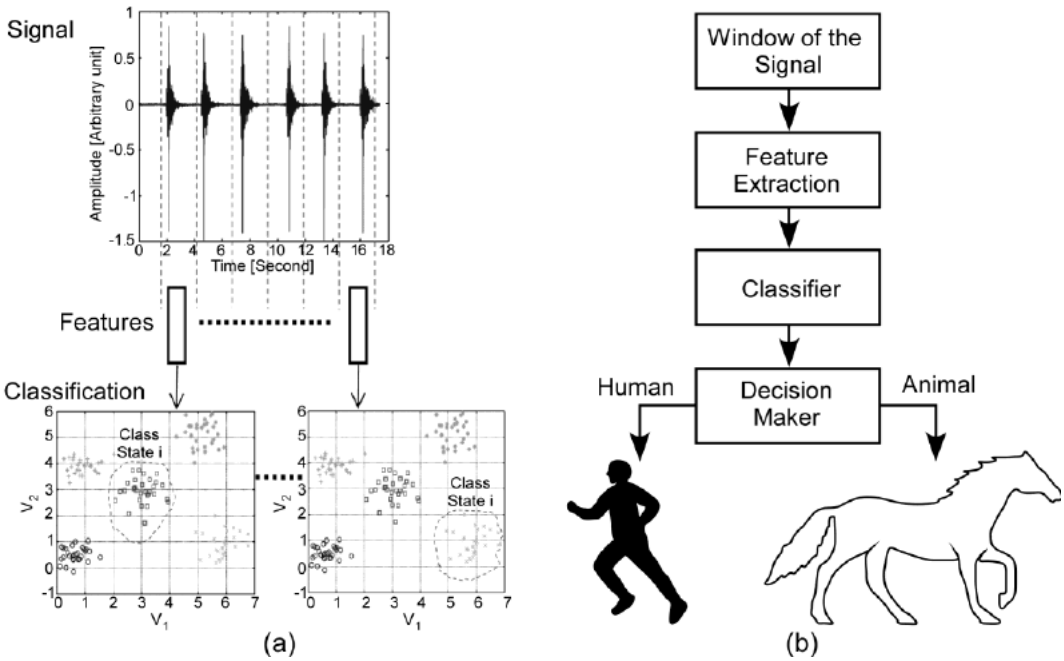
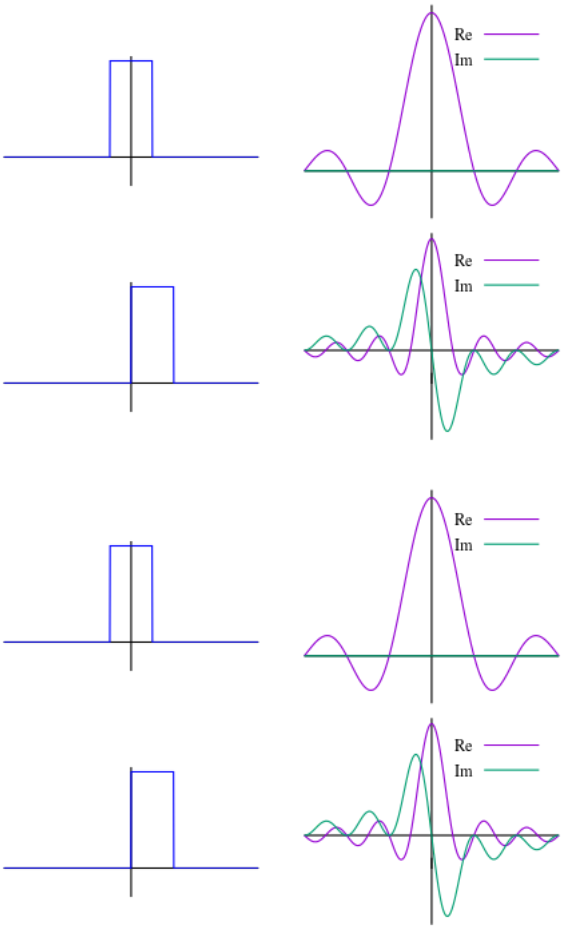
Signal To Features (Feature Extraction & Feature Selection)

Complex Signal To Data(Features)



Fourier transform (FT) is a transform that converts a function into a form that describes the frequencies present in the original function.



Fourier transform

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-i2\pi\xi x}\,dx.$$

In mathematics, the **discrete-time Fourier transform** (**DTFT**), also called the finite Fourier transform, is a form of Fourier analysis that is applicable to a sequence of values.

$$X_{2\pi}(\omega) = \sum_{n=-\infty}^{\infty} x[n]\, e^{-i\omega n}. \quad \textbf{(Eq.1)}$$

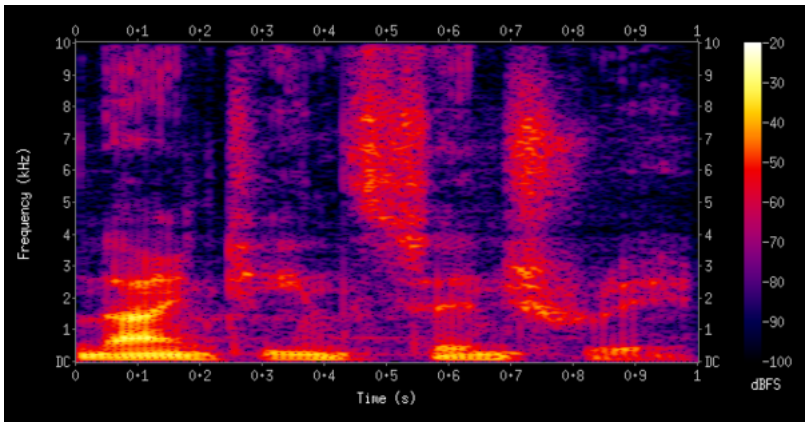
In mathematics, the **discrete Fourier transform** (DFT) converts a finite sequence of equally-spaced samples of a function into a same-length sequence of equally-spaced samples of the discrete-time Fourier transform (DTFT), which is a complex-valued function of frequency.

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-\frac{i2\pi}{N} kn}$$

A **Fast Fourier transform** (**FFT**) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence

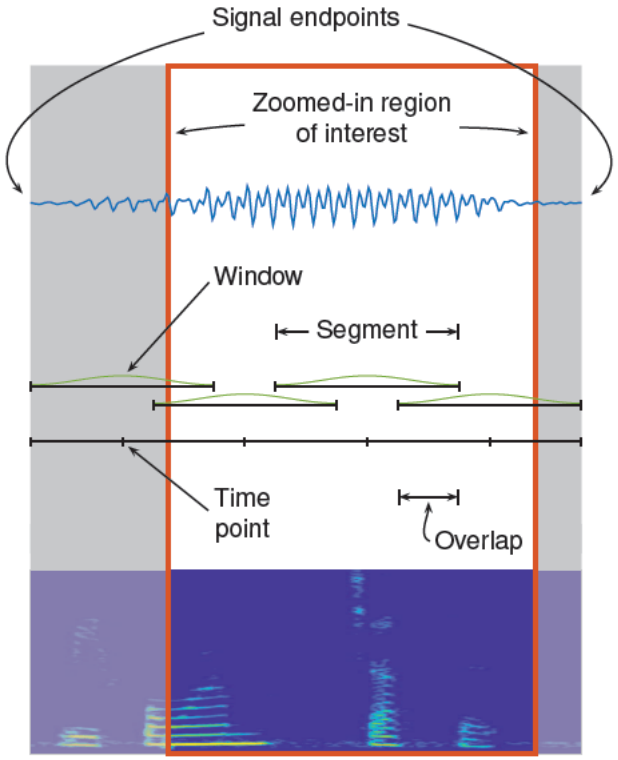
$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} \qquad k = 0, \ldots, N-1,$$

A **Spectrogram** is a visual representation of the spectrum of frequencies of a signal as it varies with time.



Create Spectrogram

1. Divide the signal into equal-length segments. The segments must be short enough that the frequency content of the signal does not change appreciably within a segment. The segments may or may not overlap.
2. Window each segment and compute its spectrum to get the **short-time Fourier transform**.
3. Display segment-by-segment the power of each spectrum in decibels. Depict the magnitudes side-by-side as an image with magnitude-dependent colormap.



The **short-time Fourier transform** (**STFT**), is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time.

$$\mathbf{STFT}\{x[n]\}(m,\omega) \equiv X(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-i\omega n}$$

$$\text{spectrogram}\{x(t)\}(\tau,\omega) \equiv |X(\tau,\omega)|^2$$

In sound processing, the **mel-frequency cepstrum** (**MFC**) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency.

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC. They are derived from a type of cepstral representation of the audio clip (a nonlinear "spectrum-of-a-spectrum").

Generate MFCC

1. Take the Fourier transform of (a windowed excerpt of) a signal.
2. Map the powers of the spectrum obtained above onto the mel scale, using triangular overlapping windows or alternatively, cosine overlapping windows.
3. Take the logs of the powers at each of the mel frequencies.
4. Take the discrete cosine transform of the list of mel log powers, as if it were a signal.
5. The MFCCs are the amplitudes of the resulting spectrum.

In [3]:
```
from IPython.display import Audio, display
import wave
import numpy as np
```

In [7]:
```
display(Audio('test.wav'))
```

0:00 / 0:00

In [8]:
```
obj = wave.open('test.wav','rb')
```

In [9]:
```
1 frames = obj.readframes(-1)
```

In [10]:
```
1 signal_array = np.frombuffer(frames,dtype=np.int16)
```

In [11]:
```
1 len(signal_array)
```

Out[11]: 617400

In [12]:
```
1 len(signal_array)/7
```

Out[12]: 88200.0

In [13]:
```
1 hoosh = signal_array[int((int(len(signal_array)/7))*2.3):int((int(len(signal_array)/7))*3.1)]
```

In [14]:
```
1 new_obj6 = wave.open('hoosh.wav','wb')
2 new_obj6.setframerate(44100)
3 new_obj6.setsampwidth(2)
4 new_obj6.setnchannels(2)
5 new_obj6.writeframes(hoosh.tobytes())
6 new_obj6.close()
```

In [15]:
```
1 display(Audio('hoosh.wav'))
```

0:00 / 0:00

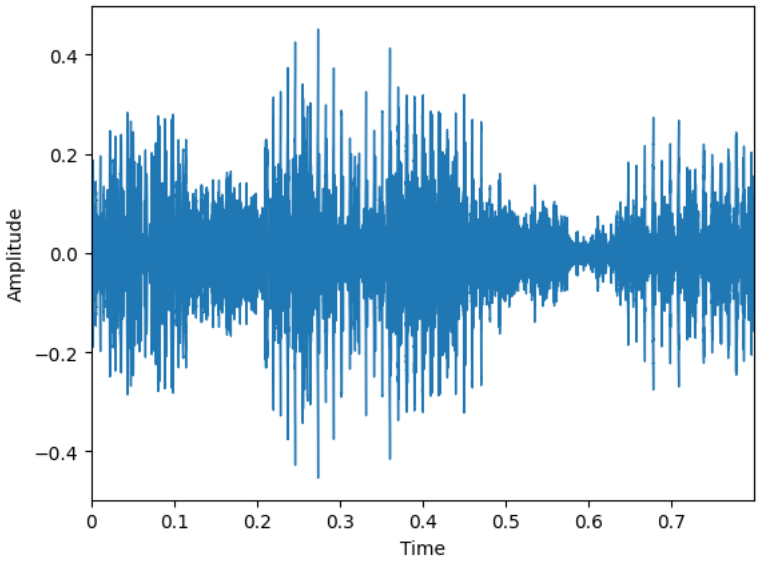
In [16]:
```
!pip uninstall librosa # 0.9.0
!pip install librosa==0.8.0
```

```
Found existing installation: librosa 0.10.0.post2
Uninstalling librosa-0.10.0.post2:
  Would remove:
    /usr/local/lib/python3.10/dist-packages/librosa-0.10.0.post2.dist-info/*
    /usr/local/lib/python3.10/dist-packages/librosa/*
Proceed (Y/n)? y
  Successfully uninstalled librosa-0.10.0.post2
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) https://us-python.pkg.dev/colab-wheels/public/simple/
(https://us-python.pkg.dev/colab-wheels/public/simple/)
Collecting librosa==0.8.0
  Downloading librosa-0.8.0.tar.gz (183 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 183.9/183.9 kB 13.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: audioread>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0) (3.0.0)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0) (1.22.4)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0) (1.10.1)
Requirement already satisfied: scikit-learn!=0.19.0,>=0.14.0 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0)
(1.2.2)
Requirement already satisfied: joblib>=0.14 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0) (1.2.0)
Requirement already satisfied: decorator>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0) (4.4.2)
Collecting resampy>=0.2.2 (from librosa==0.8.0)
  Downloading resampy-0.4.2-py3-none-any.whl (3.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.1/3.1 MB 79.6 MB/s eta 0:00:00
Requirement already satisfied: numba>=0.43.0 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0) (0.56.4)
Requirement already satisfied: soundfile>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0) (0.12.1)
Requirement already satisfied: pooch>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa==0.8.0) (1.6.0)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.43.0->libro
sa==0.8.0) (0.39.1)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from numba>=0.43.0->librosa==0.8.0) (67.
7.2)
Requirement already satisfied: appdirs>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.0->librosa==0.8.0) (1.
4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.0->librosa==0.8.0) (2
3.1)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.0->librosa==0.8.0)
(2.27.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn!=0.19.0,>=0.1
4.0->librosa==0.8.0) (3.1.0)
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from soundfile>=0.9.0->librosa==0.8.0) (1.
15.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0->soundfile>=0.9.0->librosa=
=0.8.0) (2.21)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=
1.0->librosa==0.8.0) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0
->librosa==0.8.0) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->poo
ch>=1.0->librosa==0.8.0) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->libr
osa==0.8.0) (3.4)
Building wheels for collected packages: librosa
  Building wheel for librosa (setup.py) ... done
  Created wheel for librosa: filename=librosa-0.8.0-py3-none-any.whl size=201393 sha256=fca7d6d2f115b5cb8a3f88831d886627937f48d
eec9405eddbf0cacc158bfb31
  Stored in directory: /root/.cache/pip/wheels/bf/b7/85/2f8044306ccec014930aea23ad4852fca9e2584e21c6972bc6
Successfully built librosa
Installing collected packages: resampy, librosa
Successfully installed librosa-0.8.0 resampy-0.4.2
```

In [17]:
```
import librosa, librosa.display
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import IPython.display as ipd
%matplotlib inline
```

In [18]:
```
signal, sr = librosa.load('hoosh.wav', sr=44100)
```

In [19]:
```
# librosa.display.waveshow(signal, sr = sr)
librosa.display.waveplot(signal, sr = sr)
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.show()
```



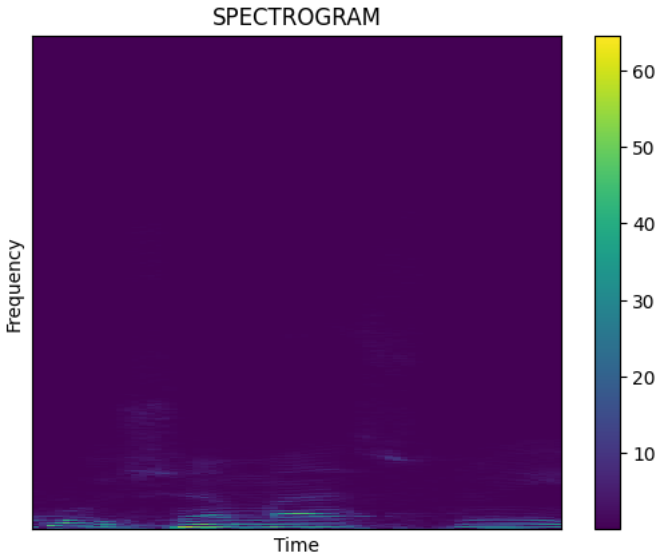
In [20]:
```
n_fft = 2048
hop_length = 512 # How much we have to shift
stft = librosa.core.stft(signal, hop_length = hop_length, n_fft = n_fft)
spectrogram = np.abs(stft)

log_spectrogram  = librosa.amplitude_to_db(spectrogram)
```

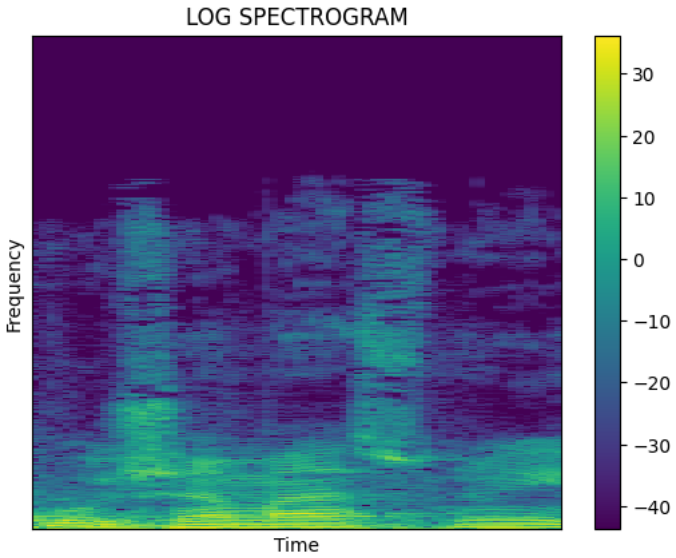
dB to ratio conversion table

dB	Amplitude ratio	Power ratio
1 dB	1.122	1.259
2 dB	1.259	1.585
3 dB	1.413	2 ≈ 1.995
6 dB	2 ≈ 1.995	3.981

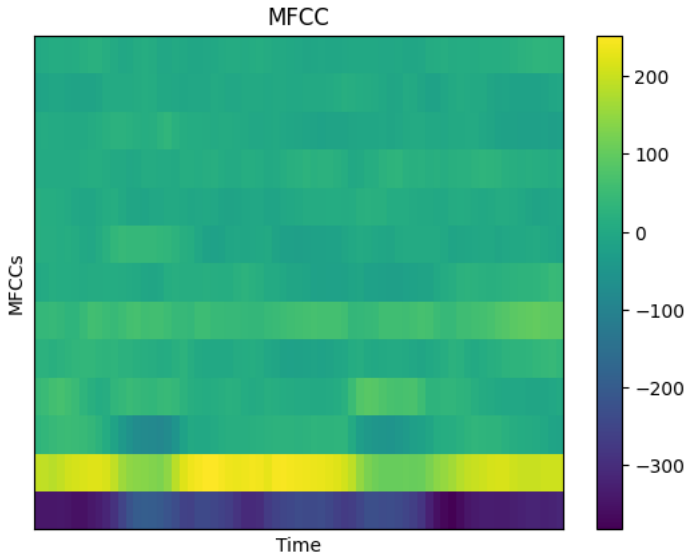
In [21]:
```
# SPECTROGRAM
librosa.display.specshow(spectrogram, sr = sr, hop_length = hop_length)
plt.xlabel("Time")
plt.ylabel("Frequency")
plt.colorbar()
plt.set_cmap("viridis")
plt.title('SPECTROGRAM')
plt.show()
```



In [22]:
```
# LOG SPECTROGRAM
librosa.display.specshow(log_spectrogram, sr = sr, hop_length = hop_length)
plt.xlabel("Time")
plt.ylabel("Frequency")
plt.set_cmap("viridis")
plt.colorbar()
plt.title('LOG SPECTROGRAM')
plt.show()
```



In [23]:
```
#MFCCs
MFCCs = librosa.feature.mfcc(signal, n_fft = n_fft, hop_length = hop_length, n_mfcc = 13)
librosa.display.specshow(MFCCs, sr = sr, hop_length = hop_length)
plt.xlabel("Time")
plt.ylabel("MFCCs")
plt.set_cmap("viridis")
plt.title('MFCC')
plt.colorbar()
plt.show()
```



In [24]:
```
spectrogram
```

Out[24]:
```
array([[1.0964104e+01, 1.2548919e+01, 4.3708107e-01, ..., 6.7684836e+00,
        4.5142803e+00, 7.2206540e+00],
       [3.6564114e+00, 7.9565511e+00, 1.5980395e+01, ..., 9.7252216e+00,
        5.7103858e+00, 3.1406274e+00],
       [6.7267284e+00, 2.8948438e+00, 1.5272521e+01, ..., 5.7925563e+00,
        2.1918626e+00, 2.5763698e+00],
       ...,
       [3.6083888e-03, 1.8274670e-03, 1.0527653e-04, ..., 8.6743414e-05,
        3.7803973e-05, 4.3897130e-04],
       [3.5854257e-03, 1.8334051e-03, 6.7456509e-05, ..., 1.8692644e-04,
        1.4025108e-04, 5.6835782e-04],
       [3.5054986e-03, 1.6777616e-03, 9.6358890e-05, ..., 2.1495954e-04,
        1.5598827e-04, 4.9479160e-04]], dtype=float32)
```

In [25]:
```
spectrogram.shape
```

Out[25]:
```
(1025, 69)
```

In [26]:
```
log_spectrogram.shape
```

Out[26]:
```
(1025, 69)
```

In [28]:
```
log_spectrogram.max()
```

Out[28]:
```
36.19528
```

In [29]:
```
log_spectrogram.min()
```

Out[29]:
```
-43.80472
```

In [30]:
```
spectrogram.max()
```

Out[30]:
```
64.53034
```

In [31]:
```
spectrogram.min()
```

Out[31]:
```
3.9766775e-08
```

In [32]:
```
1 MFCCs.shape
```

Out[32]: (13, 69)

In [33]:
```
1 MFCCs.max()
```

Out[33]: 251.96802

In [34]:
```
1 MFCCs.min()
```

Out[34]: -382.60916

In [35]:
```
1 MFCCs.shape
```

Out[35]: (13, 69)

In [36]:
```
1 spectrogram.shape
```

Out[36]: (1025, 69)

In [37]:
```
1 signal.shape
```

Out[37]: (35280,)

In [38]:
```
1 spectrogram.shape[0]*spectrogram.shape[1]
```

Out[38]: 70725

In [40]:
```
1 signal.shape[0]/(spectrogram.shape[0]*spectrogram.shape[1])
```

Out[40]: 0.49883351007423116

In [41]:
```
1 signal.shape[0]/(MFCCs.shape[0]*MFCCs.shape[1])
```

Out[41]: 39.331103678929765

In [ ]:
```
1
```