Jason C. Casas
Student, PHD-CS
1st Semester, A.Y. 2023 – 2024

Mr. Larry Vea, PhD
Professor
PHDCS 105 – Affective Computing

Feature Extraction from Audio/Sound:

Code Snippets:

In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import librosa
audio_path = 'download.wav'
x , sr = librosa.load(audio_path)
```

Playing Audio Using IPython.display.Audio, to play the audi

In [3]:
```
import IPython.display as ipd
ipd.Audio(audio_path)
```

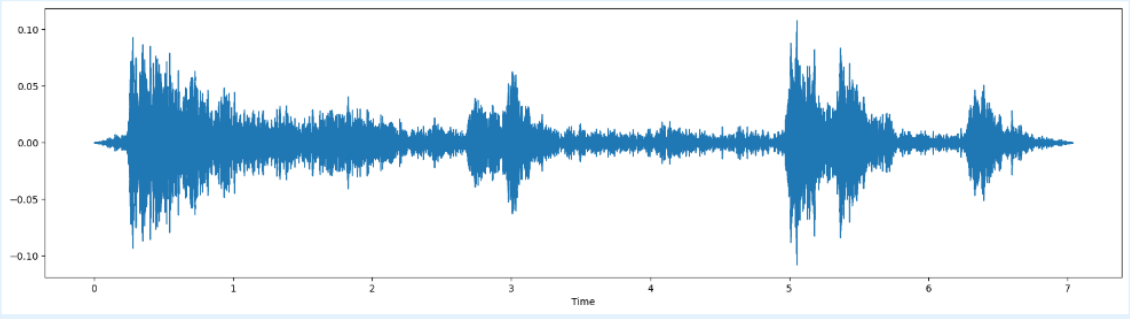
Out[3]:

Visualizing Audio

```
# Waveform
# We can plot the audio array using librosa.display.waveplot:
```

In [9]:
```
%matplotlib inline
import sklearn
import matplotlib.pyplot as plt
import librosa.display

plt.figure(figsize=(20, 5))
librosa.display.waveshow(x, sr=sr) ## instead of waveplot use waveshow as parameter
```

Out[9]: <librosa.display.AdaptiveWaveplot at 0x1d3f2e992d0>



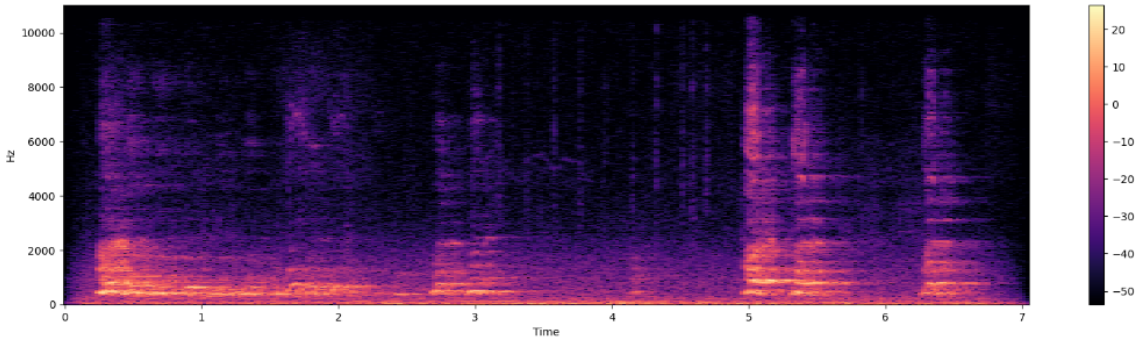
Jason C. Casas
Student, PHD-CS
1st Semester, A.Y. 2023 – 2024

Mr. Larry Vea, PhD
Professor
PHDCS 105 – Affective Computing

Spectrogram: display a spectrogram using librosa.display.specshow

In [10]:
```
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(20, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

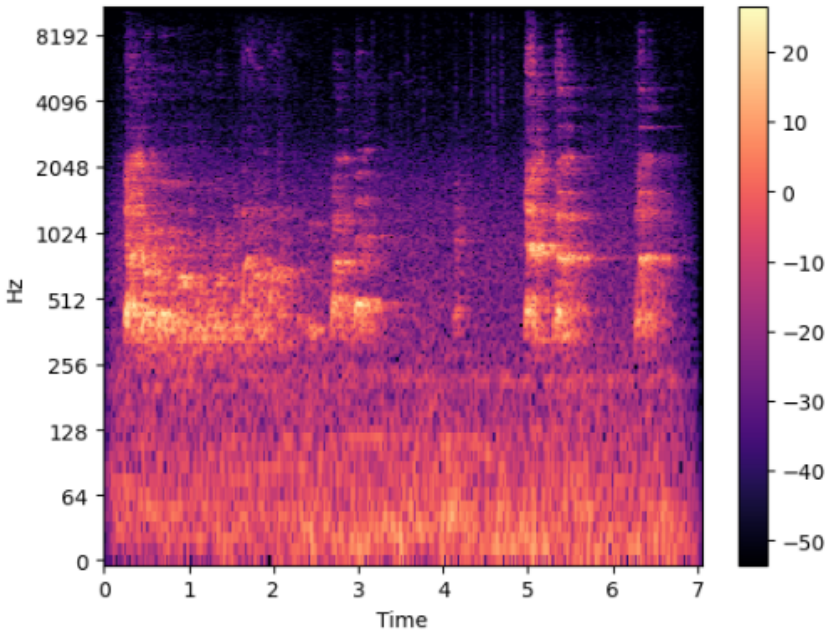
Out[10]: <matplotlib.colorbar.Colorbar at 0x1d3f2a3e8d0>



Log Frequency axis

In [11]:
```
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
plt.colorbar()
```

Out[11]: <matplotlib.colorbar.Colorbar at 0x1d3f51d6590>



Jason C. Casas
Student, PHD-CS
1st Semester, A.Y. 2023 – 2024

Mr. Larry Vea, PhD
Professor
PHDCS 105 – Affective Computing

Creating an audio signal

In [12]:
```
# create an audio signal at 220Hz. We know an audio signal is a numpy array,
# so we shall create one and pass it on to the audio function

import numpy as np
sr = 22050 # sample rate
T = 5.0    # seconds
t = np.linspace(0, T, int(T*sr), endpoint=False) # time variable
x = 0.5*np.sin(2*np.pi*220*t)# pure sine wave at 220 Hz
```

Playing the sound

In [13]:
```
ipd.Audio(x, rate=sr) # load a NumPy array
```

Out[13]:

Saving the signal

In [18]:
```
# With modern librosa, you should instead use soundfile.write to write audio output.
#librosa.output.write_wav('tone_440.wav', x, sr) # writing wave file in tone440.wav format


import soundfile as sf
sf.write('stereo_file1.wav', x, sr,  'PCM_24')
```

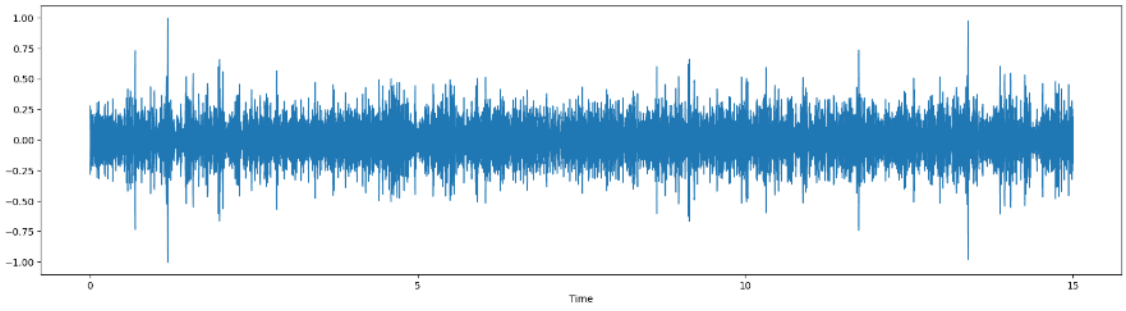
Feature Extraction

In [21]:
```
x, sr = librosa.load('download2.wav')
ipd.Audio(x, rate=sr)
```

Out[21]:

In [23]:
```
#Plot the signal:
plt.figure(figsize=(20, 5))
librosa.display.waveshow(x, sr=sr)  #instead of waveplot use waveshow as parameter
```

Out[23]:
```
<librosa.display.AdaptiveWaveplot at 0x1d3f5b69c10>
```

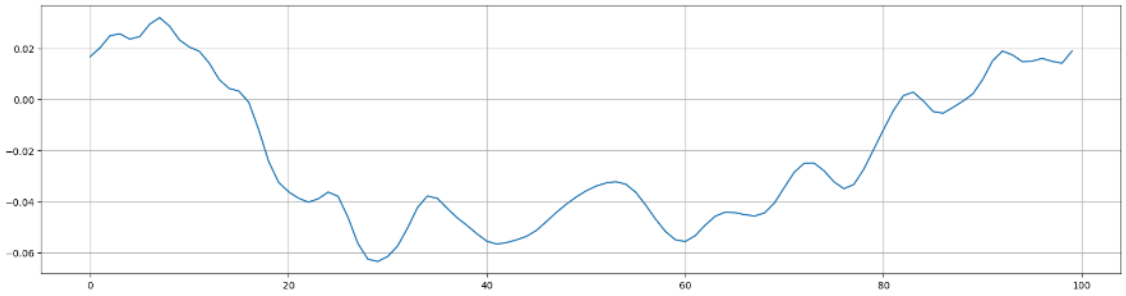


Jason C. Casas
Student, PHD-CS
1st Semester, A.Y. 2023 – 2024

Mr. Larry Vea, PhD
Professor
PHDCS 105 – Affective Computing

1. Zero Crossing Rate

In [24]:
```
# Zooming in
n0 = 9000
n1 = 9100
plt.figure(figsize=(20, 5))
plt.plot(x[n0:n1])
plt.grid()
```



In [25]:
```
# I count 6 zero crossings. Let's compute the zero crossings using librosa.

zero_crossings = librosa.zero_crossings(x[n0:n1], pad=False)
zero_crossings.shape
```

Out[25]: (100,)

In [26]:
```
print(sum(zero_crossings))
```

```
4
```

2.Spectral Centroid

In [28]:
```
spectral_centroids = librosa.feature.spectral_centroid(y=x, sr=sr)[0]  # include y as argument and input
spectral_centroids.shape
```

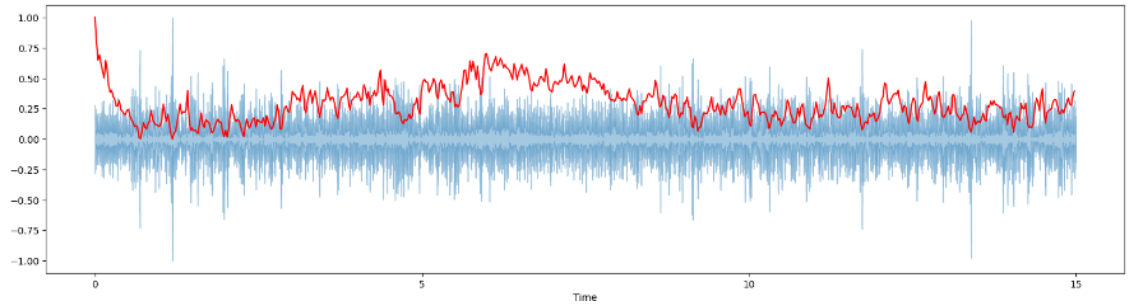
Out[28]: (646,)

In [46]:
```
from sklearn import preprocessing ## add this line



# Computing the time variable for visualization
plt.figure(figsize=(20,5))
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)

# Normalising the spectral centroid for visualisation
def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)

#Plotting the Spectral Centroid along the waveform
librosa.display.waveshow(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='r')
```

Jason C. Casas
Student, PHD-CS
1st Semester, A.Y. 2023 – 2024

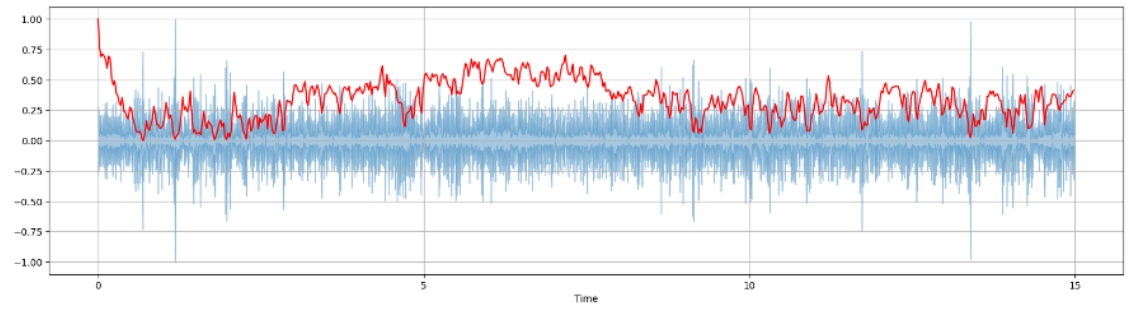
Mr. Larry Vea, PhD
Professor
PHDCS 105 – Affective Computing

Out[46]:
```
[<matplotlib.lines.Line2D at 0x1d38214be90>]
```



3.Spectral Rolloff

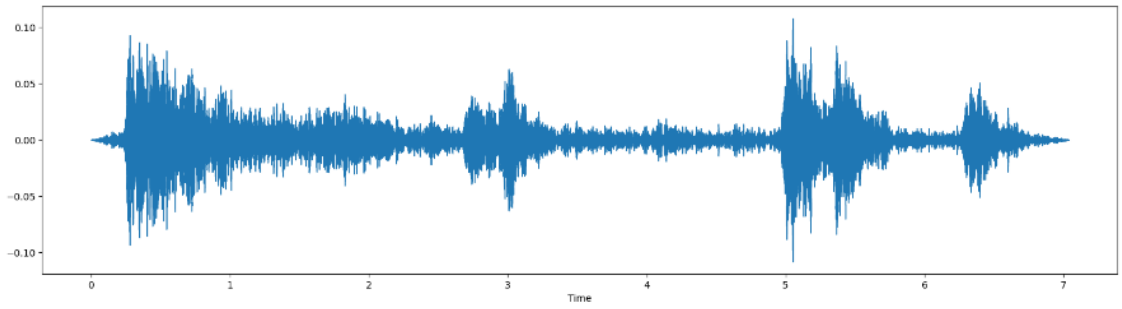
In [55]:
```
plt.figure(figsize=(20,5))
spectral_rolloff = librosa.feature.spectral_rolloff(y=x + 0.01, sr=sr)[0]  ## y=x argument here
librosa.display.waveshow(y=x, sr=sr, alpha=0.4)    ## y=x argument here
plt.plot(t, normalize(spectral_rolloff), color='r')
plt.grid()
```



4.MFCC

In [56]:
```
plt.figure(figsize=(20,5))
x, fs = librosa.load('download.wav')
librosa.display.waveshow(x, sr=sr)
```

Out[56]:
```
<librosa.display.AdaptiveWaveplot at 0x1d3f907a790>
```



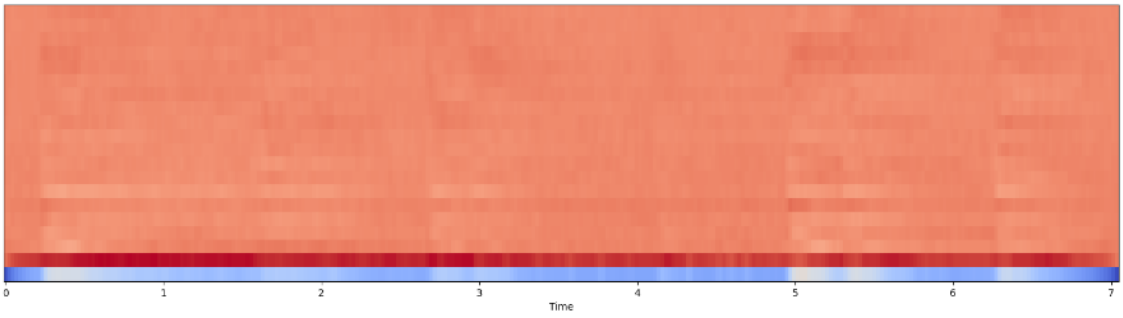
Jason C. Casas
Student, PHD-CS
1st Semester, A.Y. 2023 – 2024

Mr. Larry Vea, PhD
Professor
PHDCS 105 – Affective Computing

In [58]:
```
# MFCC
plt.figure(figsize=(20,5))
mfccs = librosa.feature.mfcc(y=x, sr=sr)
print(mfccs.shape)

librosa.display.specshow(mfccs, sr=sr, x_axis='time')
```

```
(20, 304)
```

Out[58]:
```
<matplotlib.collections.QuadMesh at 0x1d38fab6010>
```



Feature Scaling

In [59]:
```
# Let's scale the MFCCs such that each coefficient dimension has zero mean and unit variance:

mfccs = sklearn.preprocessing.scale(mfccs, axis=1)
print(mfccs.mean(axis=1))
print(mfccs.var(axis=1))
```

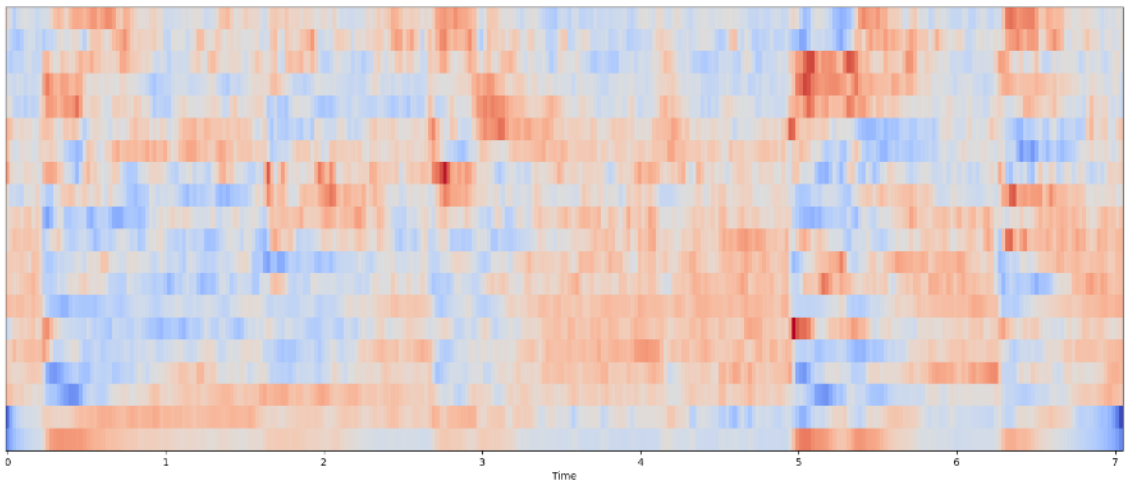
```
[ 1.2548346e-08  5.0193385e-08  3.7645037e-08  0.0000000e+00
  1.2548346e-08  2.5096693e-08  0.0000000e+00 -2.5096693e-08
  2.5096693e-08  0.0000000e+00  1.2548346e-08 -6.2741732e-09
  1.2548346e-08 -2.5096693e-08  0.0000000e+00  3.1370866e-09
  0.0000000e+00 -7.8427167e-09  0.0000000e+00  0.0000000e+00]
[1.        1.        1.        1.        1.        0.9999999 1.
 1.        1.0000001 1.        1.        1.        1.0000001 1.0000001
 1.        1.        1.        1.0000001 1.0000001 1.0000001]
```

In [60]:
```
plt.figure(figsize=(20,8))
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
```

Jason C. Casas
Student, PHD-CS
1st Semester, A.Y. 2023 – 2024

Mr. Larry Vea, PhD
Professor
PHDCS 105 – Affective Computing

Out[60]:
```
<matplotlib.collections.QuadMesh at 0x1d3900d2710>
```



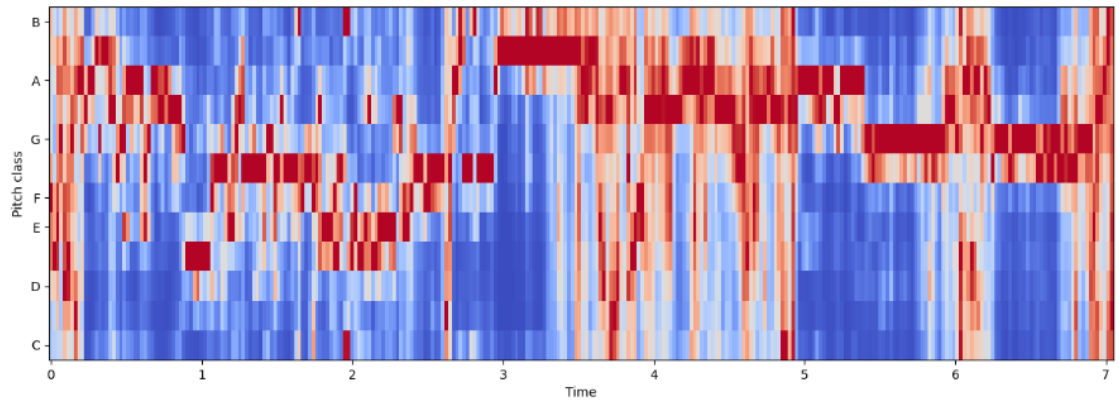
Chroma Frequencies

In [61]:
```
# Loadign the file
x, sr = librosa.load('download.wav')
ipd.Audio(x, rate=sr)
```

Out[61]:

In [63]:
```
hop_length = 512
chromagram = librosa.feature.chroma_stft(y=x, sr=sr, hop_length=hop_length) ## y=x argument
plt.figure(figsize=(15, 5))
librosa.display.specshow(chromagram, x_axis='time', y_axis='chroma', hop_length=hop_length, cmap='coolwarm')
```

Out[63]:
```
<matplotlib.collections.QuadMesh at 0x1d390129dd0>
```



reference:
https://www.kaggle.com/code/ashishpatel26/feature-extraction-from-audio/notebook