FİNAL PROJECT

COM421_ IMAGE PROCESSING

PROJECT TITLE: SPEECH EMOTION RECOGNITION

ADVİSOR: DR.ÖGR.ÜYESİ PERİ GÜNEŞ

GROUP MEMBERS:

NURŞAH DEMİRPOLAT – B1605.010037

ZEYNEP GİZEM ÇETİNCİ – B1605.010034

TABLE OF CONTENTS

Introductıon

There are many ways people communicate with each other. However, the most important way of communication is talking. Talking is the fastest and most sincere communication between people. People can express their feelings more easily by talking. They can detect their feelings among each other. However, emotion detection is a bit difficult for the machine. For this reason, the purpose of our sound and emotion analysis project is to use the information about the emotion of the sound recording that we will analyze by improving the human machine communication.

Emotions from the speech of human speakers are found in emotion recognition in our project. These emotions are collected in h5 format. Using these data, the emotion in the sound is reflected on the screen visually.

1.1 Litrature review

Every living thing has its own unique voice, so communication with other creatures is established. Human voice is the most important factor for communication between people, as well as each individual's fingerprint is unique, and voice signals are unique. Emotional states of people can be predicted from sound signals during their speech. In this project assignment we do, we aim to analyze emotion from human voice.[1]

The deep learning paradigm has emerged in the past years, the feature of the deep learning paradigm is that it is an end-to-end speaking model that learns from raw speech signals. On the other hand, local feature learning blocks are also used to reduce the dimensionality of a signal and reveal the information that is essential, thus providing better accuracy.[2]

Speaking is the fastest and most understandable method of communication between people. This case software developers have explored what kinds of efficient situations can be achieved in human and machine interaction. First, they sought to do sound-emotion analysis with machine learning. Emotion consists of angry, neutral, happiness and sad emotion classes. And, according to research, 393 data were collected from these classifications. These data tests are kept and used in the EmoDB dataset set. At the same time, verification is performed between different data sets using the EmoDB data set.[3]

Analyzing emotions with sound generally provides many advantages in situations where there are visual obstacles. For example, because we cannot see the face of the other person while making a phone call, it is very useful to analyze emotions with voice. The DVM method is used in emotion analysis with machine learning. DVM stands for Support Vector Machines. The way sound analysis works, analyzes are made using Support Vector Machines over 20 sound attributes that will show that the sound belongs to the human. And by comparing these analyzes with the emotions in the database, the closest feeling is determined.[4]

To give an example of the paragraph written above, the pictures in figure 1 and figure 2 show the vocal event of an angry person.[5]

Figure 1

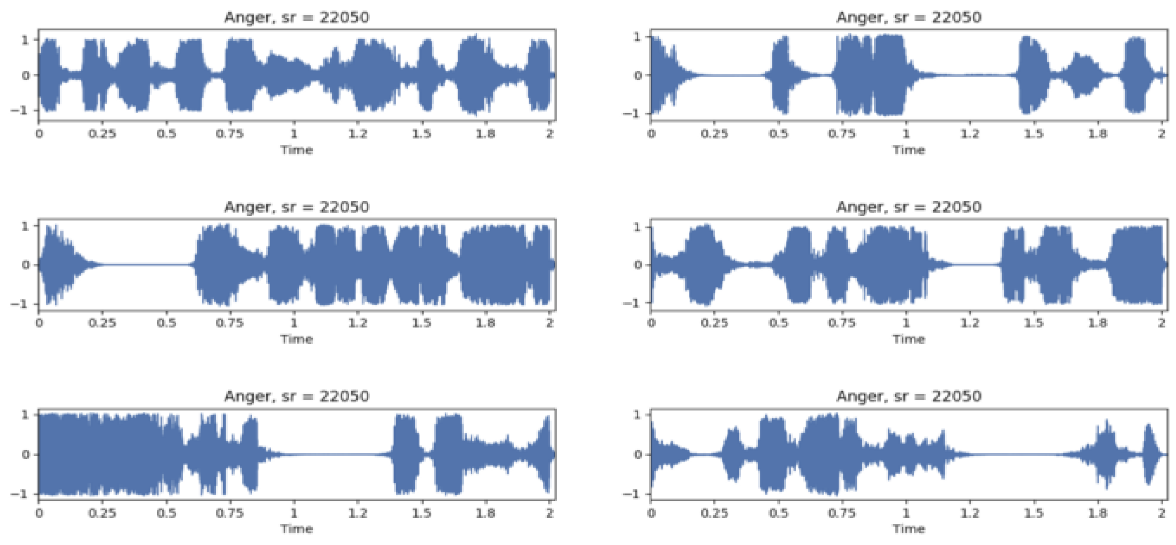


Image (by Author): Visualisation for raw audio time series for various emotions (Natural audios)[5]

Figure 2

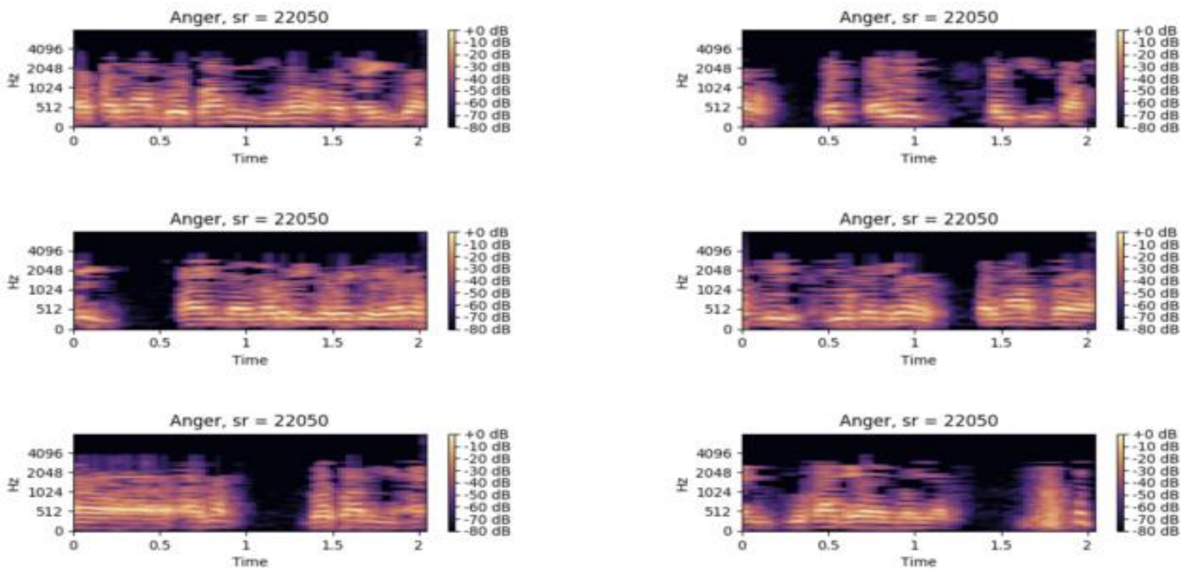


Image (by Author): Visualisation for Mel Spectrogram (n_mels = 128) for various emotions (Natural audios) [5]

The emotion recognition system without speech is actually similar to the pattern recognition system. The steps taken in these two systems are very similar. The emotion recognition system from speech consists of 5 main items, which is shown in figure 3.[6]

Figure 3

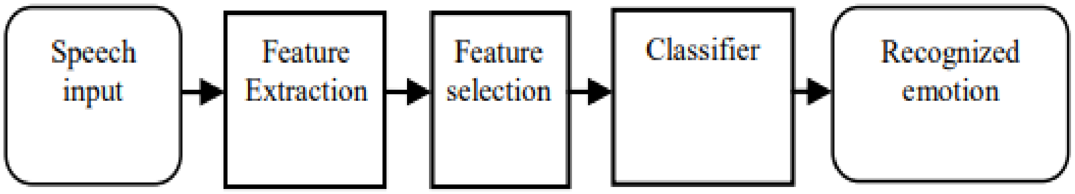
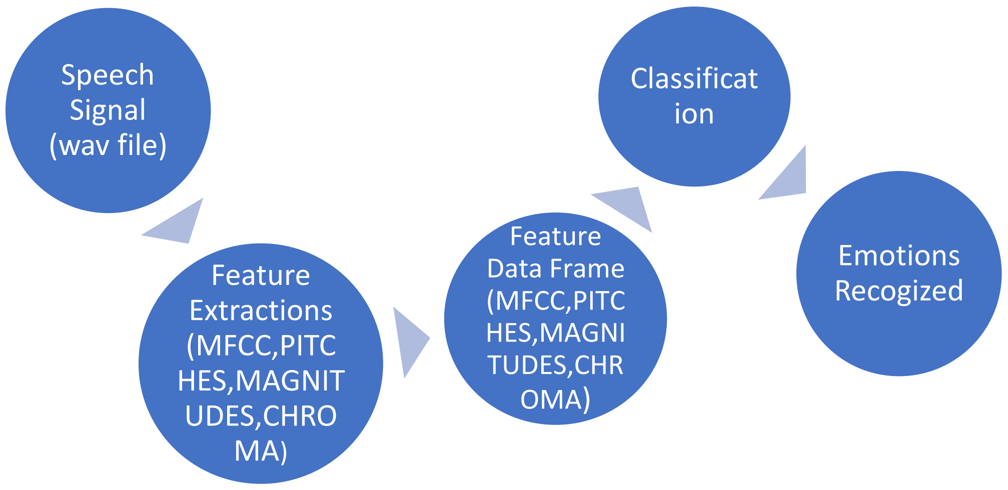


Figure 1. Structure of the Speech Emotion Recognition System.[6]

It is the most important thing to find the main emotion while recognizing the emotion in the emotion recognition system without speech. Emotion recognition sequences contain 300 emotional states, so they are a somewhat complex structure. According to "Palette Theory", there are so many colors in our lives, but all of them have emerged with the blue-red-yellow mixture of the primary colors. When we consider this theory according to the emotion recognition system without speaking, we can actually say that our main emotions are anger, fear, sadness, surprise, disgust, and fear. [6]

2. System Design

2.1 Design Method and Standards

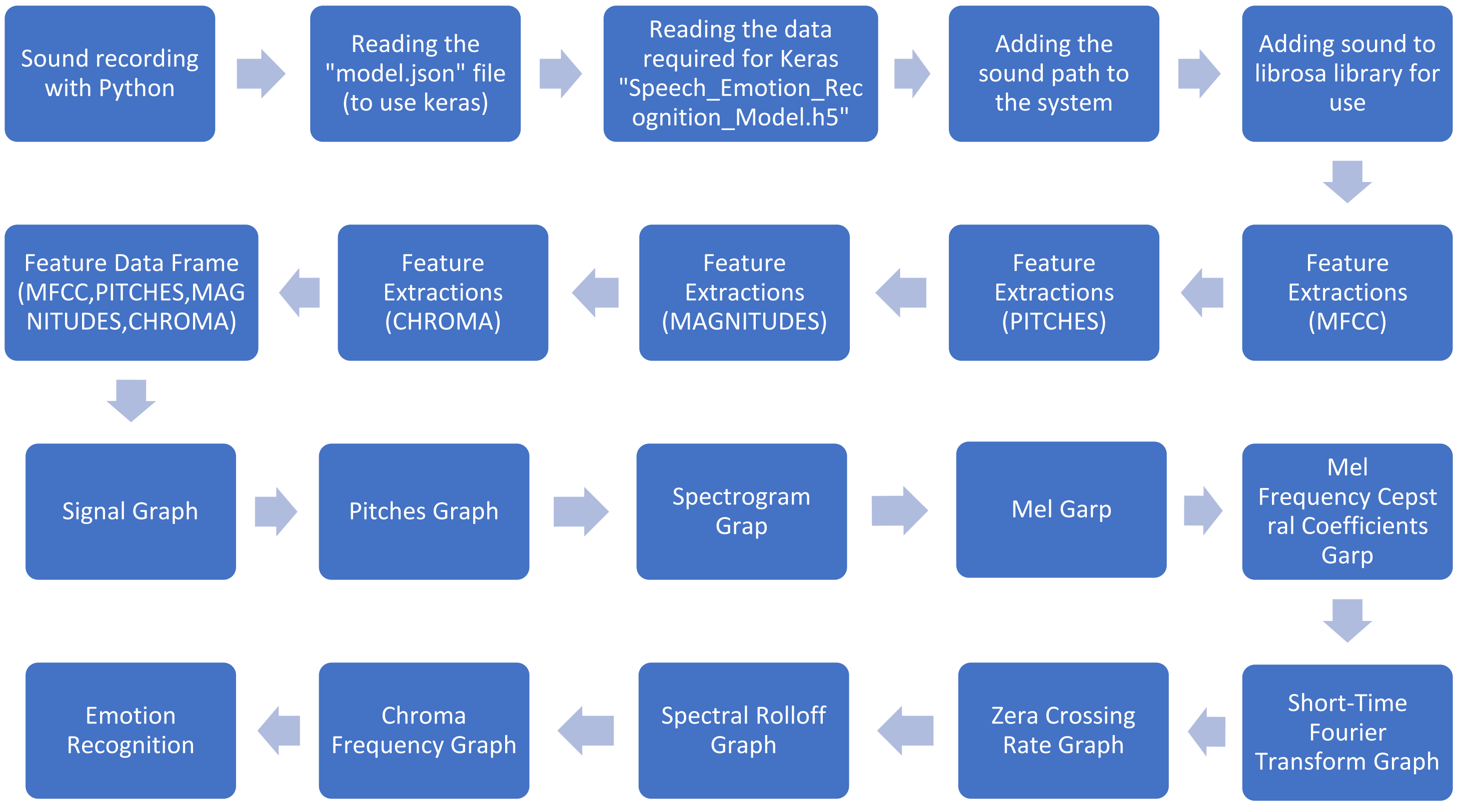

Speech Signal (wav file)
Feature Extractions (MFCC,PITCHES,MAGNITUDES,CHROMA)
Feature Data Frame (MFCC,PITCHES,MAGNITUDES,CHROMA)
Classification
Emotions Recogized

2.2 Software development tools

We used Colab and pyCharm platforms to make our project. The reason we use these platforms is because we used our project in the python software language. The reason we use Colab is that we have free access to GPUs. Since we made our project with keras, it allows us to use it easily. At the same time, we used Co Kabin since we wrote our main code for sound analysis on IPYNB. An IPYNB file allows it to work with the Python language and data. The PyCharm platform is already used for python. That's why we wrote the audio recording code here.

3. Results

3.1 Syntax of Algorithm

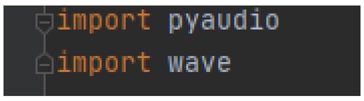

Sound recording with Python
Reading the "model.json" file (to use keras)
Reading the data required for Keras "Speech_Emotion_Recognition_Model.h5"
Adding the sound path to the system
Adding sound to librosa library for use
Feature Extractions (MFCC)
Feature Extractions (PITCHES)
Feature Extractions (MAGNITUDES)
Feature Extractions (CHROMA)
Feature Data Frame (MFCC,PITCHES,MAGNITUDES,CHROMA)
Signal Graph
Pitches Graph
Spectrogram Grap
Mel Garp
Mel Frequency Cepstral Coefficients Garp
Short-Time Fourier Transform Graph
Zera Crossing Rate Graph
Spectral Rolloff Graph
Chroma Frequency Graph
Emotion Recognition

3.2 Output Explanation

Record Sound

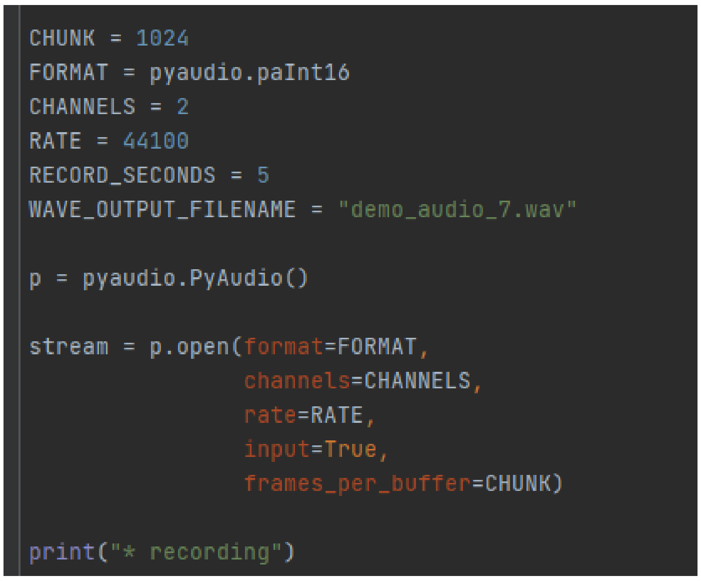
In order to do sound analysis, we must first have a sound recording at hand. For this reason, we did the audio recording process first. We used the pyaudio package to make audio input and audio recording output from the PyCharm software platform. PyAudio is used to play and record sound in python.

Wave module is a library of python. It is an interface that allows us to record the sound recording we will create in the "wav" extension. The Wave module can write audio data to a raw format file. And then it can read it as a wav file.



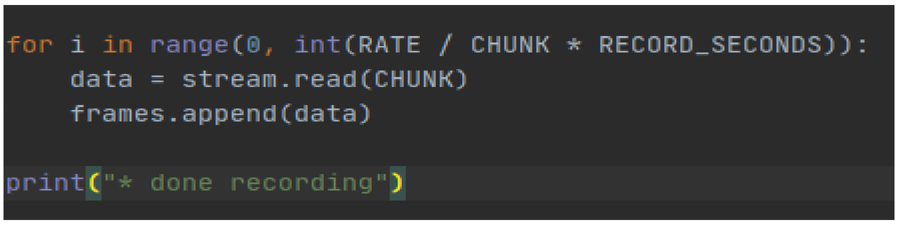
```
import pyaudio
import wave
```

In order to use the PyAudio module, we first write this code (variable p). In this way, we have embodied the module. We also added the open () command to record audio. We created a stream in our file with the sound parameters we set. We named the sound recording with the Filename variable. We set the audio recording time to 5 seconds. Depending on the situation, this period can be increased or decreased.



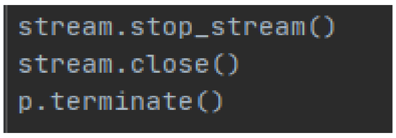
```
CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 2
RATE = 44100
RECORD_SECONDS = 5
WAVE_OUTPUT_FILENAME = "demo_audio_7.wav"

p = pyaudio.PyAudio()

stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK)

print("* recording")
```

By giving the read () command to the variable named steam, which I have assigned by ourselves, we have enabled to read the audio data in the stream. In addition, it enables us to create audio data very quickly.



```
for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data)

print("* done recording")
```

We stopped recording audio with the stop () command and ended the stream with the close () command. Finally, we ended the portaudio session.



```
stream.stop_stream()
stream.close()
p.terminate()
```

This function opens a file for reading / writing audio data. The function needs two parameters - filename first and mode second. The mod can be 'wb' for writing audio data or 'rb' for reading.

The function of wave.open() code opens a file for us to read or write audio data. In this way, it creates a memory for us to record sound. To print the sound data with the name we give to the sound recording that we want to do in our code. We do this printing with 'wb'.

```
wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

Packages We Use In Our Code

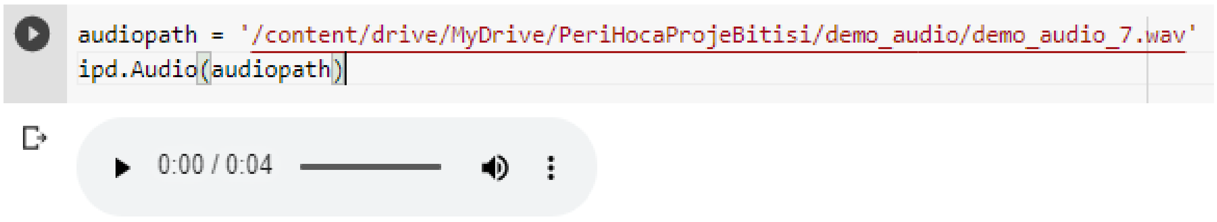
Librosa is a Python package for music and sound analysis. We used this package because we wanted to do sound analysis in our project. In order to use this package, we first download it as follows.

```
!pip install librosa
```

We use the IPython library to play the audio recording created with our code. Thanks to this package, an interface comes and we can listen to the recorded sound.

Thanks to the imports we have written, the interface that allows us to play the sound recording that we will analyze the sound is coming. Thus, we can listen to the audio recording before analyzing.

```
import librosa
import librosa.display
import IPython.display as ipd
```



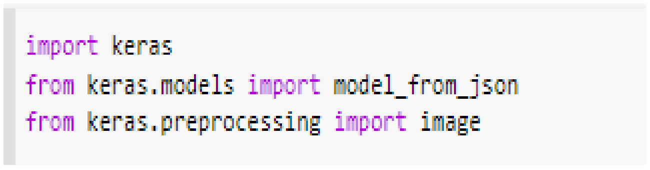
```
audiopath = '/content/drive/MyDrive/PeriHocaProjeBitisi/demo_audio/demo_audio_7.wav'
ipd.Audio(audiopath)
```

We use the code below to connect Google Drive to Google Colab. Thanks to Google Colab, we made it possible to save data sets in our project to Google Drive. At the same time, we store the voice recordings and some pictures we have created for our project on Google Driver. For this reason, with this code, we can access our information in Google Driver and use it in our code.

```
[ ] # Load the Drive helper and mount
    from google.colab import drive
    # This will prompt for authorization.
    drive.mount('/content/drive/')
```

We are doing our project with keras deep learning. Because we need to have multidimensional data in order to make sound analysis. As an example, we created a sound recording in an angry way. When analyzing this, he must first understand whether this voice is male, female or child. Because the pitch may differ as thinner or thicker. Likewise, not everyone's angry voices are the same. Some people may speak angrily by shouting, others by talking fast, etc. More options can be added to these examples. Thus, we have a great deal of data in our hands.

We saved and read the Keras model in JSON format. Thanks to this model, we recorded the big data for sound analysis in HDF5 format. Here, model json is of great help to us. Because thanks to this model, we can easily read the data in h5.



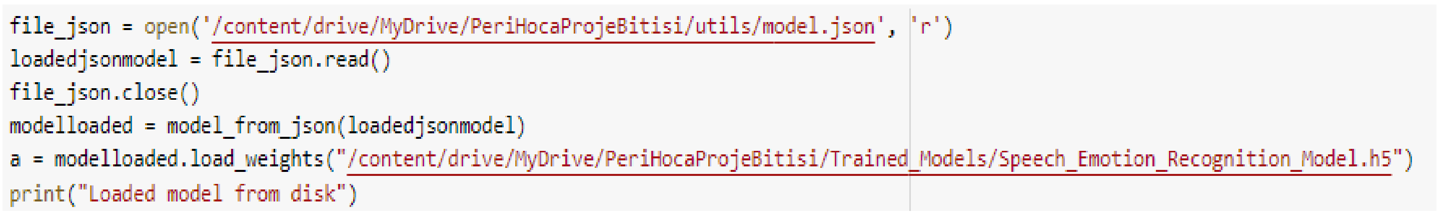
```
import keras
from keras.models import model_from_json
from keras.preprocessing import image
```

Our data set contains 2411 wav extension files. These files consist of 7 sections as anger, disgust, fear, happy, neutral, sad, surprise and these sections contain these wav files.

- anger: 400 samples
- disgust: 325 samples
- fear: 394 samples
- happy: 390 samples
- neutral: 308 samples
- sad: 326 samples
- surprise: 268 samples

Total: 2411 Samples

This creates a classification thanks to the data set we have made, so that we can compare our audio signals with these files and reach a more accurate result.



```
file_json = open('/content/drive/MyDrive/PeriHocaProjeBitisi/utils/model.json', 'r')
loadedjsonmodel = file_json.read()
file_json.close()
modelloaded = model_from_json(loadedjsonmodel)
a = modelloaded.load_weights("/content/drive/MyDrive/PeriHocaProjeBitisi/Trained_Models/Speech_Emotion_Recognition_Model.h5")
print("Loaded model from disk")
```

Pandas is a Python package that provides fast, flexible and impressive data structures designed to make working with "relational" and "tagged" data easy and intuitive.

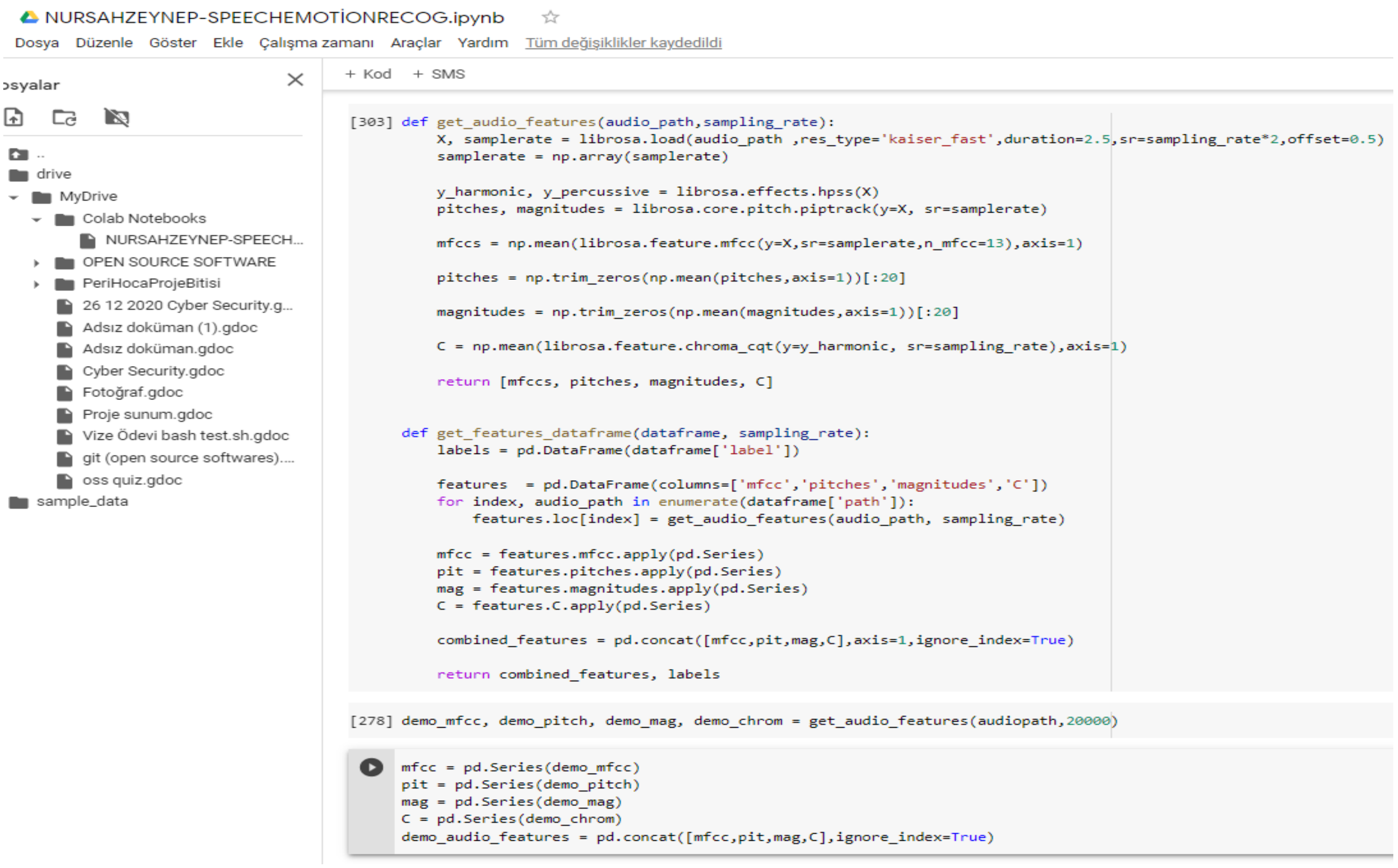
The NumPy package is a math python library that enables fast scientific calculations.

Matplotlib is a pyhton library that is used to visualize the data from which we get sound for sound analysis. Visual data allows us to make 2 or 3 dimensional drawings. This way we can download the matplotlib library;

```
pip install matplotlib
```

Feature Extractions & Feature Data Frame

The section here includes mfcc, pitches, magnitudes, chrome settings. Thanks to this section, we take the values in the sound signals and analyze the emotions.



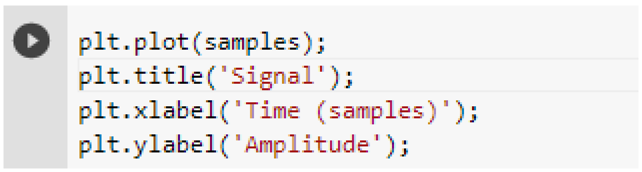
```
[303] def get_audio_features(audio_path,sampling_rate):
          X, samplerate = librosa.load(audio_path ,res_type='kaiser_fast',duration=2.5,sr=sampling_rate*2,offset=0.5)
          samplerate = np.array(samplerate)

          y_harmonic, y_percussive = librosa.effects.hpss(X)
          pitches, magnitudes = librosa.core.pitch.piptrack(y=X, sr=samplerate)

          mfccs = np.mean(librosa.feature.mfcc(y=X,sr=samplerate,n_mfcc=13),axis=1)

          pitches = np.trim_zeros(np.mean(pitches,axis=1))[:20]

          magnitudes = np.trim_zeros(np.mean(magnitudes,axis=1))[:20]

          C = np.mean(librosa.feature.chroma_cqt(y=y_harmonic, sr=sampling_rate),axis=1)

          return [mfccs, pitches, magnitudes, C]


      def get_features_dataframe(dataframe, sampling_rate):
          labels = pd.DataFrame(dataframe['label'])

          features  = pd.DataFrame(columns=['mfcc','pitches','magnitudes','C'])
          for index, audio_path in enumerate(dataframe['path']):
              features.loc[index] = get_audio_features(audio_path, sampling_rate)

          mfcc = features.mfcc.apply(pd.Series)
          pit = features.pitches.apply(pd.Series)
          mag = features.magnitudes.apply(pd.Series)
          C = features.C.apply(pd.Series)

          combined_features = pd.concat([mfcc,pit,mag,C],axis=1,ignore_index=True)

          return combined_features, labels
```

```
[278] demo_mfcc, demo_pitch, demo_mag, demo_chrom = get_audio_features(audiopath,20000)
```

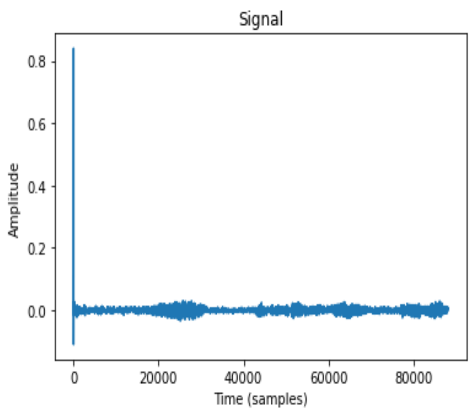
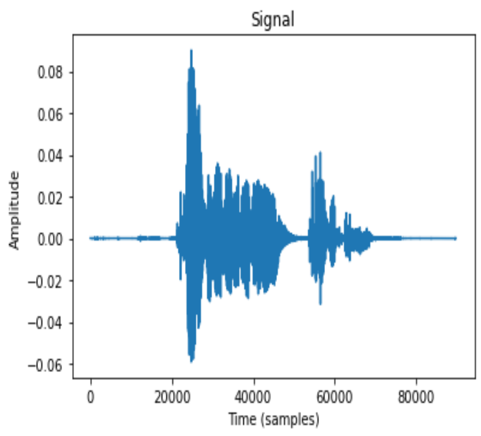
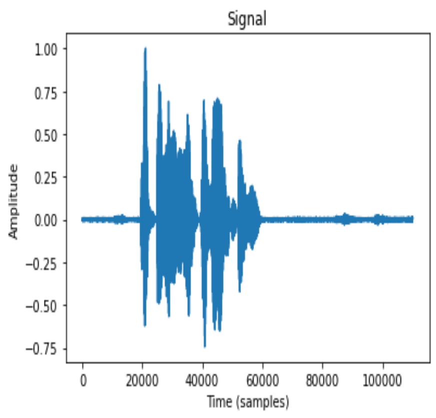
```
mfcc = pd.Series(demo_mfcc)
pit = pd.Series(demo_pitch)
mag = pd.Series(demo_mag)
C = pd.Series(demo_chrom)
demo_audio_features = pd.concat([mfcc,pit,mag,C],ignore_index=True)
```

Signal

It is the visual signal image of the received audio file. Below are the code and sample graphics we use.

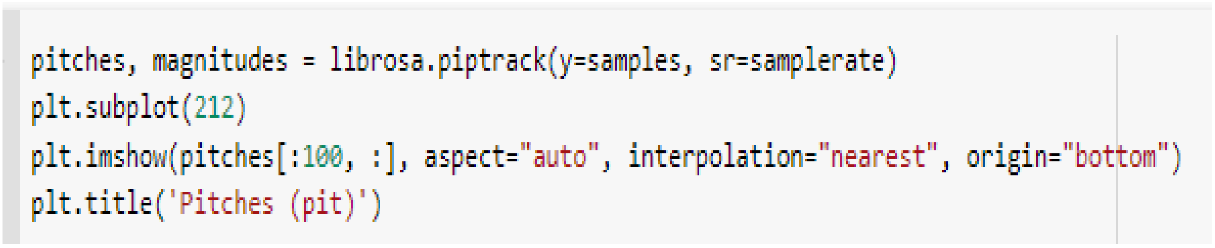


```
plt.plot(samples);
plt.title('Signal');
plt.xlabel('Time (samples)');
plt.ylabel('Amplitude');
```

 demo_audio_3.wav Emotions: disgust	 demo_audio_1.wav Emotions: sad	 demo_audio_7.wav Emotions: anger

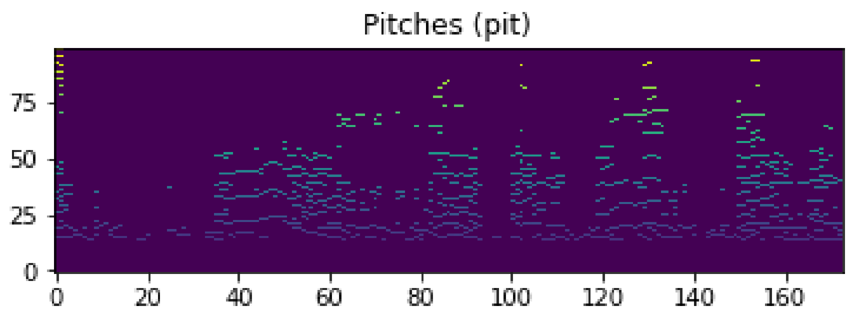
Pitches

Pitch is the cornerstone of the speech signal. It is the perception relationship of fundamental frequency. It represents the frequency during sound production that causes the vocal cords to vibrate.

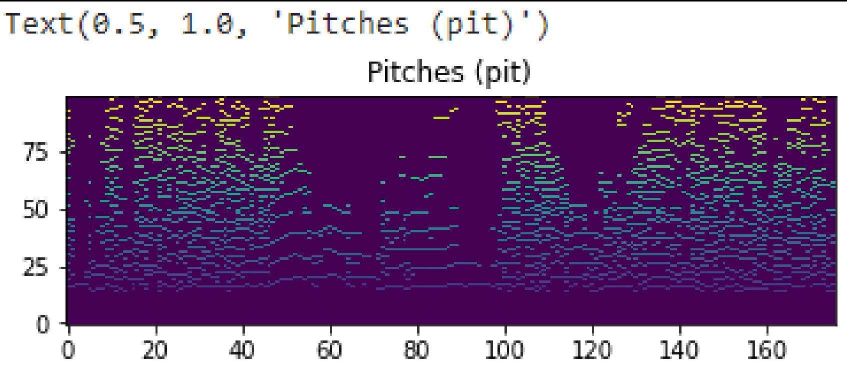
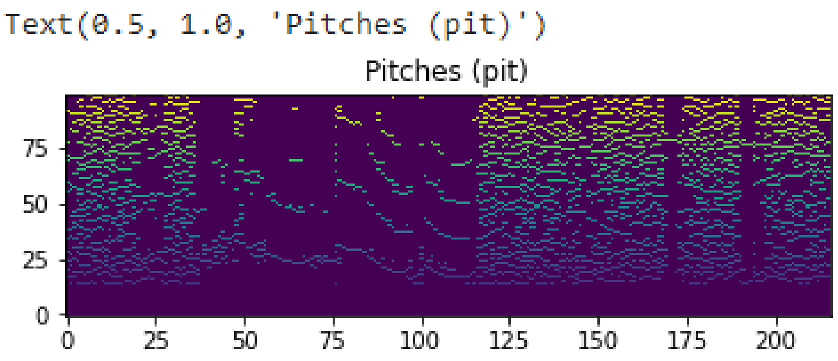


```
pitches, magnitudes = librosa.piptrack(y=samples, sr=samplerate)
plt.subplot(212)
plt.imshow(pitches[:100, :], aspect="auto", interpolation="nearest", origin="bottom")
plt.title('Pitches (pit)')
```

```
Text(0.5, 1.0, 'Pitches (pit)')
```

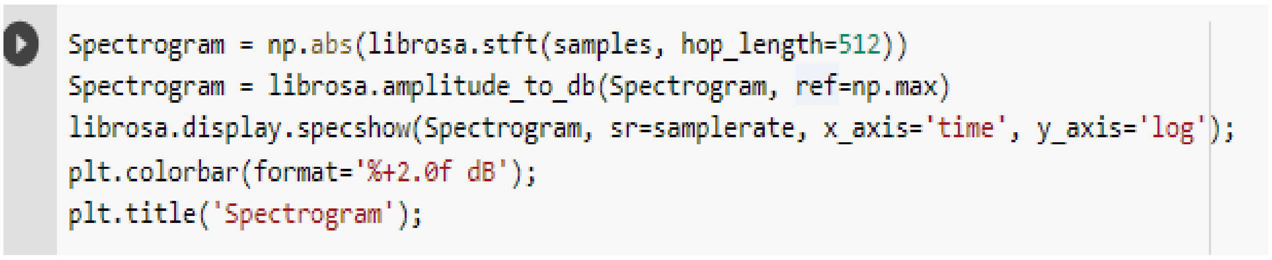


demo_audio_3.wav
Emotions: disgust

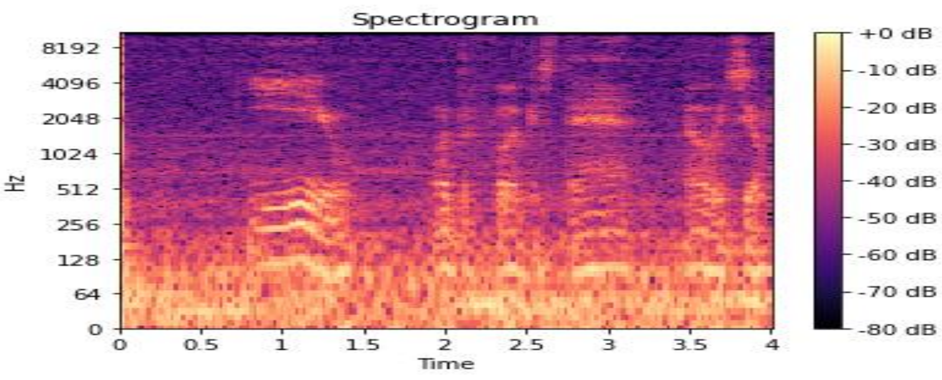
 `Text(0.5, 1.0, 'Pitches (pit)')`   demo_audio_1.wav Emotions: sad
 `Text(0.5, 1.0, 'Pitches (pit)')`   demo_audio_7.wav Emotions: anger

Spectrogram

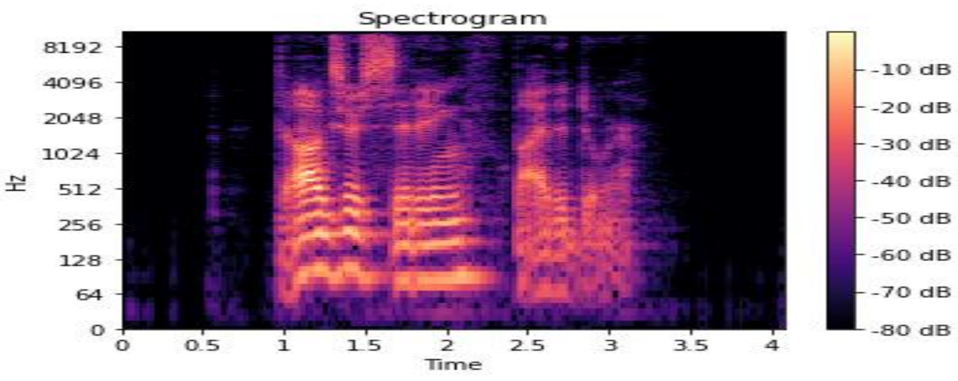
It is a method of visualizing sound. Actually, visualizing sound here means visualizing airwaves, so what we do is visualizing waves in the air. We also realized this visualization in our project. As can be seen from the picture here, under the titles of "time" and "Hz":



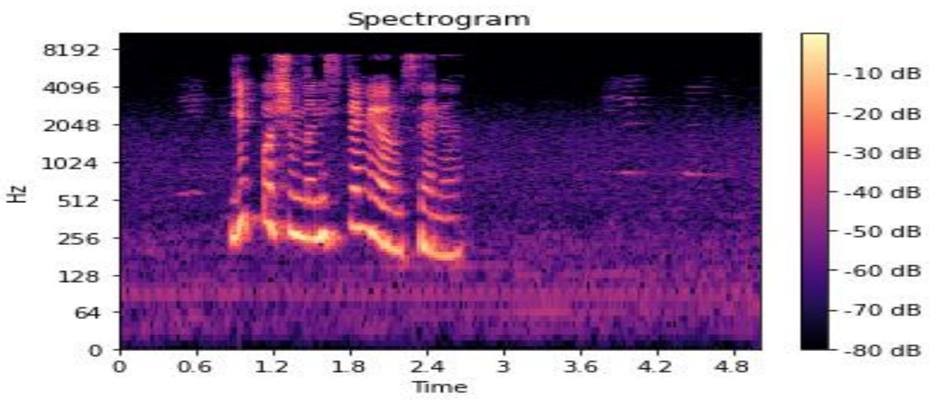
```
Spectrogram = np.abs(librosa.stft(samples, hop_length=512))
Spectrogram = librosa.amplitude_to_db(Spectrogram, ref=np.max)
librosa.display.specshow(Spectrogram, sr=samplerate, x_axis='time', y_axis='log');
plt.colorbar(format='%+2.0f dB');
plt.title('Spectrogram');
```



demo_audio_3.wav
Emotions: disgust



demo_audio_1.wav
Emotions: sad



demo_audio_7.wav
Emotions: anger

Mel Frequency Scale

The Mel frequency scale is a type of scale that shows the perception of a change in sound frequency (below 1HZ) that the human ear can hear. Today, the formula is like this; **(m=mels,f=hertz)**

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

However, although it does not contain only this formula, it is expressed with different algorithmic figures;

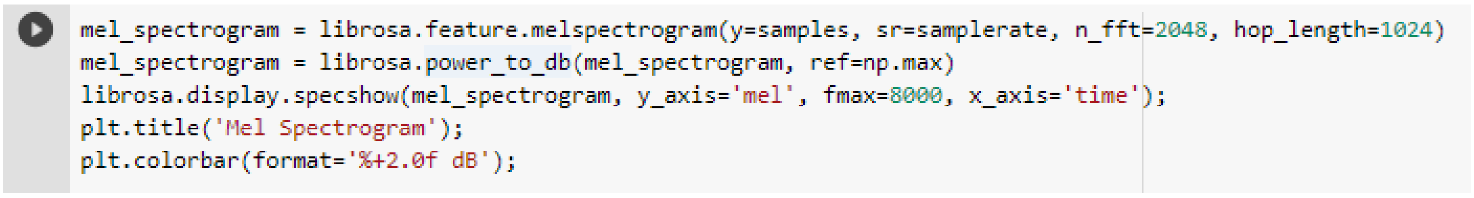
$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \ln\left(1 + \frac{f}{700}\right)$$

Likewise, this expression can be written as follows;

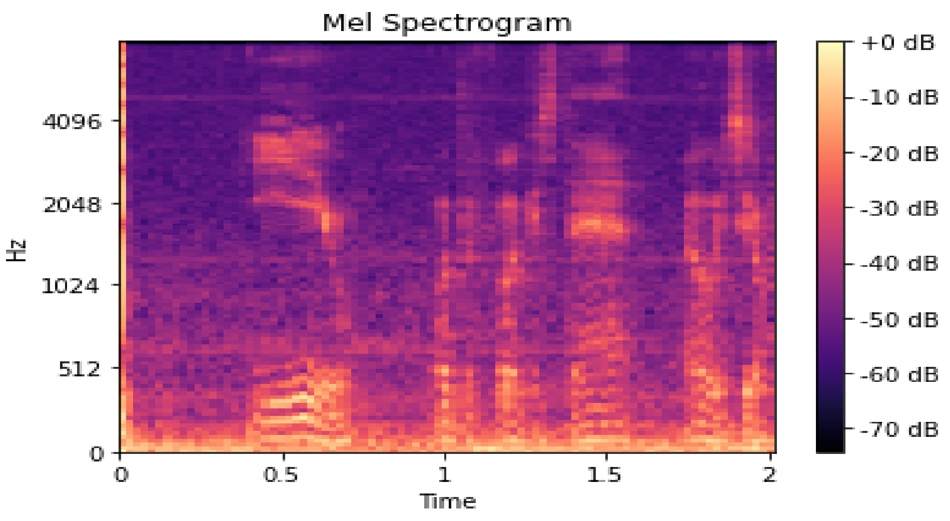
$$f = 700\left(10^{\frac{m}{2595}} - 1\right) = 700\left(e^{\frac{m}{1127}} - 1\right)$$

Mel Spectrogram

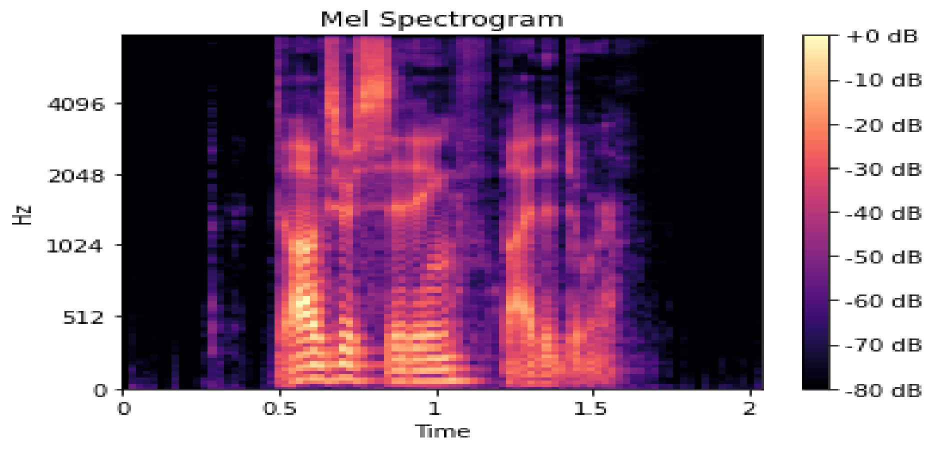
In fact, it is a spectrogram similar to the spectrom, with the y-axis mel scale.



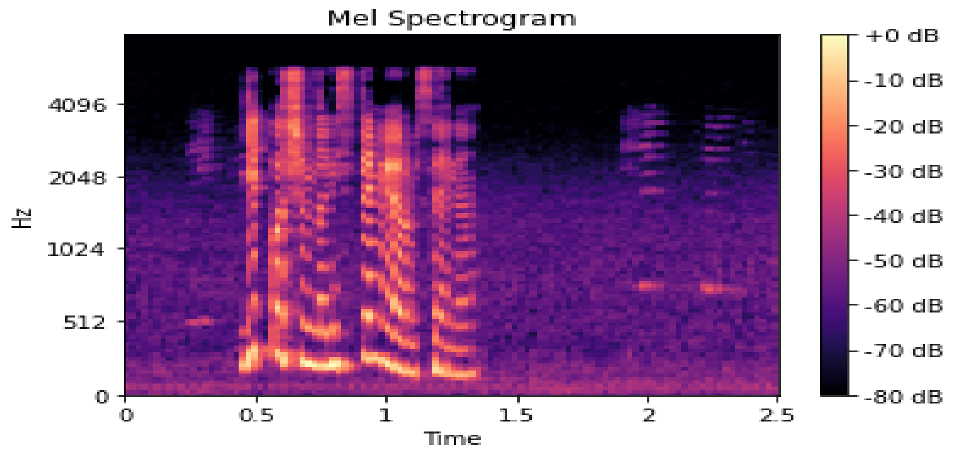
```
mel_spectrogram = librosa.feature.melspectrogram(y=samples, sr=samplerate, n_fft=2048, hop_length=1024)
mel_spectrogram = librosa.power_to_db(mel_spectrogram, ref=np.max)
librosa.display.specshow(mel_spectrogram, y_axis='mel', fmax=8000, x_axis='time');
plt.title('Mel Spectrogram');
plt.colorbar(format='%+2.0f dB');
```



demo_audio_3.wav
Emotions: disgust



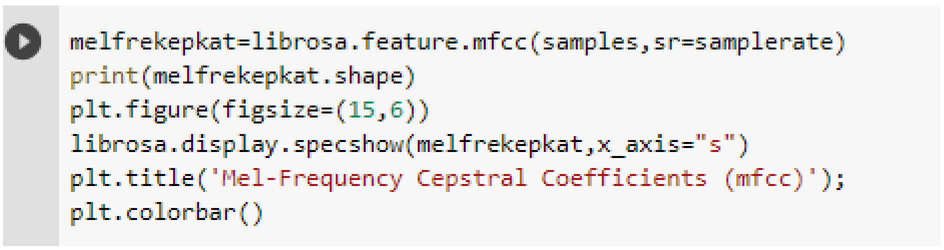
demo_audio_1.wav
Emotions: sad



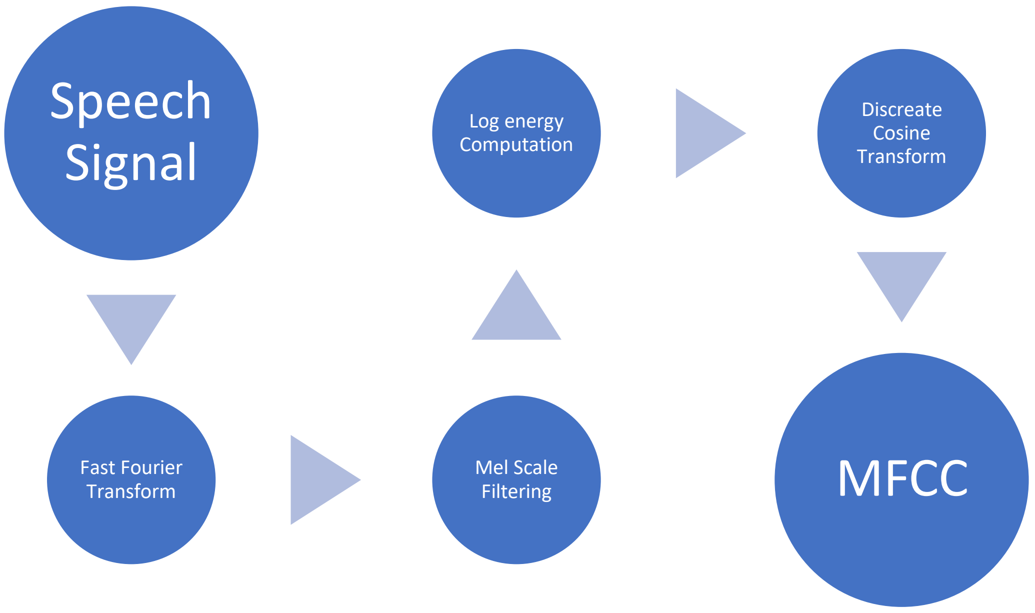
demo_audio_7.wav
Emotions: anger

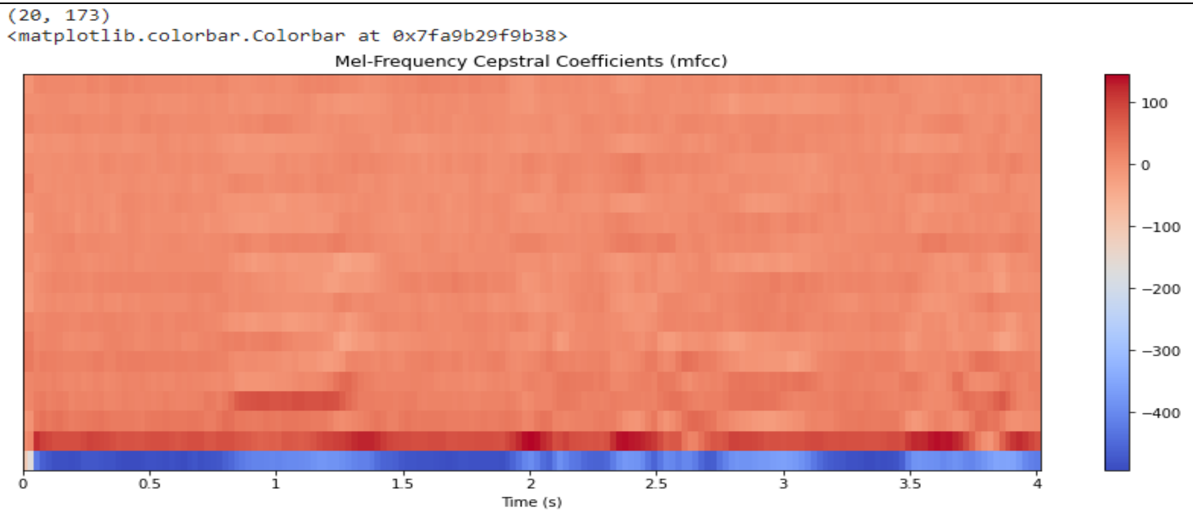
Mel-Frequency Cepstral Coefficients

The Mel frequency scale is a scale that expresses the perception of sound frequency changes in the human ear. MFCC (Mel Frequency Cepstral Coefficient) is an expression of the short-term power spectrum of an audio signal on the Mel frequency scale and is based on a known frequency change of the critical bandwidth of the human ear. The MFCC technique uses two types of filters, linearly distributed filters and logarithmically distributed filters. The Mel frequency scale is used to capture important features of sound recording and speech. The used MFCC scale has a linear frequency range below 1000 Hz and a logarithmic frequency range above it. The transform formula else is as follows;



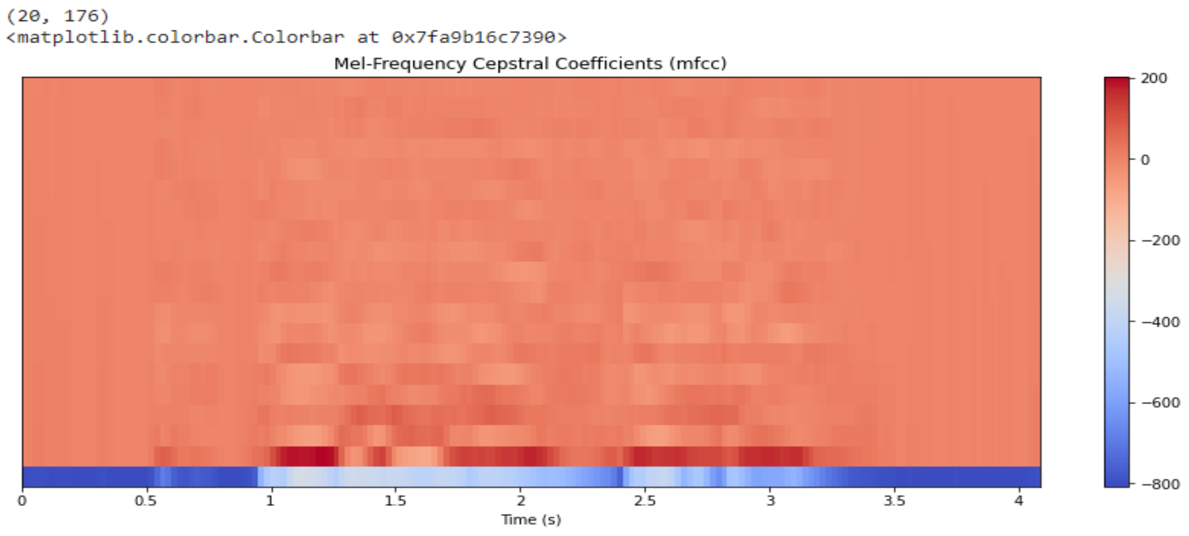
```
melfrekepkat=librosa.feature.mfcc(samples,sr=samplerate)
print(melfrekepkat.shape)
plt.figure(figsize=(15,6))
librosa.display.specshow(melfrekepkat,x_axis="s")
plt.title('Mel-Frequency Cepstral Coefficients (mfcc)');
plt.colorbar()
```





```
(20, 173)
<matplotlib.colorbar.Colorbar at 0x7fa9b29f9b38>
```

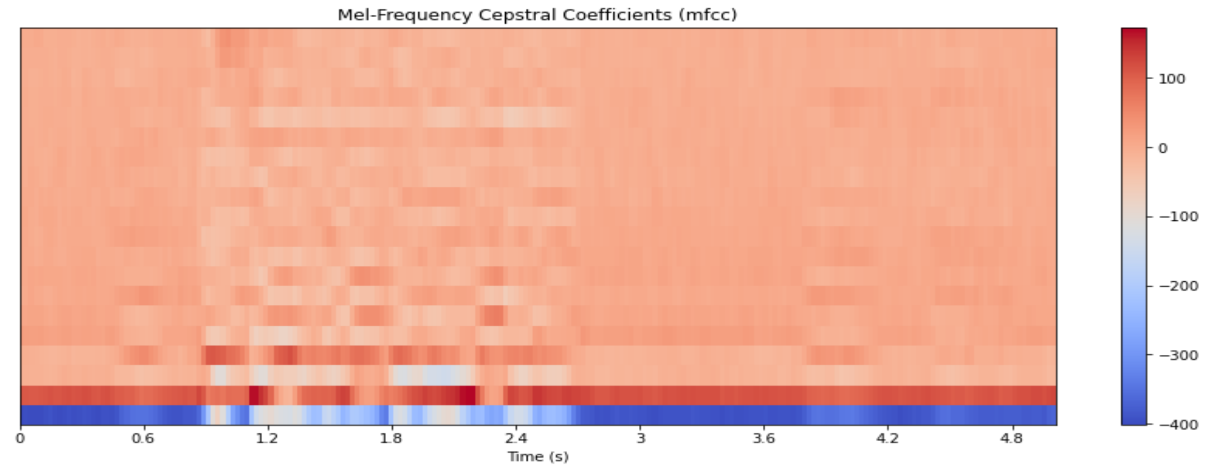
demo_audio_3.wav
Emotions: disgust



```
(20, 176)
<matplotlib.colorbar.Colorbar at 0x7fa9b16c7390>
```

demo_audio_1.wav
Emotions: sad

```
(20, 216)
<matplotlib.colorbar.Colorbar at 0x7fa9b0be2668>
```



demo_audio_7.wav
Emotions: anger

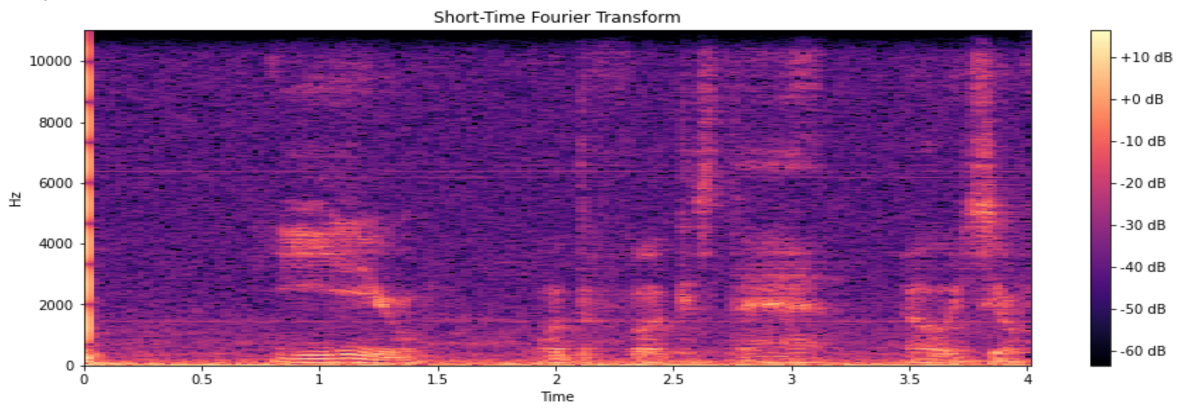
Fast Fourier Transform

The Fourier transform is essentially a mathematical treatment of sound. It is an algorithm that can quickly calculate the Fourier transform process that expands any complex signal to the frequency axis, a function that takes the signal as a time domain input and decomposes it into frequency. Its formula is generally as follows;

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\left(\frac{2\pi}{N}\right)nk} \quad (k = 0,1,\ldots,N-1)$$

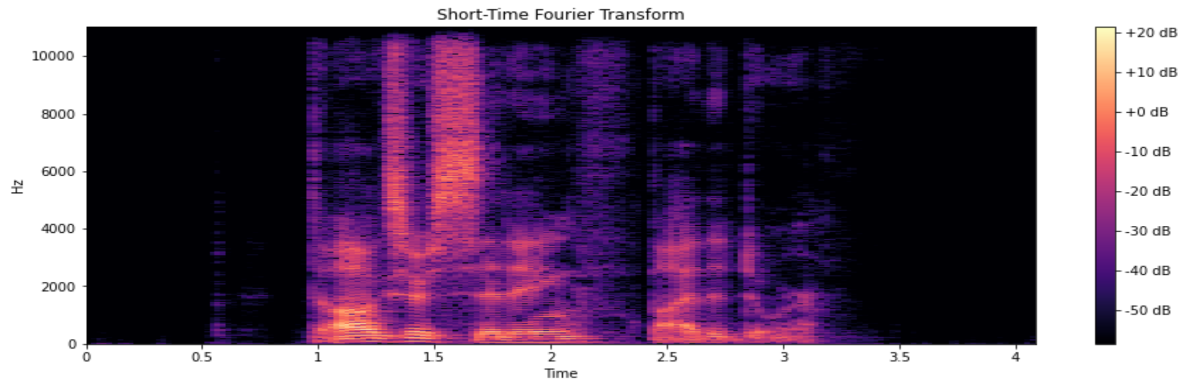
```
hop_length = 512
n_fft = 2048
X = librosa.stft(samples, n_fft=n_fft, hop_length=hop_length)
S = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(15, 5))
librosa.display.specshow(S, sr=samplerate, hop_length=hop_length, x_axis='time', y_axis='linear')
plt.title('Short-Time Fourier Transform');

plt.colorbar(format='%+2.0f dB')
```

```
<matplotlib.colorbar.Colorbar at 0x7fa9bd1e1f98>
```



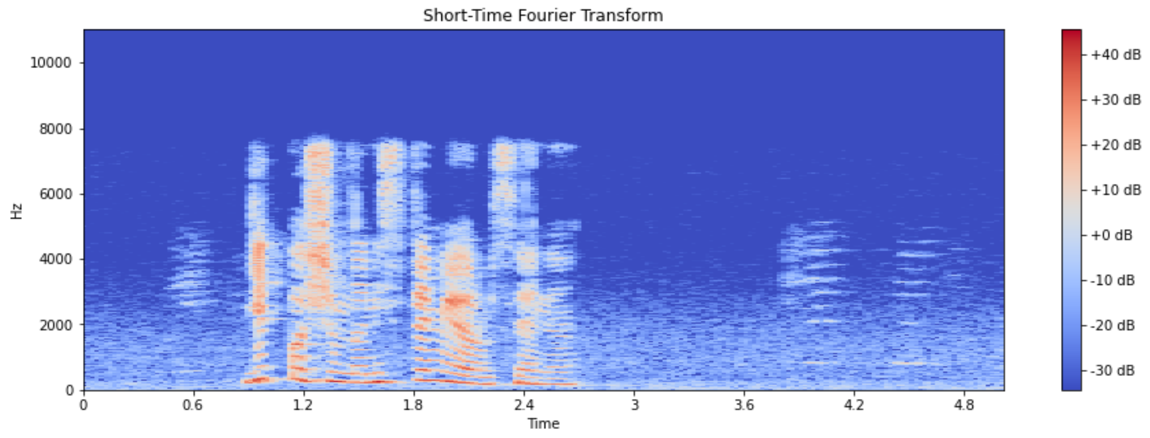
demo_audio_3.wav
Emotions: disgust

```
<matplotlib.colorbar.Colorbar at 0x7fa9b196d780>
```



demo_audio_1.wav
Emotions: sad

```
<matplotlib.colorbar.Colorbar at 0x7fa9b0b805f8>
```

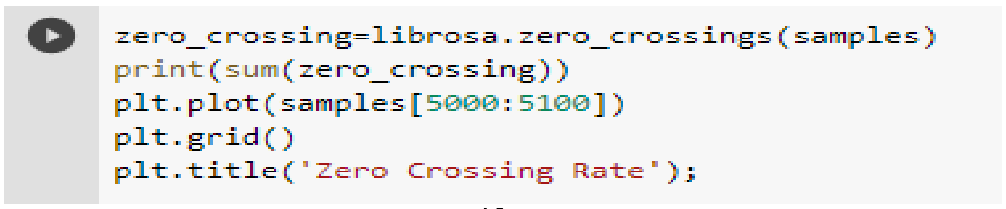


demo_audio_7.wav
Emotions: anger

Zero Crossing Rate

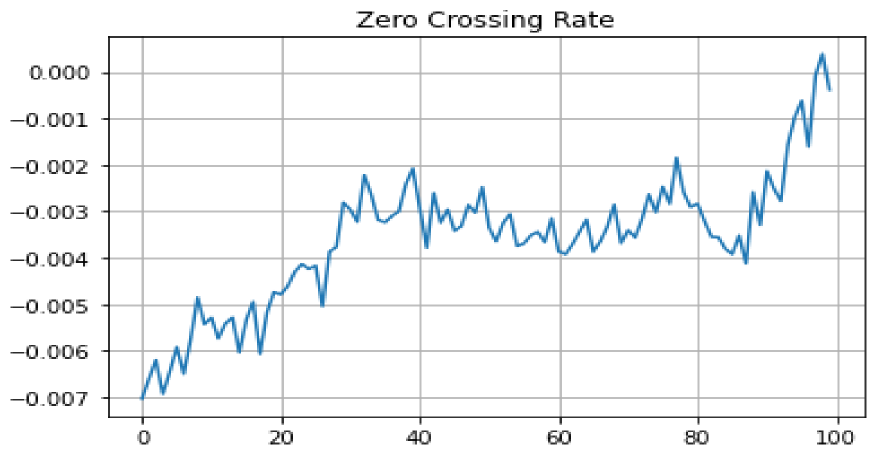
The transition from negative to positive or vice versa in an audio signal. Such a determination is made to start driver circuits as a result of any signal reaching zero. The excess of these transitions may indicate that a conversation is taking place in the area. Once the possible language is recognized by the sudden increase and decrease in sound intensity, the number of these transitions is checked to check, eg Where it begins and where it ends. Zero Crossing Rate formula is as follows;

$$zcr = \frac{1}{T-1}\sum_{t=1}^{T-1} 1_{\mathbb{R}_{<0}}(s_t s_{t-1})$$



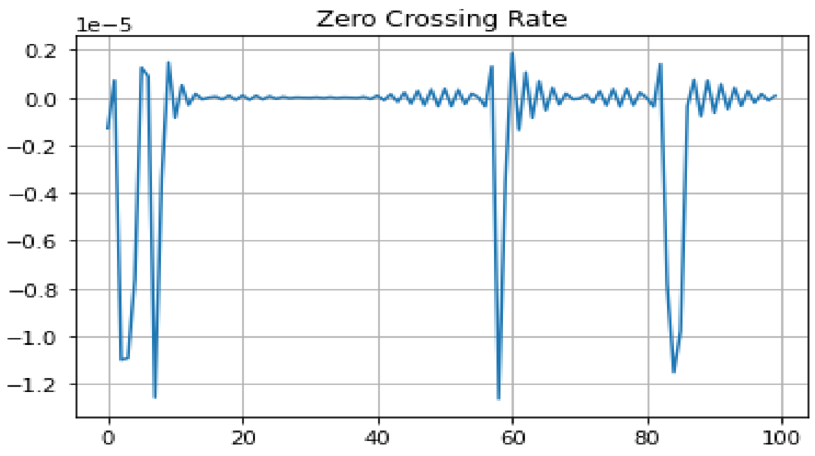
```
zero_crossing=librosa.zero_crossings(samples)
print(sum(zero_crossing))
plt.plot(samples[5000:5100])
plt.grid()
plt.title('Zero Crossing Rate');
```

```
5074
```



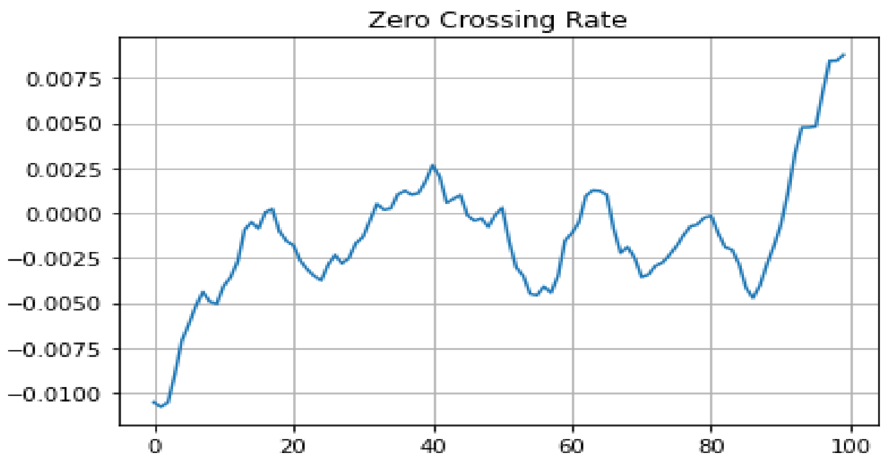
demo_audio_3.wav
Emotions: disgust

```
28793
```



demo_audio_1.wav
Emotions: sad

```
11895
```

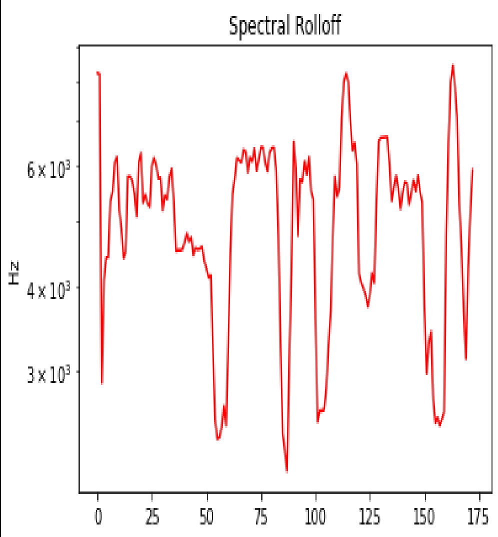
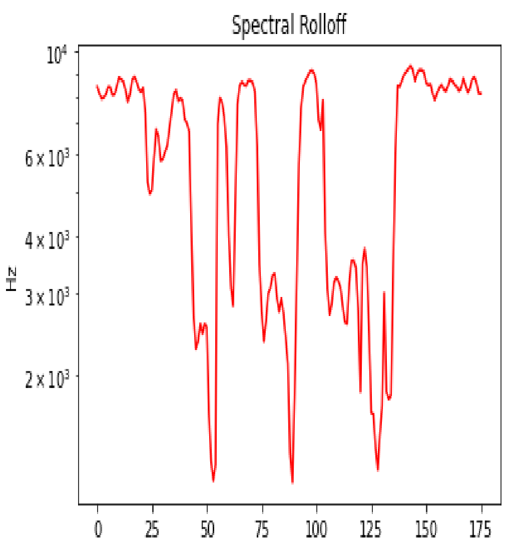
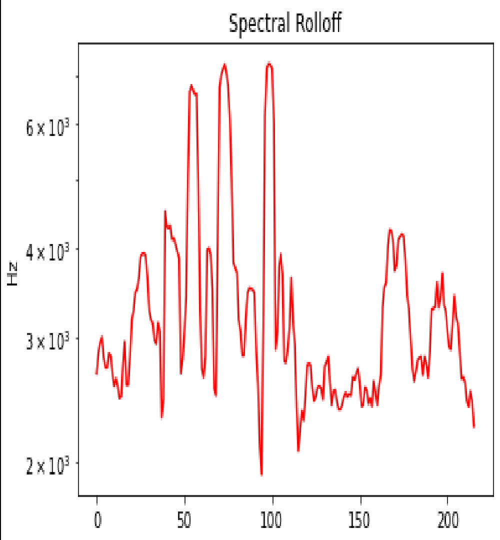


demo_audio_7.wav
Emotions: anger

SPECTRAL ROLLOFF

A signal shape measure. It represents a certain percentage of its total spectral energy.

```
spec_roll=librosa.feature.spectral_rolloff(samples,sr=samplerate)
print(spec_roll.shape)
plt.semilogy(spec_roll.T,"r")
plt.ylabel("Hz")
plt.title('Spectral Rolloff');
```

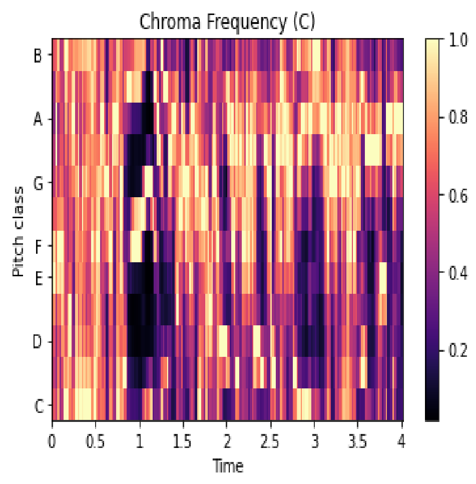
(1, 173)  demo_audio_3.wav Emotions: disgust	(1, 176)  demo_audio_1.wav Emotions: sad	(1, 216)  demo_audio_7.wav Emotions: anger

Chroma Frequency

It is a powerful representation of the spectrum musical octave for sound with 12 tracks representing 12 different tan content colors ("chroma"). It is checked under the name "pitch class" and "Time".

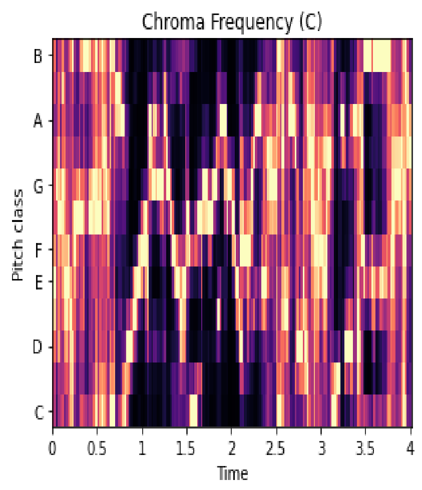
```
chroma=librosa.feature.chroma_stft(samples,sr=samplerate)
print(chroma.shape)
librosa.display.specshow(chroma,y_axis="chroma",x_axis="time")
plt.colorbar()
plt.title('Chroma Frequency (C)');
```

```
(12, 173)
```



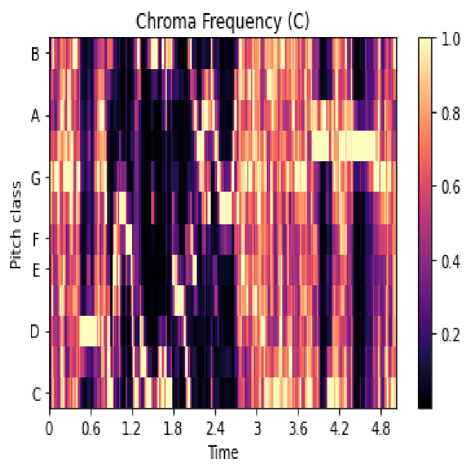
demo_audio_3.wav
Emotions: disgust

```
(12, 173)
```



demo_audio_1.wav
Emotions: sad

```
(12, 216)
```



demo_audio_7.wav
Emotions: anger

Emotions

Emotion The most important part of our work is which emotion is found from the sound. As a result of the "Feature Extractions" section and "Feature Data Frame" section that we have used above, we see value as a result of the values recorded in "livepreds". Using these values in our array called emotions, we return both the emotion and a png of the emotion.

Use for find which emotion:

```
emotions=["anger","disgust","fear","happy","neutral", "sad", "surprise"]
index = livepreds.argmax(axis=1).item()
emotions[index]
```

Use for Png of emotion

```
emojis_img = image.load_img('/content/drive/MyDrive/PeriHocaProjeBitisi/EmotionPic/%s.png'% emotions[index])
plt.imshow(emojis_img)
plt.axis('off')
plt.show()
```

```
'disgust'
```

demo_audio_3.wav
Emotions: disgust

```
'sad'
```

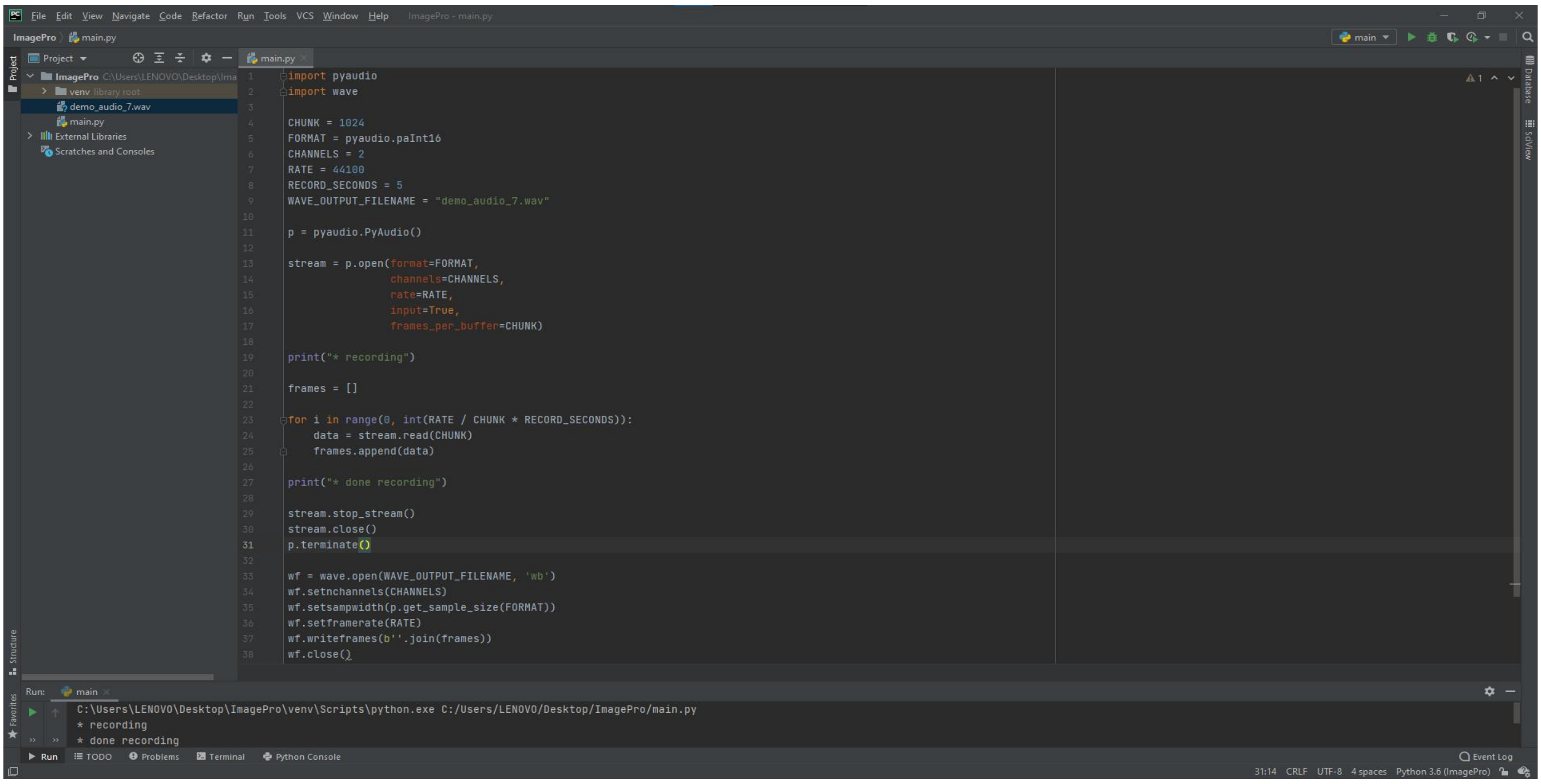
demo_audio_1.wav
Emotions: sad

```
'anger'
```

demo_audio_7.wav
Emotions: anger

Full code

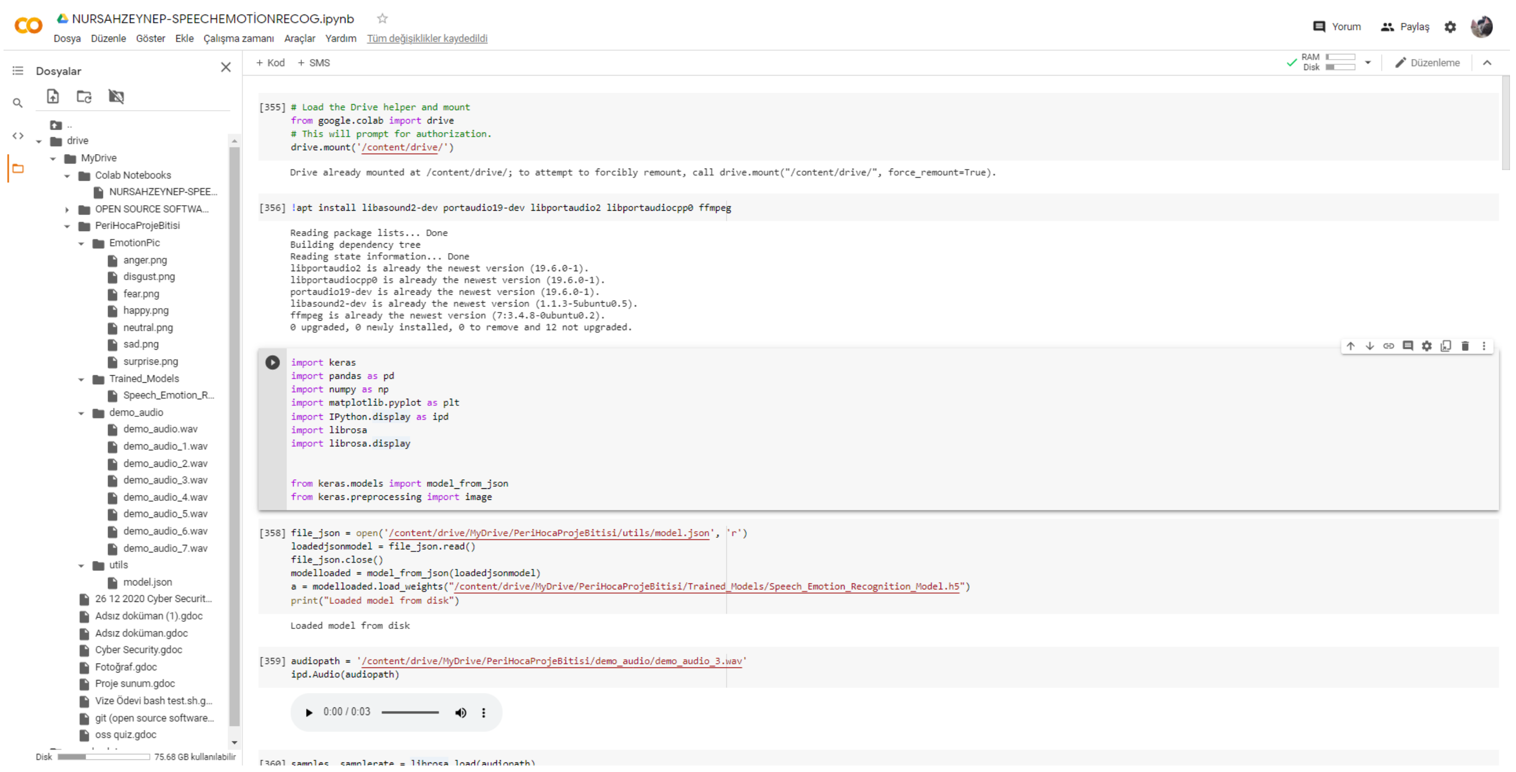
Recording Sound



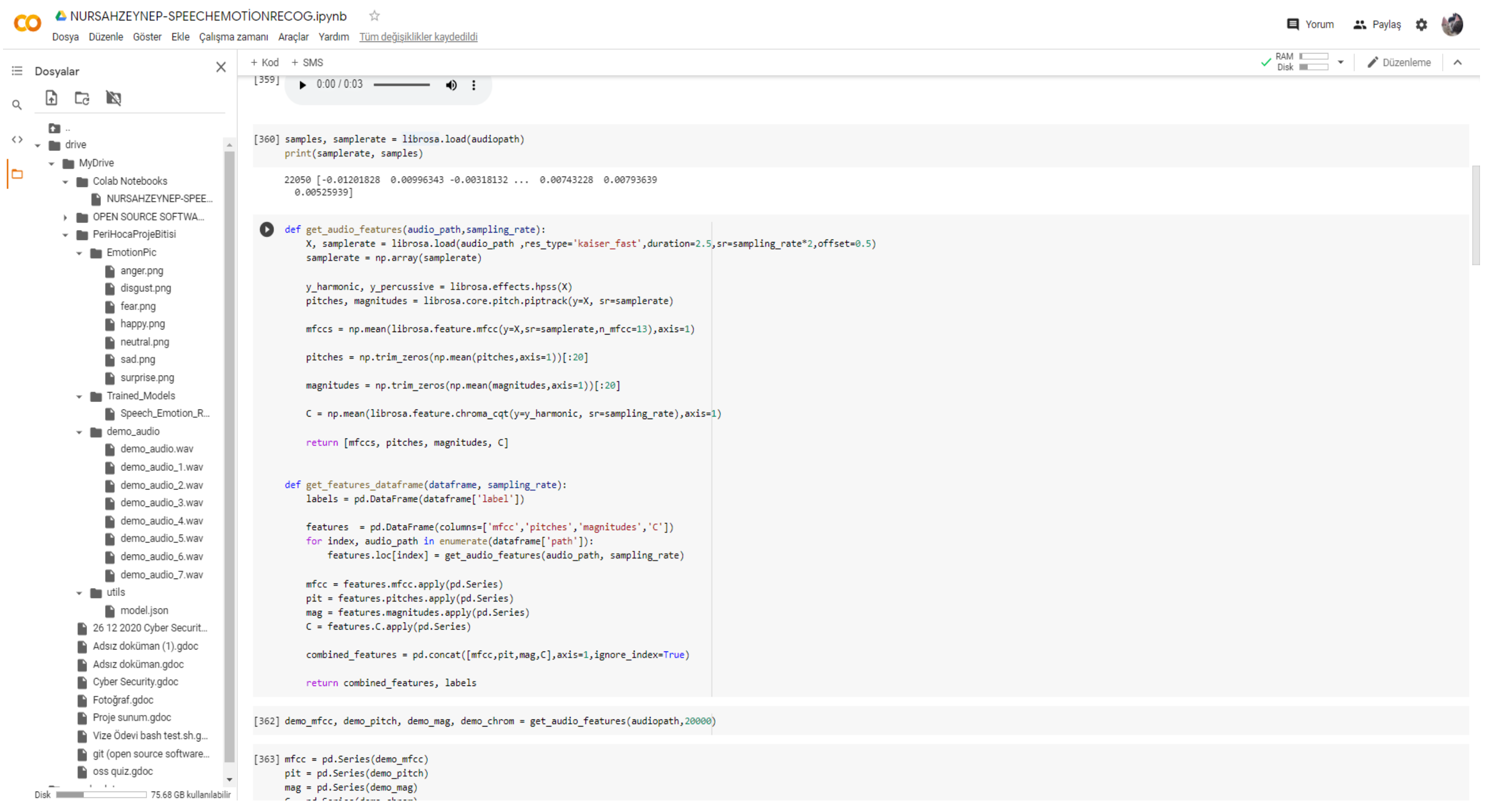
```
import pyaudio
import wave

CHUNK = 1024
FORMAT = pyaudio.paInt16
CHANNELS = 2
RATE = 44100
RECORD_SECONDS = 5
WAVE_OUTPUT_FILENAME = "demo_audio_7.wav"

p = pyaudio.PyAudio()

stream = p.open(format=FORMAT,
                channels=CHANNELS,
                rate=RATE,
                input=True,
                frames_per_buffer=CHUNK)

print("* recording")

frames = []

for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
    data = stream.read(CHUNK)
    frames.append(data)

print("* done recording")

stream.stop_stream()
stream.close()
p.terminate()

wf = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()
```

```
C:\Users\LENOVO\Desktop\ImagePro\venv\Scripts\python.exe C:/Users/LENOVO/Desktop/ImagePro/main.py
* recording
* done recording
```

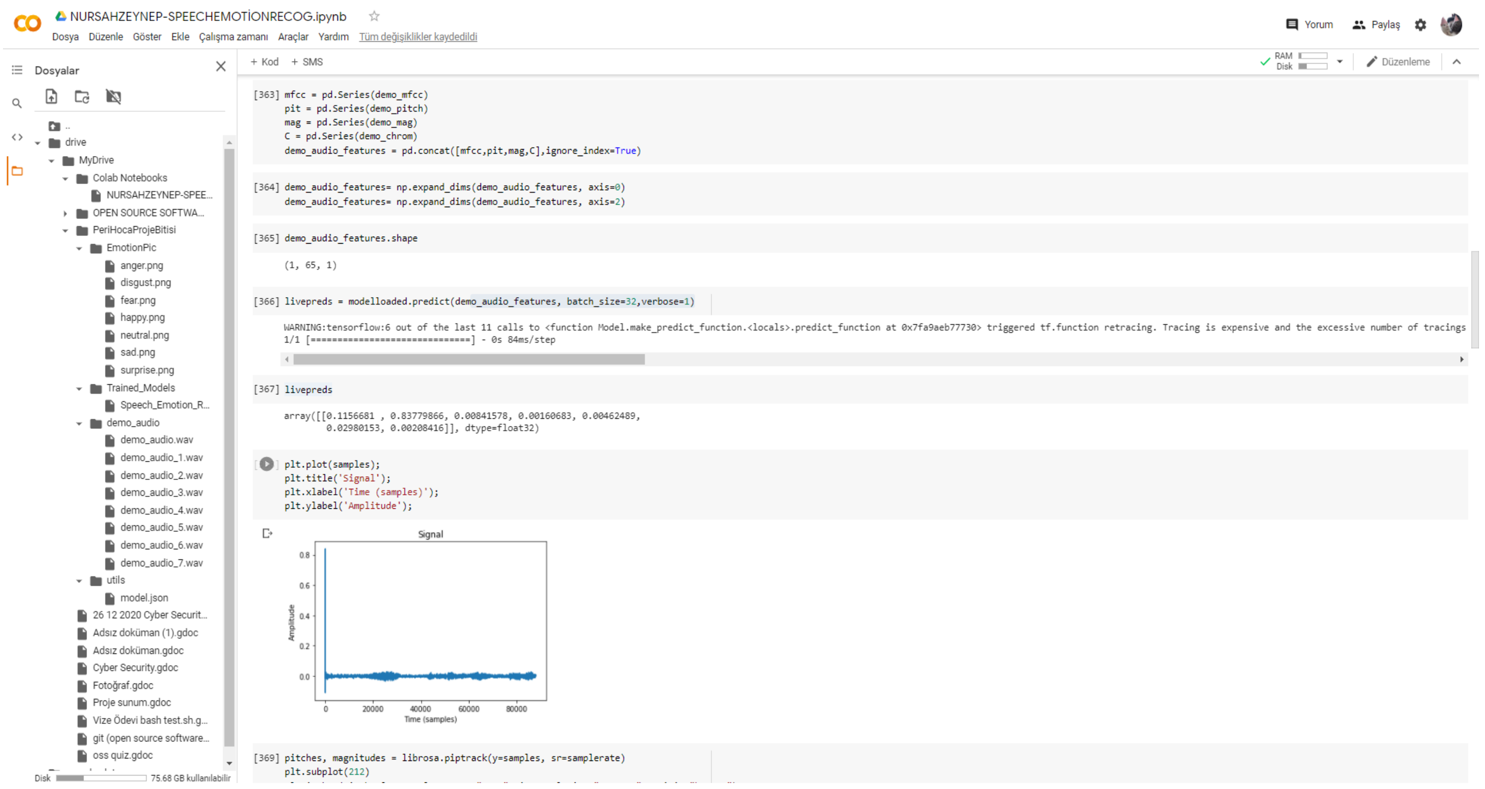
Speech Emotion Recognition



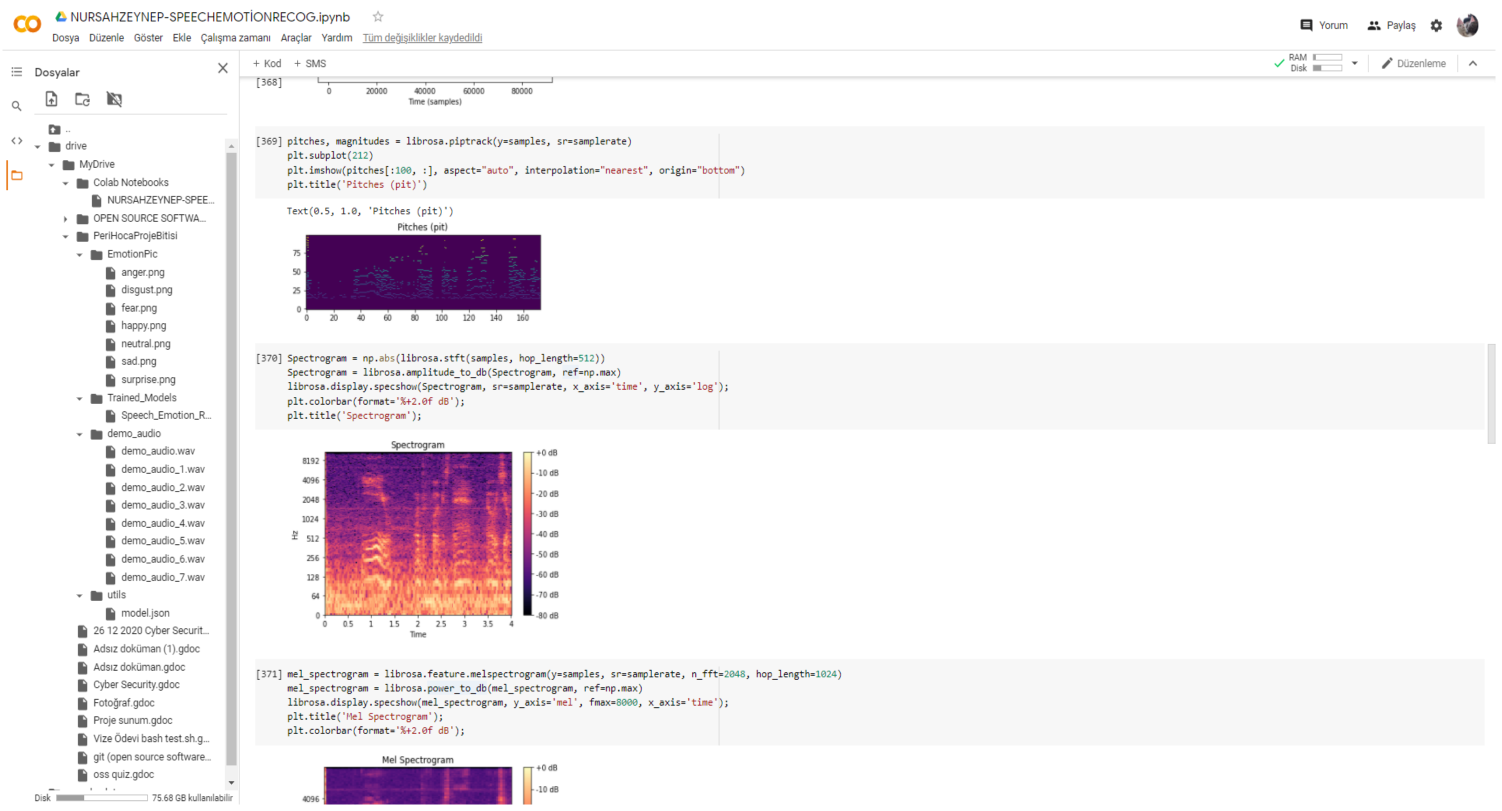
```
[355] # Load the Drive helper and mount
      from google.colab import drive
      # This will prompt for authorization.
      drive.mount('/content/drive/')

      Drive already mounted at /content/drive/; to attempt to forcibly remount, call drive.mount("/content/drive/", force_remount=True).

[356] !apt install libasound2-dev portaudio19-dev libportaudio2 libportaudiocpp0 ffmpeg

      Reading package lists... Done
      Building dependency tree
      Reading state information... Done
      libportaudio2 is already the newest version (19.6.0-1).
      libportaudiocpp0 is already the newest version (19.6.0-1).
      portaudio19-dev is already the newest version (19.6.0-1).
      libasound2-dev is already the newest version (1.1.3-5ubuntu0.5).
      ffmpeg is already the newest version (7:3.4.8-0ubuntu0.2).
      0 upgraded, 0 newly installed, 0 to remove and 12 not upgraded.

      import keras
      import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import IPython.display as ipd
      import librosa
      import librosa.display


      from keras.models import model_from_json
      from keras.preprocessing import image

[358] file_json = open('/content/drive/MyDrive/PeriHocaProjeBitisi/utils/model.json', 'r')
      loadedjsonmodel = file_json.read()
      file_json.close()
      modelloaded = model_from_json(loadedjsonmodel)
      a = modelloaded.load_weights("/content/drive/MyDrive/PeriHocaProjeBitisi/Trained_Models/Speech_Emotion_Recognition_Model.h5")
      print("Loaded model from disk")

      Loaded model from disk

[359] audiopath = '/content/drive/MyDrive/PeriHocaProjeBitisi/demo_audio/demo_audio_3.wav'
      ipd.Audio(audiopath)

      0:00 / 0:03

[360] samples, samplerate = librosa.load(audiopath)
```



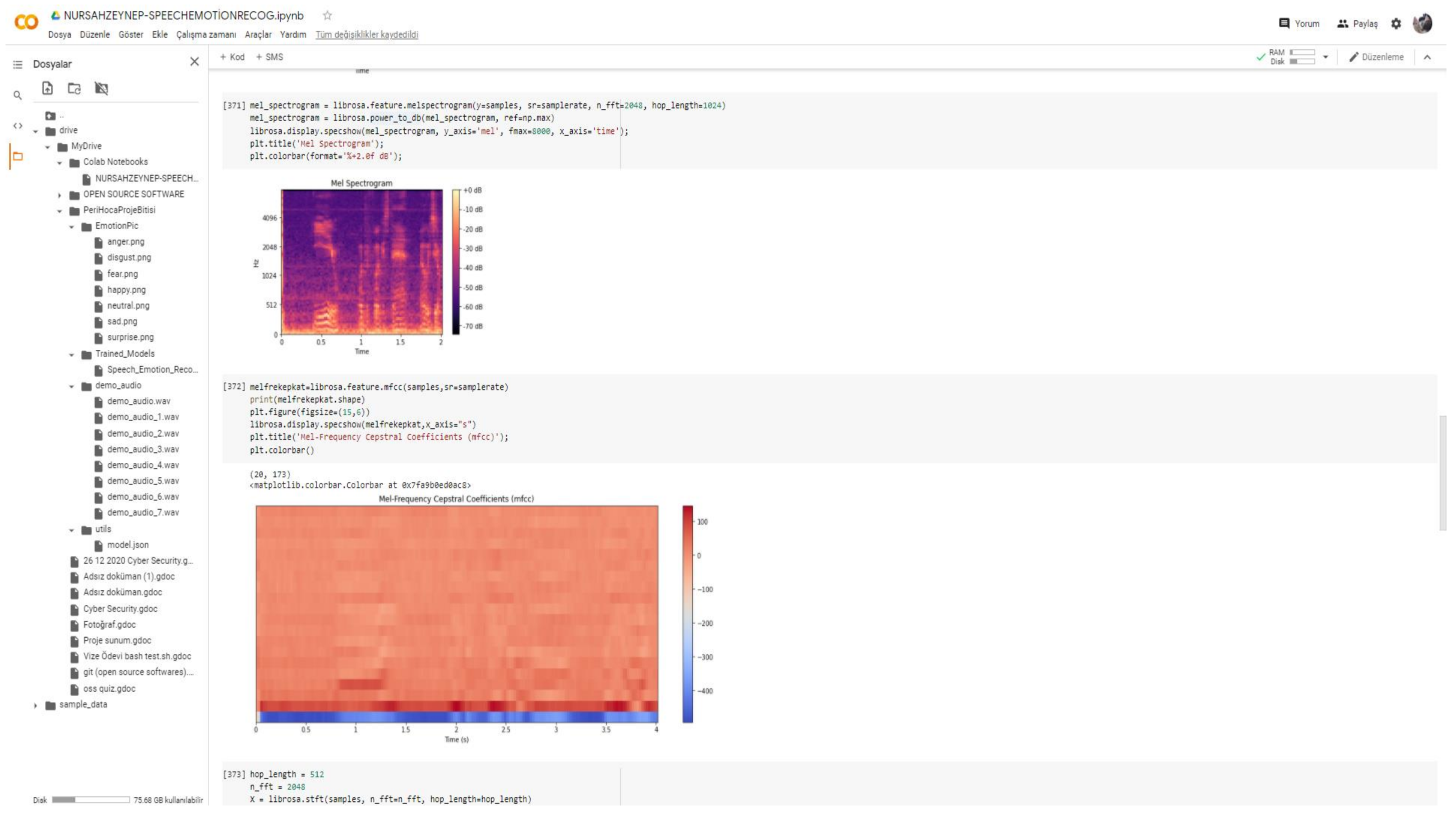
```
[359]

[360] samples, samplerate = librosa.load(audiopath)
      print(samplerate, samples)

      22050 [-0.01201828  0.00996343 -0.00318132 ...  0.00743228  0.00793639
        0.00525939]

      def get_audio_features(audio_path,sampling_rate):
          X, samplerate = librosa.load(audio_path ,res_type='kaiser_fast',duration=2.5,sr=sampling_rate*2,offset=0.5)
          samplerate = np.array(samplerate)

          y_harmonic, y_percussive = librosa.effects.hpss(X)
          pitches, magnitudes = librosa.core.pitch.piptrack(y=X, sr=samplerate)

          mfccs = np.mean(librosa.feature.mfcc(y=X,sr=samplerate,n_mfcc=13),axis=1)

          pitches = np.trim_zeros(np.mean(pitches,axis=1))[:20]

          magnitudes = np.trim_zeros(np.mean(magnitudes,axis=1))[:20]

          C = np.mean(librosa.feature.chroma_cqt(y=y_harmonic, sr=sampling_rate),axis=1)

          return [mfccs, pitches, magnitudes, C]


      def get_features_dataframe(dataframe, sampling_rate):
          labels = pd.DataFrame(dataframe['label'])

          features  = pd.DataFrame(columns=['mfcc','pitches','magnitudes','C'])
          for index, audio_path in enumerate(dataframe['path']):
              features.loc[index] = get_audio_features(audio_path, sampling_rate)

          mfcc = features.mfcc.apply(pd.Series)
          pit = features.pitches.apply(pd.Series)
          mag = features.magnitudes.apply(pd.Series)
          C = features.C.apply(pd.Series)

          combined_features = pd.concat([mfcc,pit,mag,C],axis=1,ignore_index=True)

          return combined_features, labels

[362] demo_mfcc, demo_pitch, demo_mag, demo_chrom = get_audio_features(audiopath,20000)

[363] mfcc = pd.Series(demo_mfcc)
      pit = pd.Series(demo_pitch)
      mag = pd.Series(demo_mag)
```


NURSAHZEYNEP-SPEECHEMOTİONRECOG.ipynb
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi
Yorum
Paylaş
+ Kod + SMS
RAM Disk
Düzenleme
Dosyalar
Disk 75.68 GB kullanılabilir
Signal
0.8
0.6
0.4
0.2
0.0
Amplitude
0
20000
40000
60000
80000
Time (samples)

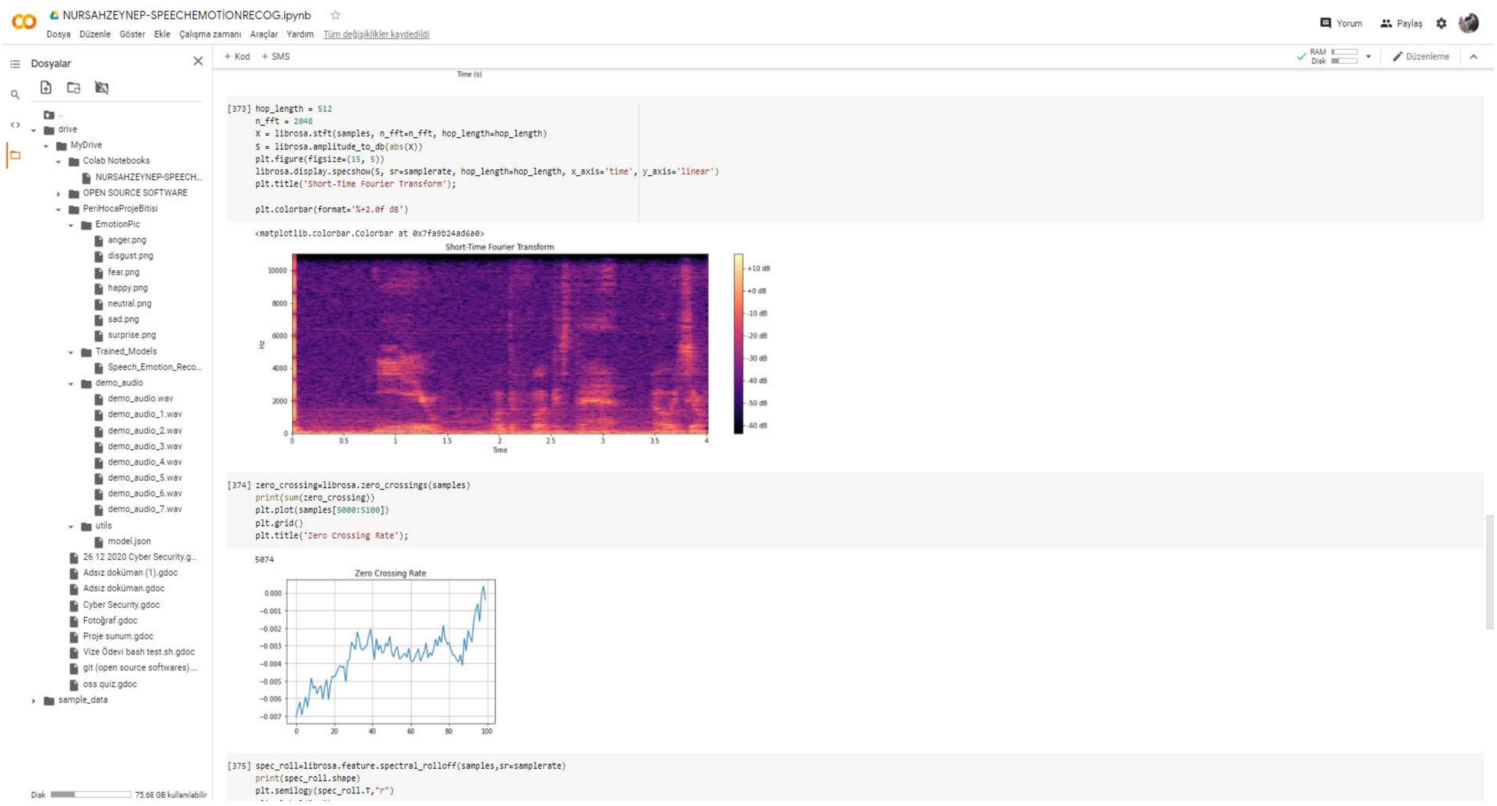
```
[363] mfcc = pd.Series(demo_mfcc)
      pit = pd.Series(demo_pitch)
      mag = pd.Series(demo_mag)
      C = pd.Series(demo_chrom)
      demo_audio_features = pd.concat([mfcc,pit,mag,C],ignore_index=True)

[364] demo_audio_features= np.expand_dims(demo_audio_features, axis=0)
      demo_audio_features= np.expand_dims(demo_audio_features, axis=2)

[365] demo_audio_features.shape

      (1, 65, 1)

[366] livepreds = modelloaded.predict(demo_audio_features, batch_size=32,verbose=1)

      WARNING:tensorflow:6 out of the last 11 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7fa9aeb77730> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings
      1/1 [==============================] - 0s 84ms/step

[367] livepreds

      array([[0.1156681 , 0.83779866, 0.00841578, 0.00160683, 0.00462489,
              0.02980153, 0.00208416]], dtype=float32)

      plt.plot(samples);
      plt.title('Signal');
      plt.xlabel('Time (samples)');
      plt.ylabel('Amplitude');

[369] pitches, magnitudes = librosa.piptrack(y=samples, sr=samplerate)
      plt.subplot(212)
```



```
[368]

[369] pitches, magnitudes = librosa.piptrack(y=samples, sr=samplerate)
      plt.subplot(212)
      plt.imshow(pitches[:100, :], aspect="auto", interpolation="nearest", origin="bottom")
      plt.title('Pitches (pit)')

      Text(0.5, 1.0, 'Pitches (pit)')

[370] Spectrogram = np.abs(librosa.stft(samples, hop_length=512))
      Spectrogram = librosa.amplitude_to_db(Spectrogram, ref=np.max)
      librosa.display.specshow(Spectrogram, sr=samplerate, x_axis='time', y_axis='log');
      plt.colorbar(format='%+2.0f dB');
      plt.title('Spectrogram');

[371] mel_spectrogram = librosa.feature.melspectrogram(y=samples, sr=samplerate, n_fft=2048, hop_length=1024)
      mel_spectrogram = librosa.power_to_db(mel_spectrogram, ref=np.max)
      librosa.display.specshow(mel_spectrogram, y_axis='mel', fmax=8000, x_axis='time');
      plt.title('Mel Spectrogram');
      plt.colorbar(format='%+2.0f dB');
```


NURSAHZEYNEP-SPEECHEMOTİONRECOG.ipynb
Dosya Düzenle Göster Ekle Çalışma zamanı Araçlar Yardım Tüm değişiklikler kaydedildi
Yorum Paylaş
+ Kod + SMS
Düzenleme
Dosyalar
sample_data
Disk 75.68 GB kullanılabilir
Mel Spectrogram
Hz
Time
Mel-Frequency Cepstral Coefficients (mfcc)
Time (s)

```
[371] mel_spectrogram = librosa.feature.melspectrogram(y=samples, sr=samplerate, n_fft=2048, hop_length=1024)
      mel_spectrogram = librosa.power_to_db(mel_spectrogram, ref=np.max)
      librosa.display.specshow(mel_spectrogram, y_axis='mel', fmax=8000, x_axis='time');
      plt.title('Mel Spectrogram');
      plt.colorbar(format='%+2.0f dB');

[372] melfrekepkat=librosa.feature.mfcc(samples,sr=samplerate)
      print(melfrekepkat.shape)
      plt.figure(figsize=(15,6))
      librosa.display.specshow(melfrekepkat,x_axis="s")
      plt.title('Mel-Frequency Cepstral Coefficients (mfcc)');
      plt.colorbar()

      (20, 173)
      <matplotlib.colorbar.Colorbar at 0x7fa9b0ed0ac8>

[373] hop_length = 512
      n_fft = 2048
      X = librosa.stft(samples, n_fft=n_fft, hop_length=hop_length)
```



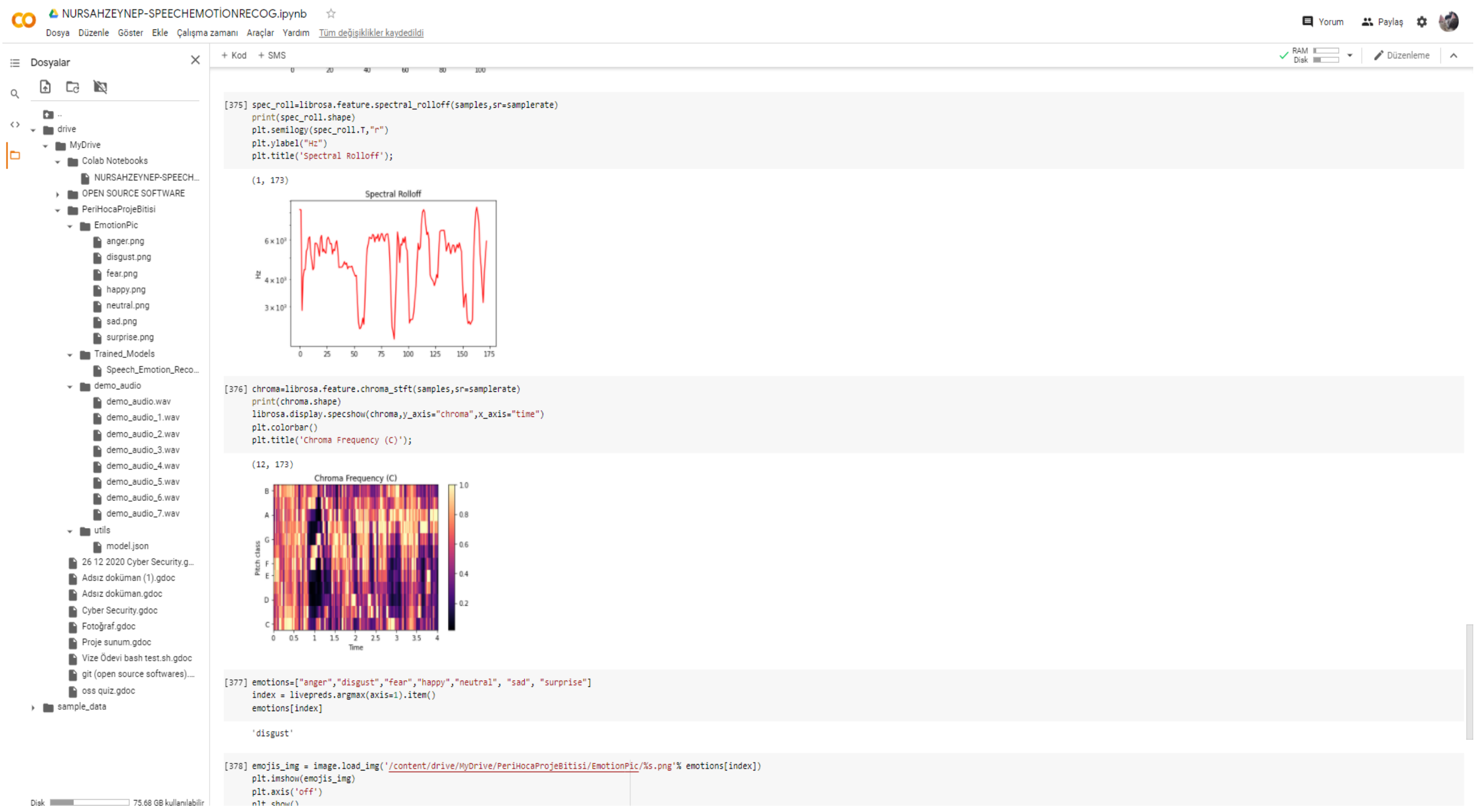
```
[373] hop_length = 512
      n_fft = 2048
      X = librosa.stft(samples, n_fft=n_fft, hop_length=hop_length)
      S = librosa.amplitude_to_db(abs(X))
      plt.figure(figsize=(15, 5))
      librosa.display.specshow(S, sr=samplerate, hop_length=hop_length, x_axis='time', y_axis='linear')
      plt.title('Short-Time Fourier Transform');

      plt.colorbar(format='%+2.0f dB')
```

```
<matplotlib.colorbar.Colorbar at 0x7fa9b24ad6a0>
```

```
[374] zero_crossing=librosa.zero_crossings(samples)
      print(sum(zero_crossing))
      plt.plot(samples[5000:5100])
      plt.grid()
      plt.title('Zero Crossing Rate');
```

```
5074
```

```
[375] spec_roll=librosa.feature.spectral_rolloff(samples,sr=samplerate)
      print(spec_roll.shape)
      plt.semilogy(spec_roll.T,"r")
```



```
[375] spec_roll=librosa.feature.spectral_rolloff(samples,sr=samplerate)
      print(spec_roll.shape)
      plt.semilogy(spec_roll.T,"r")
      plt.ylabel("Hz")
      plt.title('Spectral Rolloff');
```

```
(1, 173)
```

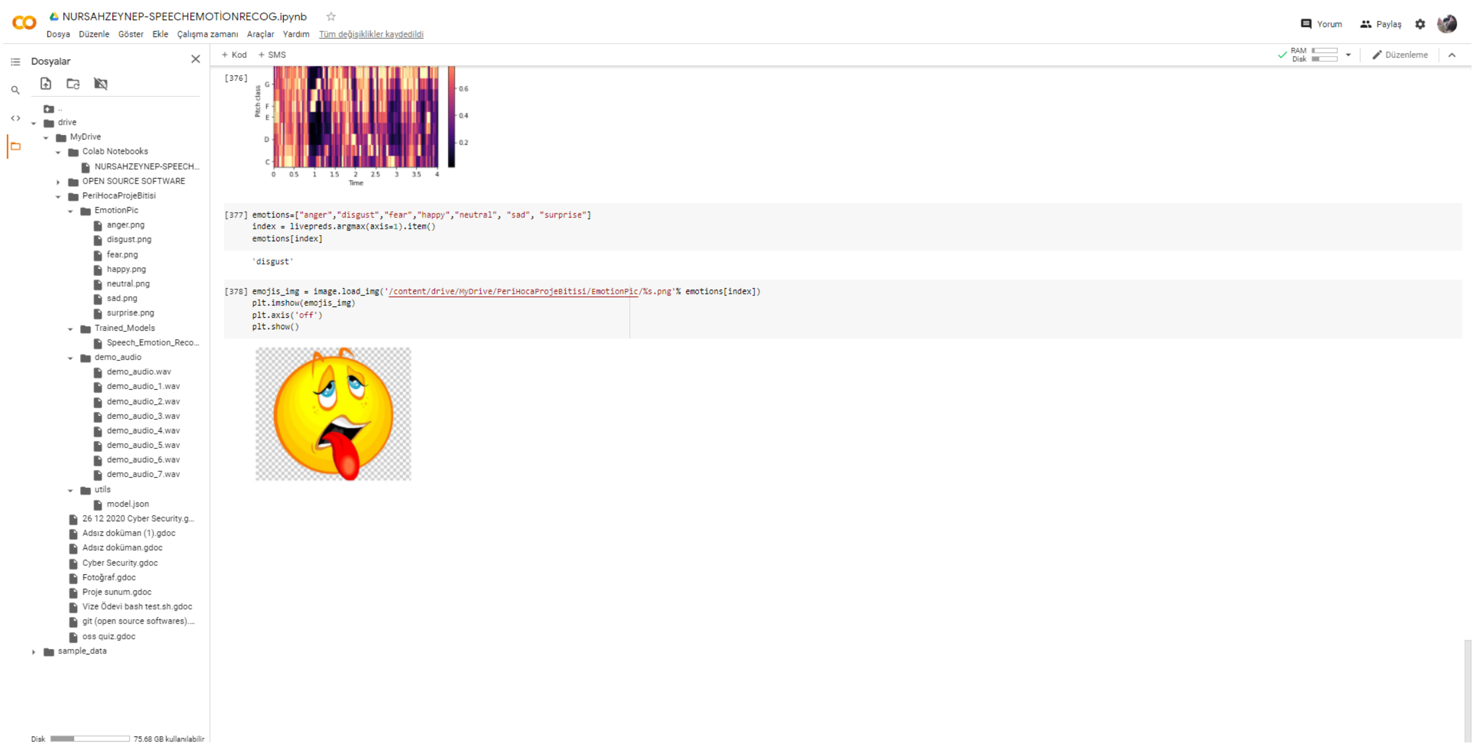
```
[376] chroma=librosa.feature.chroma_stft(samples,sr=samplerate)
      print(chroma.shape)
      librosa.display.specshow(chroma,y_axis="chroma",x_axis="time")
      plt.colorbar()
      plt.title('Chroma Frequency (C)');
```

```
(12, 173)
```

```
[377] emotions=["anger","disgust","fear","happy","neutral", "sad", "surprise"]
      index = livepreds.argmax(axis=1).item()
      emotions[index]
```

```
'disgust'
```

```
[378] emojis_img = image.load_img('/content/drive/MyDrive/PeriHocaProjeBitisi/EmotionPic/%s.png'% emotions[index])
      plt.imshow(emojis_img)
      plt.axis('off')
```



```
[376]

[377] emotions=["anger","disgust","fear","happy","neutral", "sad", "surprise"]
      index = livepreds.argmax(axis=1).item()
      emotions[index]

      'disgust'

[378] emojis_img = image.load_img('/content/drive/MyDrive/PeriHocaProjeBitisi/EmotionPic/%s.png'% emotions[index])
      plt.imshow(emojis_img)
      plt.axis('off')
      plt.show()
```

4. Presentation Link

https://www.youtube.com/watch?v=EPZ8hMZayzM

5. Conlusion

As a result, our project is a way of teaching the machine the phenomenon of determining emotion analysis with the help of human sound waves. In short, we gave the machine a human task. In our project, using MFCC, PITCHES and chroma algorithms to do emotion analysis with sound helped us a lot. In this way, working parents can easily leave their children alone with their nannies. Their parents will be able to follow their children's every emotional moment step by step. As soon as they see them crying, they can call the house and find out why.

6. References

[1] C. Parlak and B. Diri, "İnsan Sesinden Duygu Çıkarma - Emotion Recognition from the Human Voice," *Bilgi. Mühendisliği Bölümü Yıldız Tek. Üniversitesi*, pp. 11–14, 2013, doi: 978-1-4673-5563-6/13/$31.00 ©2013 IEEE.

[2] J. Rintala, "Speech Emotion Recognition from Raw Audio using Deep Learning from Raw Audio using Deep Learning," KTH ROYAL INSTITUTE OF TECHNOLOGY SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, 2020.

[3] O. E. Korkmaz, "EMOTION RECOGNITION FROM SPEECH SIGNAL," Karadeniz Technical University The Graduate School of Natural and Applied Sciences, 2016.

[4] E. S. Erdem, "EMOTION RECOGNITION AND RETRIEVAL IN AUDIO SIGNALS," Başkent Üniversitesi, 2014.

[5] V. V. Sharma, "Getting started with Speech Emotion Recognition | Visualising Emotions," *Analytics Vidhya*, 2020.

[6] S. Lalitha, A. Madhavan, B. Bhushan, and S. Saketh, "Speech emotion recognition," *2014 Int. Conf. Adv. Electron. Comput. Commun. ICAECC 2014*, no. 1, pp. 235–238, 2015, doi: 10.1109/ICAECC.2014.7002390.

[7] M. Kattel, A. Nepal, A. K. Shah, and D. C. Shrestha, "Chroma Feature Extraction," *Encycl. GIS*, no. January, pp. 1–9, 2019.

[8] R. Hasan, M. Jamil, G. Rabbani, and S. Rahman, "Speaker Identification Using Mel Frequency Cepstral Coefficients," *3rd Int. Conf. Electr. Comput. Eng. ICECE 2004*, no. December, pp. 28–30, 2004, doi: 984-32-1804-4.

[9] Y. Lee, "Method and apparatus for estimating pitch of signal," vol. 2, no. 12, p. 1, 2010, [Online]. Available: https://patentimages.storage.googleapis.com/f0/23/b7/12f9e545fefb6e/US7672836.pdf.

[10] H. Nussbaumer, "Fast Fourier Transform and Convolution Algorithms," pp. 3–4, 1981, doi: https://doi.org/10.1007/978-3-662-00551-4_4.