Speech Tone Prediction

INDEX

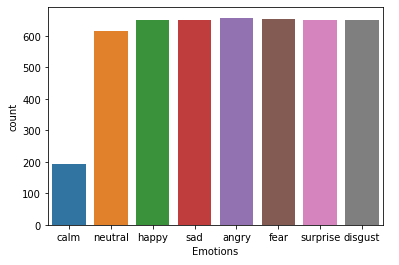
|  |  |
| --- | --- |
| Content | Pages |
| Introduction | 1 |
| EDA | 2 - 5 |
| Balancing Data | 6 - 7 |
| Feature Extraction and Preprocessing | 8 - 10 |
| Model Building and Training | 11 - 12 |
| Model Prediction pipeline and Data Ingestion | 13 |

# Introduction:-

As human beings speech is amongst the most natural way to express ourselves. emotions play a vital role in communication, the detection and analysis of the same is of vital importance in today’s digital world of remote communication. Emotion detection is a challenging task, because emotions are subjective. There is no common consensus on how to measure or categorize them. We define a SER system as a collection of methodologies that process and classify speech signals to detect emotions embedded in them.

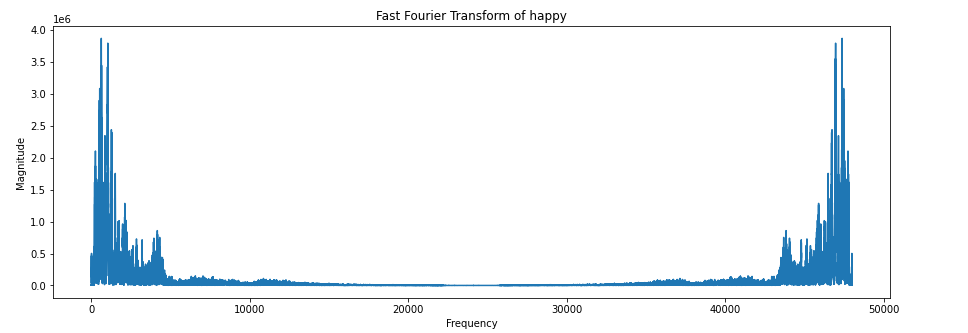
# Exploratory Data Analysis

## Distribution of data in the dataset :-

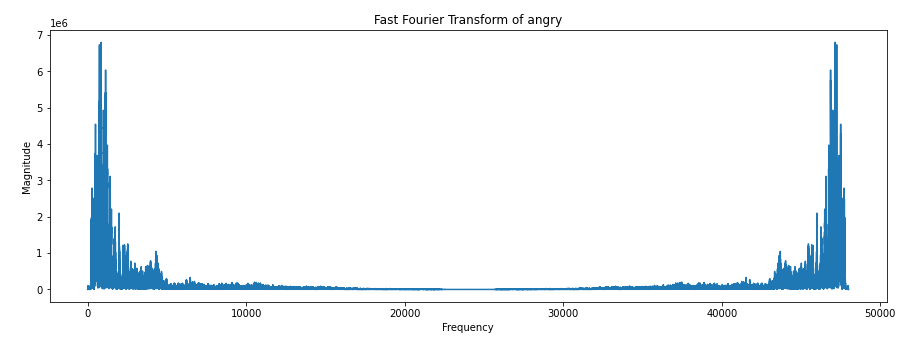


## Fast Fourier Transform to find the different magnitude of each frequency for different Emotions :-

1. **Happy**



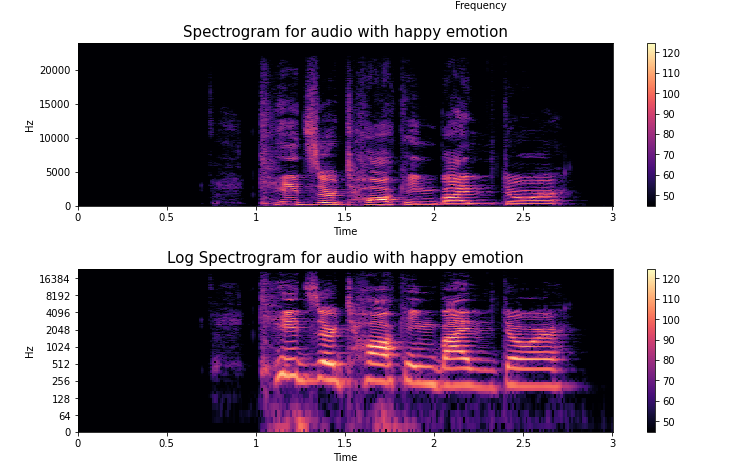
1. **Angry**

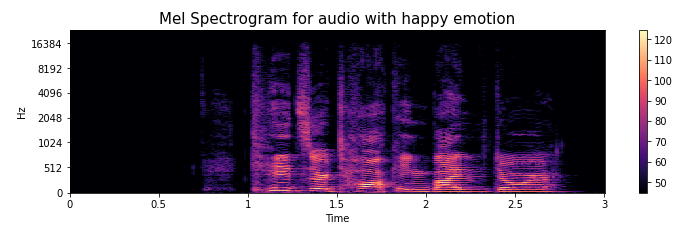


As we can see here the magnitude on each frequency changes based on emotions

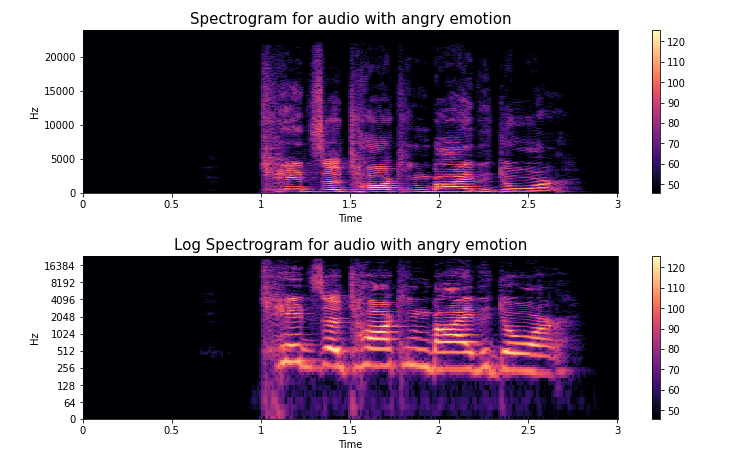
## Visualizing Data Using Spectrogram, Log-Spectrogram and Mel –Spectrogram time seperated :-

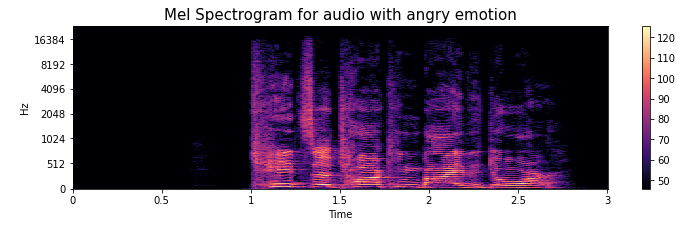
1. **Happy**



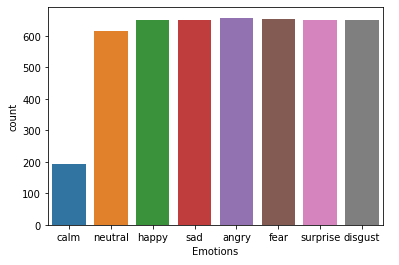


1. **Angry**





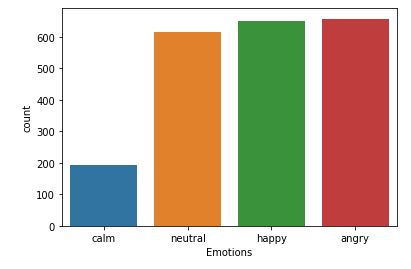
# Balancing Data



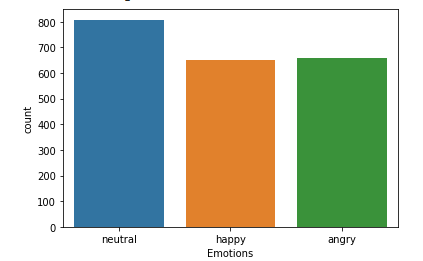
From the above graph we can see the amount of data for each emotion we have:

**Emotions that we need are:**  Happy, Angry and Neutral

Eliminating the other emotions,

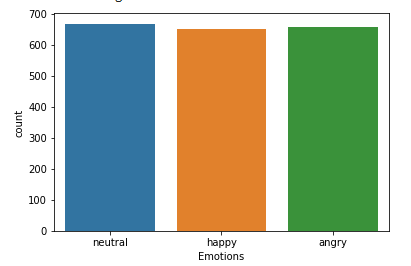


As we have a little less data in Neutral emotion we can group two emotions that were close to each other such as (Calm and Neutral)



After grouping Calm and Neutral we need to balance the data

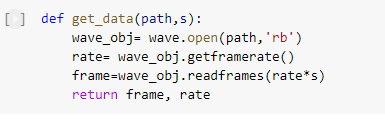
Since in audio data using SMOTE for oversampling increased a lot of noise in data, we prefer to eliminate few data from Neutral emotion to closely balance the dataset.



# Feature Extraction and Preprocessing

## Data Extraction

We can use librosa library for extraction of audio data from .wave files, but Wave library provides a little bit more flexibility while extracting audio data, as we can get the parameters such as Sample Rate, Time of the audio, Length of the full data etc.



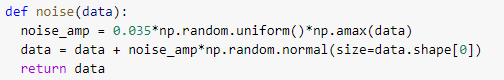
### **Preprocessing**

## Data Augmentation

There are 4 ways of data augmentation of audio data:-

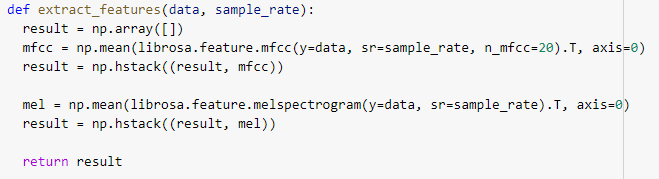
1. Noise injection
2. Shifting time
3. Changing pitch
4. Speed

We use noise injection for augmenting data:-



## Extract Features

There are various features that can be extracted form audio files but, In our case stacking the mean values of (mfcc and melspectogram) features of the audio file horizontally gives us the better accuracy



## Create The Feature Data Frame

To create the feature Data Frame, we need to: -

**Step1: -** Extract audio data

**Step2: -** Data Augmentation

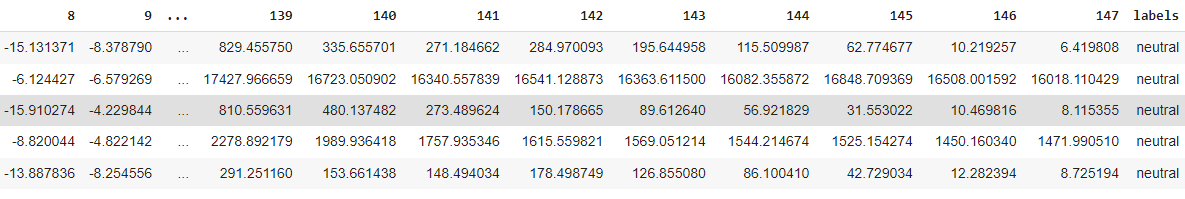
**Step3: -** Feature Extraction

**Step4: -** Stack Extracted features vertically

**Step5: -** Store extracted features and emotions on two list, i.e X and Y

**Step6: -** Create a Data frame of list X and add a feature named ‘label’ from list Y

**Data Frame will look like: -**



## Split Data

Splitting the data into (Train and Test)

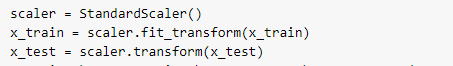
Train – 75%

Test – 25%

Random State - 5

## Data Normalization

Before feeding the data into the model we need to perform Data Normalization using Standard Scalar-



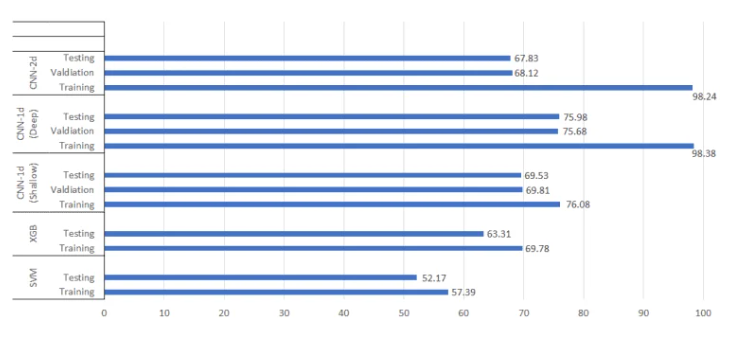
## Dimensionality Expansion

Since we will be using a deep learning CNN-1D model we need to expanding one dimension to the data.



# Model Building and Training

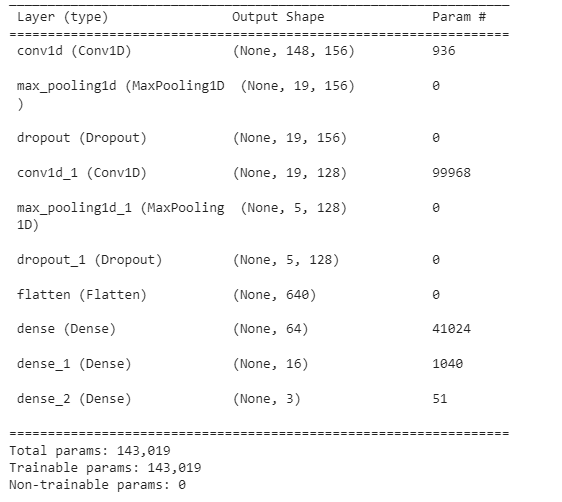
## Choosing Model



From the above graph acquired from web, we can identify the most valid model with a better accuracy is provided by the CNN-1D (Deep).

Model we choose to train is CNN followed by Deep Neural network.

## Model Summary



This CNN model had the following architectural complexity:

* 1 convolution layers of 156 channels, 5×5 kernel size, 1x1 stride and same padding followed by a max-pooling layer of size 8×8 and same padding followed by Dropout of 10%.
* 1 convolution layers of 128 channels, 5×5 kernel size and same padding followed by a max-pooling layer of size 4×4 and same padding followed by Dropout of 10%.
* Each convolution layer had the ‘relu’ activation function.
* After flattening, two dense layers of 64 units and 16 units were added where activation function is ‘relu’.
* Finally, the output layer was added with a ‘softmax’ activation function.

Compiling the model, with ‘adam optimizer’ and loss function as ‘categorical crossentropy’ and accuracy metrics.

## Model Training

Model is trained on batch size 48, epochs 40, callback API ‘ReduceLROnPlateau’

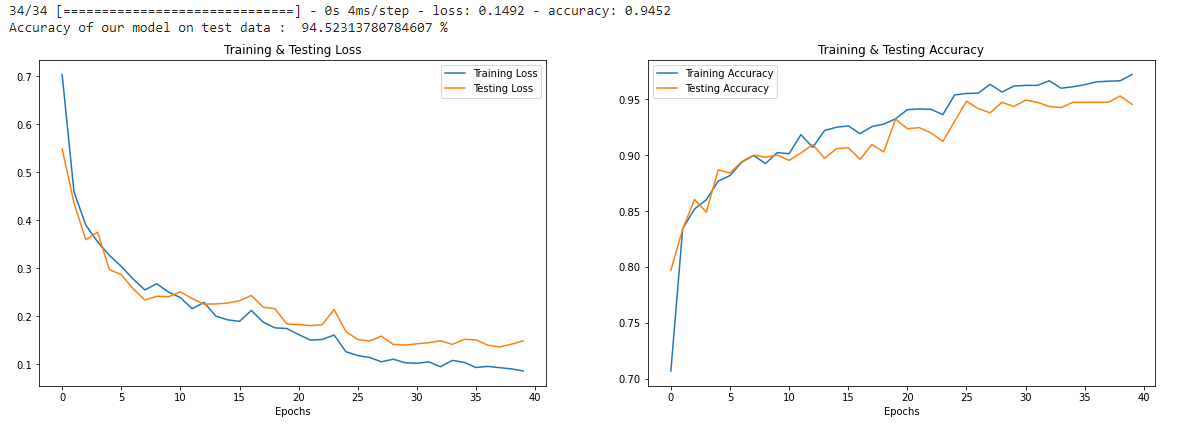
Callback API- Monitors ‘loss’, Learning Rate is reduced by factor 0.2 when once the learning stagnates, patience is 2 and the minimum learning rate is set to 3e-4.

**ReduceLROnPlateau -**

Reduce learning rate when a metric has stopped improving.

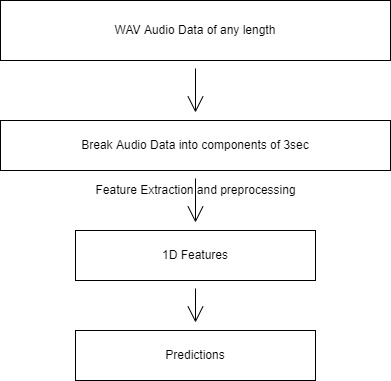
Models often benefit from reducing the learning rate by a factor of 2-10 once learning stagnates. This callback monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

## Model learning curve



# Model Prediction Pipeline

**From Audio File**



**From Base64 Data in Realtime**

