

# CMPE-256 ADVANCED DATA MINING

## Assignment 2 : Feature Engineering Unstructured Data (Audio & Wave Files)

**Learning Objective:** The purpose of the assignment is to derive feature engineering spectrograms of wave files:

- Display a spectrogram/chromagram/cqt/etc.
- Compute a mel-scaled spectrogram.
- Short-time Fourier transform (STFT).
- Beat tracker
- Compute the constant-Q transform of an audio signal.

Please re-use in class demo files on wave features as a reference!

Please extract following features of the Car NASA

- Kepler: Star KIC12268220C Light Curve Waves to Sound
- NASA - Kepler: Star KIC7671081B Light Curve Waves to
- NASA - Whistler Waves
- Johns Hopkins APL - Parker Solar Probe - Whistler Mode Waves 2
- NASA - Audio from NASA's Juno Mission: Europa Flyby

### Solution:

**Feature Engineering:** Feature engineering involves transforming raw data (in this case, audio waveforms) into features that can be used for machine learning models or further analysis. When working with audio data, one powerful way to extract meaningful information is through spectrograms.

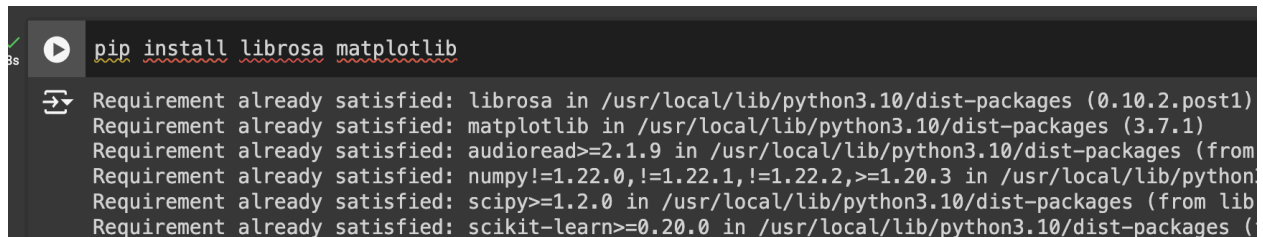
**Spectrograms** are visual representations of the spectrum of frequencies of a signal as it varies with time. They are a key tool in audio processing because they allow you to visualize how sound frequencies evolve over time, which is essential for understanding the structure of an audio signal.

In this assignment, I'm using Kepler: Star KIC12268220C Light Curve Waves to Sound audio file.

Install matplotlib and librosa libraries

**Matplotlib** - Matplotlib is a powerful Python library used for creating static, interactive, and animated visualizations in Python. It is particularly useful for plotting data in various forms (e.g., line plots, bar charts, scatter plots, histograms)

**Librosa** - It is used for audio and music analysis. It provides tools for loading audio files, extracting features, and performing transformations like spectrograms, mel-scaled spectrograms, Short-Time Fourier Transform (STFT), and more. It's widely used for tasks in music information retrieval, speech recognition, and general audio analysis.



```
pip install librosa matplotlib
Requirement already satisfied: librosa in /usr/local/lib/python3.10/dist-packages (0.10.2.post1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/python3.10/dist-packages (from librosa)
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20.3 in /usr/local/lib/python3.10/dist-packages (from librosa)
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from librosa)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from librosa)
```

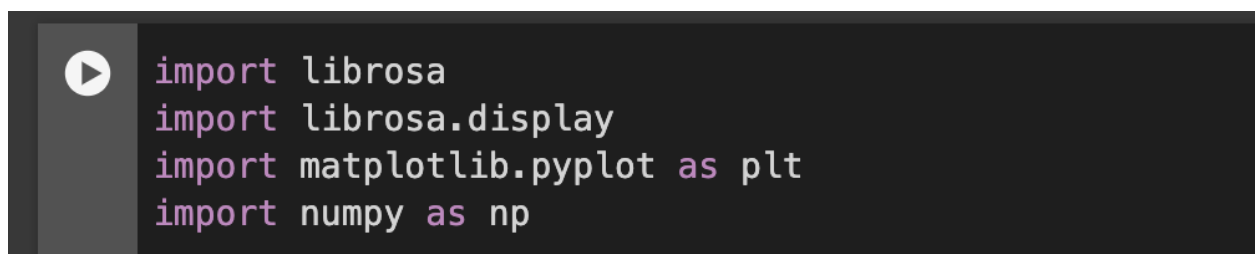
### Importing the libraries:

**librosa**: A Python library for analyzing and processing audio files. It is commonly used for tasks like audio analysis, feature extraction, and visualizations.

**librosa.display**: This module provides utilities for displaying waveforms, spectrograms, and other visual representations of audio data.

**matplotlib.pyplot**: This is a plotting library, used here to visualize the audio data (e.g., to plot waveforms).

**numpy**: A fundamental package for numerical computing in Python. It's commonly used for working with arrays, performing mathematical operations, and data manipulation.



```
import librosa
import librosa.display
import matplotlib.pyplot as plt
import numpy as np
```

## Load the audio file

```
a='/content/578358main_kepler_star_KIC12268220C.mp3'

y,sr=librosa.load(a)
print(f"Sample Rate: {sr} Hz")
print(f"Audio Length: {len(y)} samples")
```

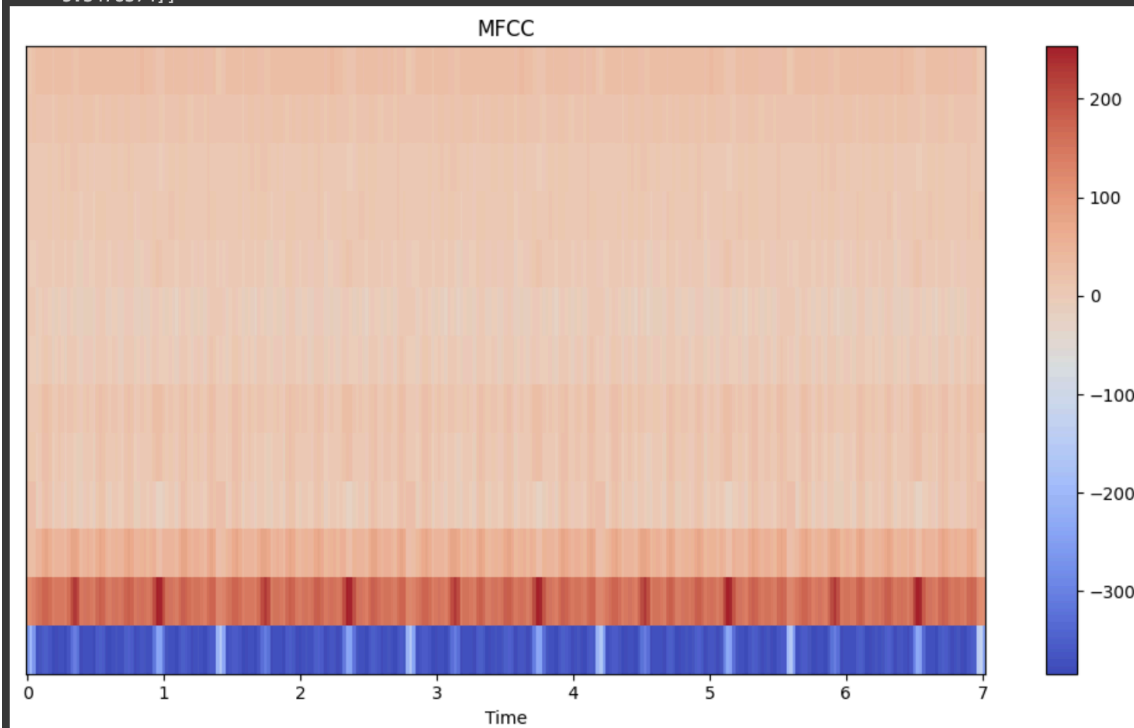
## Extract and display Mel Frequency Cepstral Coefficients

MFCC - Short term power spectrum of audio signals

```
mfcc=librosa.feature.mfcc(y=y,sr=sr,n_mfcc=13)
print(mfcc)

plt.figure(figsize=(10,6))
librosa.display.specshow(mfcc,sr=sr,x_axis='time')
plt.colorbar()
plt.title('MFCC')
plt.tight_layout()
plt.show()
```

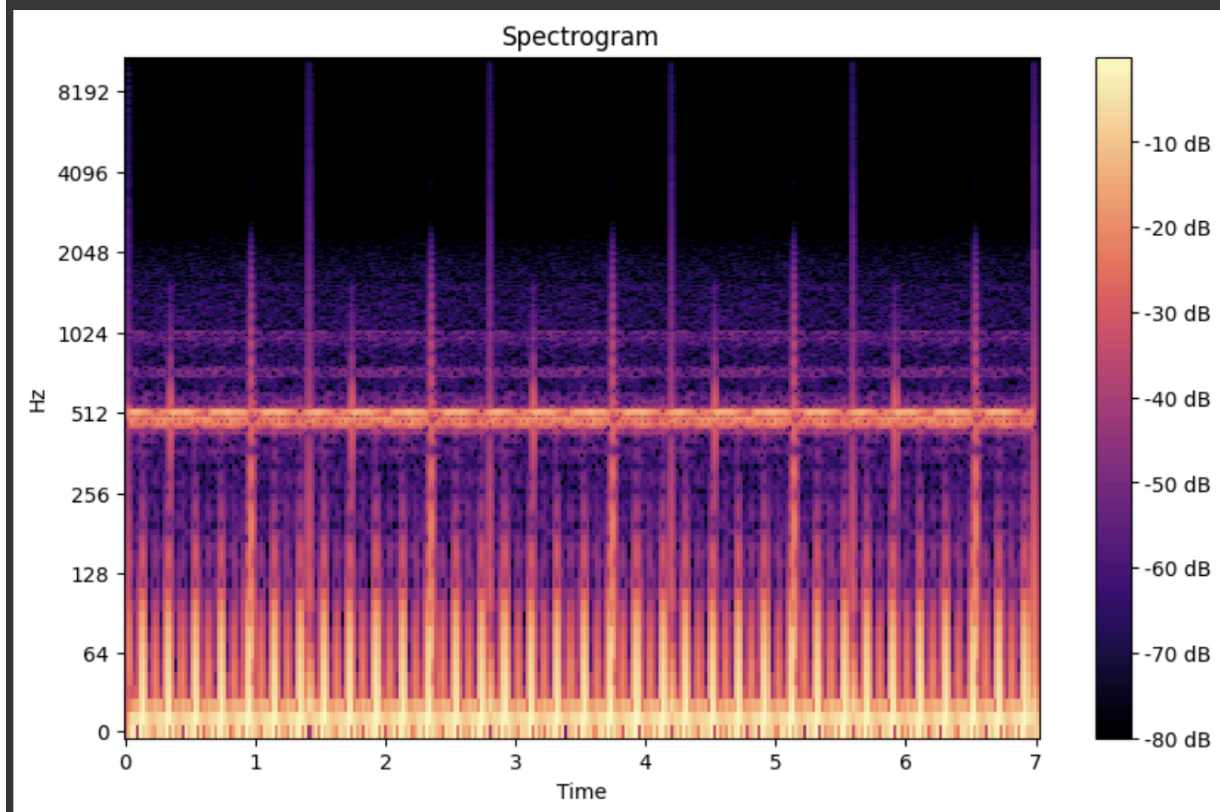
```
[[-262.36493  -200.15802  -265.783   ... -148.55527  -122.4602
  -240.7314   ]
 [ 112.71478  120.50728  132.74704   ...  121.4913   116.59863
  111.42892   ]
 [  26.88695   28.958977  33.986534   ...  23.877289   20.78935
  18.07006   ]
 ...
 [  6.9142     6.139324   5.9285846   ...   2.3152978   3.3653848
  4.5064073   ]
 [ 13.4522705  16.300701  17.38763   ...   9.078508   8.105353
  6.195675   ]
 [  8.349279   11.5777645  18.30165   ...  11.050024   8.421319
  5.3478374   ]]
```



## Compute and display spectrogram

Spectrogram shows amplitude of different frequencies over time.

```
D = librosa.amplitude_to_db(np.abs(librosa.stft(y)), ref=np.max)
plt.figure(figsize=(10, 6))
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='log')
plt.colorbar(format='%+2.0f dB')
plt.title('Spectrogram')
plt.show()
```

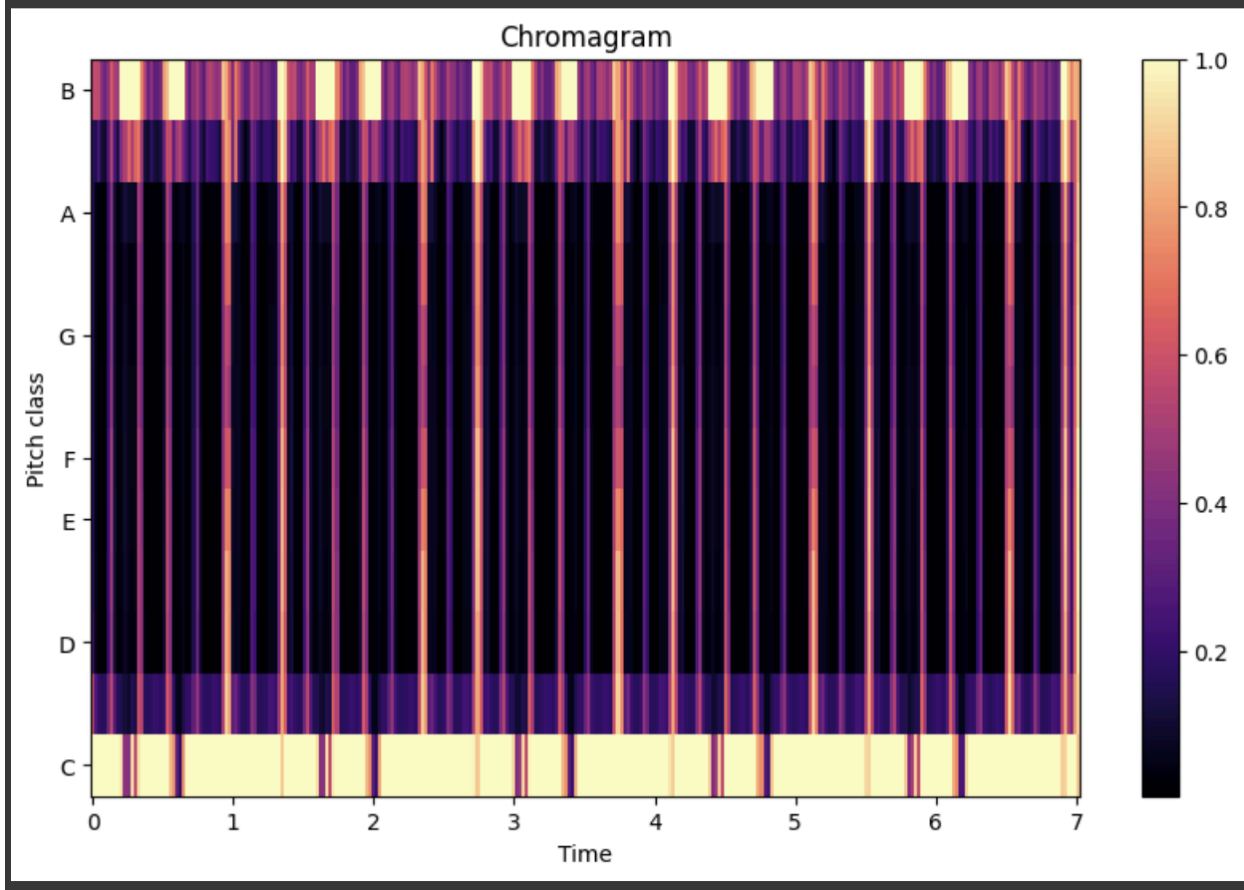


## Compute and Display Chromagram

It shows how much energy is present in each pitch class at any given time.

```
chroma = librosa.feature.chroma_stft(y=y, sr=sr)

plt.figure(figsize=(10, 6))
librosa.display.specshow(chroma, y_axis='chroma', x_axis='time', sr=sr)
plt.colorbar()
plt.title('Chromagram')
plt.show()
```



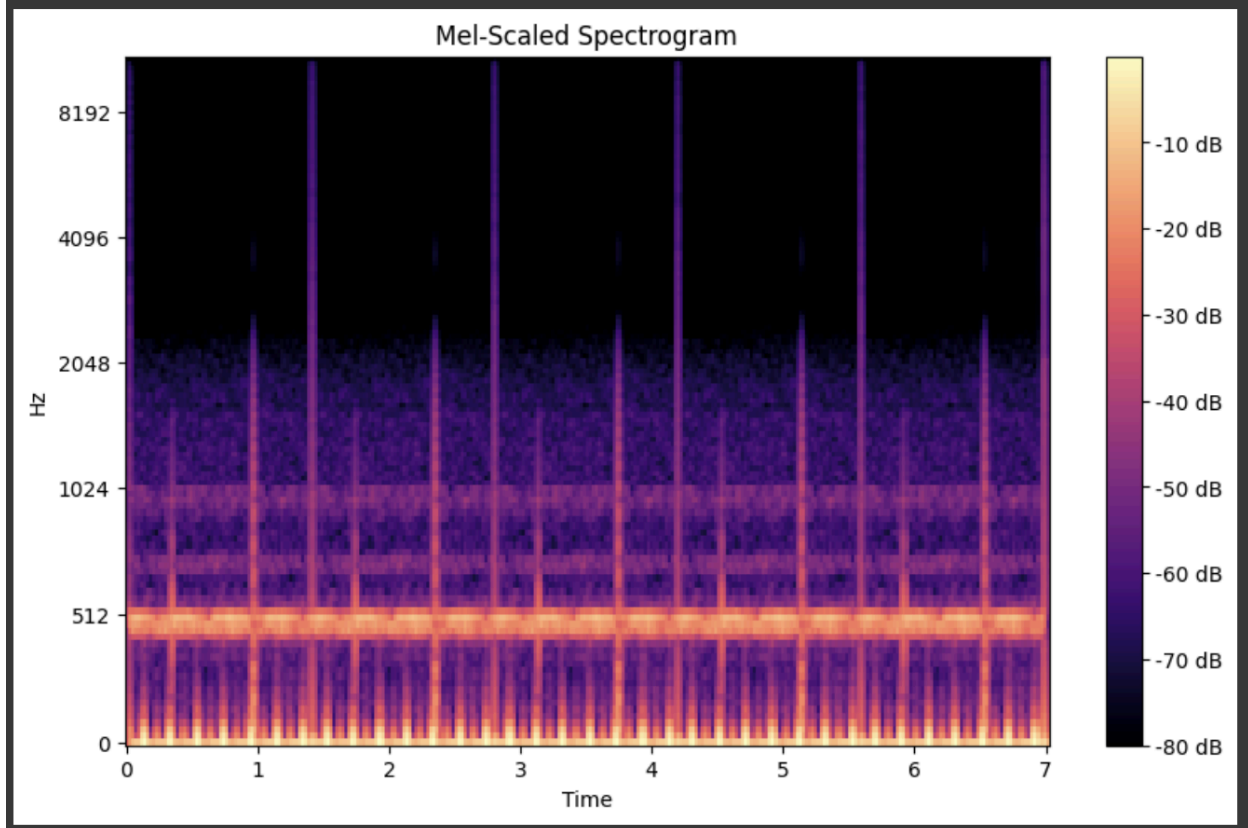
## Compute and Display Mel-Scaled Spectrogram

A mel-spectrogram is a spectrogram where the frequency axis is converted to the mel scale (a scale based on how humans perceive pitch).

```
mel_spectrogram = librosa.feature.melspectrogram(y=y, sr=sr)

mel_spectrogram_db = librosa.power_to_db(mel_spectrogram, ref=np.max)

plt.figure(figsize=(10, 6))
librosa.display.specshow(mel_spectrogram_db, sr=sr, x_axis='time', y_axis='mel')
plt.colorbar(format='%+2.0f dB')
plt.title('Mel-Scaled Spectrogram')
plt.show()
```



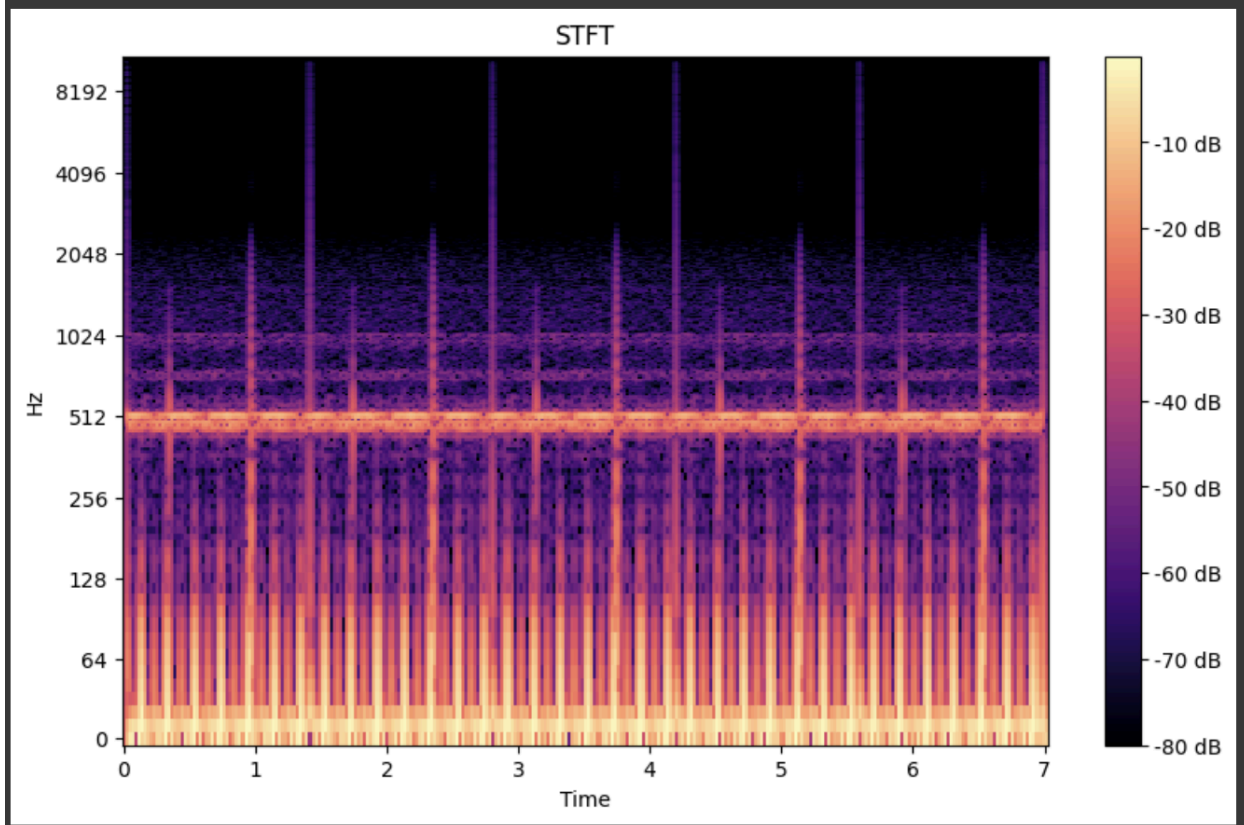
## Compute Short-Time Fourier Transform

STFT divides the audio signal into short overlapping segments and applies a Fourier Transform to each segment.

```
stft_result = librosa.stft(y)

stft_db = librosa.amplitude_to_db(np.abs(stft_result), ref=np.max)

plt.figure(figsize=(10, 6))
librosa.display.specshow(stft_db, sr=sr, x_axis='time', y_axis='log')
plt.colorbar(format='%+2.0f dB')
plt.title('STFT')
plt.show()
```





## Compute Beat Tracking

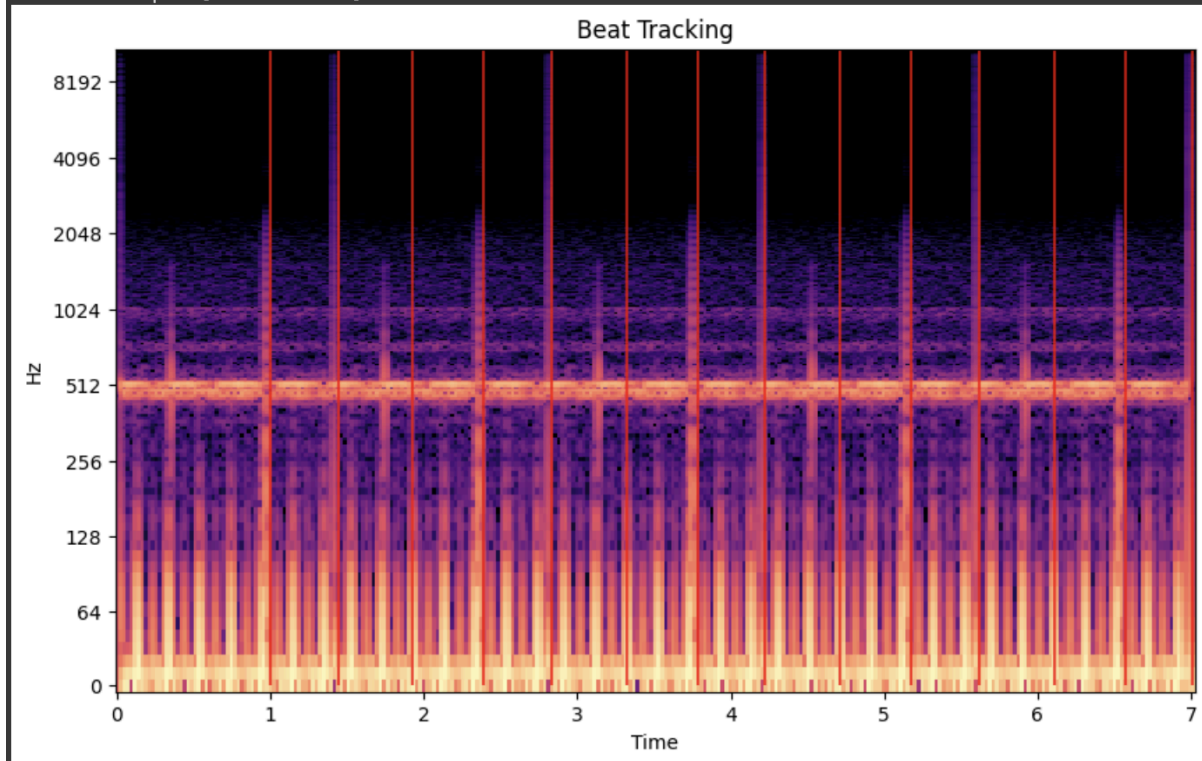
Beat tracking identifies the locations of rhythmic beats in the audio.

```
tempo, beats = librosa.beat.beat_track(y=y, sr=sr)

print(f"Detected Tempo: {tempo} BPM")

plt.figure(figsize=(10, 6))
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='log')
plt.vlines(librosa.frames_to_time(beats), 0, sr / 2, color='r', alpha=0.75)
plt.title('Beat Tracking')
plt.show()
```

Detected Tempo: [135.99917763] BPM



## Compute Constant-Q Transform (CQT)

CQT provides a frequency analysis with logarithmically spaced frequency bins, which are more appropriate for analyzing musical signals.

```
CQT = librosa.cqt(y, sr=sr)

CQT_db = librosa.amplitude_to_db(np.abs(CQT), ref=np.max)

plt.figure(figsize=(10, 6))
librosa.display.specshow(CQT_db, sr=sr, x_axis='time', y_axis='cqt_note')
plt.colorbar(format='%+2.0f dB')
plt.title('Constant-Q Transform (CQT)')
plt.show()
```

