DFT)

Signal from the time domain is converted to frequency domain by this transform.

Mel-Filter Bank

Our ear’s perception of sound is very different from machine’s perception. It is noticed that modeling the human hearing property at the feature extraction improves the performance of perception models. So we will use the mel scale to map the actual frequency to the frequency that humans will perceive.

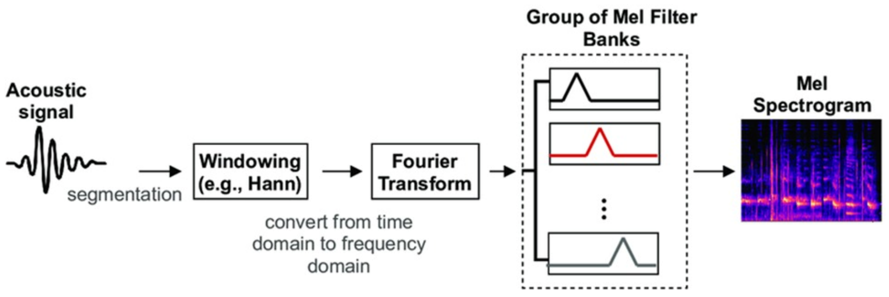


Fig 4.2 Mel Filterbank application on DFT signal

In python mel-spectrogram can be applied to a signal with the function available in librosa library *librosa.feature.melspectrogram()*

MEL FREQUENCY CEPSTRAL CO-EFFICIENT (MFCC)

MFCC follows the same steps as mel-spectrogram. The mel spectrogram is log scaled and DCT is computed to obtain MFCC

Discrete Cosine Transform (DCT)

Cosine function has very large compression compared to sine functions thus requiring fewer functions to approximate a typical signal.

In python mfcc can be applied to a signal with the function available in librosa library *librosa.feature.mfcc()*

Applying Log

Log functions have high gradient for lower inputs while lower gradient for high values of input. This property is similar to human perception of sound. Thus log is applied after Mel-filter to replicate.

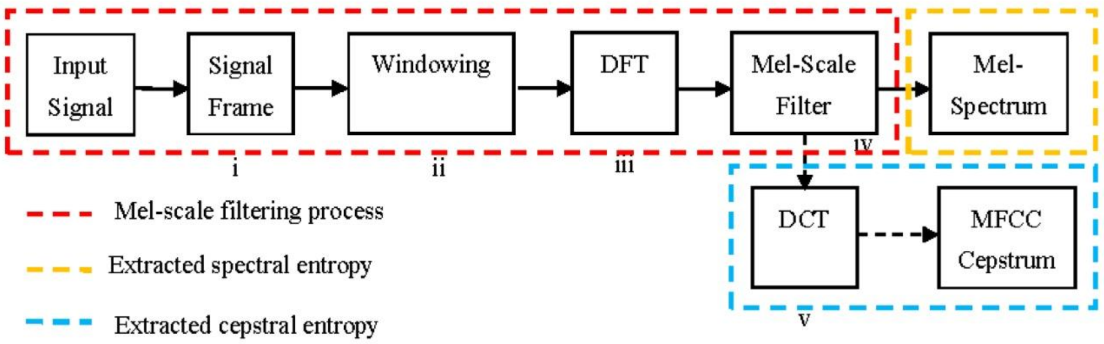


Fig 4.3 Differennce between MFCC and Mel-spectrogram process

CHROMA FEATURE

EXTRACTION PROCESS

This feature extraction process condenses the tonal content of the audio signal thus making this an important pre-processing step in analysis of harmonic similarities between two or more audio sources.

Pitch

It is a perceptual property of sound that allows it to be ordered on a frequency related scale. This allows sound to be judged on a scale as “higher” or “lower”.

Frequency

It determines the pitch of sound. A higher frequency signal has a higher perceived pitch.

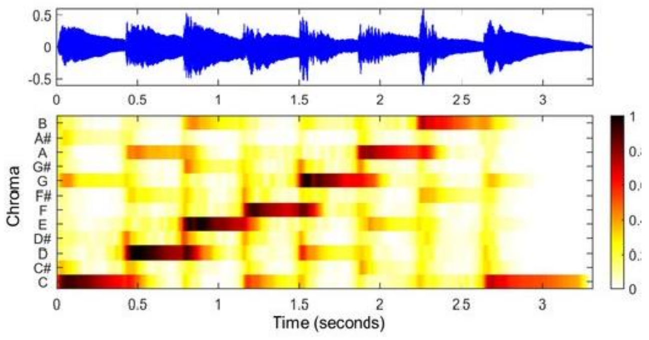


Fig 4.4 (top) Audiorecording of C-Major (Bottom) Chromatic representation of the audio

Amplitude

It determines the loudness of the sound.

A chromagram of a signal can be generated by using the function *librosa.feature.chroma_stft()* in the librosa audio analysis library.

CHAPTER 5

PROBLEM STATEMENT

Although the logical intelligence of personal computers have developed beyond what anyone could imagine the efficiency of computers still take a hit due to their lack of emotional intelligence. The human-computer interaction seems uninteresting at times as is monotonic. SER systems can improve the emotional intelligence of computers by training to detect emotions from voice using many parameters. They also come with may added benefits.

MERITS

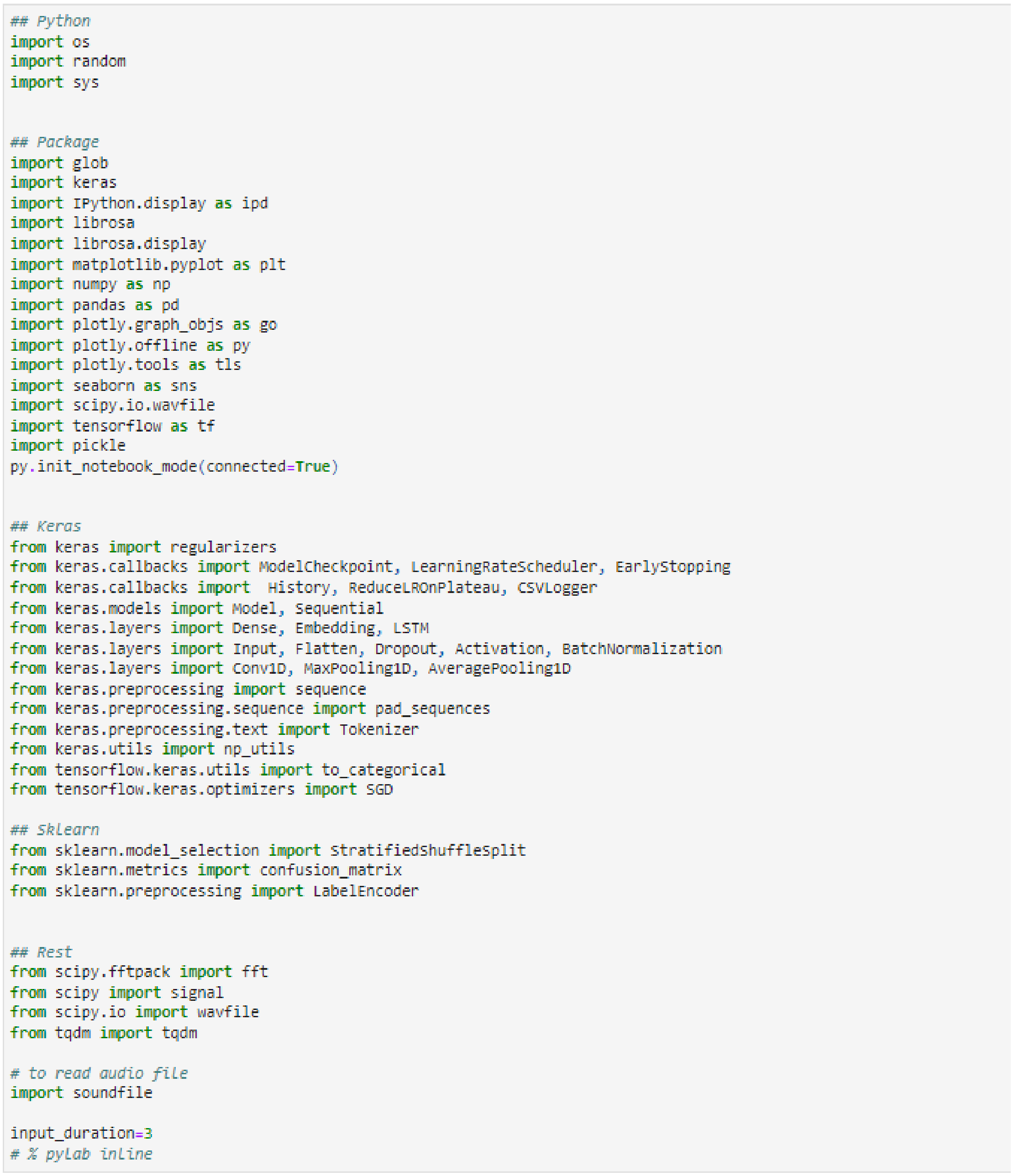
- In psychiatric analysis computers can aid in detecting lies.
- Call Center could analyze user's voice to gauge their emotion and respond accordingly to improve quality of service.
- In transports the drivers emotions can be analyzed for signs of stress and fatigue.
- The naturalness of computer generated speech can be vastly imporved with feedback from the emotional recognition system.

TECHNICAL LIMITATIONS & OTHER CHALLENGES FACED:

- Different regional accents in a language make the process of training the system difficult.
- Very fewer comprehensive studies are found on Indian regional language
- Finding and importing specific versions of several python libraries is also one of the difficult challenges.
- The training and modeling is computationally intensive and requires powerful hardware. The requirement of GPU is still a challenge to process the dataset as it required an efficient GPU.
- The emotional characteristics are sometimes language specific. It means that the system that works for one language might not work well for another. So a universal model for emotion recognition is hard to implement.

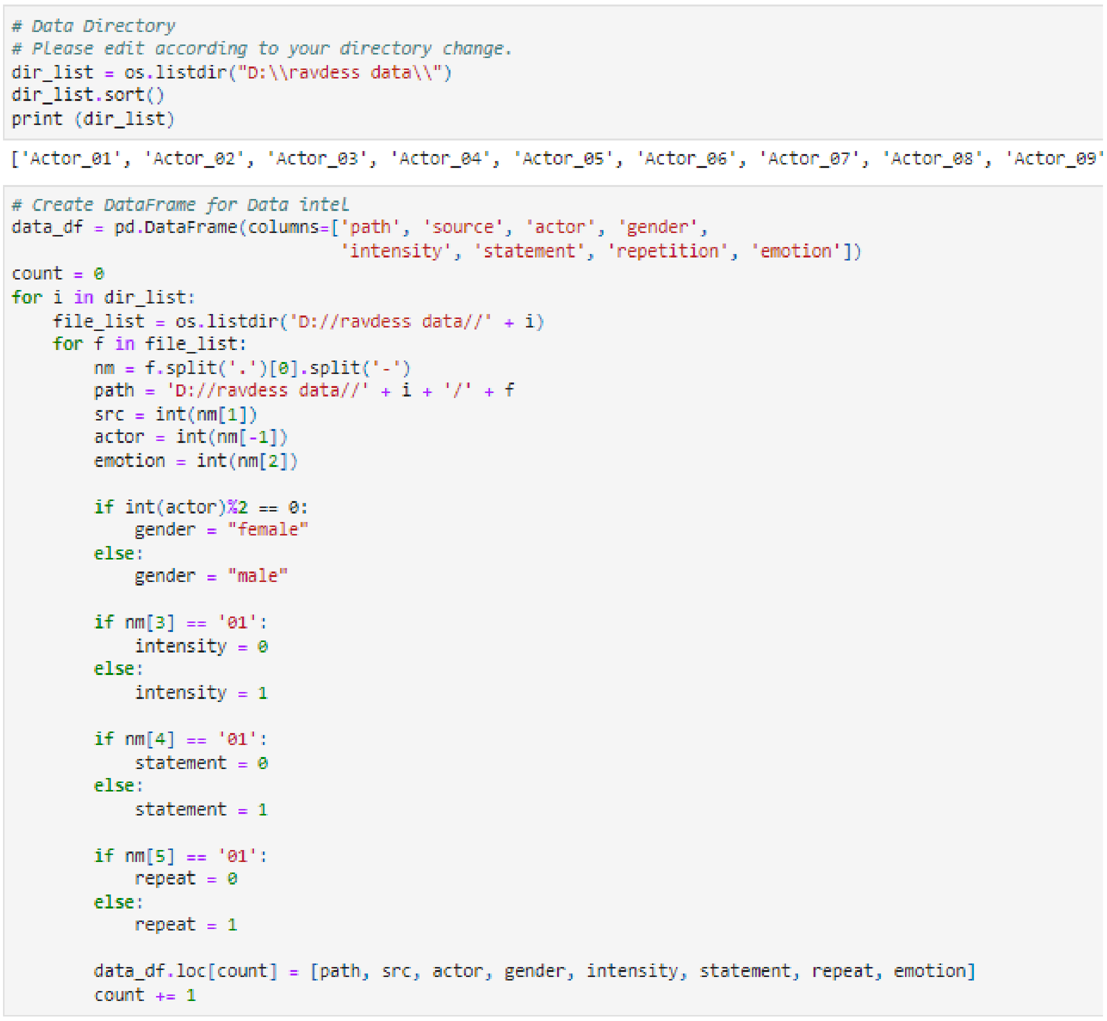
CHAPTER 6

CODE ANALYSIS



```python
## Python
import os
import random
import sys


## Package
import glob
import keras
import IPython.display as ipd
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import plotly.graph_objs as go
import plotly.offline as py
import plotly.tools as tls
import seaborn as sns
import scipy.io.wavfile
import tensorflow as tf
import pickle
py.init_notebook_mode(connected=True)


## Keras
from keras import regularizers
from keras.callbacks import ModelCheckpoint, LearningRateScheduler, EarlyStopping
from keras.callbacks import  History, ReduceLROnPlateau, CSVLogger
from keras.models import Model, Sequential
from keras.layers import Dense, Embedding, LSTM
from keras.layers import Input, Flatten, Dropout, Activation, BatchNormalization
from keras.layers import Conv1D, MaxPooling1D, AveragePooling1D
from keras.preprocessing import sequence
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
from keras.utils import np_utils
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import SGD

## Sklearn
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder


## Rest
from scipy.fftpack import fft
from scipy import signal
from scipy.io import wavfile
from tqdm import tqdm

# to read audio file
import soundfile

input_duration=3
# % pylab inline
```

We are importing the required libraries.



```
# Data Directory
# Please edit according to your directory change.
dir_list = os.listdir("D:\\ravdess data\\")
dir_list.sort()
print (dir_list)
```

```
['Actor_01', 'Actor_02', 'Actor_03', 'Actor_04', 'Actor_05', 'Actor_06', 'Actor_07', 'Actor_08', 'Actor_09'
```

```
# Create DataFrame for Data intel
data_df = pd.DataFrame(columns=['path', 'source', 'actor', 'gender',
                                'intensity', 'statement', 'repetition', 'emotion'])
count = 0
for i in dir_list:
    file_list = os.listdir('D://ravdess data//' + i)
    for f in file_list:
        nm = f.split('.')[0].split('-')
        path = 'D://ravdess data//' + i + '/' + f
        src = int(nm[1])
        actor = int(nm[-1])
        emotion = int(nm[2])

        if int(actor)%2 == 0:
            gender = "female"
        else:
            gender = "male"

        if nm[3] == '01':
            intensity = 0
        else:
            intensity = 1

        if nm[4] == '01':
            statement = 0
        else:
            statement = 1

        if nm[5] == '01':
            repeat = 0
        else:
            repeat = 1

        data_df.loc[count] = [path, src, actor, gender, intensity, statement, repeat, emotion]
        count += 1
```

In dir_list we are storing the list of all directories which in this case are for each of the actors. Then a dataframe is created. We are storing the files in each of the author's folder in file_list. The files in the list are processed to find whether the audio is speech or song also the actor, emotion, gender, intensity of emotion, the statement that was uttered and the repetition count. This is then stored in the dataframe in its appropriate coloumn

```python
def log_specgram(audio, sample_rate, window_size=20,
                 step_size=10, eps=1e-10):
    nperseg = int(round(window_size * sample_rate / 1e3))
    noverlap = int(round(step_size * sample_rate / 1e3))
    freqs, times, spec = signal.spectrogram(audio,
                                    fs=sample_rate,
                                    window='hann',
                                    nperseg=nperseg,
                                    noverlap=noverlap,
                                    detrend=False)
    return freqs, times, np.log(spec.T.astype(np.float32) + eps)

```

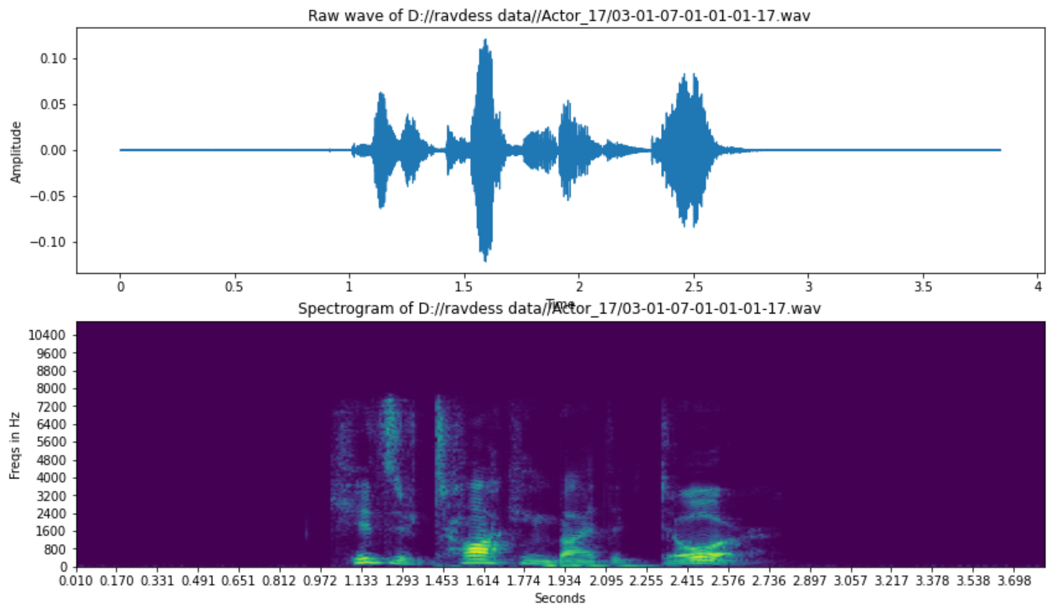
```python
sample_rate/ len(samples)
```

```
0.26060749320411297
```

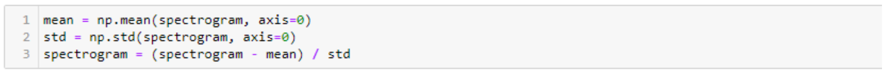
```python
# Plotting Wave Form and Spectrogram
freqs, times, spectrogram = log_specgram(samples, sample_rate)

fig = plt.figure(figsize=(14, 8))
ax1 = fig.add_subplot(211)
ax1.set_title('Raw wave of ' + filename)
ax1.set_ylabel('Amplitude')
librosa.display.waveshow(samples, sr=sample_rate)

ax2 = fig.add_subplot(212)
ax2.imshow(spectrogram.T, aspect='auto', origin='lower',
           extent=[times.min(), times.max(), freqs.min(), freqs.max()])
ax2.set_yticks(freqs[::16])
ax2.set_xticks(times[::16])
ax2.set_title('Spectrogram of ' + filename)
ax2.set_ylabel('Freqs in Hz')
ax2.set_xlabel('Seconds')
```

```
Text(0.5, 0, 'Seconds')
```

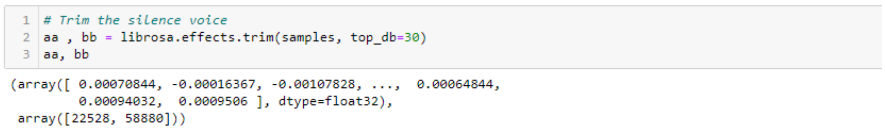
A spectrogram of the input signl is generated.



The generated spectrogram and wave is seen above.

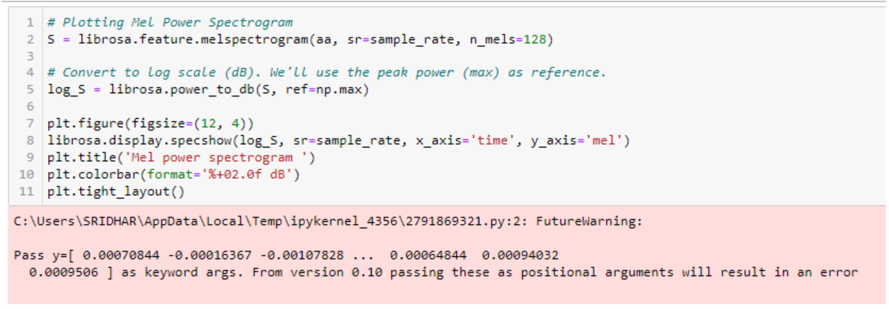


```
mean = np.mean(spectrogram, axis=0)
std = np.std(spectrogram, axis=0)
spectrogram = (spectrogram - mean) / std
```



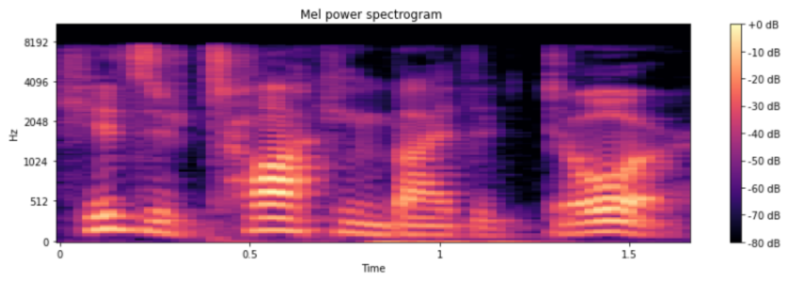
```
# Trim the silence voice
aa , bb = librosa.effects.trim(samples, top_db=30)
aa, bb
```

```
(array([ 0.00070844, -0.00016367, -0.00107828, ...,  0.00064844,
         0.00094032,  0.0009506 ], dtype=float32),
 array([22528, 58880]))
```

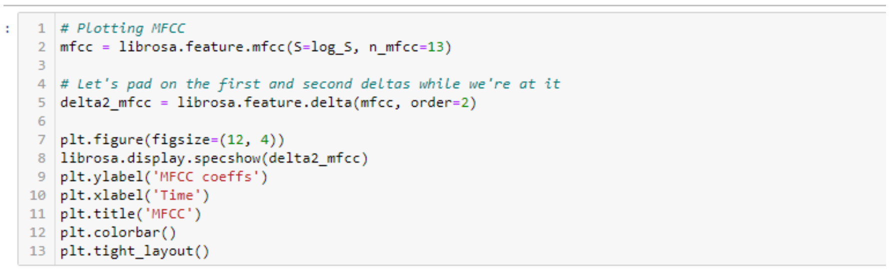


```
# Plotting Mel Power Spectrogram
S = librosa.feature.melspectrogram(aa, sr=sample_rate, n_mels=128)

# Convert to log scale (dB). We'll use the peak power (max) as reference.
log_S = librosa.power_to_db(S, ref=np.max)

plt.figure(figsize=(12, 4))
librosa.display.specshow(log_S, sr=sample_rate, x_axis='time', y_axis='mel')
plt.title('Mel power spectrogram ')
plt.colorbar(format='%+02.0f dB')
plt.tight_layout()
```

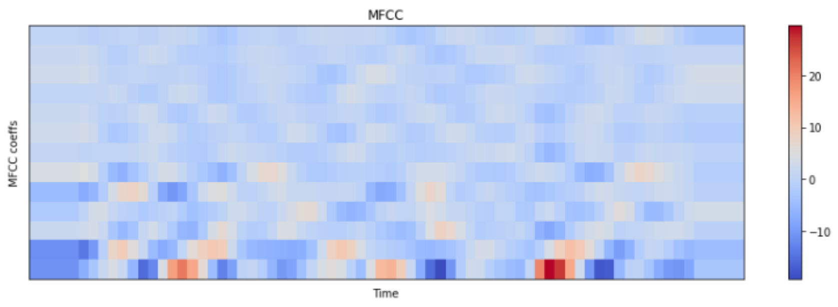
```
C:\Users\SRIDHAR\AppData\Local\Temp\ipykernel_4356\2791869321.py:2: FutureWarning:

Pass y=[ 0.00070844 -0.00016367 -0.00107828 ...  0.00064844  0.00094032
  0.0009506 ] as keyword args. From version 0.10 passing these as positional arguments will result in an error
```



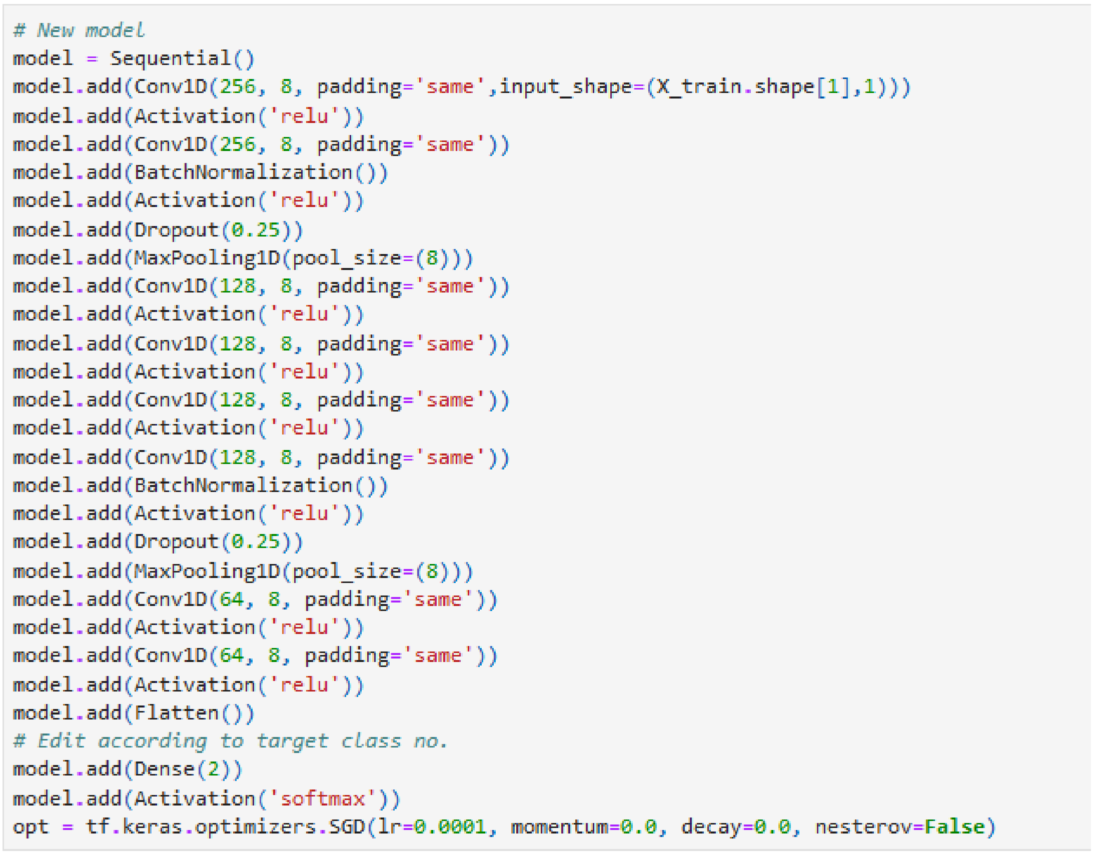
The mel power spectrogram of the previously obtained spectrogram is generated with *librosa.feature.melspectrogram()* and plotted with *librosa.display.specshow()*. It can be noted the log values of samples were taken to represent in decibel scale.



```
# Plotting MFCC
mfcc = librosa.feature.mfcc(S=log_S, n_mfcc=13)

# Let's pad on the first and second deltas while we're at it
delta2_mfcc = librosa.feature.delta(mfcc, order=2)

plt.figure(figsize=(12, 4))
librosa.display.specshow(delta2_mfcc)
plt.ylabel('MFCC coeffs')
plt.xlabel('Time')
plt.title('MFCC')
plt.colorbar()
plt.tight_layout()
```



The mfcc extraction has been made with *librosa.feature.mfcc()* and plotted with *librosa.display.specshow()*.



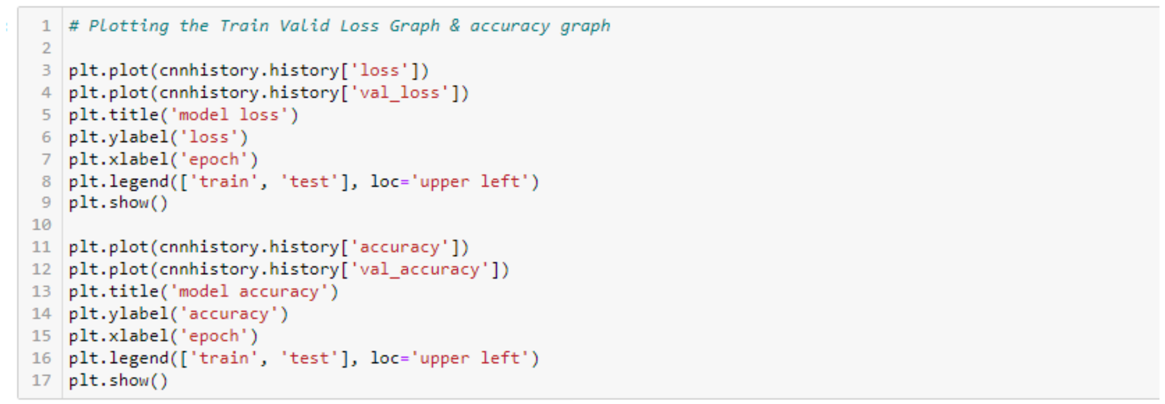
```
# New model
model = Sequential()
model.add(Conv1D(256, 8, padding='same',input_shape=(X_train.shape[1],1)))
model.add(Activation('relu'))
model.add(Conv1D(256, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(128, 8, padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))
model.add(MaxPooling1D(pool_size=(8)))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Conv1D(64, 8, padding='same'))
model.add(Activation('relu'))
model.add(Flatten())
# Edit according to target class no.
model.add(Dense(2))
model.add(Activation('softmax'))
opt = tf.keras.optimizers.SGD(lr=0.0001, momentum=0.0, decay=0.0, nesterov=False)
```

We will create a CNN SER model in order to categorise the photos into their appropriate clases because CNN is very well known and popular classigication netural network

The proposed model architecture has a…

1. Sequential model:
   A plain stack of layers with one input tensor and one output tensor for every layer
2. Convolutional 1D Layer:
   It has a filter of 256 and a kernel size of 8 and “same” padding which results in padding with zero evenly to left/right or up/down such that output has same height and width demension as input. The activation layer used is ReLU.
3. Batch Normalization:
   This layer is applied to maintain the mean output close to zero and the standard deviation close to one.
4. Dropout layer:
   This layer has a rate of 0.25. This means that input units are set to 0 with a frequency of specified rate(here 0.25) at each step during the training time. This helps prevent overfitting.
5. Maxpooling 1D layer:
   This layer downsamples the input by taking the max value over a spatial window of size specified (here pool size is 8)
6. Flattening Layer:
   After multiple passes throught convolution, activation, droput and pooling layers we finally have a flattening layer. This layer compresses the output of the previous layer into a single dimensional vector.

CHAPTER 7
RESULT AND SNAPSHOTS



```python
# Plotting the Train Valid Loss Graph & accuracy graph

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

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

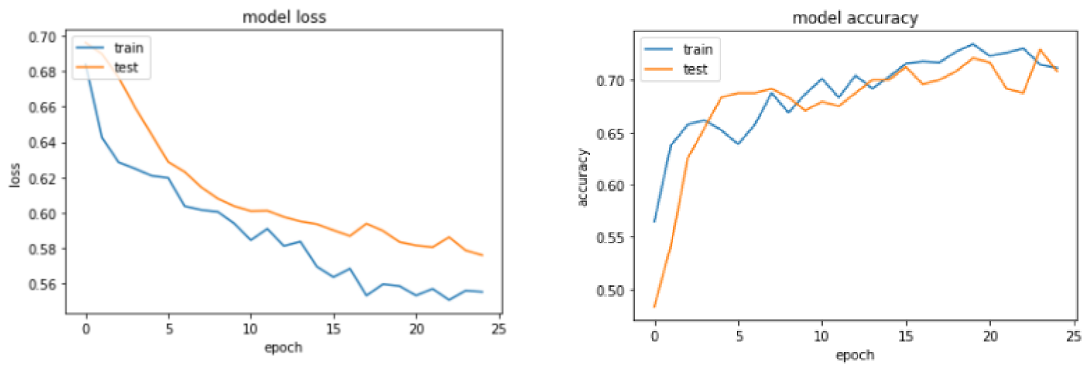
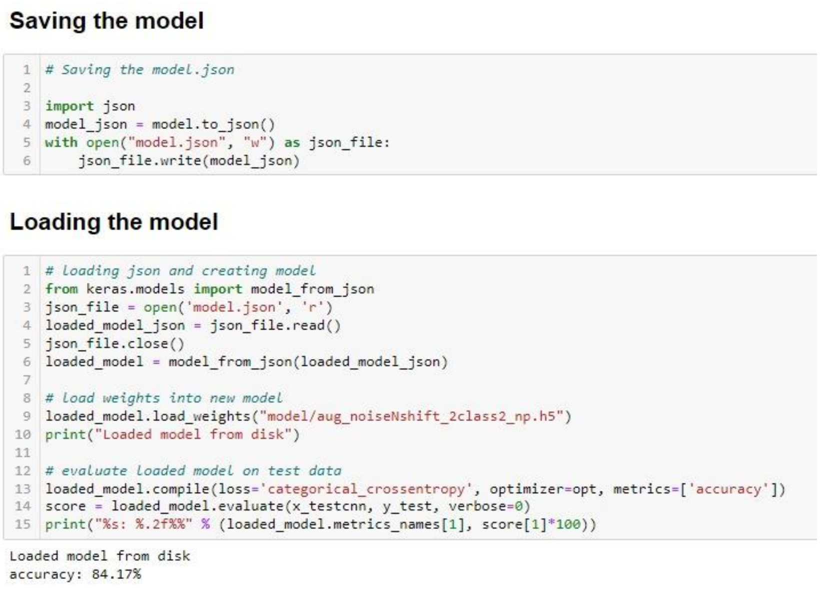


Fig 6.1 Loss and Accuracy Graphs

These are the accuracy and loss graph of the model on the training dataset given as input. The graphs were created with *matplotlib* package.



Saving the model

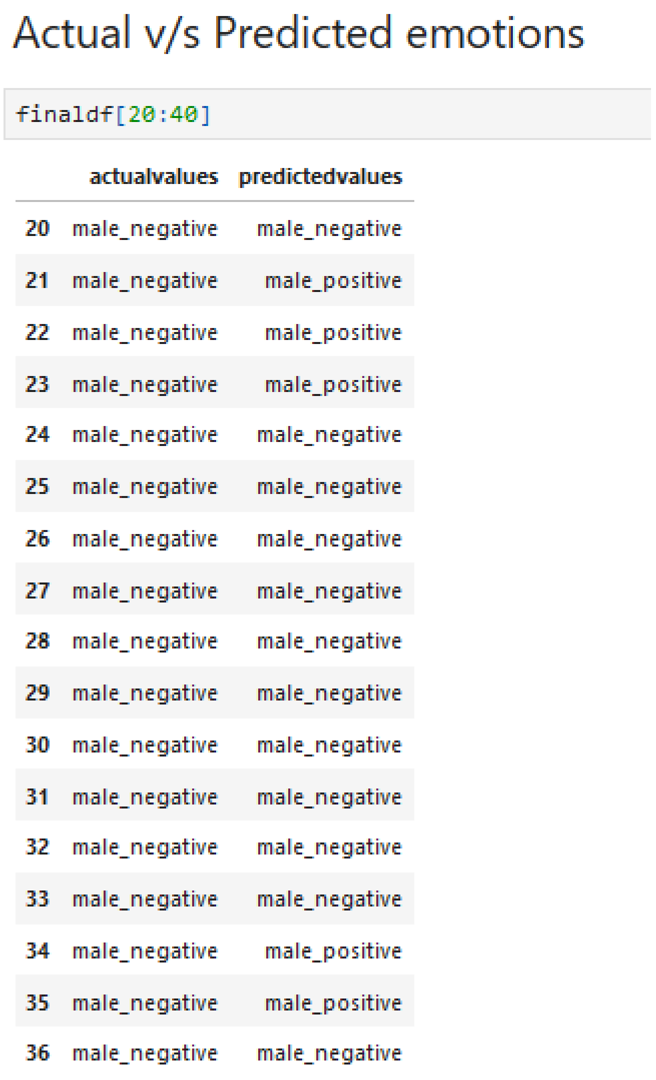
```python
# Saving the model.json

import json
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

Loading the model

```python
# loading json and creating model
from keras.models import model_from_json
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)

# load weights into new model
loaded_model.load_weights("model/aug_noiseNshift_2class2_np.h5")
print("Loaded model from disk")

# evaluate loaded model on test data
loaded_model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
score = loaded_model.evaluate(x_testcnn, y_test, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], score[1]*100))
```

```
Loaded model from disk
accuracy: 84.17%
```

Fig 6.2 Accuracy of model



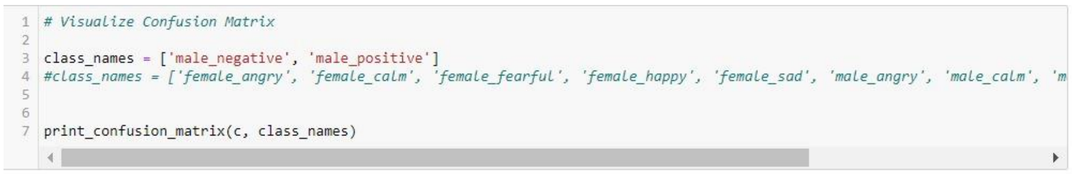
Actual v/s Predicted emotions

```
finaldf[20:40]
```

	actualvalues	predictedvalues
20	male_negative	male_negative
21	male_negative	male_positive
22	male_negative	male_positive
23	male_negative	male_positive
24	male_negative	male_negative
25	male_negative	male_negative
26	male_negative	male_negative
27	male_negative	male_negative
28	male_negative	male_negative
29	male_negative	male_negative
30	male_negative	male_negative
31	male_negative	male_negative
32	male_negative	male_negative
33	male_negative	male_negative
34	male_negative	male_positive
35	male_negative	male_positive
36	male_negative	male_negative

Fig 6.3 Actual vs Predicted Emotions

This is a sample of predicted emotions by the model against the actual emotions.



```
# Visualize Confusion Matrix

class_names = ['male_negative', 'male_positive']
#class_names = ['female_angry', 'female_calm', 'female_fearful', 'female_happy', 'female_sad', 'male_angry', 'male_calm', 'm


print_confusion_matrix(c, class_names)
```

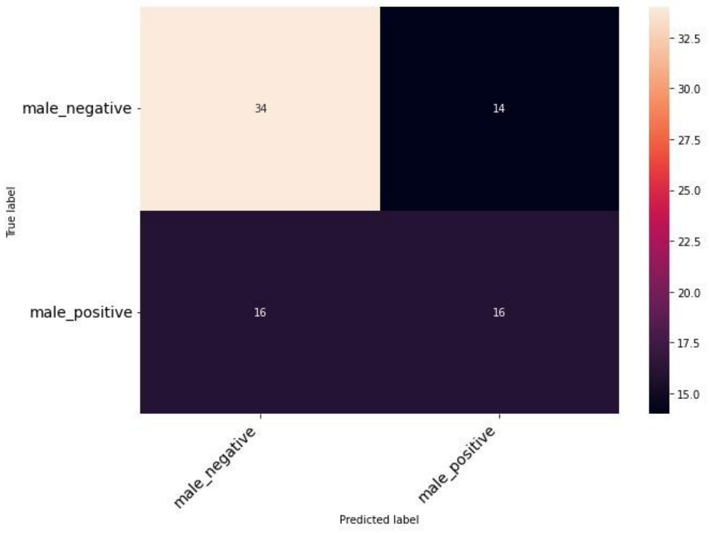
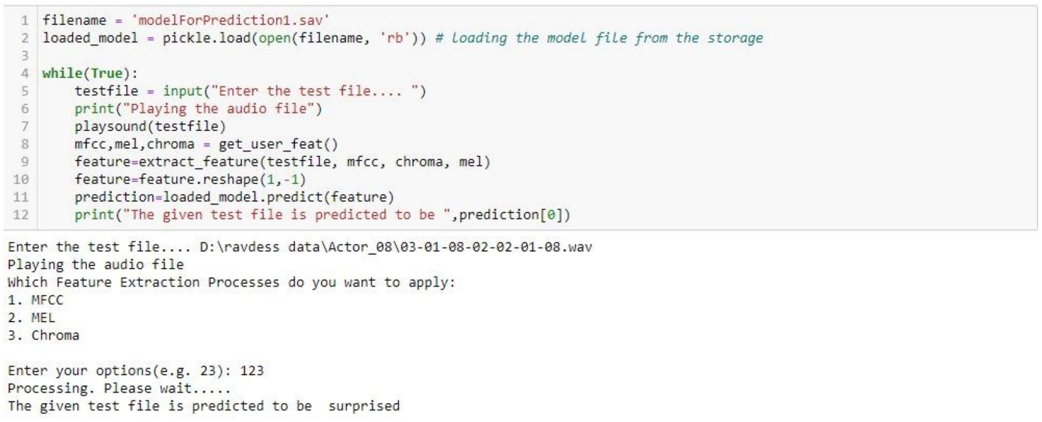


Fig 6.4 Confusion Matrix

Confusion matrix is the table that is used to define the performance of classification and analysis algorithm. This is the confusion matrix of the emotions predicted by the model.



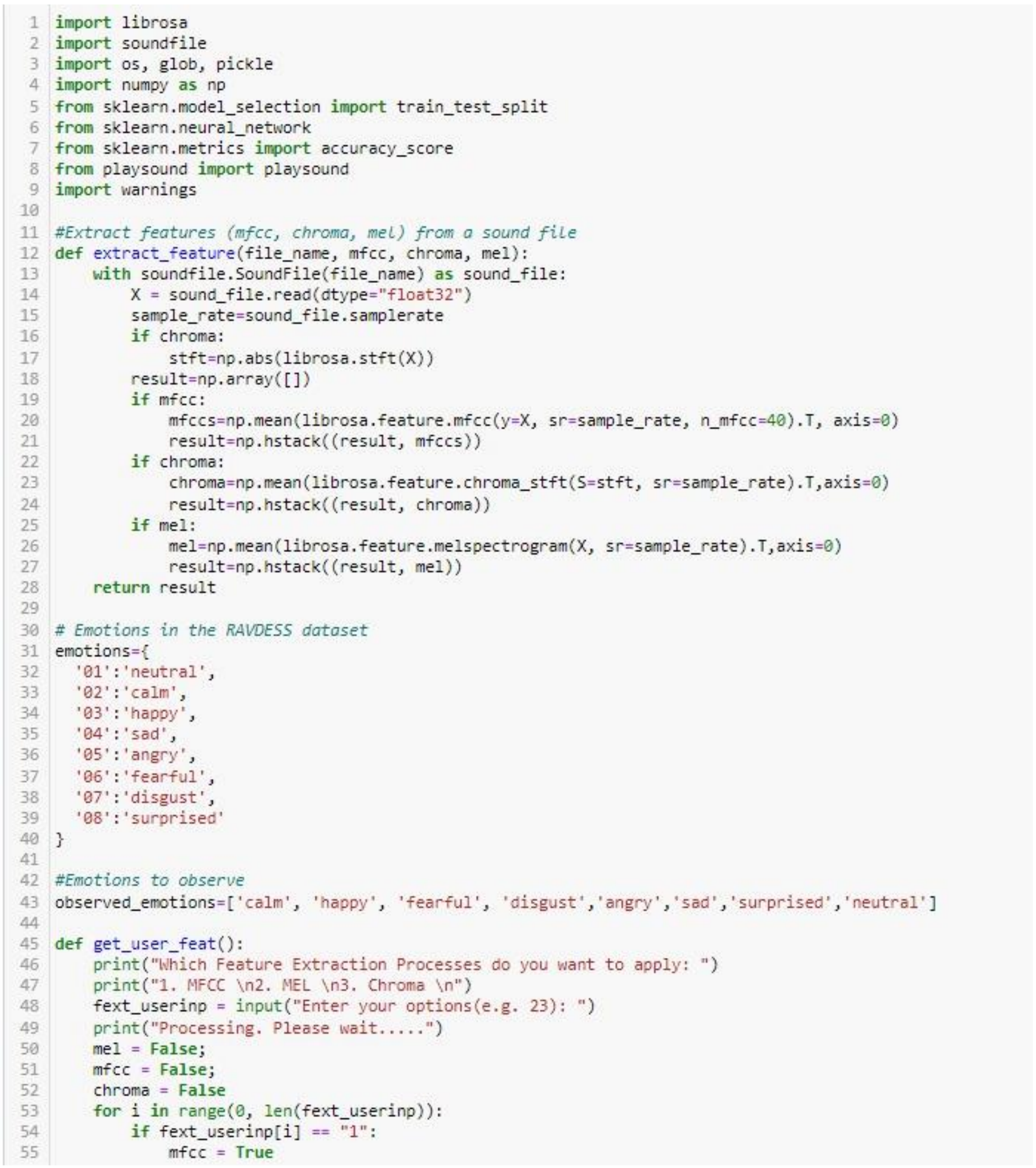
```
filename = 'modelForPrediction1.sav'
loaded_model = pickle.load(open(filename, 'rb')) # Loading the model file from the storage

while(True):
    testfile = input("Enter the test file.... ")
    print("Playing the audio file")
    playsound(testfile)
    mfcc,mel,chroma = get_user_feat()
    feature=extract_feature(testfile, mfcc, chroma, mel)
    feature=feature.reshape(1,-1)
    prediction=loaded_model.predict(feature)
    print("The given test file is predicted to be ",prediction[0])
```

```
Enter the test file.... D:\ravdess data\Actor_08\03-01-08-02-02-01-08.wav
Playing the audio file
Which Feature Extraction Processes do you want to apply:
1. MFCC
2. MEL
3. Chroma

Enter your options(e.g. 23): 123
Processing. Please wait.....
The given test file is predicted to be  surprised
```

Fig 6.5 User input and emotion prediction

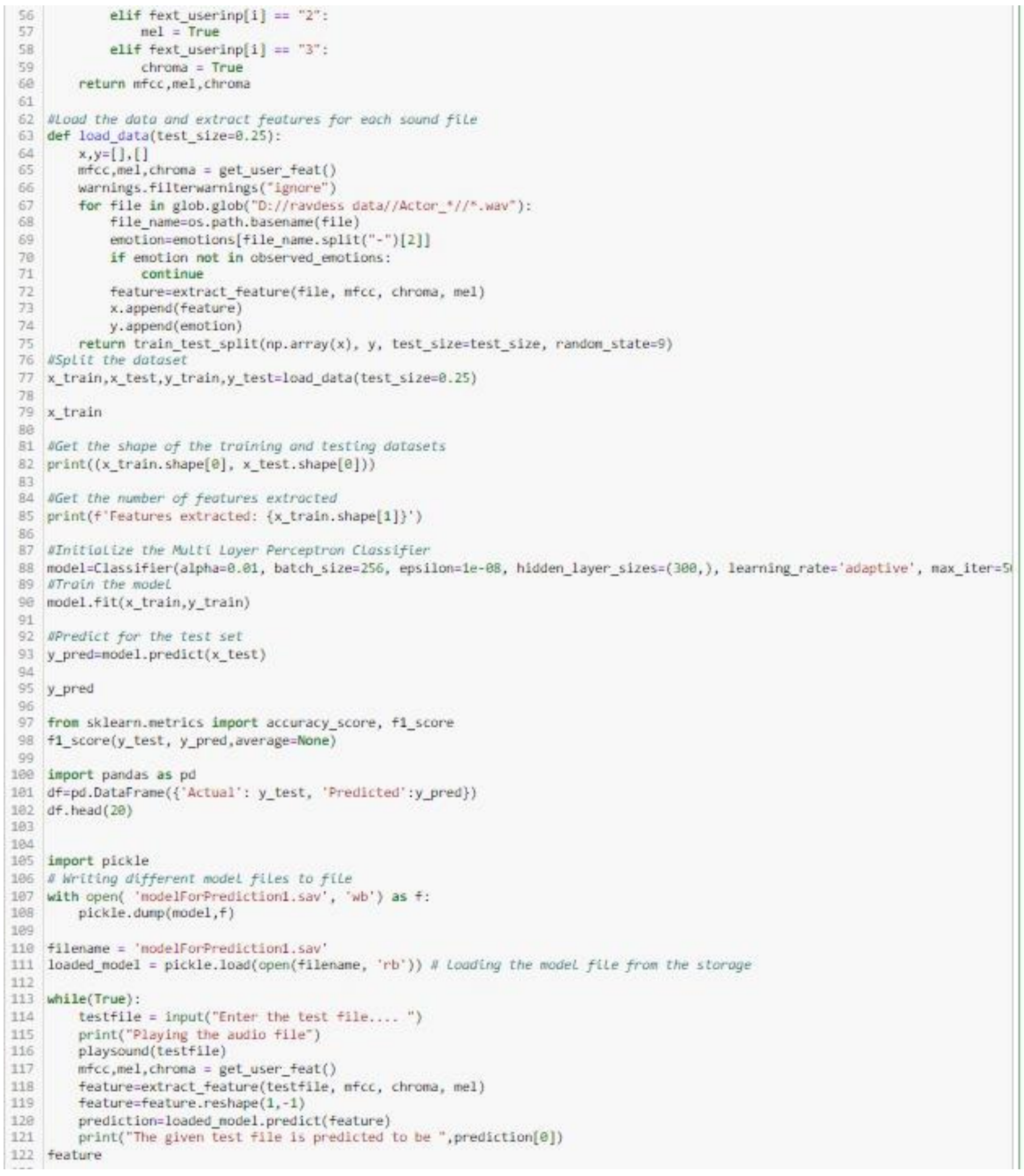
CHAPTER 8

SOURCE CODE



```python
import librosa
import soundfile
import os, glob, pickle
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network
from sklearn.metrics import accuracy_score
from playsound import playsound
import warnings

#Extract features (mfcc, chroma, mel) from a sound file
def extract_feature(file_name, mfcc, chroma, mel):
    with soundfile.SoundFile(file_name) as sound_file:
        X = sound_file.read(dtype="float32")
        sample_rate=sound_file.samplerate
        if chroma:
            stft=np.abs(librosa.stft(X))
        result=np.array([])
        if mfcc:
            mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T, axis=0)
            result=np.hstack((result, mfccs))
        if chroma:
            chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
            result=np.hstack((result, chroma))
        if mel:
            mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
            result=np.hstack((result, mel))
    return result

# Emotions in the RAVDESS dataset
emotions={
  '01':'neutral',
  '02':'calm',
  '03':'happy',
  '04':'sad',
  '05':'angry',
  '06':'fearful',
  '07':'disgust',
  '08':'surprised'
}

#Emotions to observe
observed_emotions=['calm', 'happy', 'fearful', 'disgust','angry','sad','surprised','neutral']

def get_user_feat():
    print("Which Feature Extraction Processes do you want to apply: ")
    print("1. MFCC \n2. MEL \n3. Chroma \n")
    fext_userinp = input("Enter your options(e.g. 23): ")
    print("Processing. Please wait.....")
    mel = False;
    mfcc = False;
    chroma = False
    for i in range(0, len(fext_userinp)):
        if fext_userinp[i] == "1":
            mfcc = True
```

Fig 8.1 Source Code 1 of 2



```
        elif fext_userinp[i] == "2":
            mel = True
        elif fext_userinp[i] == "3":
            chroma = True
    return mfcc,mel,chroma

#Load the data and extract features for each sound file
def load_data(test_size=0.25):
    x,y=[],[]
    mfcc,mel,chroma = get_user_feat()
    warnings.filterwarnings("ignore")
    for file in glob.glob("D://ravdess data//Actor_*//*.wav"):
        file_name=os.path.basename(file)
        emotion=emotions[file_name.split("-")[2]]
        if emotion not in observed_emotions:
            continue
        feature=extract_feature(file, mfcc, chroma, mel)
        x.append(feature)
        y.append(emotion)
    return train_test_split(np.array(x), y, test_size=test_size, random_state=9)
#Split the dataset
x_train,x_test,y_train,y_test=load_data(test_size=0.25)

x_train

#Get the shape of the training and testing datasets
print((x_train.shape[0], x_test.shape[0]))

#Get the number of features extracted
print(f'Features extracted: {x_train.shape[1]}')

#Initialize the Multi Layer Perceptron Classifier
model=Classifier(alpha=0.01, batch_size=256, epsilon=1e-08, hidden_layer_sizes=(300,), learning_rate='adaptive', max_iter=5
#Train the model
model.fit(x_train,y_train)

#Predict for the test set
y_pred=model.predict(x_test)

y_pred

from sklearn.metrics import accuracy_score, f1_score
f1_score(y_test, y_pred,average=None)

import pandas as pd
df=pd.DataFrame({'Actual': y_test, 'Predicted':y_pred})
df.head(20)


import pickle
# Writing different model files to file
with open( 'modelForPrediction1.sav', 'wb') as f:
    pickle.dump(model,f)

filename = 'modelForPrediction1.sav'
loaded_model = pickle.load(open(filename, 'rb')) # Loading the model file from the storage

while(True):
    testfile = input("Enter the test file.... ")
    print("Playing the audio file")
    playsound(testfile)
    mfcc,mel,chroma = get_user_feat()
    feature=extract_feature(testfile, mfcc, chroma, mel)
    feature=feature.reshape(1,-1)
    prediction=loaded_model.predict(feature)
    print("The given test file is predicted to be ",prediction[0])
feature
```

Fig 8.2 Source Code 2 of 2

CHAPTER 9
CONCLUSION & FUTURE PLANS

CONCLUSION

After constructing various models, we got the better CNN model for the emotion detection task. We reached 84.17% accuracy from the previously available model. Our model would’ve performed better with high volume of dataset and by using more feature extraction methods. This project could be extended to integrate with a robot to help it to understand the mood dynamics of humans, which will help it to have a better conversation.

FUTURE PLANS

- By training the model with different Dataset we can achieve various levels of accuracy for various accents.
- Work needs to be done in the creation of a extensive database of indian regional audio visual emotion database.
- In this project we have used only 3 types of Feature extraction methods. Multiple methods can be implement we can try various methods.
- Acoustic features of sound data other than the ones used in this project could be examined for its relevance in the field of speech emotion recognition.
- Lexical and acoustic based models can be trained and tested for accuracy and other parameters.

CHAPTER 10

REFERENCES

1. Singla, Chaitanya and Singh, Sukhdev and Pathak, Monika, Automatic Audio Based Emotion Recognition System: Scope and Challenges (April 1, 2020). Proceedings of the International Conference on Innovative Computing & Communications (ICICC) 2020, Available at SSRN: https://ssrn.com/abstract=3565861

2. Dong Yu and Li Deng. AUTOMATIC SPEECH RECOGNITION. Springer, 2016.

3. N. Morgan, "Deep and wide: Multiple layers in automatic speechrecognition," IEEE Trans. Audio, Speech, Lang. Process., vol. 20, no.1, pp. 7–13, Jan. 2012. Mohamed, G. Dahl, and G. Hinton, "Deep belief networks for phonerecognition," in Proc. NIPS Workshop Deep Learn. Speech Recognition Related Applicat., 2009.

4. Mohamed, D. Yu, and L. D eng, "Investigation of full-sequence training of deep belief networks for speech recognition," in Proc.Interspeech, 2010, pp. 2846–2849.

5. L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning forbuilding deep architectures," in Proc. IEEE Int. Conf. Acoustics,Speech, Signal Process., 2012, pp. 2133–2136.

6. G. Dahl,D.Yu, L.Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," IEEETrans. Audio, Speech, Lang. Process., vol. 20, no. 1, pp. 30–42, Jan.2012.
7. Artificial intelligence, and the future of the human mind. Simon and Schuster, 2007.
8. Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. Artificial intelligence: a modern approach, volume 2. Prentice hall Upper Saddle River, 2003.
9. Lawrence R Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition, volume14. PTR Prentice Hall Englewood Clis, 1993.
10. Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. In Readings in speech recognition, pages 308 - 319. Elsevier, 1990.
11. Stephen E Levinson, Lawrence R Rabiner, and Man Mohan Sondhi. An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. The Bell System Technical Journal, 62(4):1035{1074, 1983.
12. RAVDESS audio-visual emotion dataset https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio