Group 7 - Pepito, Alyssa Mae & Soleño, Keziah Antonette 06-14-2022

CS171 | BM1

Assignment 4 (M1)

```
[94] # Loading an audio file
     import warnings
     warnings.filterwarnings('ignore')

     import librosa
```

```
[3] from google.colab import files
    uploaded = files.upload()
```

Browse... No files selected. Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

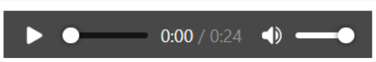
```
Saving genres.zip to genres.zip
```

```
[73] !unzip genres.zip
```

```
Archive:  genres.zip
replace genres/pop/only_the_young_chorus.wav? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace genres/rock/dont_stop_me_now_chorus.wav? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
```

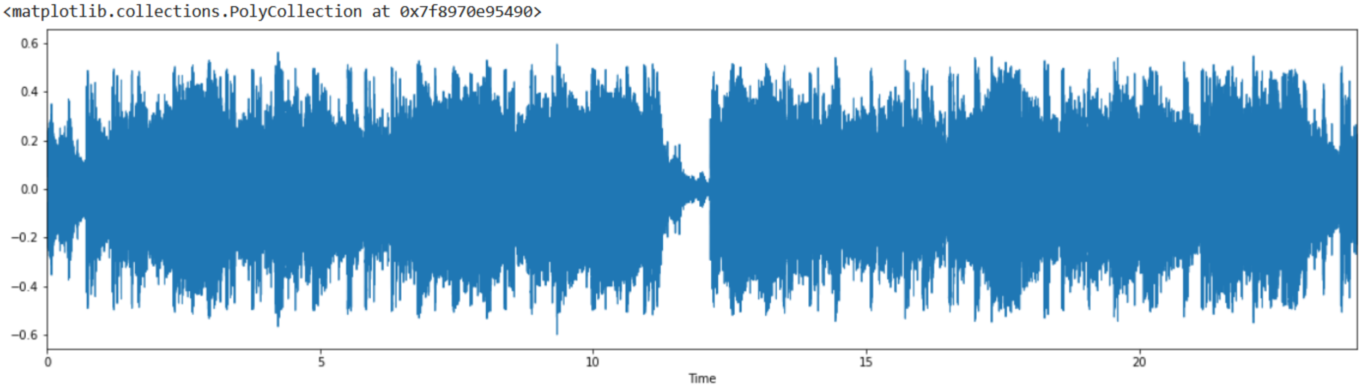
```
[155] ##############################################################################################################################
      # First wav music file
      audio_path = 'genres/pop/only_the_young_chorus.wav'
      x , sr = librosa.load(audio_path)
```

```
[98] # Playing Audio
     # Using IPython.display.Audio, to play the audio
     import IPython.display as ipd
     ipd.Audio(audio_path)
```



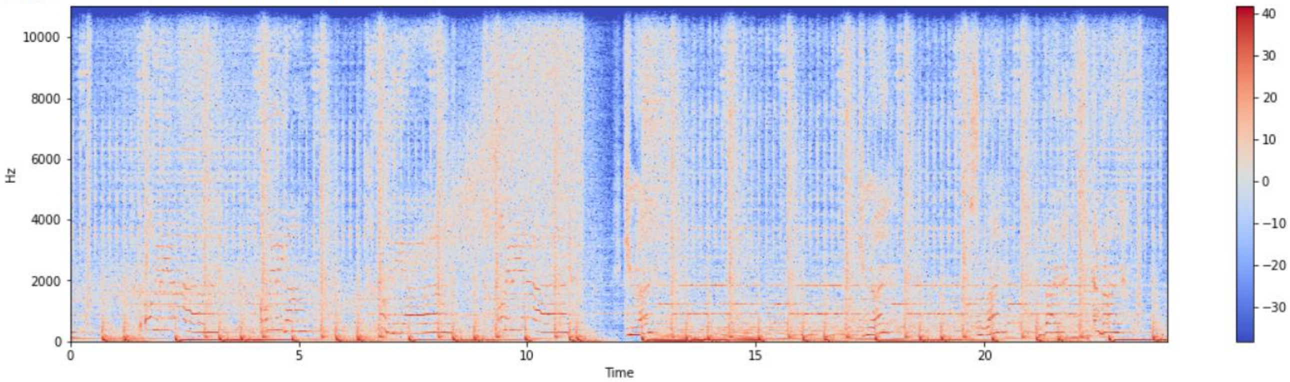

```
[99] # Visualizing Audio
     ## 1. Waveform ##
     %matplotlib inline
     import sklearn
     import matplotlib.pyplot as plt
     import librosa.display

     plt.figure(figsize=(20, 5))
     librosa.display.waveplot(x, sr=sr)
```

```
<matplotlib.collections.PolyCollection at 0x7f8970e95490>
```



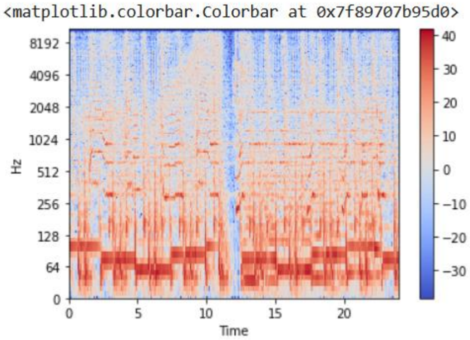
[100]
```python
## 2. Spectogram ##
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(20, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7f897036cf10>
```

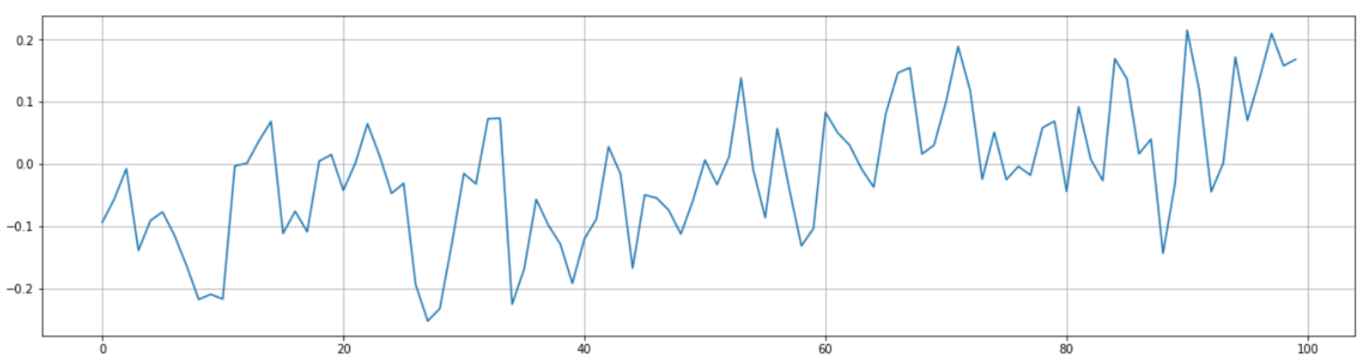


[101]
```python
## 3. Log Frequency axis ##
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7f89707b95d0>
```



[136]
```python
# Feature Extraction
## 1. Zero Crossing Rate ##
# Zooming in
# Here we will zoom or print spectrum for 100 array columns only.
n0 = 9000
n1 = 9100
plt.figure(figsize=(20, 5))
plt.plot(x[n0:n1])
plt.grid()
```



[141]
```
zero_crossings = librosa.zero_crossings(x[n0:n1], pad=False)
zero_crossings.shape
```

```
(100,)
```

[142]
```
# We can also calculate zero crossings using a given code:
print(sum(zero_crossings))
```

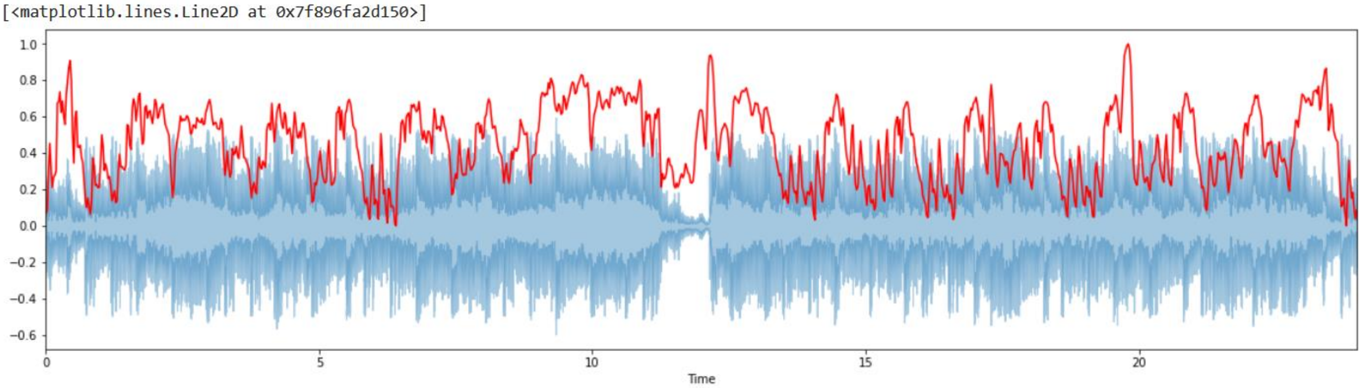
```
31
```

[145]
```
## 2. Spectral Centroid ##
#spectral centroid -- centre of mass -- weighted mean of the frequencies present in the sound
spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
spectral_centroids.shape
```

```
(1034,)
```

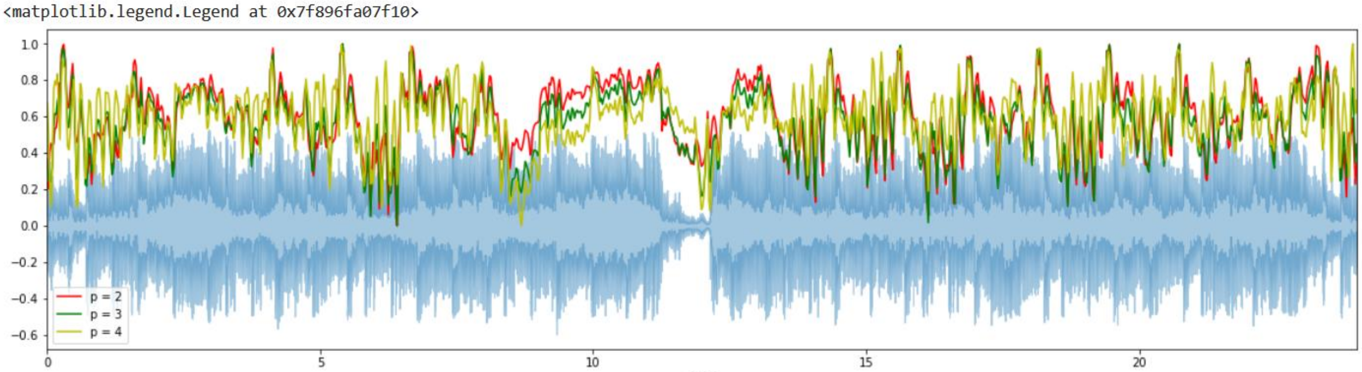
[146]
```
# Computing the time variable for visualization
plt.figure(figsize=(20,5))
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)

# 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.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='r')
```

```
[<matplotlib.lines.Line2D at 0x7f896fa2d150>]
```

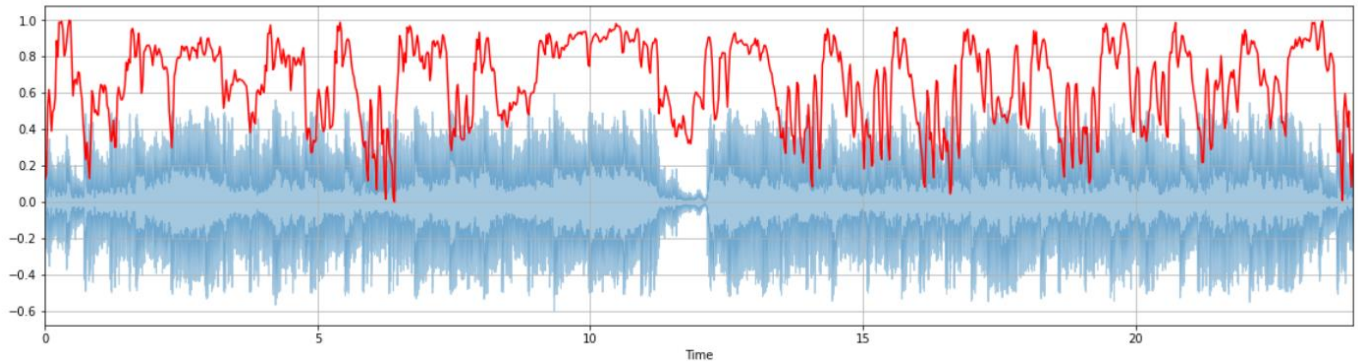


[147]
```
## 3. Spectral Bandwidth ##
plt.figure(figsize=(20,5))
spectral_bandwidth_2 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr)[0]
spectral_bandwidth_3 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr, p=3)[0]
spectral_bandwidth_4 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr, p=4)[0]
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_bandwidth_2), color='r')
plt.plot(t, normalize(spectral_bandwidth_3), color='g')
plt.plot(t, normalize(spectral_bandwidth_4), color='y')
plt.legend(('p = 2', 'p = 3', 'p = 4'))
```

```
<matplotlib.legend.Legend at 0x7f896fa07f10>
```

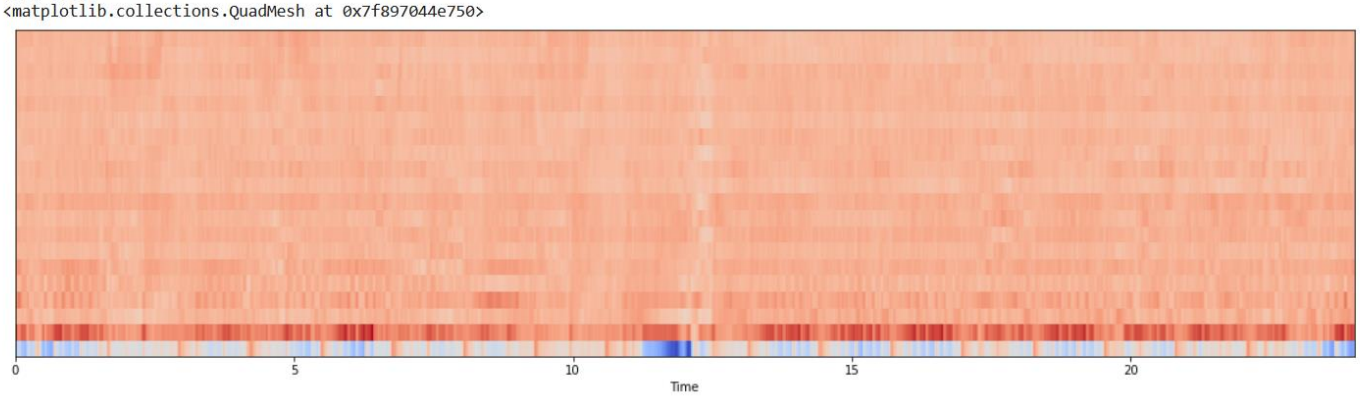


```
[149] ## 4. Spectral Rolloff ##
      plt.figure(figsize=(20,5))
      spectral_rolloff = librosa.feature.spectral_rolloff(x+0.01, sr=sr)[0]
      librosa.display.waveplot(x, sr=sr, alpha=0.4)
      plt.plot(t, normalize(spectral_rolloff), color='r')
      plt.grid()
```




```
[150] ## 5. Mel-Frequency Cepstral Coefficients ##
      plt.figure(figsize=(20,5))
      mfccs = librosa.feature.mfcc(x, sr=sr)
      print(mfccs.shape)

      #Displaying  the MFCCs:
      librosa.display.specshow(mfccs, sr=sr, x_axis='time')
```

```
(20, 1034)
<matplotlib.collections.QuadMesh at 0x7f897044e750>
```

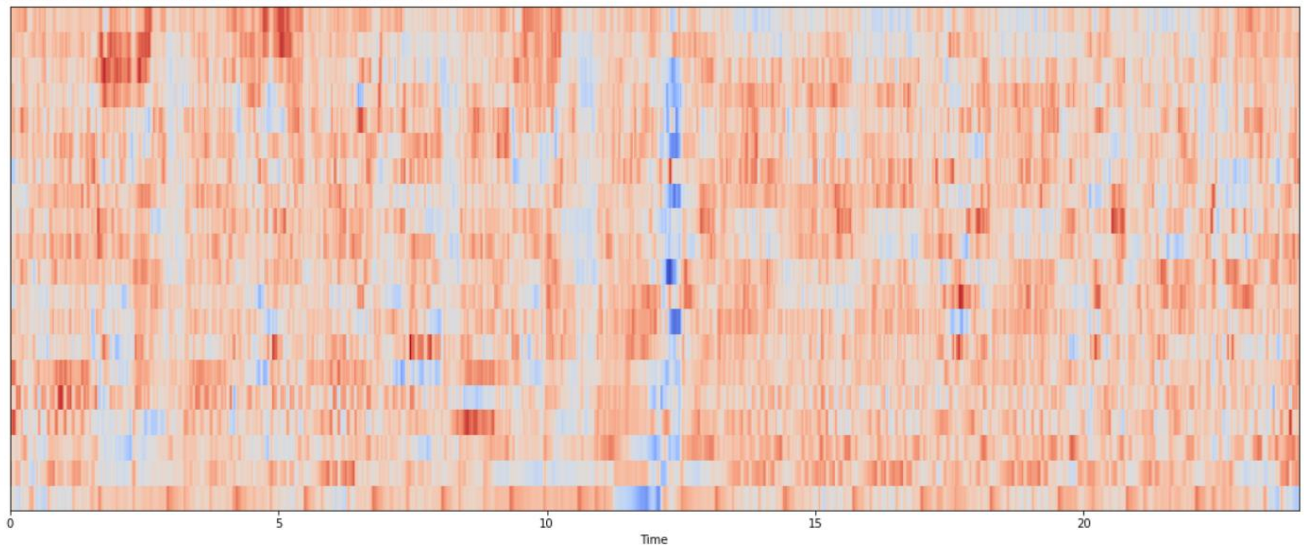



```
[151] # MFCC Feature Scaling
      # Let's scale the MFCCs such that each coefficient dimension has zero mean and unit variance:
      mfccs = sklearn.preprocessing.scale(mfccs, axis=1)
      print(mfccs.mean(axis=1))
      print(mfccs.var(axis=1))
```

```
[ 7.8396827e-09  0.0000000e+00 -1.4757050e-08  7.3785249e-09
 -7.3785249e-09  7.3785249e-09 -7.3785249e-09  1.1067787e-08
 -7.3785249e-09  0.0000000e+00  0.0000000e+00 -5.5338933e-09
 -7.3785249e-09  7.3785249e-09 -7.3785249e-09  0.0000000e+00
  0.0000000e+00  7.3785249e-09  0.0000000e+00  0.0000000e+00]
[1.        1.        1.        1.        0.9999999 1.        1.
 1.        1.0000001 1.        1.        1.        1.        1.
 0.9999999 1.        1.        1.        1.0000001 1.        ]
```

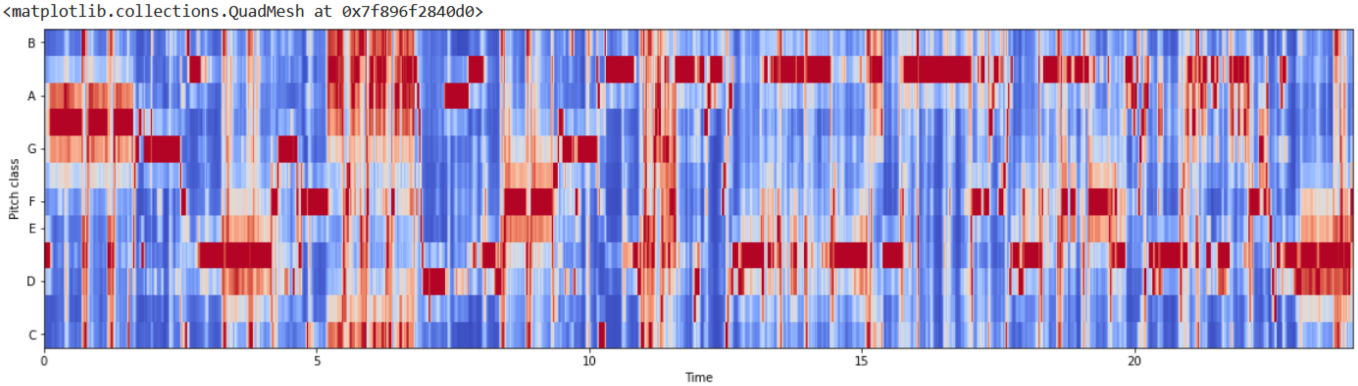
[153]
```
plt.figure(figsize=(20,8))
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
```

```
<matplotlib.collections.QuadMesh at 0x7f896fb63210>
```



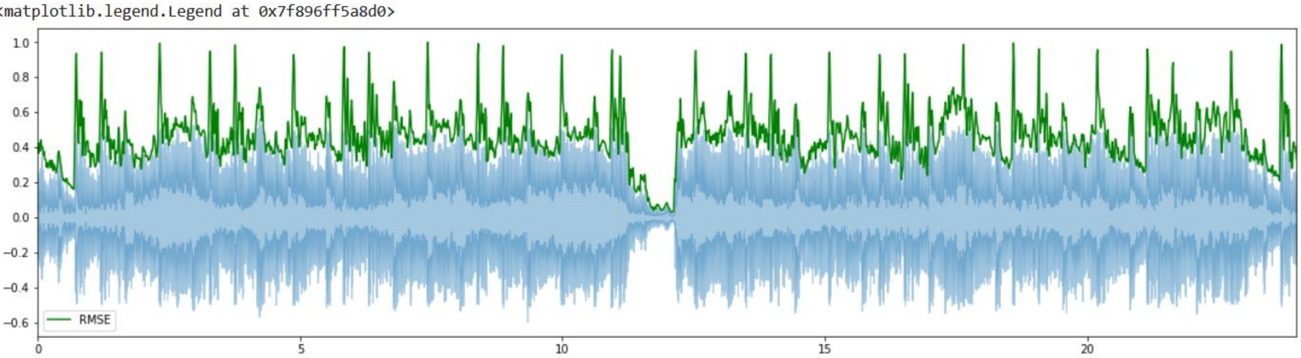
[180]
```
## 6. Chroma Frequencies ##
hop_length = 512
chromagram = librosa.feature.chroma_stft(x, sr=sr, hop_length=hop_length)
plt.figure(figsize=(20, 5))
librosa.display.specshow(chromagram, x_axis='time', y_axis='chroma', hop_length=hop_length, cmap='coolwarm')
```

```
<matplotlib.collections.QuadMesh at 0x7f896f2840d0>
```



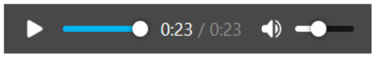
[157]
```
## 7. Root-mean-square energy (RMSE) ##
plt.figure(figsize=(20,5))
hop_length = 256
frame_length = 512
rmse = librosa.feature.rms(x, frame_length=frame_length, hop_length=hop_length, center=True)[0]
frames = range(len(rmse))
t = librosa.frames_to_time(frames, sr=sr, hop_length=hop_length)
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t[:len(rmse)], rmse/rmse.max(), color='green') # normalized for visualization
plt.legend(["RMSE"], loc=0, frameon=True)
```

```
<matplotlib.legend.Legend at 0x7f896ff5a8d0>
```



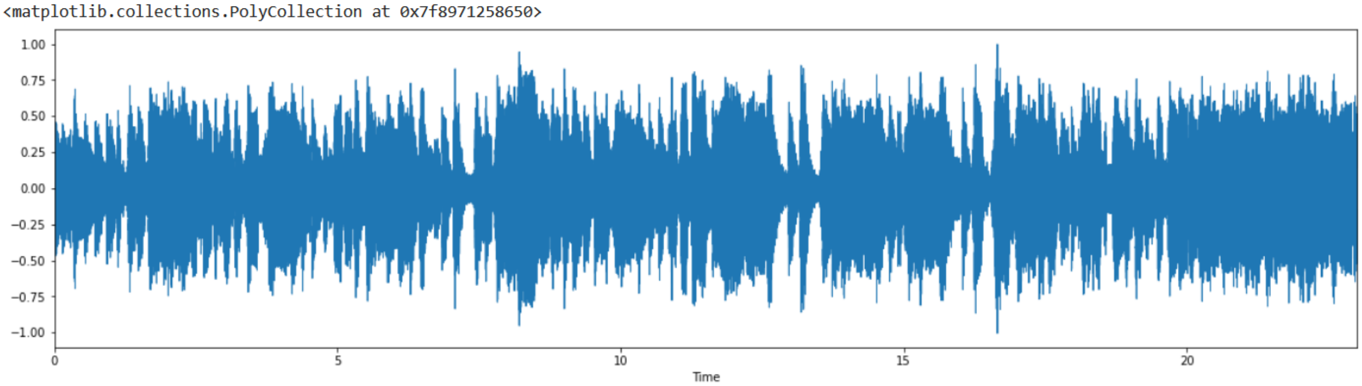
[158]
```
##########################################################################################################################
# Second wav music file
audio_path = 'genres/rock/dont_stop_me_now_chorus.wav'
x , sr = librosa.load(audio_path)
```

[159]
```
# Playing Audio
# Using IPython.display.Audio, to play the audio
import IPython.display as ipd
ipd.Audio(audio_path)
```



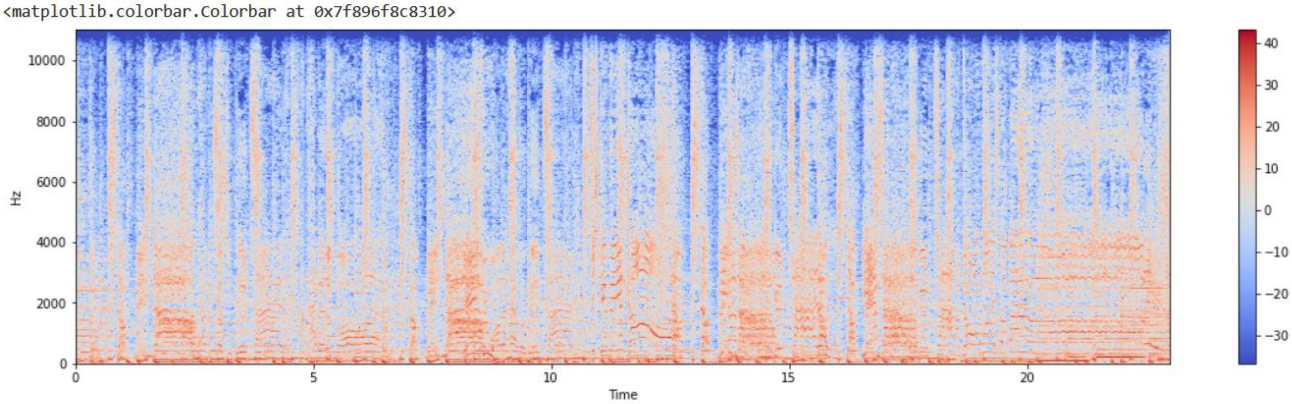
[160]
```
# Visualizing Audio
## 1. Waveform ##
%matplotlib inline
import sklearn
import matplotlib.pyplot as plt
import librosa.display

plt.figure(figsize=(20, 5))
librosa.display.waveplot(x, sr=sr)
```

```
<matplotlib.collections.PolyCollection at 0x7f8971258650>
```



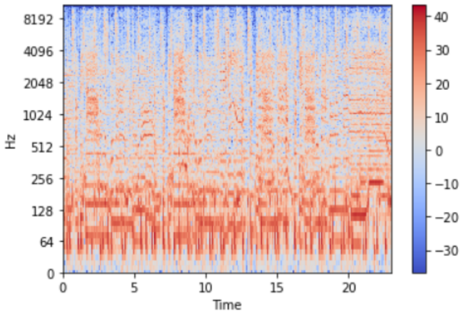
[161]
```
## 2. Spectogram ##
X = librosa.stft(x)
Xdb = librosa.amplitude_to_db(abs(X))
plt.figure(figsize=(20, 5))
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='hz')
plt.colorbar()
```

```
<matplotlib.colorbar.Colorbar at 0x7f896f8c8310>
```

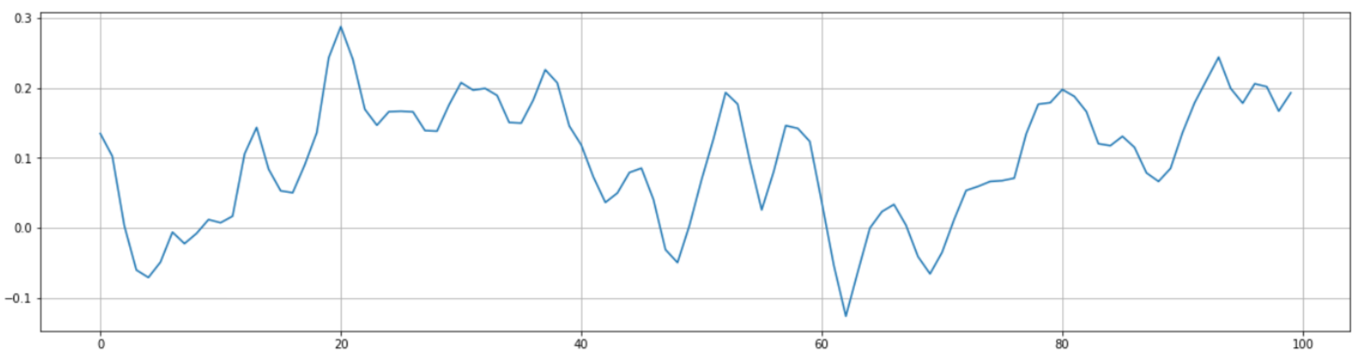


[162]
```
## 3. Log Frequency axis ##
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
plt.colorbar()
```

<matplotlib.colorbar.Colorbar at 0x7f896f8f5590>



[163]
```
# Feature Extraction
## 1. Zero Crossing Rate ##
# Zooming in
# Here we will zoom or print spectrum for 100 array columns only.
n0 = 9000
n1 = 9100
plt.figure(figsize=(20, 5))
plt.plot(x[n0:n1])
plt.grid()
```



[164]
```
zero_crossings = librosa.zero_crossings(x[n0:n1], pad=False)
zero_crossings.shape
```

(100,)

[165]
```
# We can also calculate zero crossings using a given code:
print(sum(zero_crossings))
```

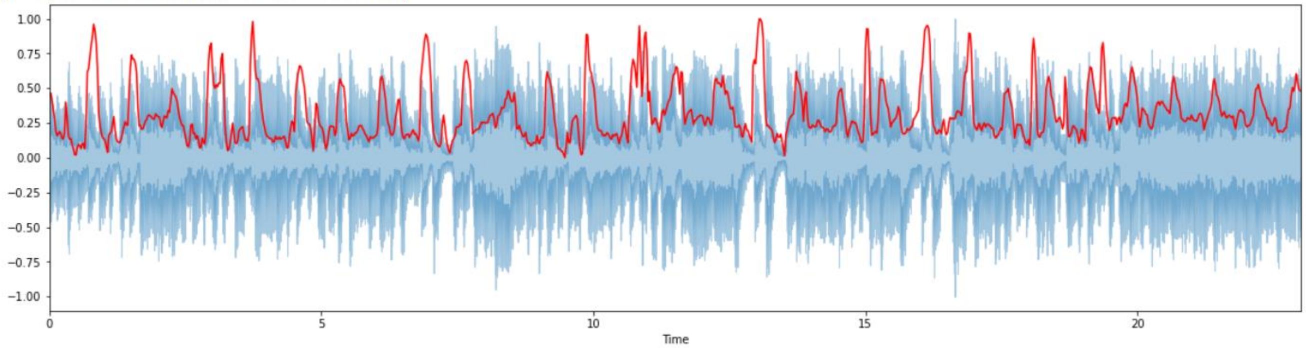
8

[166]
```
## 2. Spectral Centroid ##
#spectral centroid -- centre of mass -- weighted mean of the frequencies present in the sound
spectral_centroids = librosa.feature.spectral_centroid(x, sr=sr)[0]
spectral_centroids.shape
```

(991,)

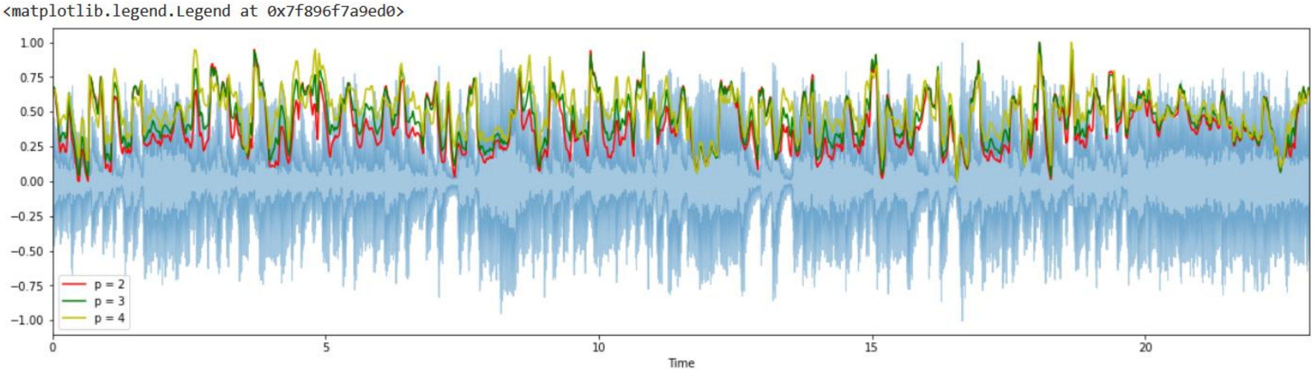
[167]
```
# Computing the time variable for visualization
plt.figure(figsize=(20,5))
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)

# 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.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='r')
```

```
[<matplotlib.lines.Line2D at 0x7f896f834bd0>]
```

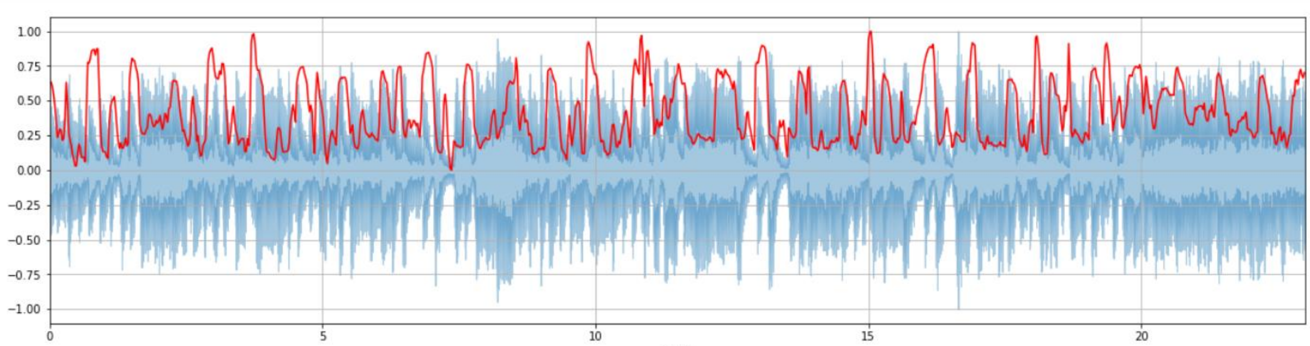


[168]
```
## 3. Spectral Bandwidth ##
plt.figure(figsize=(20,5))
spectral_bandwidth_2 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr)[0]
spectral_bandwidth_3 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr, p=3)[0]
spectral_bandwidth_4 = librosa.feature.spectral_bandwidth(x+0.01, sr=sr, p=4)[0]
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_bandwidth_2), color='r')
plt.plot(t, normalize(spectral_bandwidth_3), color='g')
plt.plot(t, normalize(spectral_bandwidth_4), color='y')
plt.legend(('p = 2', 'p = 3', 'p = 4'))
```

```
<matplotlib.legend.Legend at 0x7f896f7a9ed0>
```



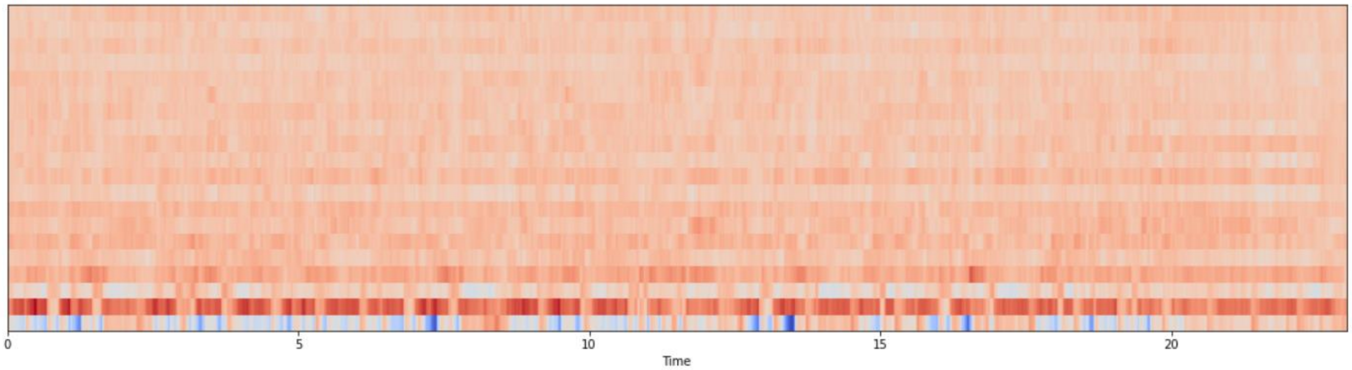
[169]
```
## 4. Spectral Rolloff ##
plt.figure(figsize=(20,5))
spectral_rolloff = librosa.feature.spectral_rolloff(x+0.01, sr=sr)[0]
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_rolloff), color='r')
plt.grid()
```



✓ [170]

```
## 5. Mel-Frequency Cepstral Coefficients ##
plt.figure(figsize=(20,5))
mfccs = librosa.feature.mfcc(x, sr=sr)
print(mfccs.shape)

#Displaying  the MFCCs:
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
```

```
(20, 991)
<matplotlib.collections.QuadMesh at 0x7f896f6a9f10>
```



✓ [171]

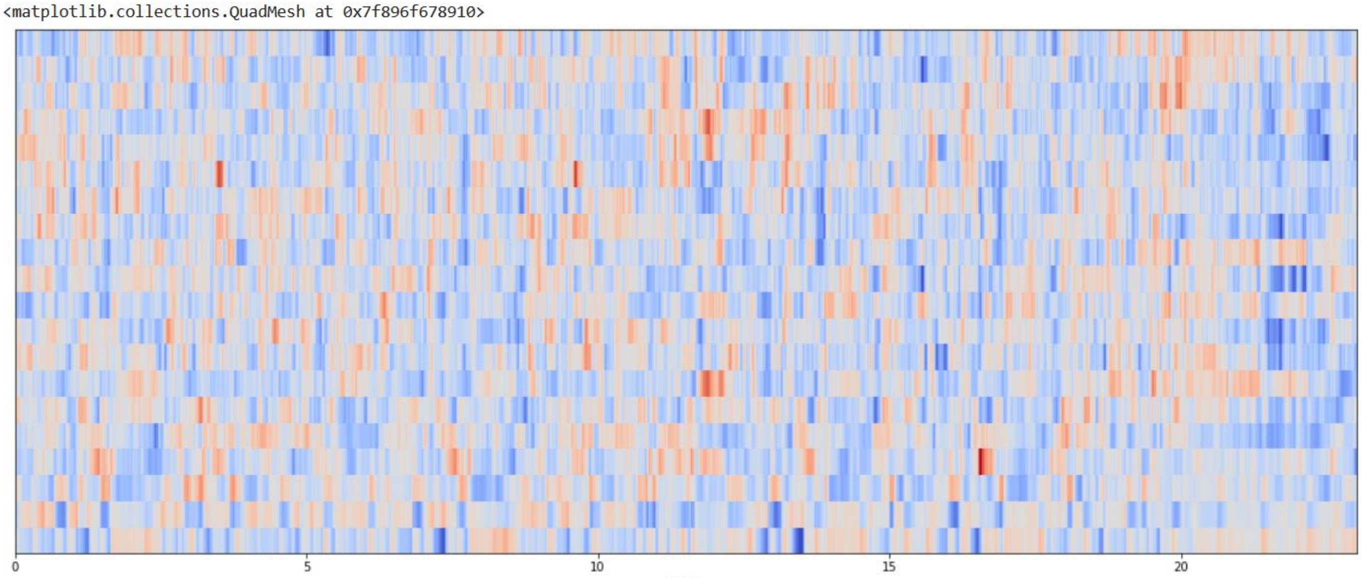
```
# MFCC Feature Scaling
# Let's scale the MFCCs such that each coefficient dimension has zero mean and unit variance:
mfccs = sklearn.preprocessing.scale(mfccs, axis=1)
print(mfccs.mean(axis=1))
print(mfccs.var(axis=1))
```

```
[ 0.0000000e+00  0.0000000e+00  0.0000000e+00 -1.9246706e-09
  7.6986826e-09  0.0000000e+00  0.0000000e+00 -7.6986826e-09
  7.6986826e-09  0.0000000e+00  7.6986826e-09  3.8493413e-09
  0.0000000e+00  0.0000000e+00  0.0000000e+00  7.6986826e-09
  5.7740119e-09  9.6233537e-09  1.1548024e-08  2.8870060e-09]
[1.         1.         1.0000001  0.9999999  1.0000001  1.0000001
 1.0000001  0.99999994 1.0000001  0.9999999  1.         1.0000001
 1.0000001  1.         0.9999999  1.         1.         0.9999998
 1.         1.0000001 ]
```

✓ [172]

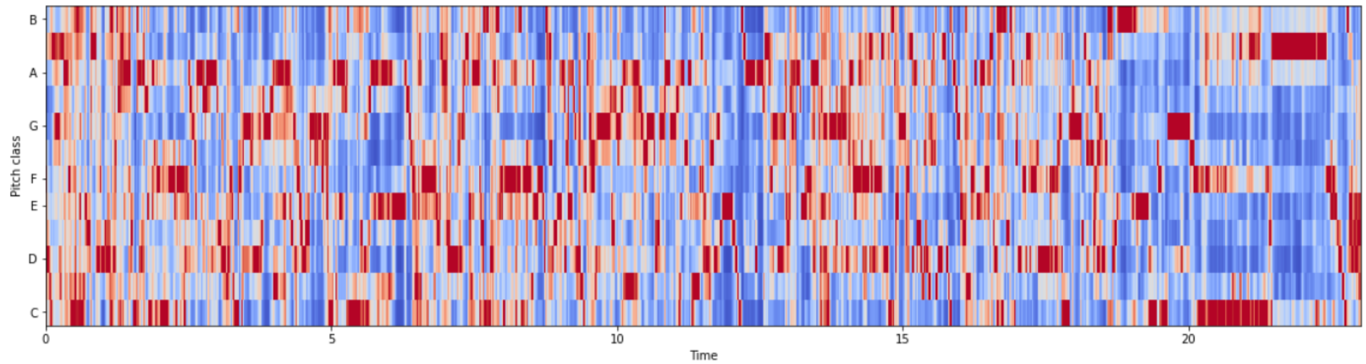
```
plt.figure(figsize=(20,8))
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
```

```
<matplotlib.collections.QuadMesh at 0x7f896f678910>
```



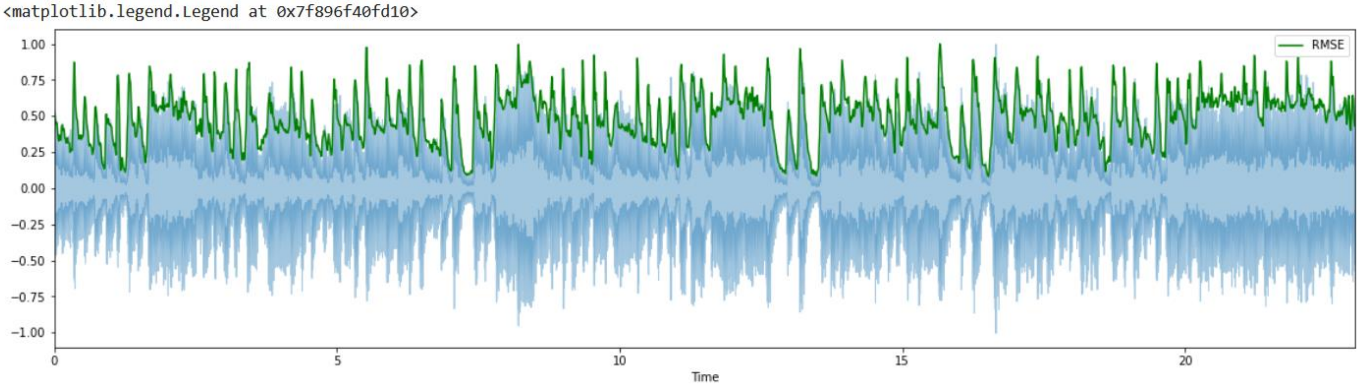
```
[175] ## 6. Chroma Frequencies ##
hop_length = 512
chromagram = librosa.feature.chroma_stft(x, sr=sr, hop_length=hop_length)
plt.figure(figsize=(20, 5))
librosa.display.specshow(chromagram, x_axis='time', y_axis='chroma', hop_length=hop_length, cmap='coolwarm')
```

<matplotlib.collections.QuadMesh at 0x7f896f4b0990>




```
[176] ## 7. Root-mean-square energy (RMSE) ##
plt.figure(figsize=(20,5))
hop_length = 256
frame_length = 512
rmse = librosa.feature.rms(x, frame_length=frame_length, hop_length=hop_length, center=True)[0]
frames = range(len(rmse))
t = librosa.frames_to_time(frames, sr=sr, hop_length=hop_length)
librosa.display.waveplot(x, sr=sr, alpha=0.4)
plt.plot(t[:len(rmse)], rmse/rmse.max(), color='green') # normalized for visualization
plt.legend(["RMSE"], loc=0, frameon=True)
```

<matplotlib.legend.Legend at 0x7f896f40fd10>




```
[184] ########################################################################################################################
# Creating the CSV Dataset
import os
import csv
import numpy as np

header = 'filename chroma_stft rmse spectral_centroid spectral_bandwidth rolloff zero_crossing_rate'
for i in range(1, 21):
    header += f' mfcc{i}'
header += ' label'
header = header.split()
```

[186]
```python
file = open('data.csv', 'w', newline='')
with file:
    writer = csv.writer(file)
    writer.writerow(header)
genres = 'pop rock'.split()

for g in genres:
    for filename in os.listdir(f'./genres/{g}'):
        songname = f'./genres/{g}/{filename}'
        y, sr = librosa.load(songname, mono=True, duration=30)
        chroma_stft = librosa.feature.chroma_stft(y=y, sr=sr)
        rmse = librosa.feature.rms(y=y)
        spec_cent = librosa.feature.spectral_centroid(y=y, sr=sr)
        spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
        rolloff = librosa.feature.spectral_rolloff(y=y, sr=sr)
        zcr = librosa.feature.zero_crossing_rate(y)
        mfcc = librosa.feature.mfcc(y=y, sr=sr)
        to_append = f'{filename} {np.mean(chroma_stft)} {np.mean(rmse)} {np.mean(spec_cent)} {np.mean(spec_bw)} {np.mean(rolloff)} {np.mean(zcr
        for e in mfcc:
            to_append += f' {np.mean(e)}'
        to_append += f' {g}'
        file = open('data.csv', 'a', newline='')
        with file:
            writer = csv.writer(file)
            writer.writerow(to_append.split())
```