Configuration Manual

MSc Research Project
Data Analytics

Bijalben Prafulchandra Bhagat
Student ID: x20167326

School of Computing
National College of Ireland

Supervisor: Dr. Christian Horn

National College of IrelandProject Submission Sheet School of Computing

Student Name:	Bijalben Prafulchandra Bhagat
Student ID:	x20167326
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Dr. Christian Horn
Submission Due Date:	16/12/2021
Project Title:	Configuration Manual
Word Count:	817
Page Count:	9

Signature:	
Date:	16th December 2021

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keepa copy on computer.	□

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

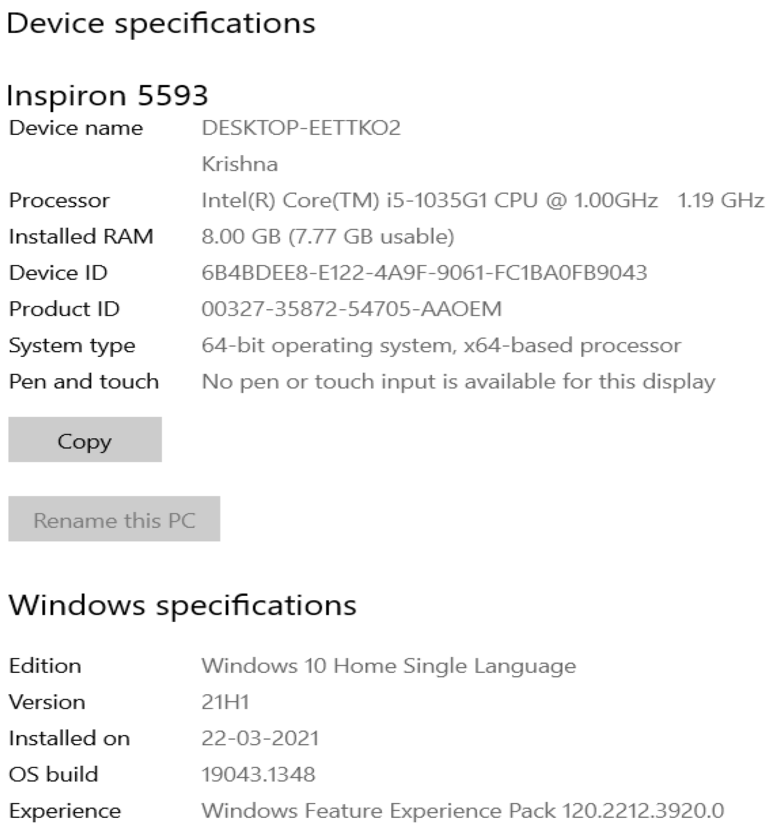
Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration manual

Bijalben Prafulchandra Bhagat
x20167326

1 Hardware Requirements

Windows 10 with a 64-bit operating system, an AMD Ryzen 5 4600HS with Radeon Graphics 3.00 GHz and 16 GB of RAM were the computer configurations used for project the implementation.



Device specifications

Inspiron 5593

Device name	DESKTOP-EETTKO2
	Krishna
Processor	Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz
Installed RAM	8.00 GB (7.77 GB usable)
Device ID	6B4BDEE8-E122-4A9F-9061-FC1BA0FB9043
Product ID	00327-35872-54705-AAOEM
System type	64-bit operating system, x64-based processor
Pen and touch	No pen or touch input is available for this display

Windows specifications

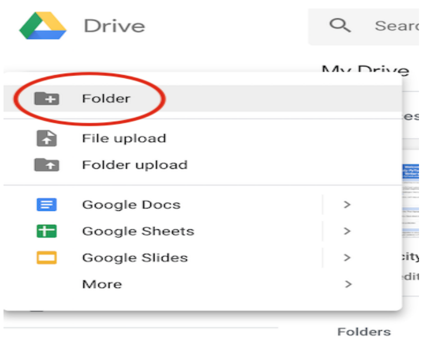
Edition	Windows 10 Home Single Language
Version	21H1
Installed on	22-03-2021
OS build	19043.1348
Experience	Windows Feature Experience Pack 120.2212.3920.0

Figure 1: Hardware Configuration

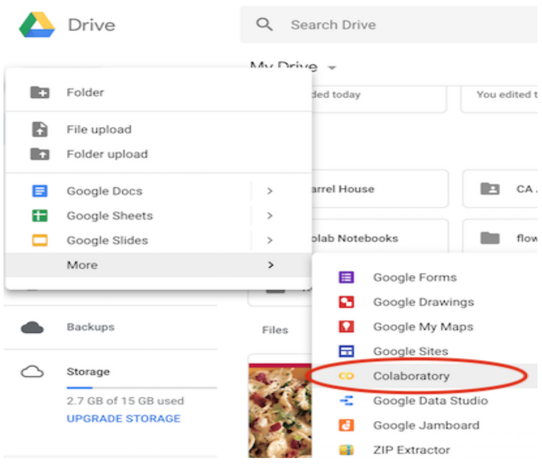
2 Software Requirement

Throughout the research, we used the Python 3.8 to write all of the coding sections. We introduced Python codes using the Google Colab platform. The first thing we need to register gmail account as it provide google colaboratory. In only a few code lines, anyone can load an audio dataset, train an audio recognition on it, and test the model with Colab. Colab notebooks run programs on Google's cloud servers, allowing you to take advantage of Google technology, such as GPUs and TPUs, independent of your machine's capabilities.

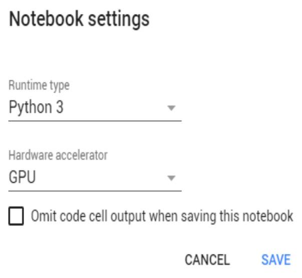
By browsing to your Google Drive and selecting "New" and then "Create a New Folder," you may create a new folder.



- Anyone may start a new Colab notebook when you're currently in your Google Drive. Simply pick "Colaboratory" from the drop-down menu after clicking "New."



- You may rename your notebook by right-clicking on it and taking it down, or by selecting "Rename" from the "File" menu.
- Selecting GPU in the hardware accelerator drop-down menu from the "runtime" drop-down menu, then selecting "change runtime type" from the "runtime" drop-down menu.


Notebook settings

Runtime type: Python 3

Hardware accelerator: GPU

☐ Omit code cell output when saving this notebook

3 Python Library Package Requirement

The pip command in the python environmental command prompt is used to install a list of python packages.

tensorflow-addons==0.8.3
tensorflow==2.2.0-rc3
tensorflow-gpu
kaggle
tf-nightly
ffmpeg-python
keras == 2.2.4
numpy==1.18
pandas

SpeechRecognition == 2.13
Scikit-learn
Matplotlib
Seaborn

4 Dataset Description

- Crowd-sourced Emotional Mutimodal Actors Dataset (Crema-D)
- Ryerson Audio-Visual Database of Emotional Speech and Song (Ravdess)
- Surrey Audio-Visual Expressed Emotion (Savee)
- Toronto emotional speech set (Tess)

Following steps to download and use kaggle dataset within Google Colab:

Download API Credentials

- Go to your account, Scroll to API section and Click **Expire API Token** to remove previous tokens
- Click on **Create New API Token -** It will download kaggle.json file on your machine.

Go to your Google Colab project file and run the following commands:

```
! pip install -q Kaggle
```

Choose the kaggle.json file that you downloaded

```
from google.colab import files
files.upload()
```

Make directory named kaggle and copy kaggle.json file there.

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
```

Change the permissions of the file.

```
! chmod 600 ~/.kaggle/kaggle.json
```

Download Data

```
!kaggle datasets download --force -d uwrfkaggler/ravdess-emotional-speech-audio
!kaggle datasets download --force -d ejlok1/cremad
!kaggle datasets download --force -d ejlok1/surrey-audiovisual-expressed-emotion-savee
!kaggle datasets download --force -d ejlok1/toronto-emotional-speech-set-tess
```

Use unzip command to unzip the dataset:

```
!unzip \*.zip
```

Removing Zip files:

```
rm *.zip
```

5 Data Augmentation

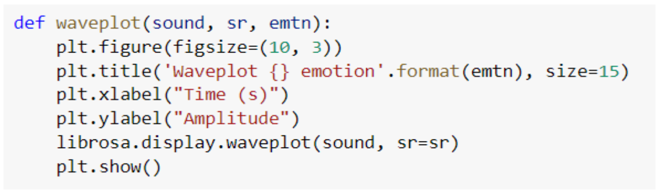
1.dataprocessing is a notebook file. The same arteacts folders will include ipynb. We have prepared the data preparation for input to the model in this file.

1. Feature extraction is the process of creating new simulated data samples from our basic training set by introducing tiny disturbances.

2. We may be using noise injection, time shifting, pitch and speed changes to provide syntactic data for audio.

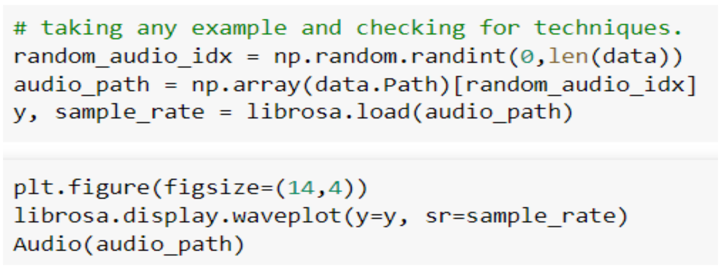
3. The goal is to make our model insensitive to those perturbations and improve its generalization capacity.

4. Applying disturbances must preserve the same label as the original training sample in order for this to operate.



```python
def waveplot(sound, sr, emtn):
    plt.figure(figsize=(10, 3))
    plt.title('Waveplot {} emotion'.format(emtn), size=15)
    plt.xlabel("Time (s)")
    plt.ylabel("Amplitude")
    librosa.display.waveplot(sound, sr=sr)
    plt.show()
```

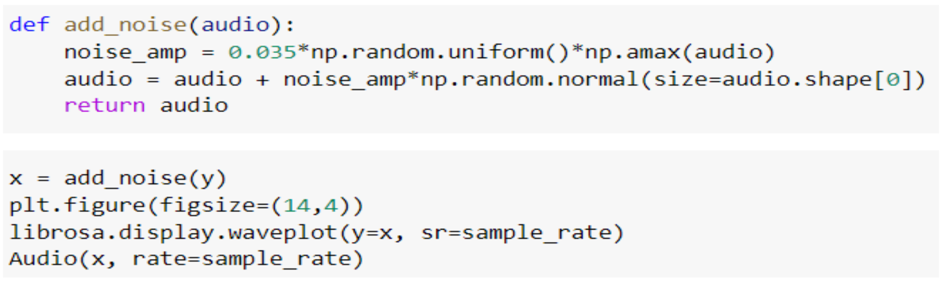
Figure 2 : Simple audio file



```python
# taking any example and checking for techniques.
random_audio_idx = np.random.randint(0,len(data))
audio_path = np.array(data.Path)[random_audio_idx]
y, sample_rate = librosa.load(audio_path)
```

```python
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=y, sr=sample_rate)
Audio(audio_path)
```

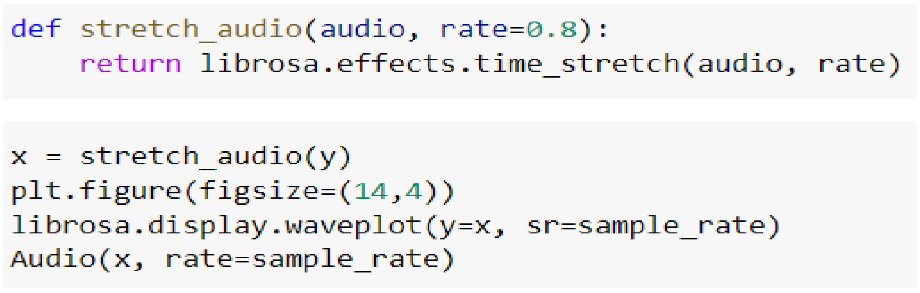
Figure 3 : Simple audio file



```python
def add_noise(audio):
    noise_amp = 0.035*np.random.uniform()*np.amax(audio)
    audio = audio + noise_amp*np.random.normal(size=audio.shape[0])
    return audio
```

```python
x = add_noise(y)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

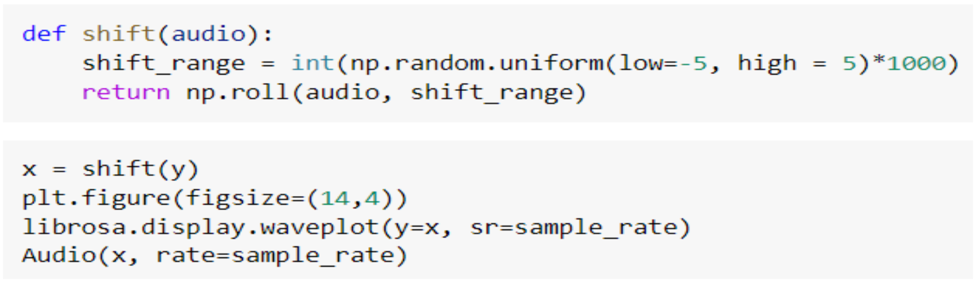
Figure 4 : noise injected audio file



```python
def stretch_audio(audio, rate=0.8):
    return librosa.effects.time_stretch(audio, rate)
```

```python
x = stretch_audio(y)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

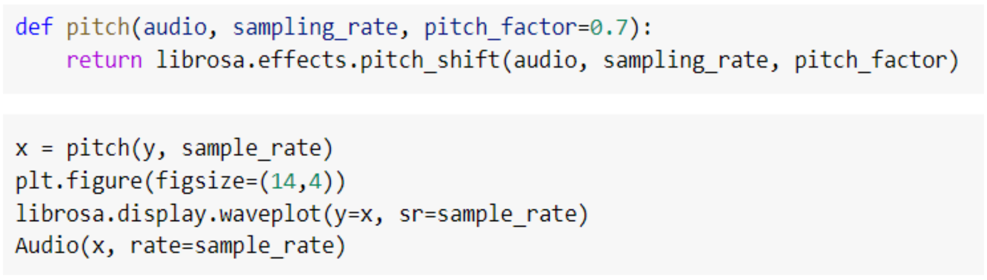
Figure 5 : stretching audio file



```python
def shift(audio):
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(audio, shift_range)
```

```python
x = shift(y)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

Figure 6 : Shifting audio file



```
def pitch(audio, sampling_rate, pitch_factor=0.7):
    return librosa.effects.pitch_shift(audio, sampling_rate, pitch_factor)
```

```
x = pitch(y, sample_rate)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

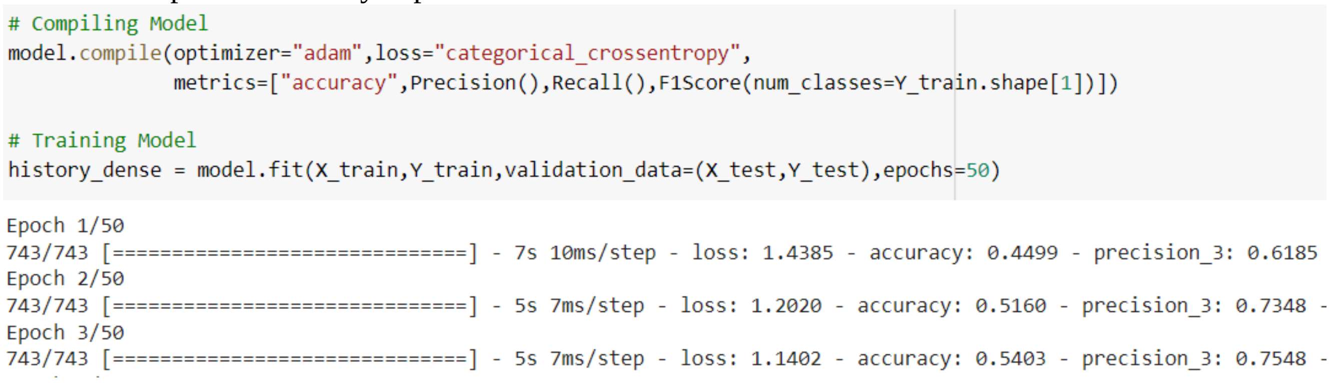
Figure 7 : Pitching audio file

6 Model Preparation

In the same artifacts directories will be the notebook files 2.NeuralNetwork-model.ipynb, 3.CNN.ipynb, and 4.LSTM.ipynb. These files use their models to train the dataset.

6.1 Deep Neural Network Model Training

- Data understating is done one more to train four audio data.
- Section 5 deals with data augmentation.
- A deep NN model has been defined and is being compiled.
- The model is trained over 50 epochs, using the optimization 'adam' and the loss value set to 'categorical-crossentropy'.
- The model is saved in the "Final_model.h5" folder.
- A 79 percent accuracy rate is reached.
- A plot of accuracy is plotted.



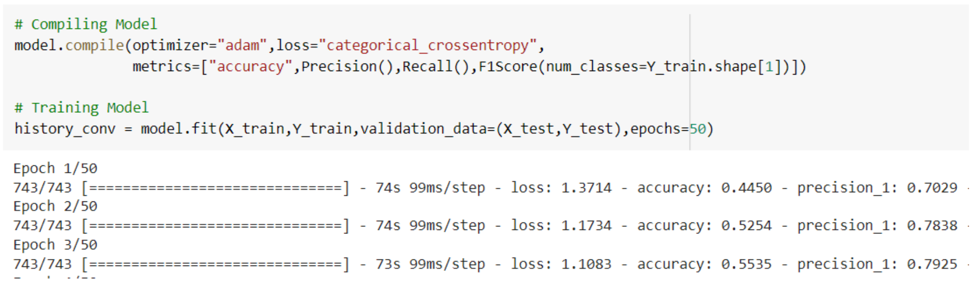
```
# Compiling Model
model.compile(optimizer="adam",loss="categorical_crossentropy",
              metrics=["accuracy",Precision(),Recall(),F1Score(num_classes=Y_train.shape[1])])

# Training Model
history_dense = model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=50)
```

```
Epoch 1/50
743/743 [==============================] - 7s 10ms/step - loss: 1.4385 - accuracy: 0.4499 - precision_3: 0.6185
Epoch 2/50
743/743 [==============================] - 5s 7ms/step - loss: 1.2020 - accuracy: 0.5160 - precision_3: 0.7348 -
Epoch 3/50
743/743 [==============================] - 5s 7ms/step - loss: 1.1402 - accuracy: 0.5403 - precision_3: 0.7548 -
```

Figure 8: Training of Deep NN Model

6.2 CNN- Model Training

- Generating a CNN layer out of four Conv1d layers.
- 1-D max pool layer with a pool size of 2 and the same padding.
- CNN is used to train and save the model CNN.h5
- Model accuracy is 90%, however with 50 epochs, val accuracy is 63 percent.
- Plotting takes place once the model has been trained.



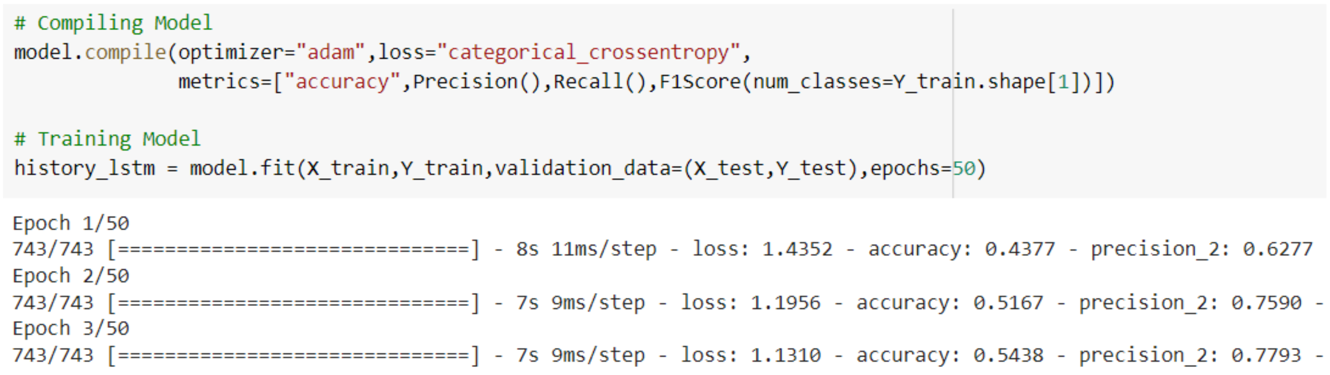
```
# Compiling Model
model.compile(optimizer="adam",loss="categorical_crossentropy",
              metrics=["accuracy",Precision(),Recall(),F1Score(num_classes=Y_train.shape[1])])

# Training Model
history_conv = model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=50)
```

```
Epoch 1/50
743/743 [==============================] - 74s 99ms/step - loss: 1.3714 - accuracy: 0.4450 - precision_1: 0.7029
Epoch 2/50
743/743 [==============================] - 74s 99ms/step - loss: 1.1734 - accuracy: 0.5254 - precision_1: 0.7838
Epoch 3/50
743/743 [==============================] - 73s 99ms/step - loss: 1.1083 - accuracy: 0.5535 - precision_1: 0.7925
```

Figure 9: Training of CNN- Model data

6.3 Long Short Term Memory (LSTM) Neural Network

- Build the model and Evaluate
- calculate accuracy ,Validation Loss, val_accuarancy and precision.
- Save the model.



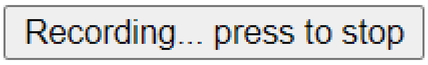
```
# Compiling Model
model.compile(optimizer="adam",loss="categorical_crossentropy",
              metrics=["accuracy",Precision(),Recall(),F1Score(num_classes=Y_train.shape[1])])

# Training Model
history_lstm = model.fit(X_train,Y_train,validation_data=(X_test,Y_test),epochs=50)
```

```
Epoch 1/50
743/743 [==============================] - 8s 11ms/step - loss: 1.4352 - accuracy: 0.4377 - precision_2: 0.6277
Epoch 2/50
743/743 [==============================] - 7s 9ms/step - loss: 1.1956 - accuracy: 0.5167 - precision_2: 0.7590 -
Epoch 3/50
743/743 [==============================] - 7s 9ms/step - loss: 1.1310 - accuracy: 0.5438 - precision_2: 0.7793 -
```

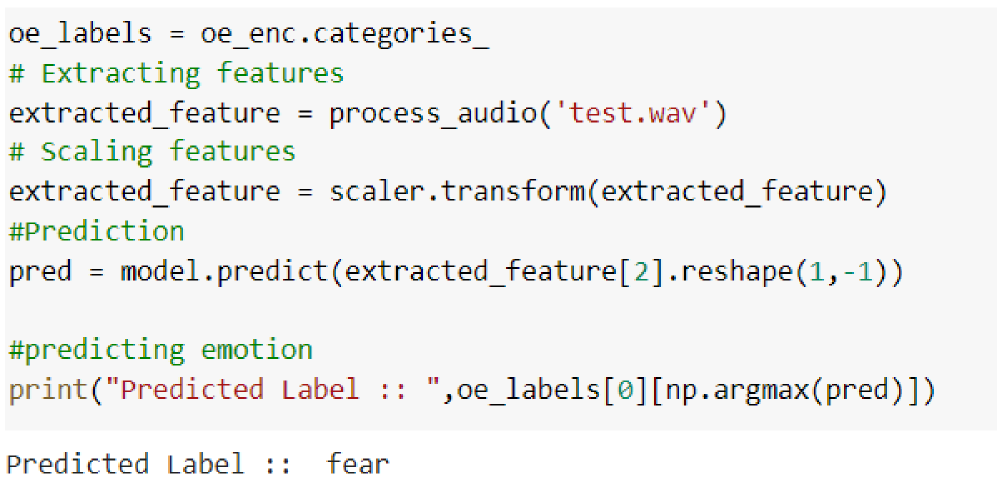
Figure 10: Training of LSTM - Model data

7 Audio Recorder Prediction

- Now to speak with recording, you have to click



- To stop recording and then by converting into wav file which is used for feature extraction and preprocessing on audio data.
- Plotting graph for recorded video and displaying prediction what user recorded.



```
oe_labels = oe_enc.categories_
# Extracting features
extracted_feature = process_audio('test.wav')
# Scaling features
extracted_feature = scaler.transform(extracted_feature)
#Prediction
pred = model.predict(extracted_feature[2].reshape(1,-1))

#predicting emotion
print("Predicted Label :: ",oe_labels[0][np.argmax(pred)])
```

```
Predicted Label ::  fear
```

Figure 11 : User recorded audio prediction