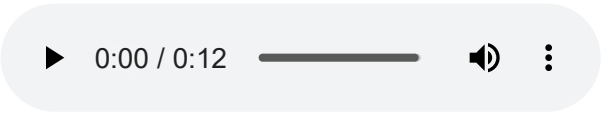
WAV file 1

Loading and playing the audio

In [ ]:
```
import warnings
import librosa
import IPython.display as ipd

warnings.filterwarnings('ignore')

audio_path = 'C:/Users/rowen/Downloads/Try/Audio/wish.wav'
x , sr = librosa.load(audio_path)

ipd.Audio(audio_path)
```

Out[ ]:

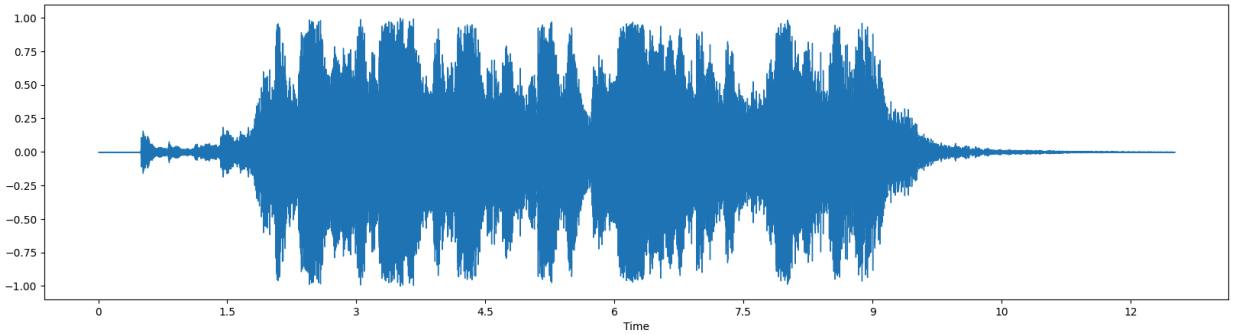


Visualising Audio

Waveform and Spectogram

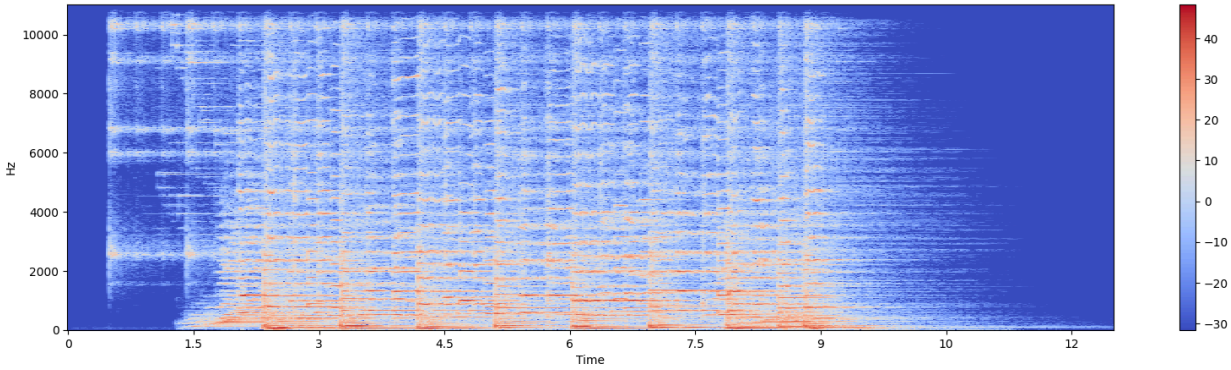
In [ ]:
```
import sklearn
import matplotlib.pyplot as plt
import librosa.display

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

Out[ ]:
```
<librosa.display.AdaptiveWaveplot at 0x1340e693d90>
```



In [ ]:
```
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()
```

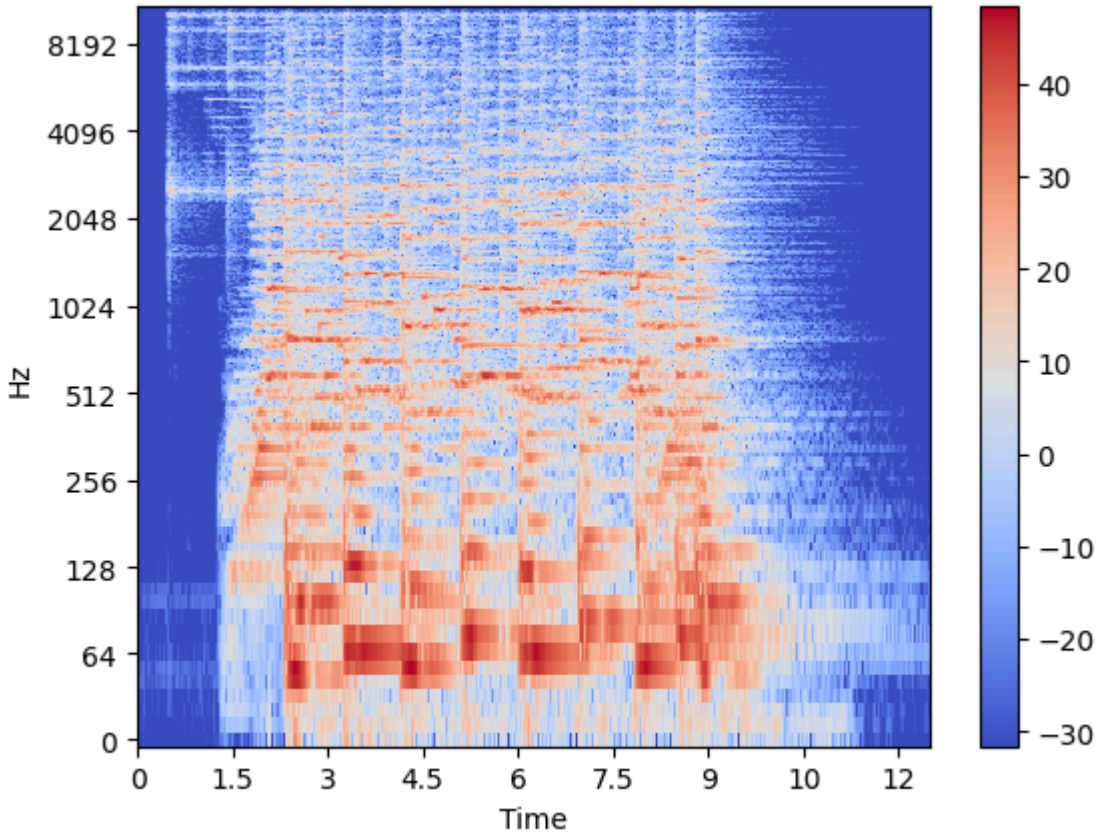
Out[ ]:
```
<matplotlib.colorbar.Colorbar at 0x134129e7f90>
```



Long Frequency axis and Audio Signal

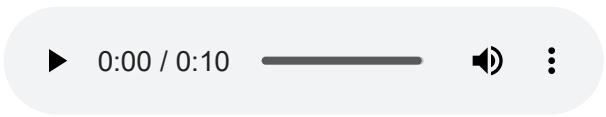
In [ ]:
```
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
plt.colorbar()
```

Out[ ]:
```
<matplotlib.colorbar.Colorbar at 0x1340d8f4fd0>
```



In [ ]:
```
import numpy as np
sr = 22050
T = 10.0
t = np.linspace(0, T, int(T*sr), endpoint=False)
x = 0.5*np.sin(2*np.pi*220*t)

ipd.Audio(x, rate=sr)
```

Out[ ]:

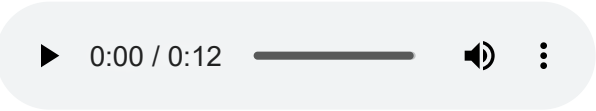


In [ ]:
```
import soundfile as sf
sf.write('wish1_440.wav', x, sr)
```

Feature Extraction

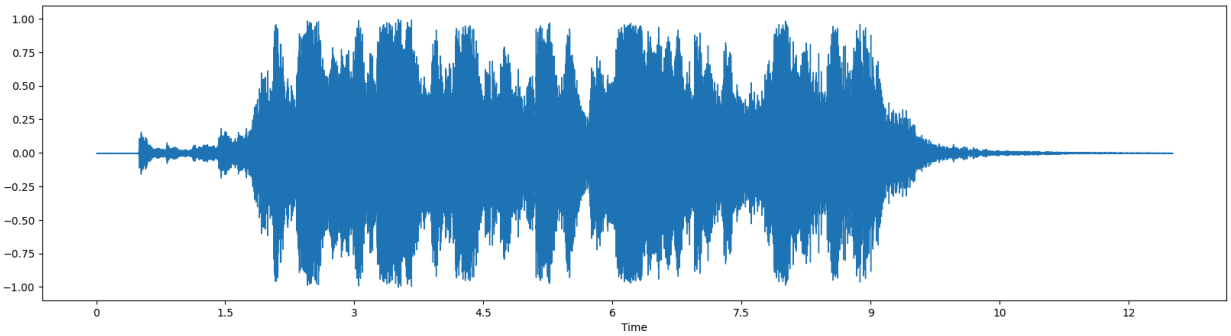
In [ ]:
```
x, sr = librosa.load('C:/Users/rowen/Downloads/Try/Audio/wish.wav')
ipd.Audio(x, rate=sr)
```

Out[ ]:



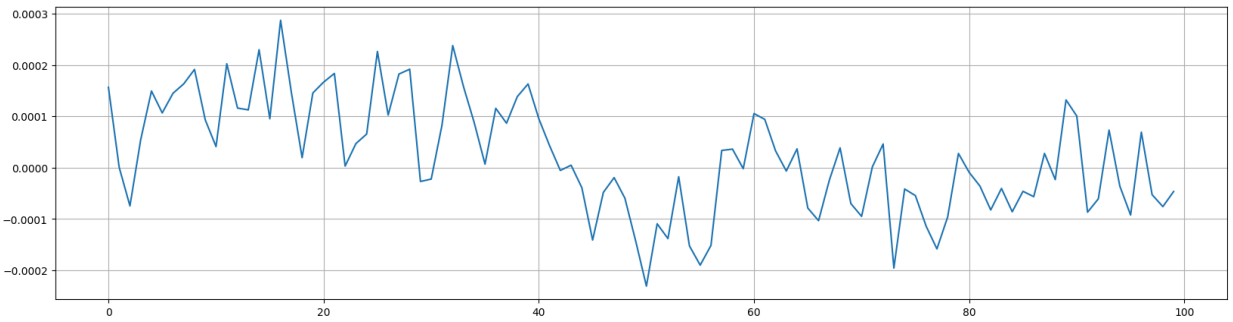
In [ ]:
```
plt.figure(figsize=(20, 5))
librosa.display.waveshow(x, sr=sr)
```

Out[ ]:
```
<librosa.display.AdaptiveWaveplot at 0x13412a53450>
```



Zero Crossing Rate

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



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

Out[ ]:
```
(100,)
```

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

```
27
```

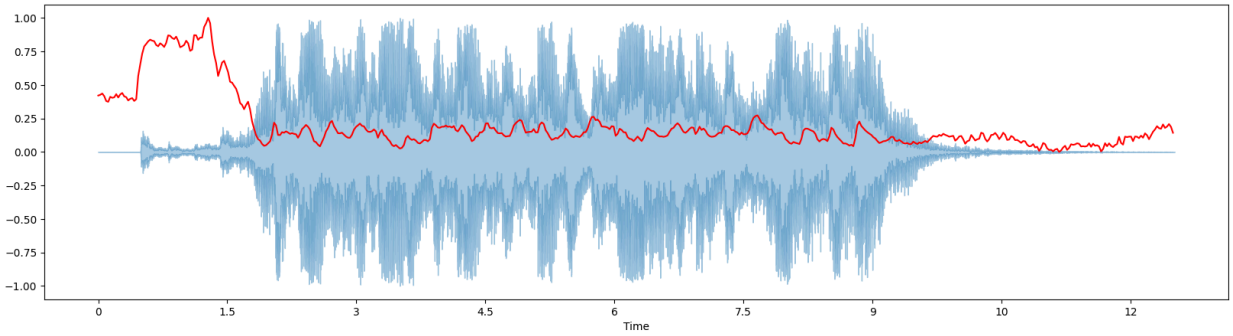
Spectral Centeroid

In [ ]:
```
spectral_centroids = librosa.feature.spectral_centroid(y=x, sr=sr)[0]
spectral_centroids.shape
```

Out[ ]:
```
(539,)
```

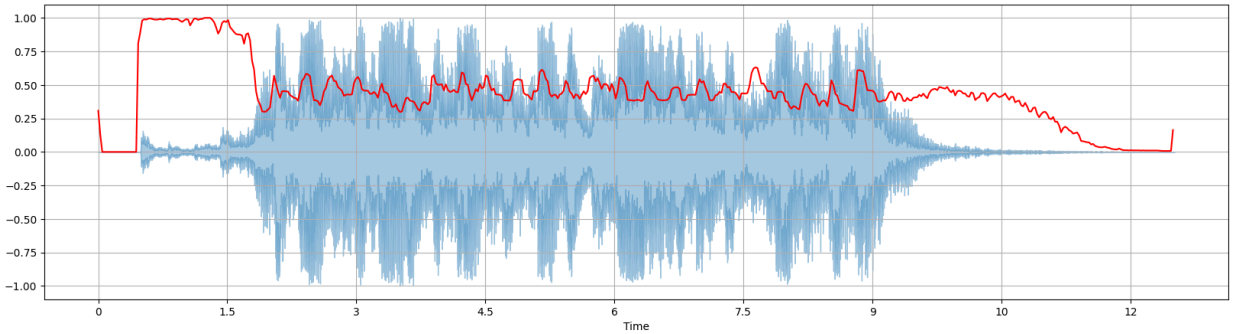
In [ ]:
```
from sklearn.preprocessing import minmax_scale

plt.figure(figsize=(20,5))
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)

def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)

librosa.display.waveshow(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='r')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1341483a6d0>]
```



Spectral Rolloff

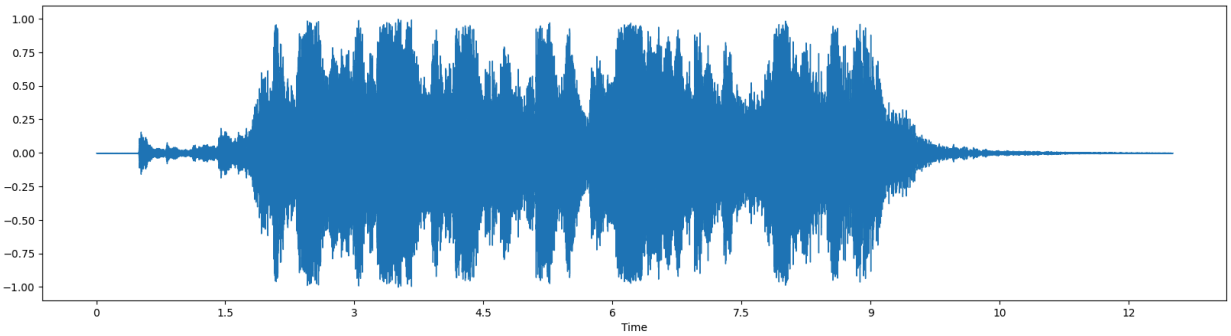
In [ ]:
```
plt.figure(figsize=(20,5))
spectral_rolloff = librosa.feature.spectral_rolloff(y=x+0.01, sr=sr)[0]
librosa.display.waveshow(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_rolloff), color='r')
plt.grid()
```



MFCC

In [ ]:
```
plt.figure(figsize=(20,5))
x, fs = librosa.load('C:/Users/rowen/Downloads/Try/Audio/wish.wav')
librosa.display.waveshow(x, sr=sr)
```

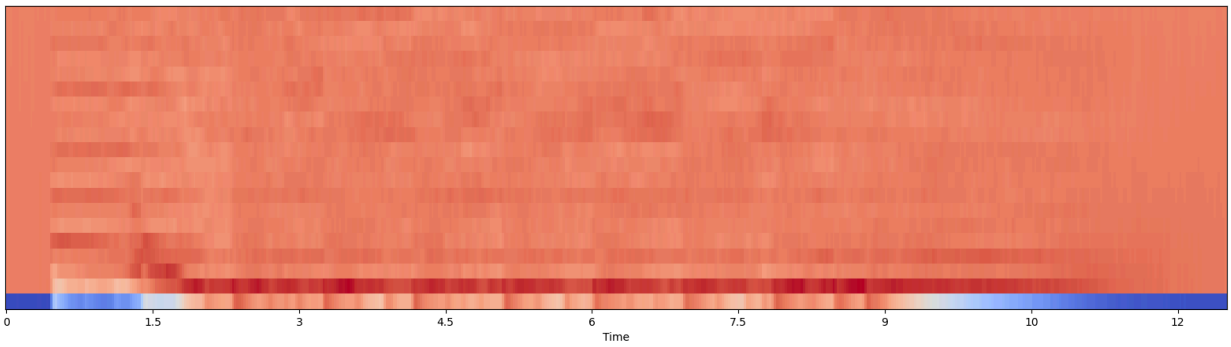
Out[ ]:
```
<librosa.display.AdaptiveWaveplot at 0x13414e91a10>
```



In [ ]:
```
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, 539)
```

Out[ ]:
```
<matplotlib.collections.QuadMesh at 0x13414ecd310>
```



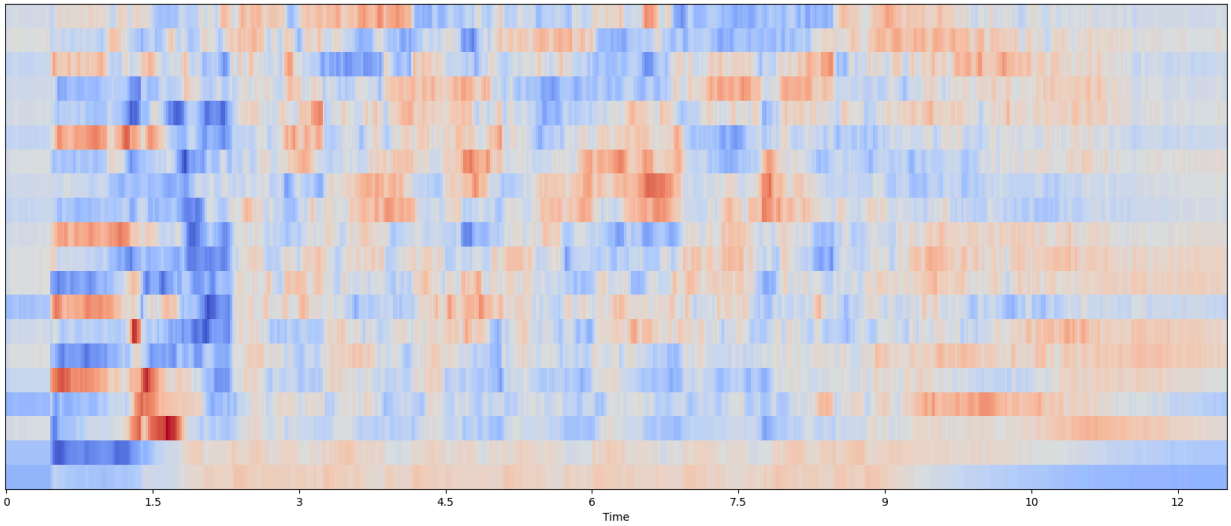
Feature Scaling

In [ ]:
```
mfccs = sklearn.preprocessing.scale(mfccs, axis=1)
print(mfccs.mean(axis=1))
print(mfccs.var(axis=1))
```

```
[ 2.8309442e-08  7.0773605e-09 -7.0773605e-09 -1.4154721e-08
  0.0000000e+00  0.0000000e+00  1.4154721e-08  0.0000000e+00
 -1.4154721e-08 -1.4154721e-08  2.1232081e-08  0.0000000e+00
  7.0773605e-09  3.5386802e-09  1.4154721e-08  0.0000000e+00
  0.0000000e+00  0.0000000e+00 -1.4154721e-08  7.0773605e-09]
[1.0000001 1.        1.        1.        1.        1.0000001 1.
 1.0000001 1.0000001 1.0000001 1.        1.        1.        0.9999999
 1.        1.0000001 1.0000002 1.        1.        1.       ]
```

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

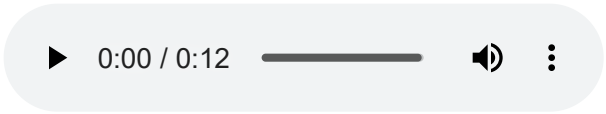
Out[ ]:
```
<matplotlib.collections.QuadMesh at 0x134153b6b90>
```



Chroma Frequencies

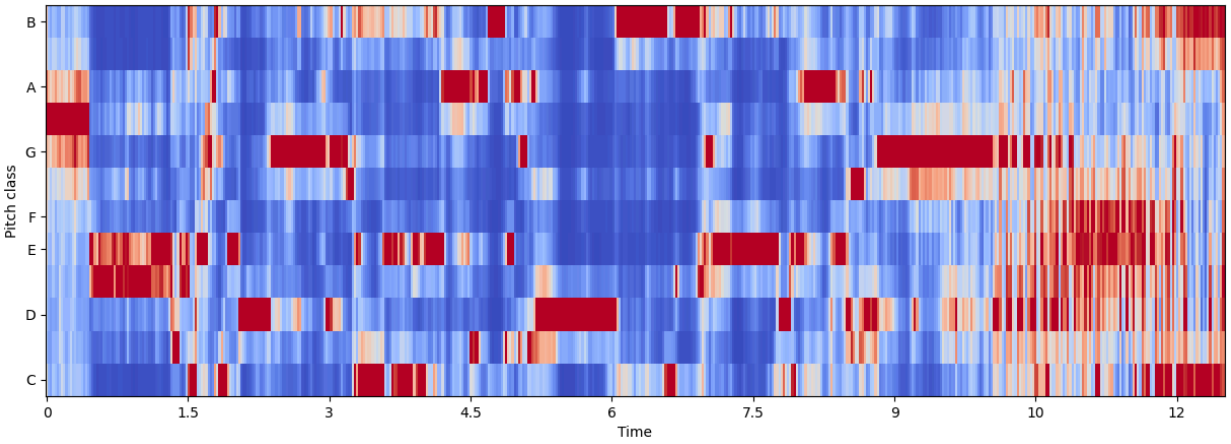
In [ ]:
```
x, sr = librosa.load('C:/Users/rowen/Downloads/Try/Audio/wish.wav')
ipd.Audio(x, rate=sr)
```

Out[ ]:



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

Out[ ]:
```
<matplotlib.collections.QuadMesh at 0x13414f3e250>
```



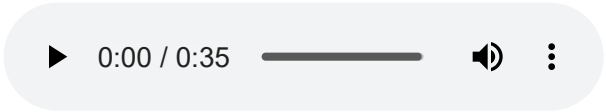
WAV file 2

Loading and playing the audio

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

audio_path = 'C:/Users/rowen/Downloads/Try/Audio/chill.wav'
x , sr = librosa.load(audio_path)
```

```
ipd.Audio(audio_path)
```

Out[ ]:

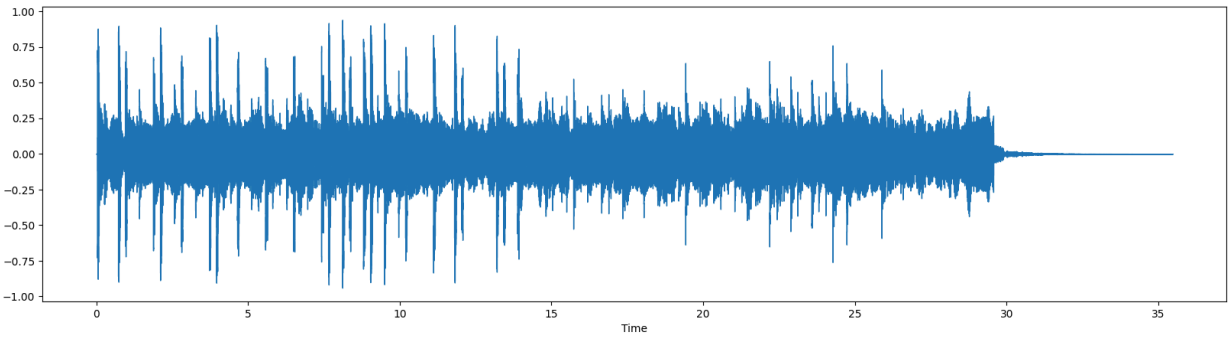


Visualising Audio

Waveform and Spectogram

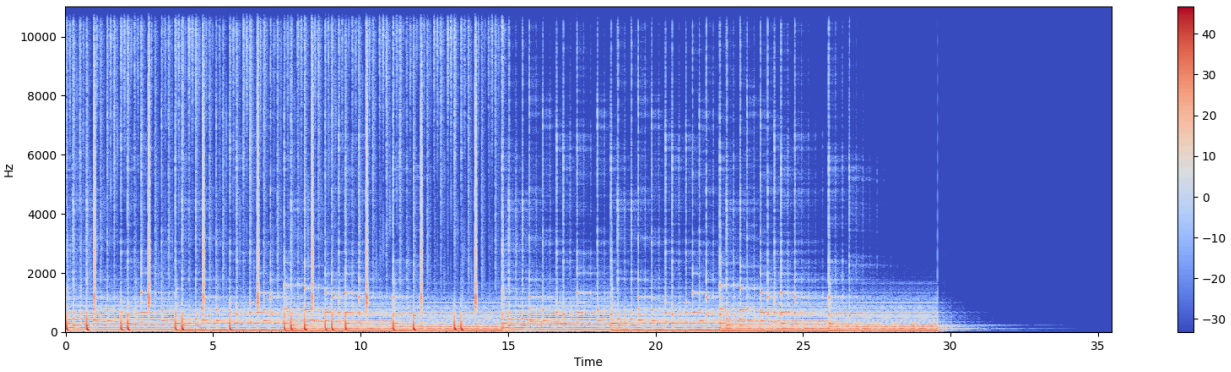
In [ ]:
```
plt.figure(figsize=(20, 5))
librosa.display.waveshow(x, sr = sr)
```

Out[ ]:
```
<librosa.display.AdaptiveWaveplot at 0x13415352c50>
```



In [ ]:
```
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()
```

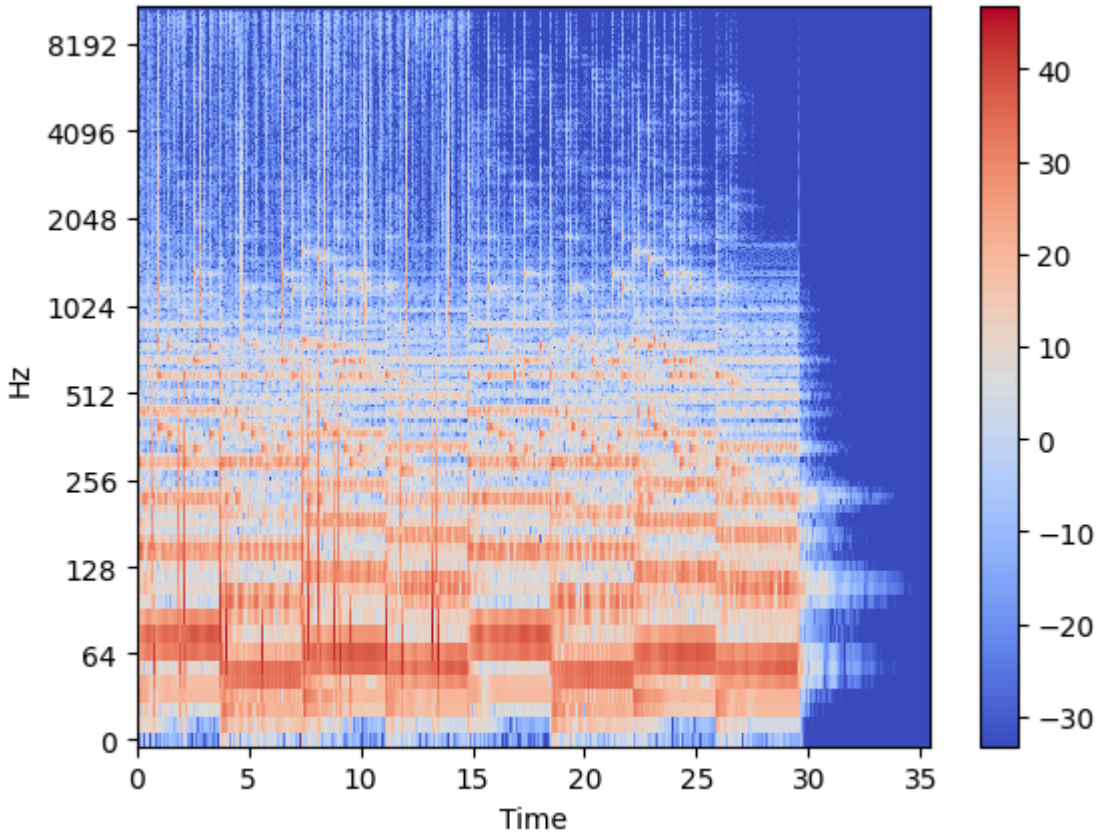
Out[ ]:
```
<matplotlib.colorbar.Colorbar at 0x13414ebd090>
```



Long Frequency axis and Audio Signal

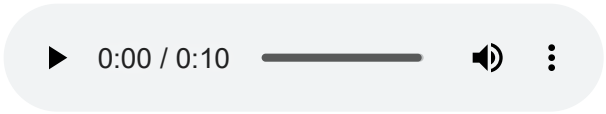
In [ ]:
```
librosa.display.specshow(Xdb, sr=sr, x_axis='time', y_axis='log')
plt.colorbar()
```

Out[ ]:
```
<matplotlib.colorbar.Colorbar at 0x1341682bd90>
```



In [ ]:
```
sr = 22050
T = 10.0
t = np.linspace(0, T, int(T*sr), endpoint=False)
x = 0.5*np.sin(2*np.pi*220*t)

ipd.Audio(x, rate=sr)
```

Out[ ]:

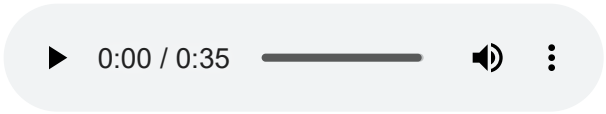


In [ ]:
```
sf.write('chill1_440.wav', x, sr)
```

Feature Extraction

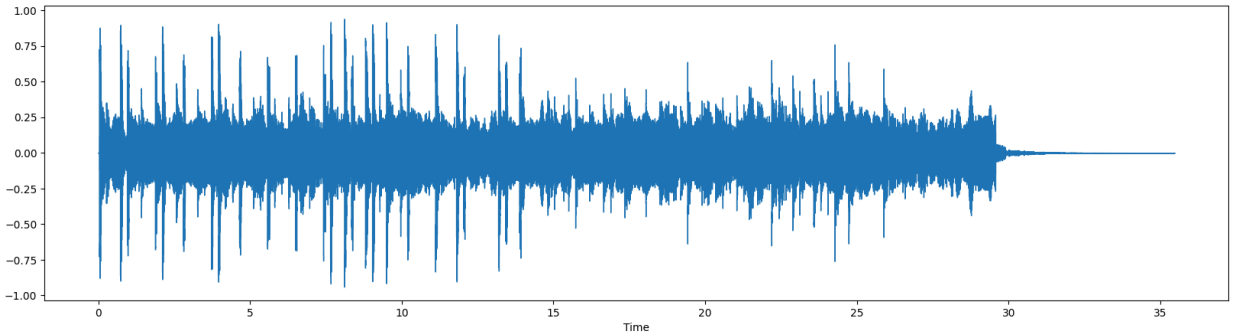
In [ ]:
```
x, sr = librosa.load('C:/Users/rowen/Downloads/Try/Audio/chill.wav')
ipd.Audio(x, rate=sr)
```

Out[ ]:



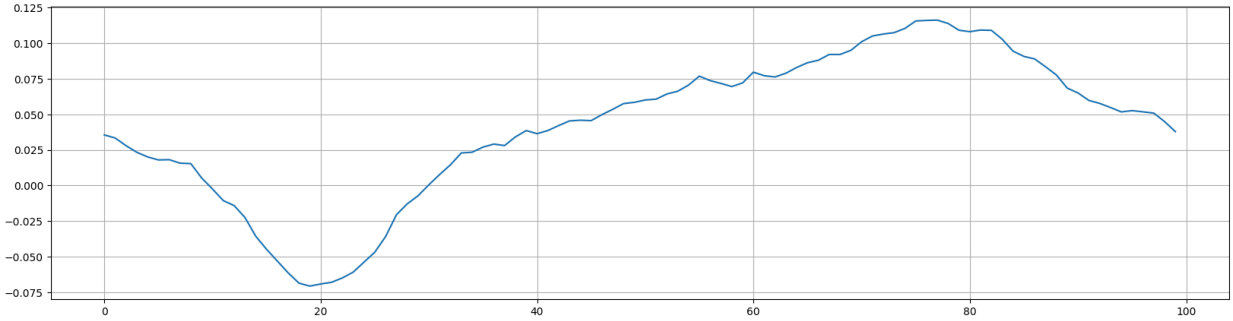
In [ ]:
```
plt.figure(figsize=(20, 5))
librosa.display.waveshow(x, sr=sr)
```

Out[ ]:
```
<librosa.display.AdaptiveWaveplot at 0x13414b29c10>
```



Zero Crossing Rate

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



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

Out[ ]:
```
(100,)
```

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

```
2
```

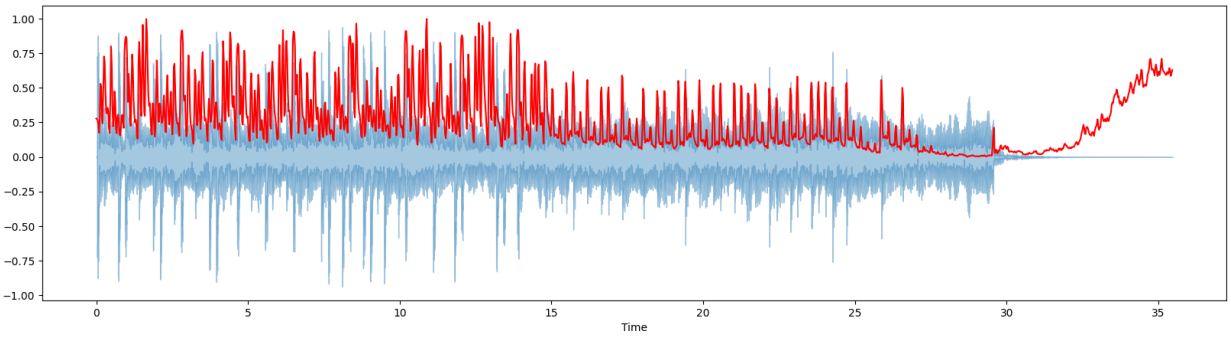
Spectral Centeroid

In [ ]:
```
spectral_centroids = librosa.feature.spectral_centroid(y=x, sr=sr)[0]
spectral_centroids.shape
```

Out[ ]:
```
(1528,)
```

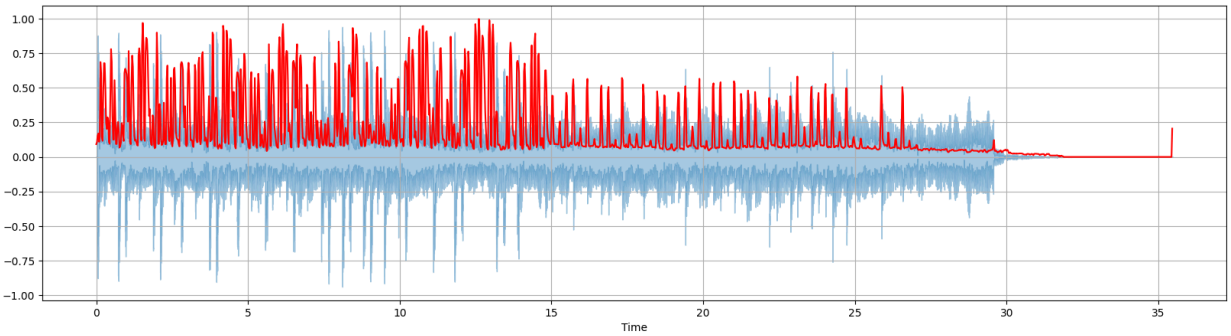
In [ ]:
```
plt.figure(figsize=(20,5))
frames = range(len(spectral_centroids))
t = librosa.frames_to_time(frames)

def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)

librosa.display.waveshow(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_centroids), color='r')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x13414b5c650>]
```



Spectral Rolloff

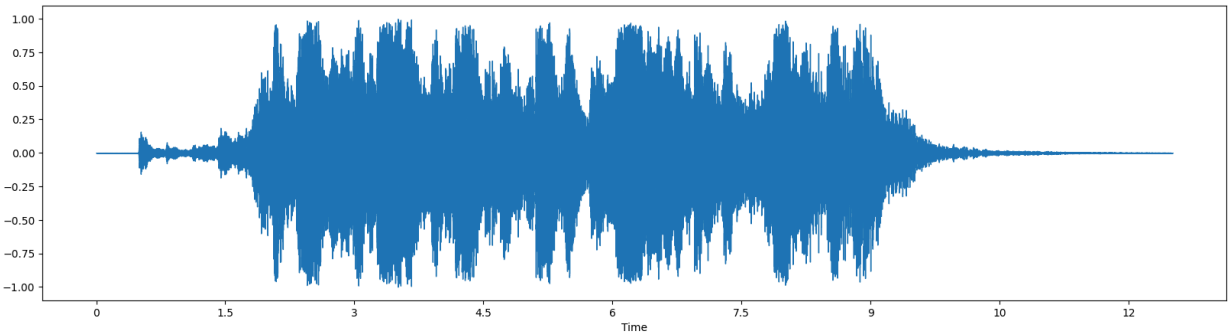
In [ ]:
```
plt.figure(figsize=(20,5))
spectral_rolloff = librosa.feature.spectral_rolloff(y=x+0.01, sr=sr)[0]
librosa.display.waveshow(x, sr=sr, alpha=0.4)
plt.plot(t, normalize(spectral_rolloff), color='r')
plt.grid()
```



MFCC

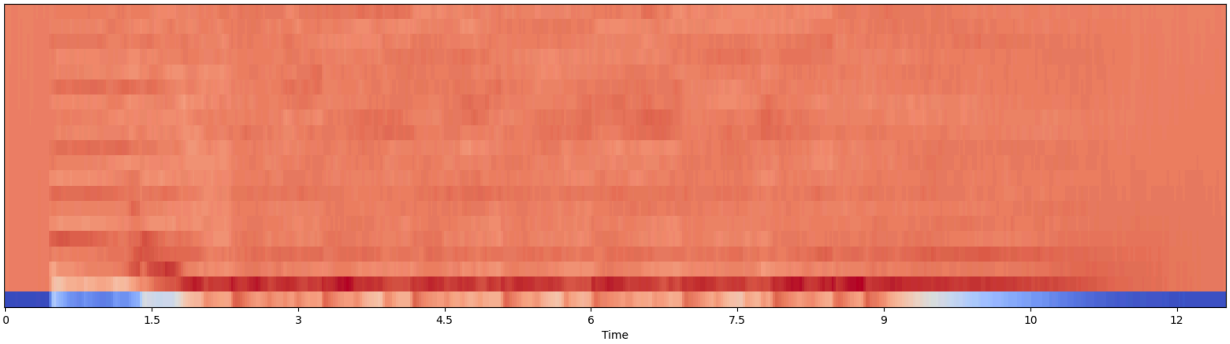
In [ ]:
```
plt.figure(figsize=(20,5))
x, fs = librosa.load('C:/Users/rowen/Downloads/Try/Audio/wish.wav')
librosa.display.waveshow(x, sr=sr)
```

Out[ ]:
```
<librosa.display.AdaptiveWaveplot at 0x1341490f990>
```



In [ ]:
```
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')
```

Out[ ]:
```
(20, 539)
<matplotlib.collections.QuadMesh at 0x13414eac050>
```



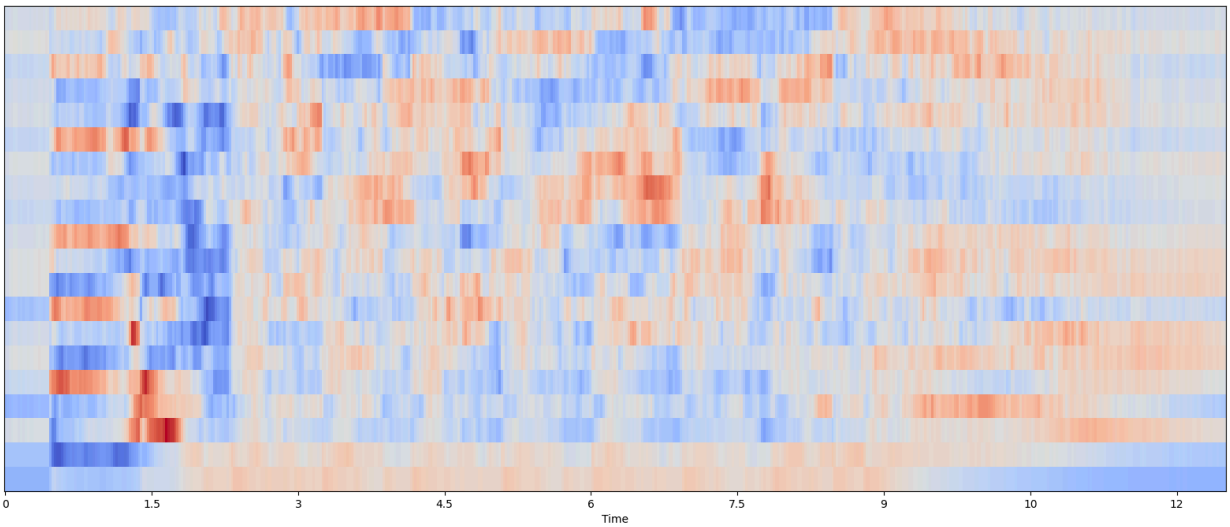
Feature Scaling

In [ ]:
```
mfccs = sklearn.preprocessing.scale(mfccs, axis=1)
print(mfccs.mean(axis=1))
print(mfccs.var(axis=1))
```

```
[ 2.8309442e-08  7.0773605e-09 -7.0773605e-09 -1.4154721e-08
  0.0000000e+00  0.0000000e+00  1.4154721e-08  0.0000000e+00
 -1.4154721e-08 -1.4154721e-08  2.1232081e-08  0.0000000e+00
  7.0773605e-09  3.5386802e-09  1.4154721e-08  0.0000000e+00
  0.0000000e+00  0.0000000e+00 -1.4154721e-08  7.0773605e-09]
[1.0000001 1.        1.        1.        1.        1.0000001 1.
 1.0000001 1.0000001 1.0000001 1.        1.        1.        0.9999999
 1.        1.0000001 1.0000002 1.        1.        1.       ]
```

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

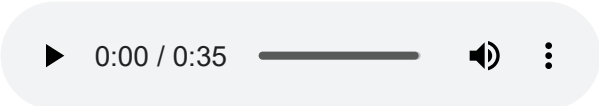
Out[ ]:
```
<matplotlib.collections.QuadMesh at 0x1341429e5d0>
```



Chroma Frequencies

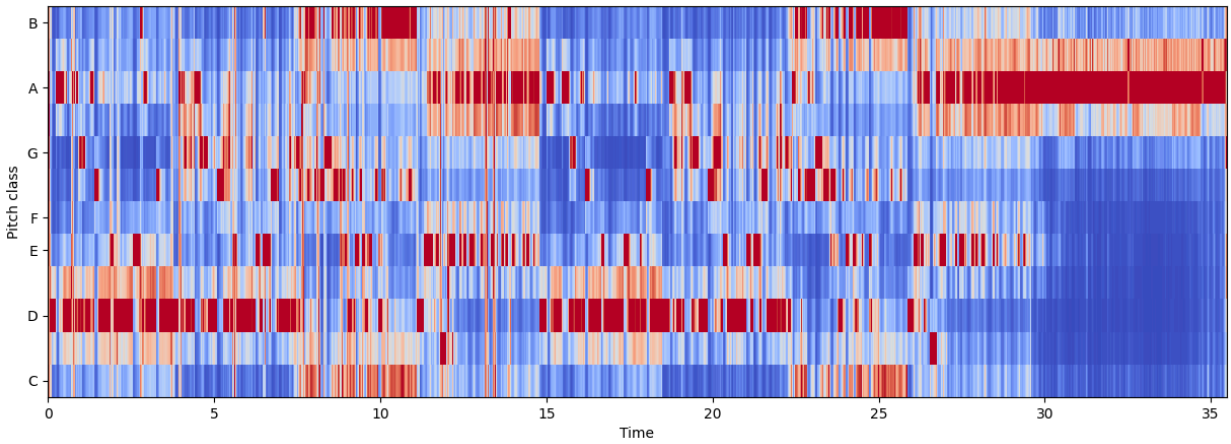
In [ ]:
```
x, sr = librosa.load('C:/Users/rowen/Downloads/Try/Audio/chill.wav')
ipd.Audio(x, rate=sr)
```

Out[ ]:



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

Out[ ]:
```
<matplotlib.collections.QuadMesh at 0x134142eee10>
```



Saving the features in a csv file with genre labels

In [ ]:
```
import pandas as pd

audio_files = ["wish.wav","chill.wav"]
labels = ["christmas music","chill"]

feature_list = []

for i, file_name in enumerate(audio_files):
    x, sr = librosa.load(file_name)

    n0, n1 = 9000, 9100
    zero= sum(librosa.zero_crossings(x[n0:n1], pad=False))

    spectral_centroids = librosa.feature.spectral_centroid(y=x, sr=sr)[0]
    spectral_c = np.mean(spectral_centroids)

    spectral_rolloff = librosa.feature.spectral_rolloff(y=x+0.01, sr=sr)[0]
    spectral_r = np.mean(spectral_rolloff)

    mfccs = librosa.feature.mfcc(y=x, sr=sr)
    mfccs_mean = np.mean(mfccs, axis=1)

    hop_length = 512
    chromagram = librosa.feature.chroma_stft(y=x, sr=sr, hop_length=hop_length)
    chroma = np.mean(chromagram, axis=1)

    features = [
        zero,
        spectral_c,
        spectral_r,
        *mfccs_mean,
        *chroma
    ]

    feature_list.append([file_name, *features, labels[i]])
```

```
columns = ['file_name', 'zero_crossings', 'spectral_centroid', 'spectral_rolloff'] + \
          [f'mfcc_{i+1}' for i in range(len(mfccs_mean))] + \
          [f'chroma_{i+1}' for i in range(len(chroma))] + ['label']
features_df = pd.DataFrame(feature_list, columns=columns)

features_df.to_csv("audio_features.csv", index=False)
```