GTZAN을 활용한 음악 장르 분류 및 추천

모두를 위한 인공지능 활용 10조

안지원 이종무 이향우

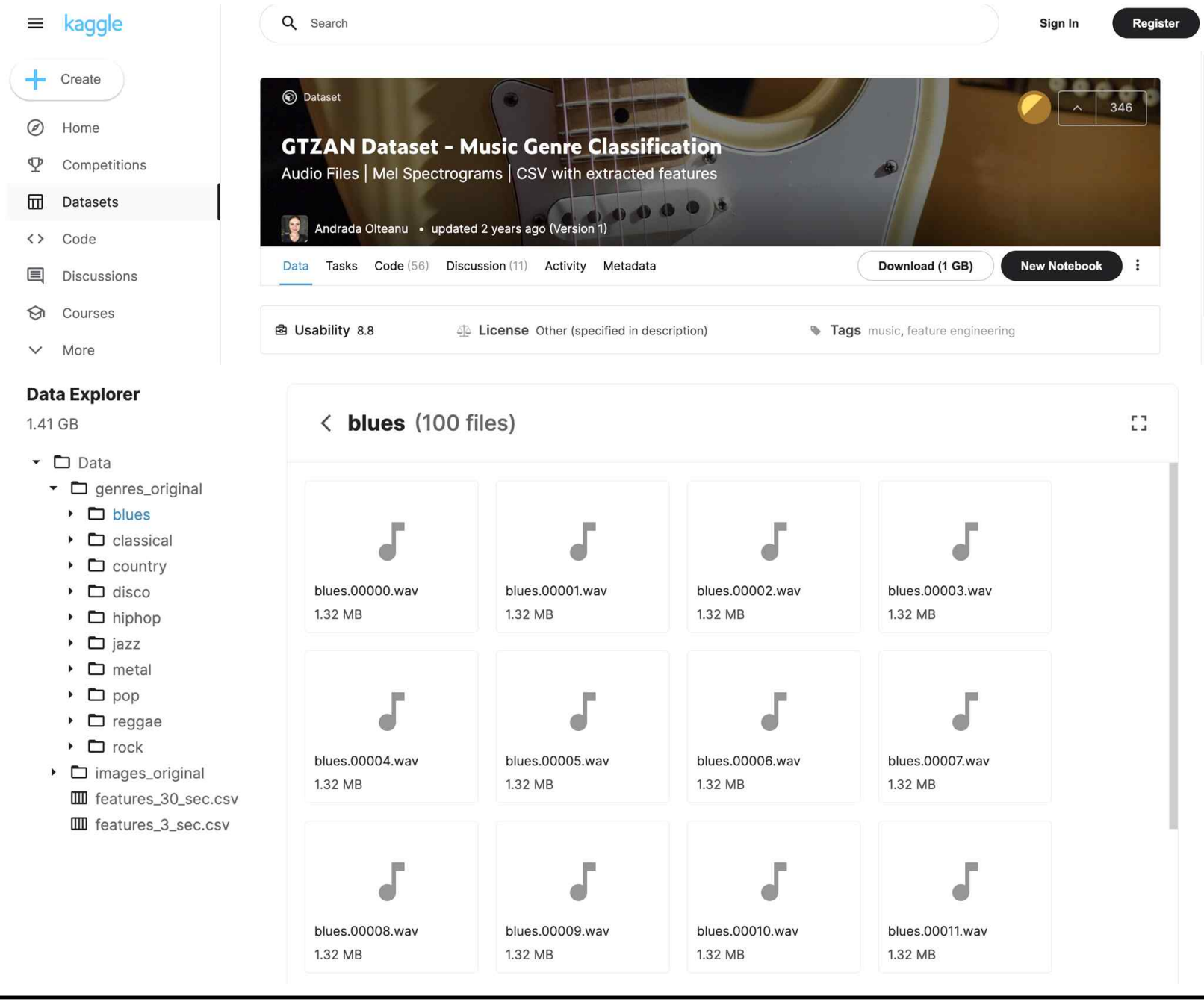
주제 선정 계기

“음악”

- 음악을 좋아한다는 공통점
- 음악 데이터를 처리하는 과정에 대한 호기심

프로젝트 시연

GTZAN


kaggle
Search
Sign In
Register
Create
Home
Competitions
Datasets
Code
Discussions
Courses
More
Dataset
346
GTZAN Dataset - Music Genre Classification
Audio Files | Mel Spectrograms | CSV with extracted features
Andrada Olteanu • updated 2 years ago (Version 1)
Data
Tasks
Code (56)
Discussion (11)
Activity
Metadata
Download (1 GB)
New Notebook
Usability 8.8
License Other (specified in description)
Tags music, feature engineering
Data Explorer
1.41 GB
Data
genres_original
blues
classical
country
disco
hiphop
jazz
metal
pop
reggae
rock
images_original
features_30_sec.csv
features_3_sec.csv
blues (100 files)
blues.00000.wav
1.32 MB
blues.00001.wav
1.32 MB
blues.00002.wav
1.32 MB
blues.00003.wav
1.32 MB
blues.00004.wav
1.32 MB
blues.00005.wav
1.32 MB
blues.00006.wav
1.32 MB
blues.00007.wav
1.32 MB
blues.00008.wav
1.32 MB
blues.00009.wav
1.32 MB
blues.00010.wav
1.32 MB
blues.00011.wav
1.32 MB

오디오 데이터에 대한 이해

[1]:
```
import librosa

y, sr = librosa.load('Data/genres_original/reggae/reggae.00036.wav')

print(y)
print(len(y))
print('Sampling rate (Hz): %d' % sr)
```

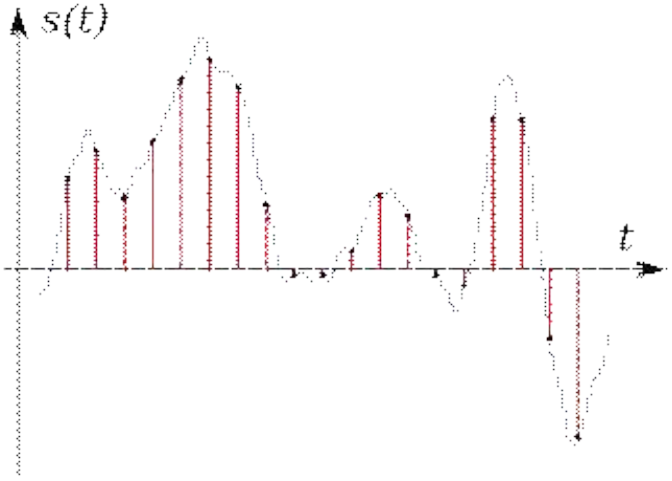
```
[0.02072144 0.04492188 0.05422974 ... 0.06912231 0.08303833 0.08572388]
661794
Sampling rate (Hz): 22050
```

[2]:
```
print('Audio length (seconds): %.2f' % (len(y) / sr))
```

```
Audio length (seconds): 30.01
```

[3]:
```
import IPython.display as ipd

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

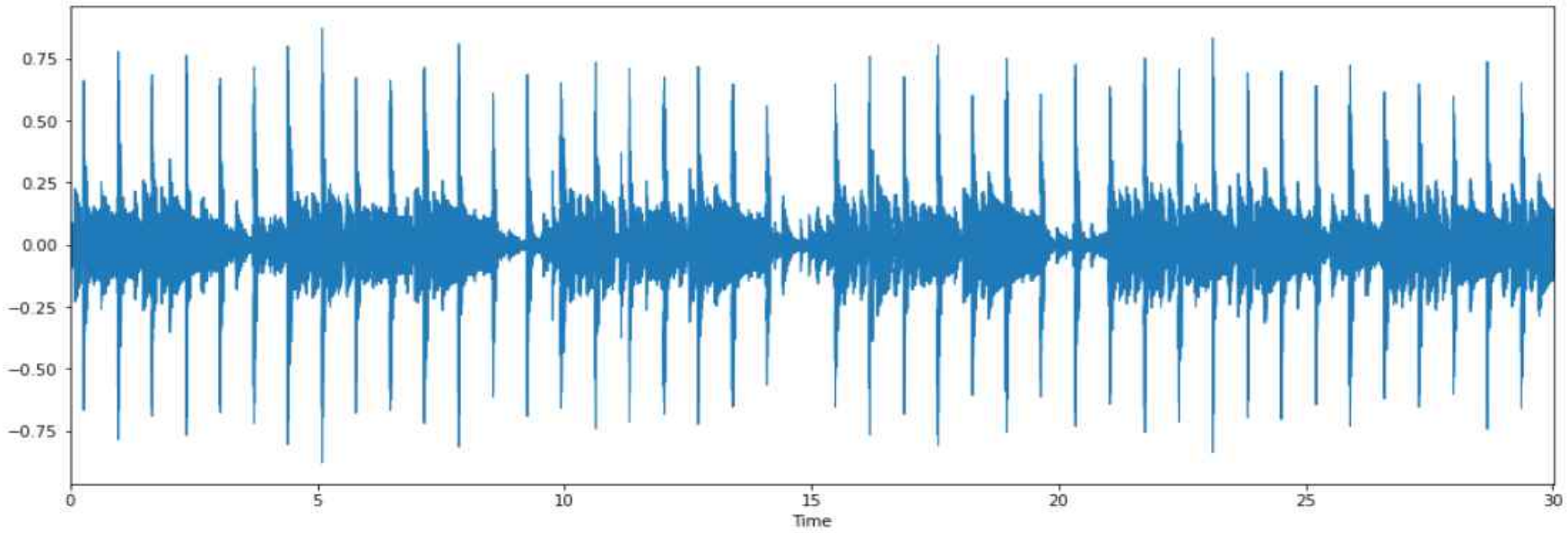
[3]:



이미지	오디오
픽셀의 값 (RGB)	소리의 세기 (y)
해상도	Sampling rate

2D 음파 그래프

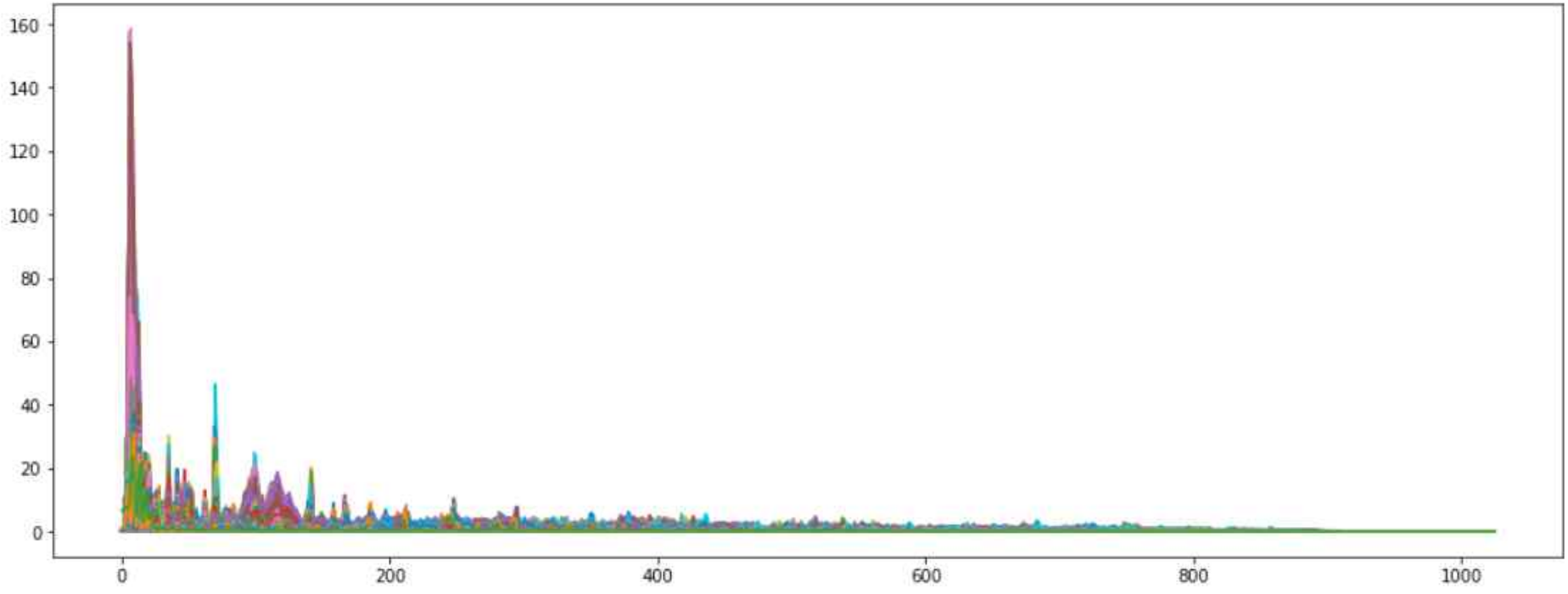
[4]:
```
import matplotlib.pyplot as plt
import librosa.display

plt.figure(figsize=(16, 6))
librosa.display.waveplot(y=y, sr=sr)
plt.show()
```



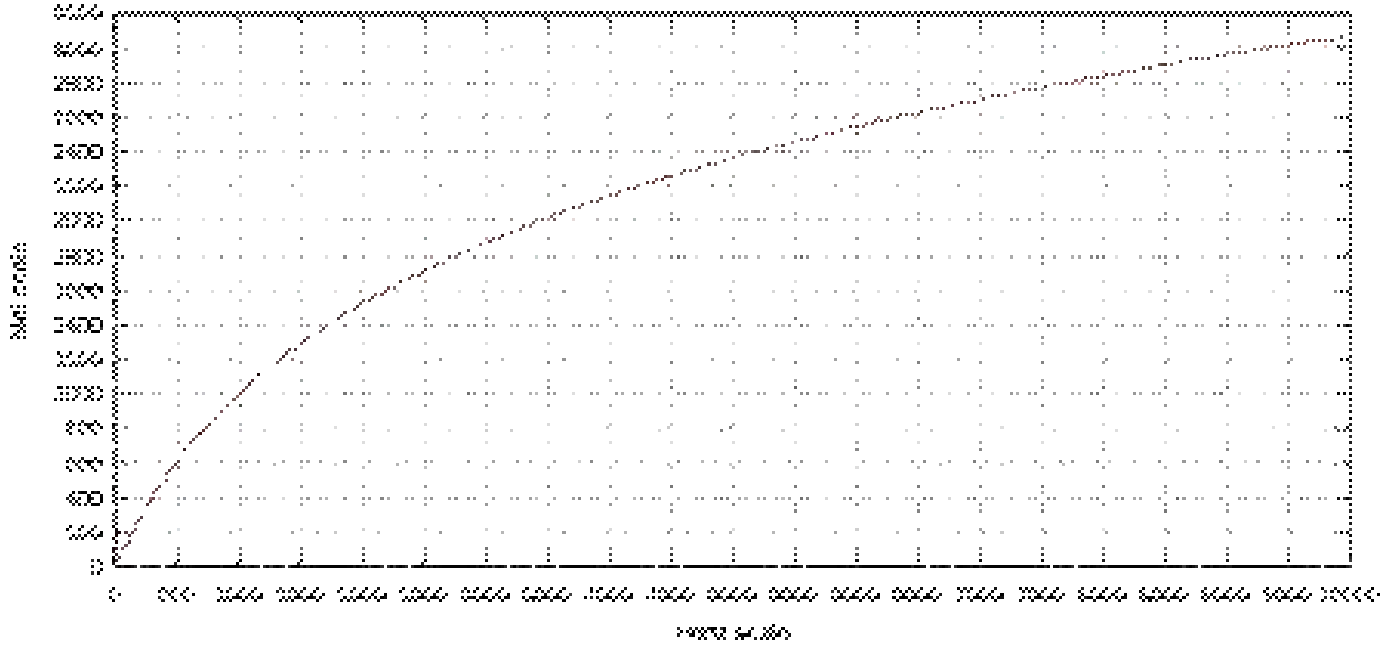
Fourier Transform (푸리에 변환)

[5]:
```
import numpy as np

D = np.abs(librosa.stft(y, n_fft=2048, hop_length=512))

print(D.shape)

plt.figure(figsize=(16, 6))
plt.plot(D)
plt.show()
```

```
(1025, 1293)
```



Mel-Scale



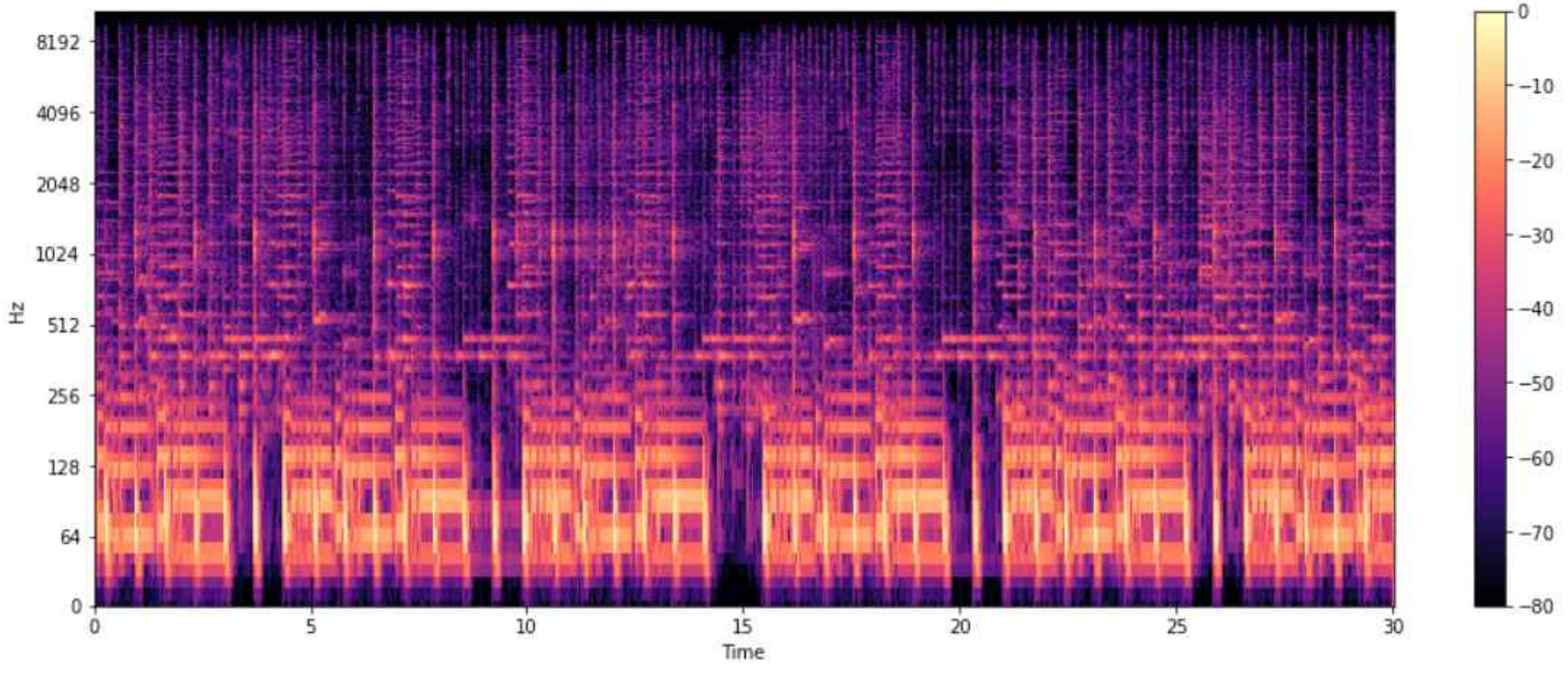
“인간이 이해하기 쉬운 mel scale”

- 높이가 다른 2개의 음을 사람에게 들려줬을 때, 사람이 인지하는 차이와 두 음의 실제 주파수 차이(컴퓨터가 인지하는 차이)를 따라가는 간단한 함수

- Mel Scale은 오디오 데이터를 분석하는 데 사용하기 위한 좋은 지표로써 다양한 음성처리 분야에서 사용.

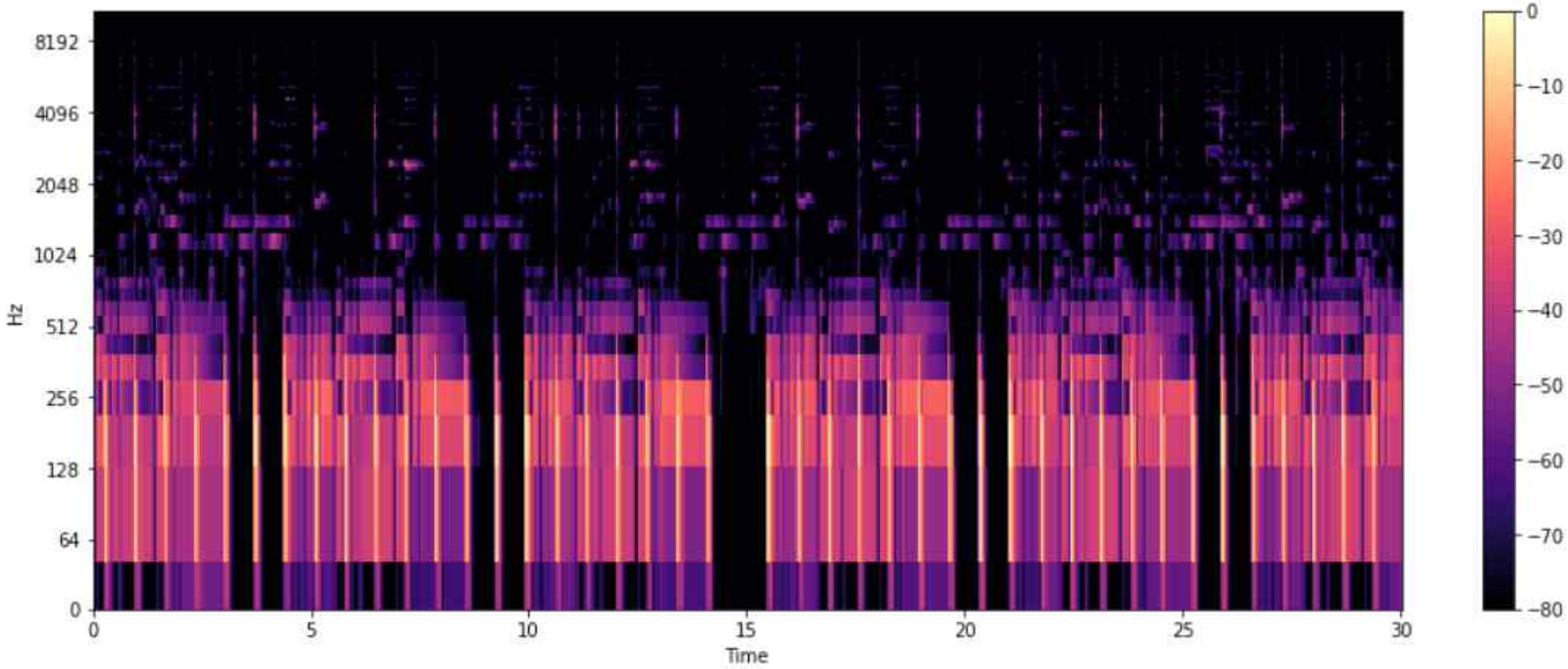
Spectogram

[6]:
```
DB = librosa.amplitude_to_db(D, ref=np.max)

plt.figure(figsize=(16, 6))
librosa.display.specshow(DB, sr=sr, hop_length=512, x_axis='time', y_axis='log')
plt.colorbar()
plt.show()
```



Mel-Spectogram

[7]:
```
S = librosa.feature.melspectrogram(y, sr=sr)
S_DB = librosa.amplitude_to_db(S, ref=np.max)

plt.figure(figsize=(16, 6))
librosa.display.specshow(S_DB, sr=sr, hop_length=512, x_axis='time', y_axis='log')
plt.colorbar()
plt.show()
```



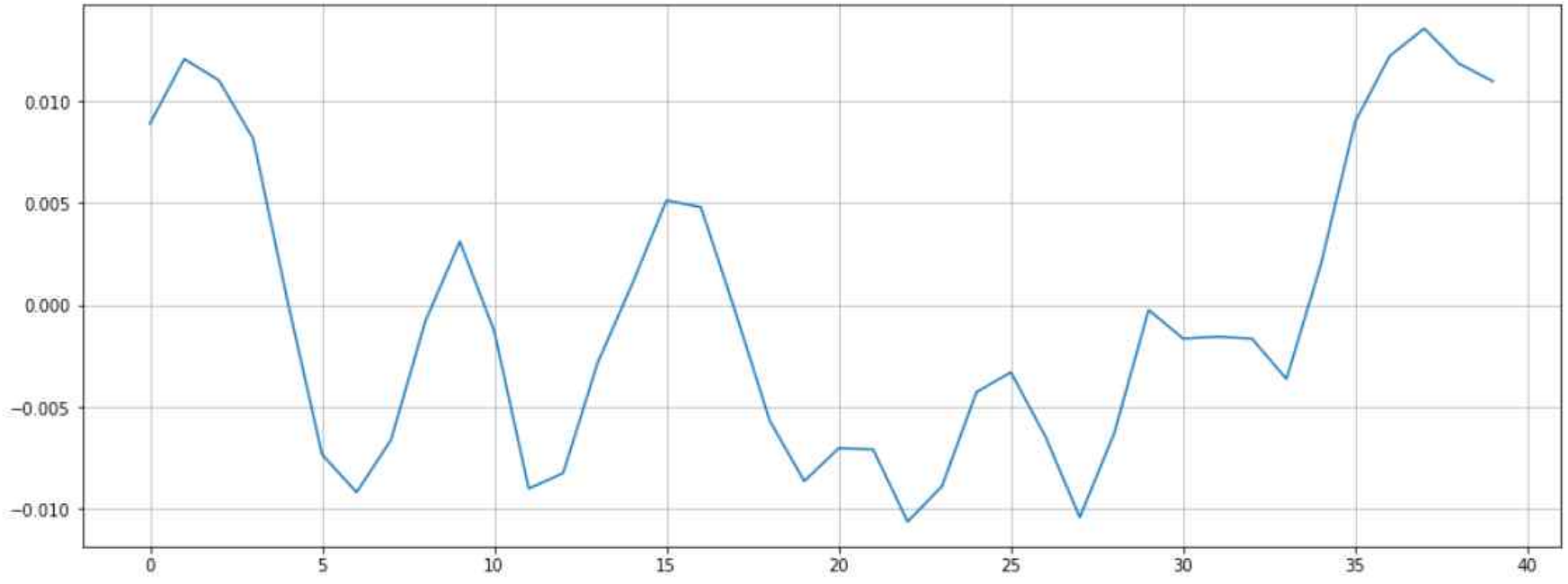
음성 특성 벡터 – Tempo , Zero Crossing Rate

[9]:
```
tempo, _ = librosa.beat.beat_track(y, sr=sr)
print(tempo)
```

```
107.666015625
```

[10]:
```
zero_crossings = librosa.zero_crossings(y, pad=False)

print(zero_crossings)
print(sum(zero_crossings))
```

```
[False False False ... False False False]
39405
```

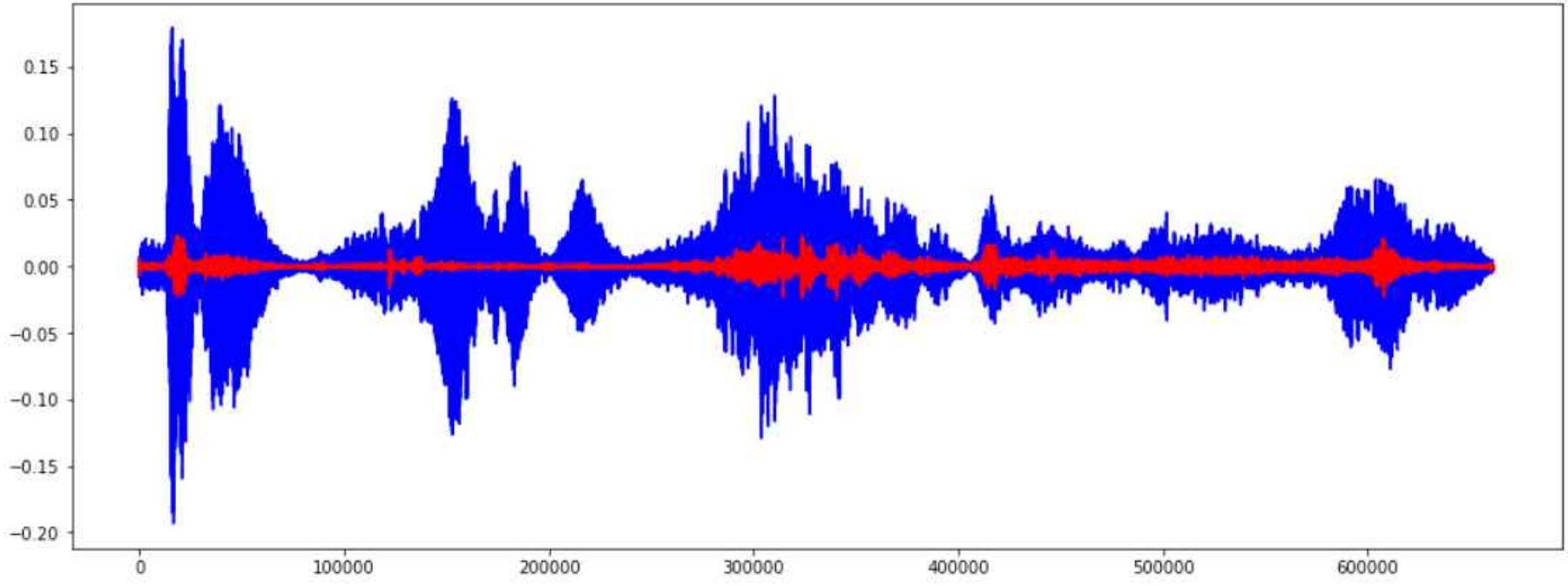


[12]:
```
zero_crossings = librosa.zero_crossings(y[n0:n1], pad=False)
print(sum(zero_crossings))
```

```
6
```

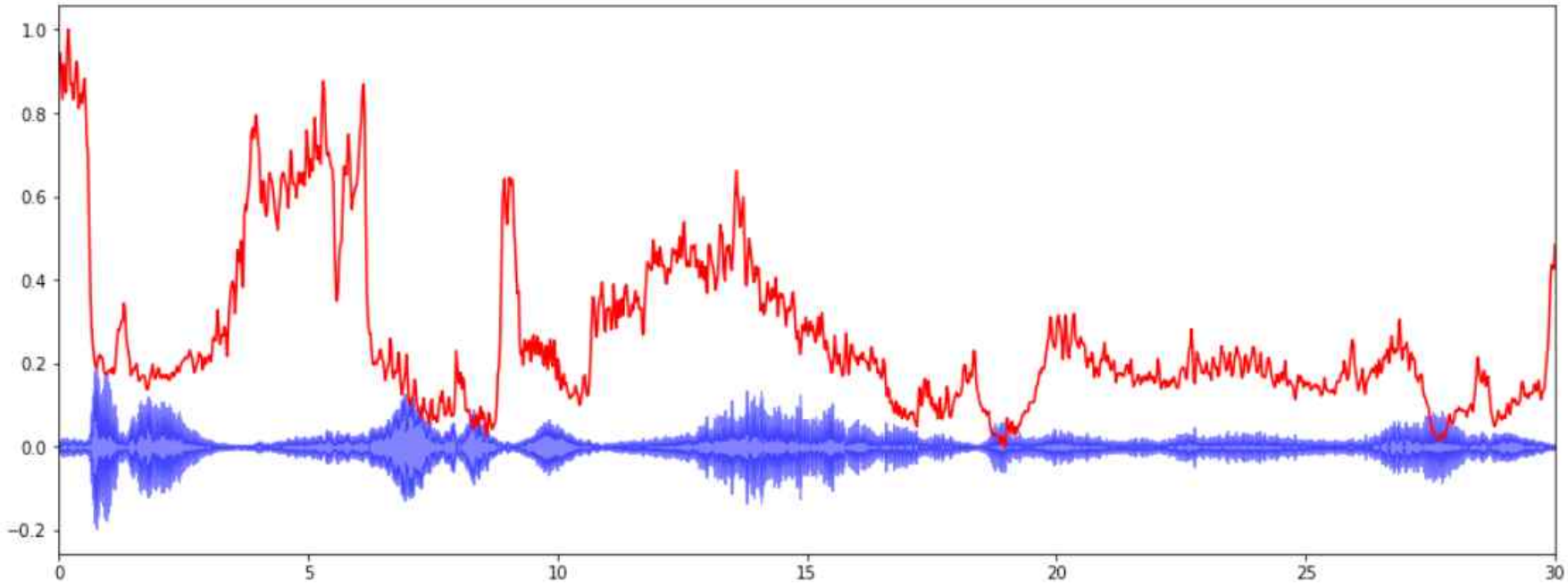
음성 특성 벡터 – Harmonic and Percussive component

[13]:
```
y_harm, y_perc = librosa.effects.hpss(y)

plt.figure(figsize=(16, 6))
plt.plot(y_harm, color='b')
plt.plot(y_perc, color='r')
plt.show()
```



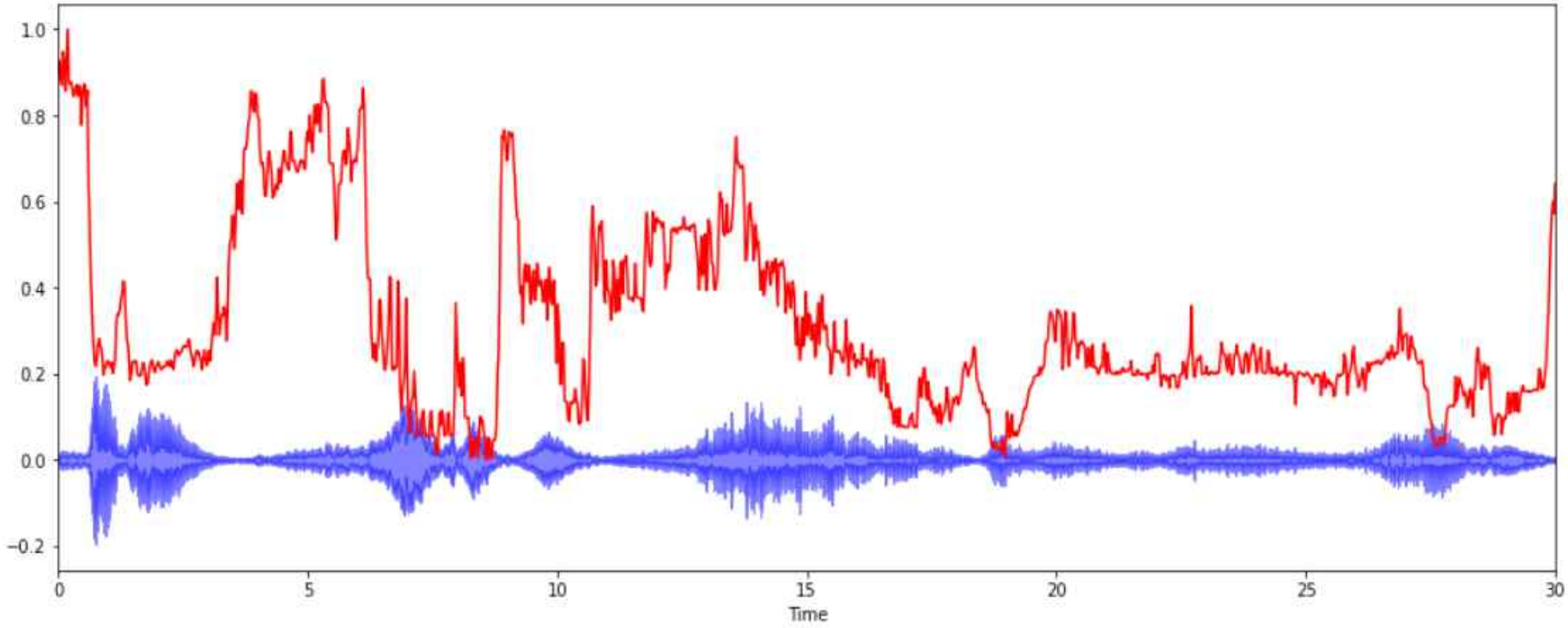
음성 특성 벡터 – Spectral Centroid

[14]:
```
spectral_centroids = librosa.feature.spectral_centroid(y, sr=sr)[0]

# Computing the time variable for visualization
frames = range(len(spectral_centroids))

# Converts frame counts to time (seconds)
t = librosa.frames_to_time(frames)

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

plt.figure(figsize=(16, 6))
librosa.display.waveplot(y, sr=sr, alpha=0.5, color='b')
plt.plot(t, normalize(spectral_centroids), color='r')
plt.show()
```



음성 특성 벡터 – Spectral Rolloff

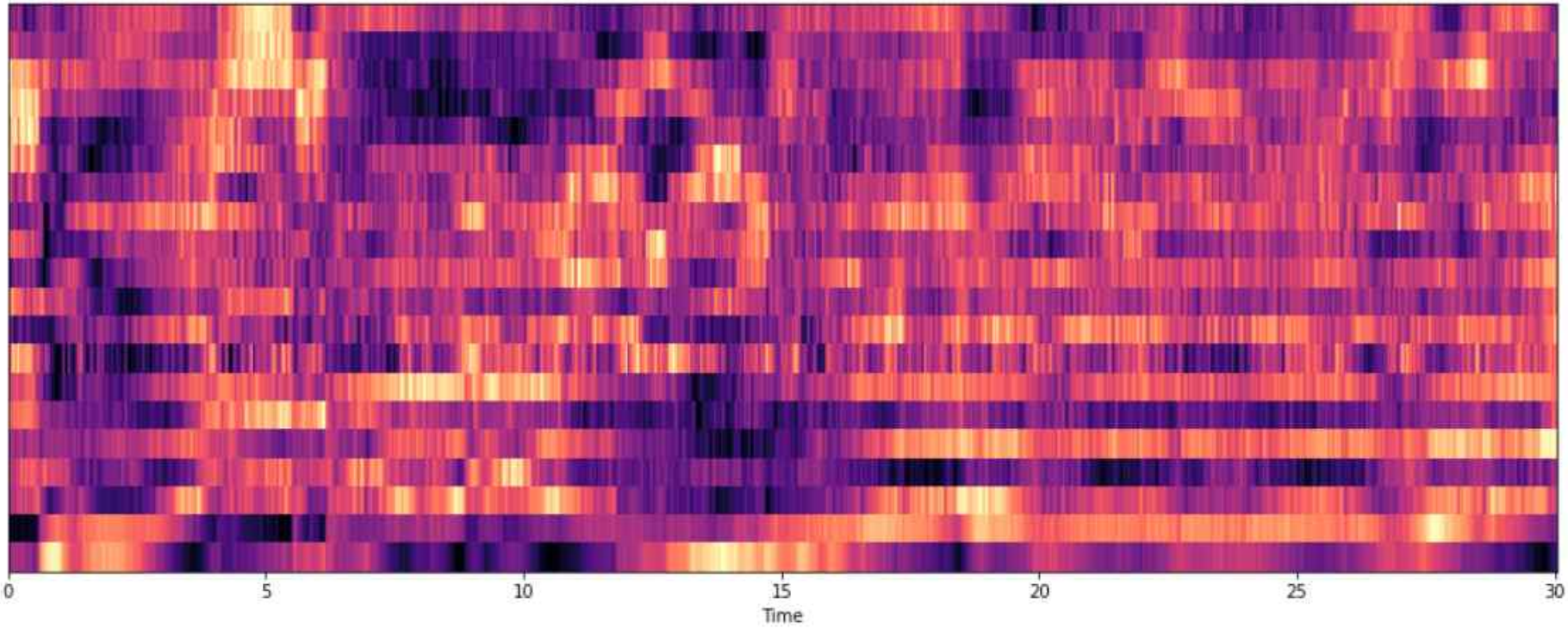
[15]:
```
spectral_rolloff = librosa.feature.spectral_rolloff(y, sr=sr)[0]

plt.figure(figsize=(16, 6))
librosa.display.waveplot(y, sr=sr, alpha=0.5, color='b')
plt.plot(t, normalize(spectral_rolloff), color='r')
plt.show()
```



음성 특성 벡터 - MFCCs

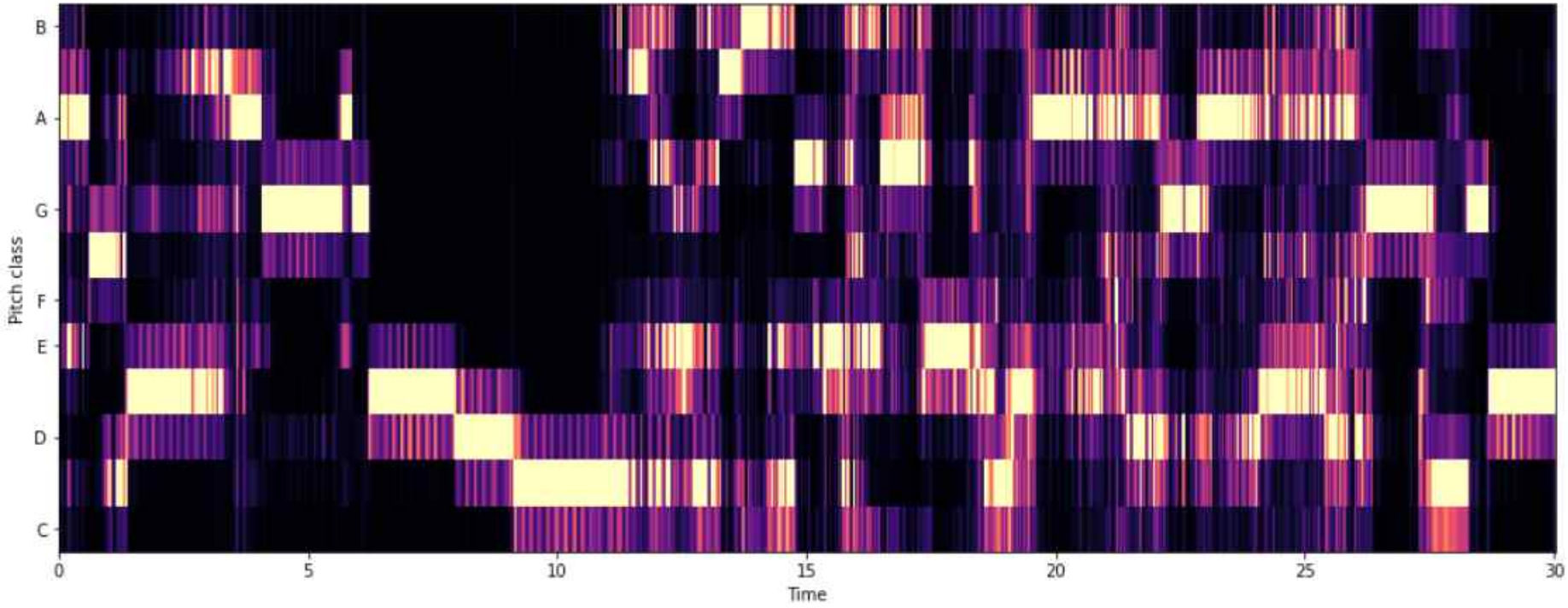
[16]:
```
mfccs = librosa.feature.mfcc(y, sr=sr)
mfccs = normalize(mfccs, axis=1)

print('mean: %.2f' % mfccs.mean())
print('var: %.2f' % mfccs.var())

plt.figure(figsize=(16, 6))
librosa.display.specshow(mfccs, sr=sr, x_axis='time')
plt.show()
```

```
mean: 0.48
var: 0.04
```



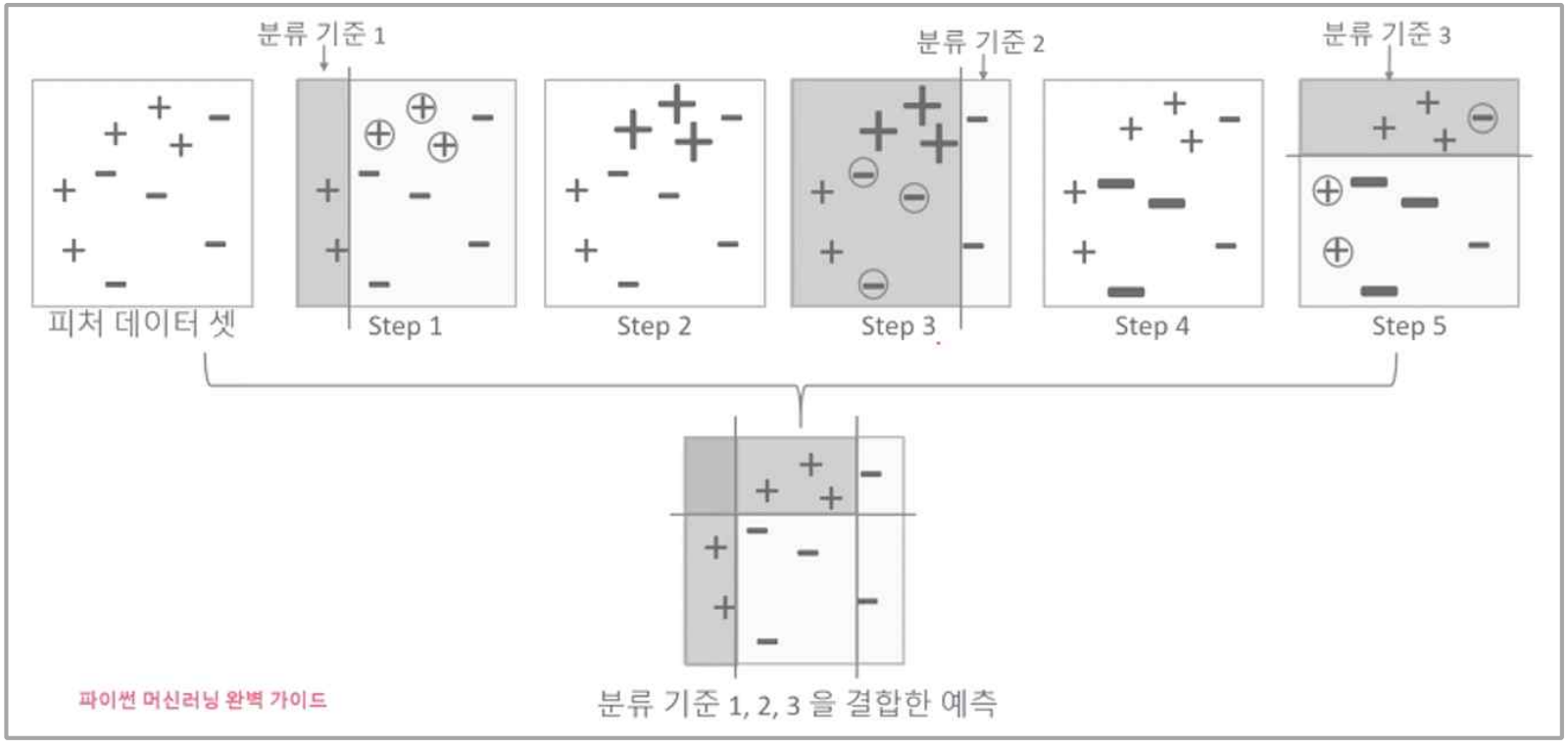
음성 특성 벡터 - Chroma

[17]:
```
chromagram = librosa.feature.chroma_stft(y, sr=sr, hop_length=512)

plt.figure(figsize=(16, 6))
librosa.display.specshow(chromagram, x_axis='time', y_axis='chroma', hop_length=512)
plt.show()
```



"XGBoost 모델"

- Gradient Boosting Model의 단점을 개선한 모델
  - 느린 속도, 과적합 문제



CODE

[18]:
```
import pandas as pd

df = pd.read_csv('Data/features_3_sec.csv')

df.head()
```

[18]:

	filename	length	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_va
0	blues.00000.0.wav	66149	0.335406	0.091048	0.130405	0.003521	1773.065032	167541.63086
1	blues.00000.1.wav	66149	0.343065	0.086147	0.112699	0.001450	1816.693777	90525.69086
2	blues.00000.2.wav	66149	0.346815	0.092243	0.132003	0.004620	1788.539719	111407.43761
3	blues.00000.3.wav	66149	0.363639	0.086856	0.132565	0.002448	1655.289045	111952.28451
4	blues.00000.4.wav	66149	0.335579	0.088129	0.143289	0.001701	1630.656199	79667.26765

5 rows × 60 columns

[19]:
```
X = df.drop(columns=['filename', 'length', 'label'])
y = df['label']

scaler = sklearn.preprocessing.MinMaxScaler()
np_scaled = scaler.fit_transform(X)

X = pd.DataFrame(np_scaled, columns=X.columns)

X.head()
```

[19]:

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_me
0	0.355399	0.716757	0.293133	0.107955	0.262173	0.034784	0.4592
1	0.367322	0.670347	0.253040	0.044447	0.270969	0.018716	0.4708
2	0.373159	0.728067	0.296753	0.141663	0.265293	0.023073	0.4940
3	0.399349	0.677066	0.298024	0.075042	0.238427	0.023187	0.4552
4	0.355668	0.689113	0.322308	0.052149	0.233460	0.016451	0.4516

5 rows × 57 columns

CODE

[20]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=2021)

print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

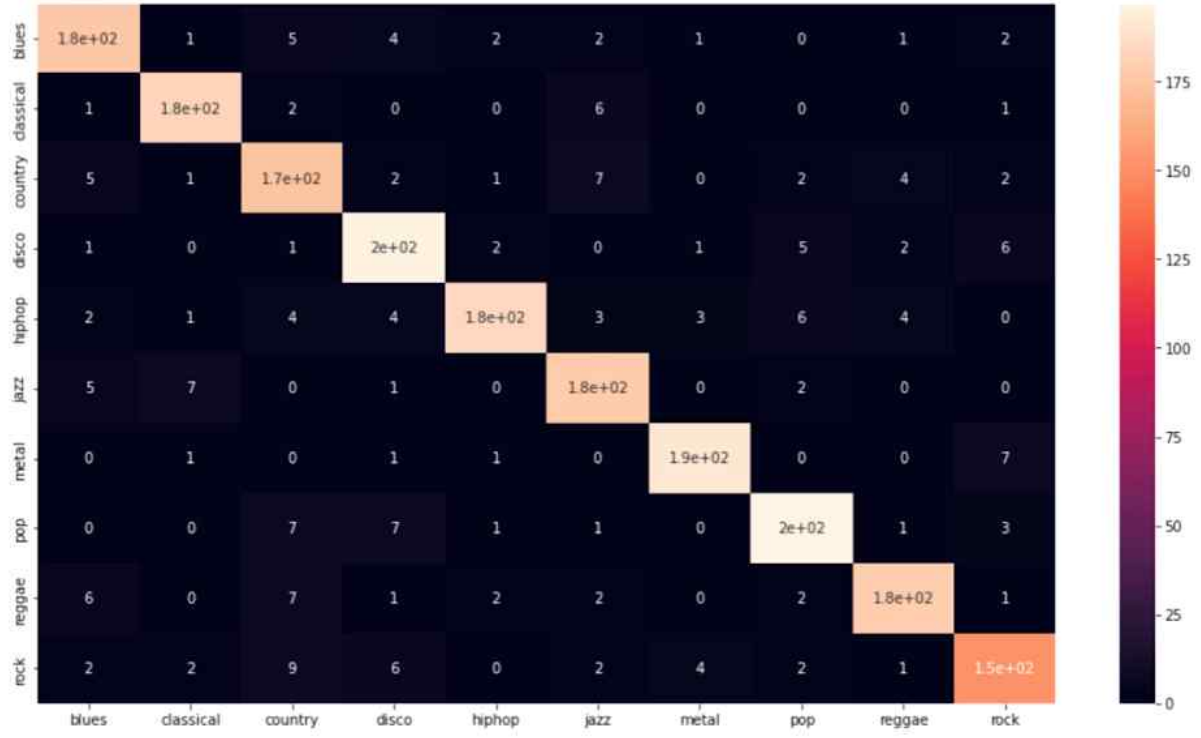
```
(7992, 57) (7992,)
(1998, 57) (1998,)
```

[22]:
```
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

xgb = XGBClassifier(n_estimators=1000, learning_rate=0.05)
xgb.fit(X_train, y_train)

y_preds = xgb.predict(X_test)

print('Accuracy: %.2f' % accuracy_score(y_test, y_preds))
```

```
Accuracy: 0.90
```

CODE

[24]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns

cm = confusion_matrix(y_test, y_preds)

plt.figure(figsize=(16, 9))
sns.heatmap(
    cm,
    annot=True,
    xticklabels=["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop", "reggae", "rock"]
    yticklabels=["blues", "classical", "country", "disco", "hiphop", "jazz", "metal", "pop", "reggae", "rock"]
)
plt.show()
```



CODE

[26]:
```
for feature, importance in zip(X_test.columns, xgb.feature_importances_):
  print('%s: %.2f' % (feature, importance))
```

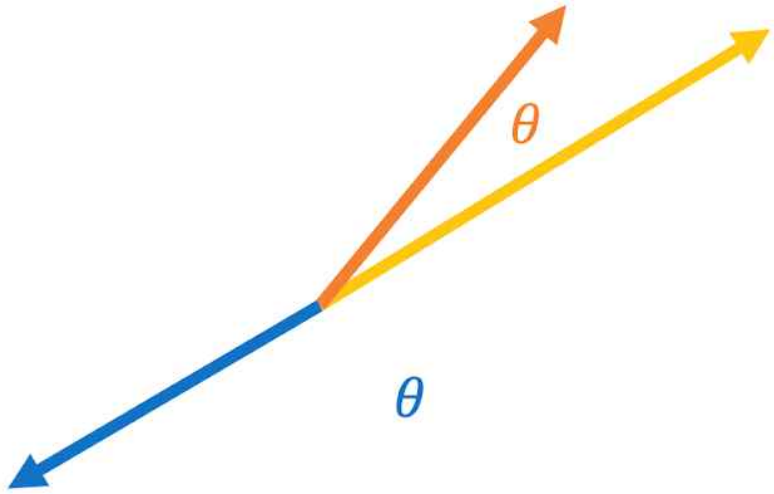
```
chroma_stft_mean: 0.04
chroma_stft_var: 0.03
rms_mean: 0.02
rms_var: 0.03
spectral_centroid_mean: 0.02
spectral_centroid_var: 0.02
spectral_bandwidth_mean: 0.06
spectral_bandwidth_var: 0.01
rolloff_mean: 0.03
rolloff_var: 0.03
zero_crossing_rate_mean: 0.02
zero_crossing_rate_var: 0.01
harmony_mean: 0.02
harmony_var: 0.02
perceptr_mean: 0.02
perceptr_var: 0.08
tempo: 0.02
mfcc1_mean: 0.03
mfcc1_var: 0.04
```

[28]:
```
df_30 = pd.read_csv('Data/features_30_sec.csv', index_col='filename')

labels = df_30[['label']]
df_30 = df_30.drop(columns=['length', 'label'])

df_30_scaled = sklearn.preprocessing.scale(df_30)

df_30 = pd.DataFrame(df_30_scaled, columns=df_30.columns)

df_30.head()
```

[28]:

	chroma_stft_mean	chroma_stft_var	rms_mean	rms_var	spectral_centroid_mean	spectral_centroid_var	spectral_bandwidth_me
0	-0.350137	0.312587	-0.010690	-0.061856	-0.583585	-0.848311	-0.456
1	-0.462482	1.117572	-0.532852	-0.186821	-0.938516	-0.234194	-0.386
2	-0.184225	-0.137701	0.679978	-0.084093	-0.906885	-0.781694	-0.940
3	0.319639	0.990659	0.154810	0.907029	-1.581429	-0.712095	-1.228
4	-0.859077	0.194163	-0.600165	-0.205909	-0.512542	-0.315178	-0.939

5 rows × 57 columns


θ
θ

CODE

[30]:
```
from sklearn.metrics.pairwise import cosine_similarity

similarity = cosine_similarity(df_30)

sim_df = pd.DataFrame(similarity, index=labels.index, columns=labels.index)

sim_df.head()
```

[30]:

filename	blues.00000.wav	blues.00001.wav	blues.00002.wav	blues.00003.wav	blues.00004.wav	blues.00005.wav	blues.0
filename							
blues.00000.wav	1.000000	0.049231	0.589618	0.284862	0.025561	-0.346688	
blues.00001.wav	0.049231	1.000000	-0.096834	0.520903	0.080749	0.307856	
blues.00002.wav	0.589618	-0.096834	1.000000	0.210411	0.400266	-0.082019	
blues.00003.wav	0.284862	0.520903	0.210411	1.000000	0.126437	0.134796	
blues.00004.wav	0.025561	0.080749	0.400266	0.126437	1.000000	0.556066	

5 rows × 1000 columns

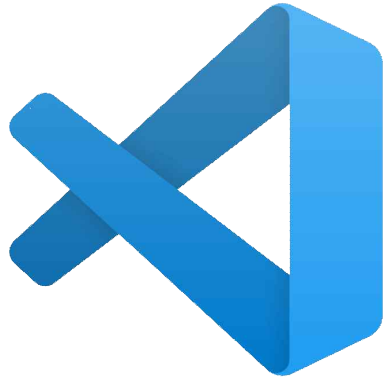
[31]:
```
def find_similar_songs(name, n=5):
  series = sim_df[name].sort_values(ascending=False)

  series = series.drop(name)

  return series.head(n).to_frame()

find_similar_songs('rock.00000.wav')
```

[31]:

filename	rock.00000.wav
rock.00079.wav	0.681819
rock.00026.wav	0.677842
country.00070.wav	0.675324
rock.00064.wav	0.662020
rock.00096.wav	0.654814

“Flask”

파이썬 웹 프레임워크



1. 가상 환경 루트 디렉토리 생성

```
C:\> mkdir venvs
C:\> cd venvs
```

2. 가상 환경 만들기

```
C:\venvs> python -m venv myproject
```

3. 가상 환경 진입 & 벗어나기

```
C:\venvs> cd C:\venvs\myproject\Scripts
C:\venvs\myproject\Scripts> activate
(myproject) C:\venvs\myproject\Scripts>
```

4. 가상 환경에 플라스크 설치하기

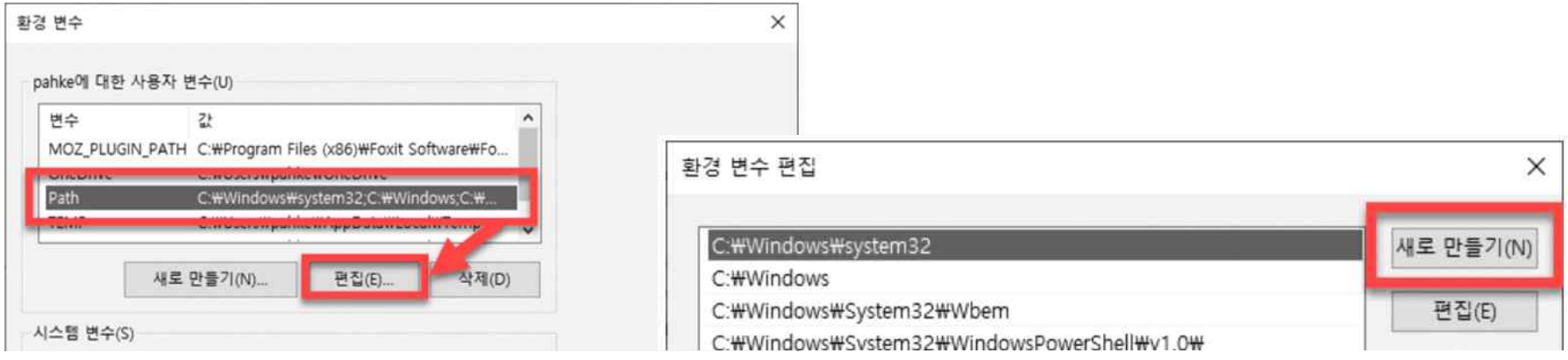
```
(myproject) C:\venvs\myproject\Scripts> pip install Flask
```

1. 프로젝트 루트 디렉토리 생성

```
C:\> mkdir projects
C:\> cd projects
```

1-1. 시스템 환경변수 설정

```
C:\projects> C:\venvs\myproject\Scripts\activate
(myproject) C:\projects>
```



C:₩venvs 가상환경 루트 디렉토리 경로를 추가

1. 프로젝트 루트 디렉토리 생성

```
C:\> mkdir projects
C:\> cd projects
```

1-1. 시스템 환경변수 설정

2. 프로젝트 루트 디렉토리 안에서 가상 환경 진입하기

```
C:\projects> C:\venvs\myproject\Scripts\activate
(myproject) C:\projects>
```

3. 프로젝트 디렉토리 만들기 (생략 가능)

```
(myproject) C:\projects> mkdir myproject
(myproject) C:\projects> cd myproject
(myproject) C:\projects\myproject>
```

4. 프로젝트 디렉토리에서 git clone

```
{프로젝트 디렉토리 경로}> git clone https://github.com/AnJW-HGU/2021AI-Music.git
```

```
{프로젝트 디렉토리 경로₩2021AI-Music₩project}> set FLASK_APP=app
{프로젝트 디렉토리 경로₩2021AI-Music₩project}> set FLASK_ENV=development
{프로젝트 디렉토리 경로₩2021AI-Music₩project}> flask run
```

```
http://localhost:5000/
```

1. 인공지능이 소리를 인식하고 분석하는 방식에 대해 배움,
모델을 학습시키는 과정을 통해 인공지능의 설계 이해 ↑.

2. 인공지능이 음악을 듣고 조화로움의 정도 또는 신남의 정도, 슬픔의 정도 등을 판단할 수 있게 하는 기술의 발판 가능

3. 물체 인식의 정확도 ↑
예) 생물 인식 – 영상 및 이미지 분석 + 오디오 분석
예) 이미지 인식이 어려운 환경 - 생물의 종 분포 및 파악에 활용 가능

Reference

Dateset : https://www.kaggle.com/andradaolteanu/gtzan-dataset-music-genre-classification

Open Source : https://www.kaggle.com/andradaolteanu/work-w-audio-data-visualise-classify-recommend

Flask : https://wikidocs.net/81039

Pickle : https://gaussian37.github.io/ml-sklearn-saving-model/

GTZAN을 활용한 음악 장르 분류 및 추천

모두를 위한 인공지능 활용 10조

안지원 이종무 이향우