Deep Learning을 이용한 음악 장르 분석

전자공학과 201724101 강감찬

목차

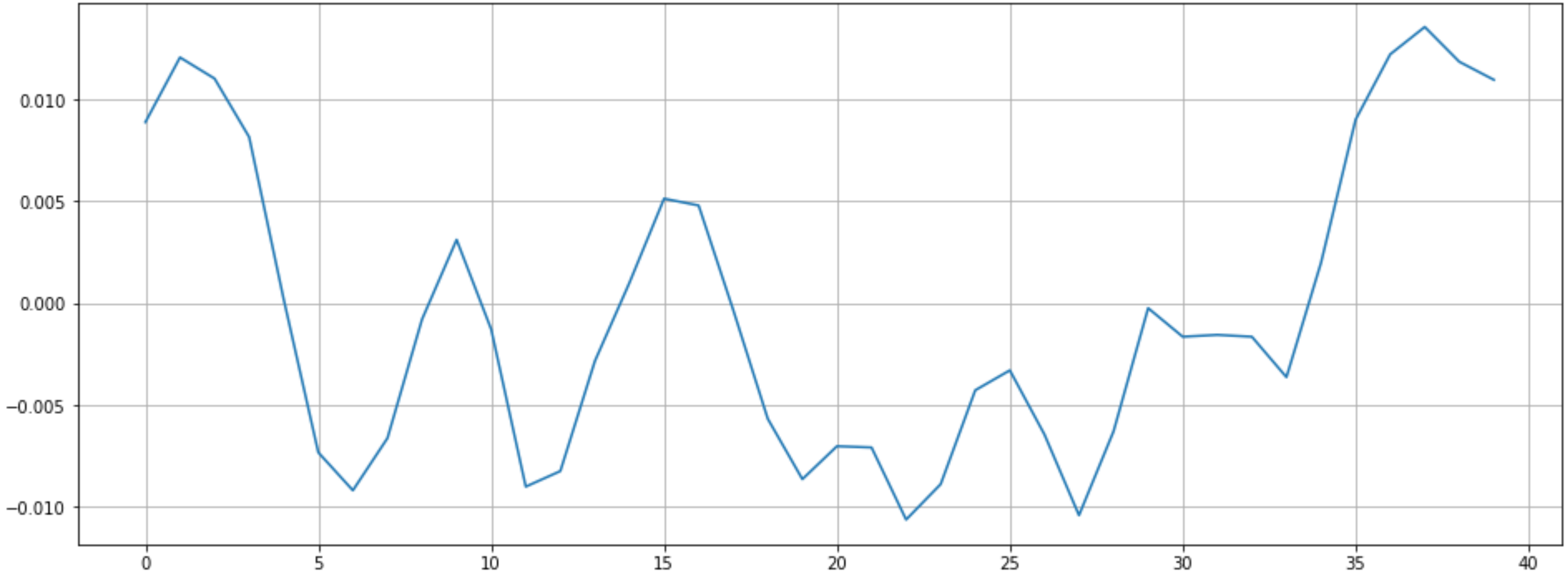
1. 과제 목표 및 필요성

음원의 파형 분석
-> BART 모델 이용

2. 이론 및 코드 개요

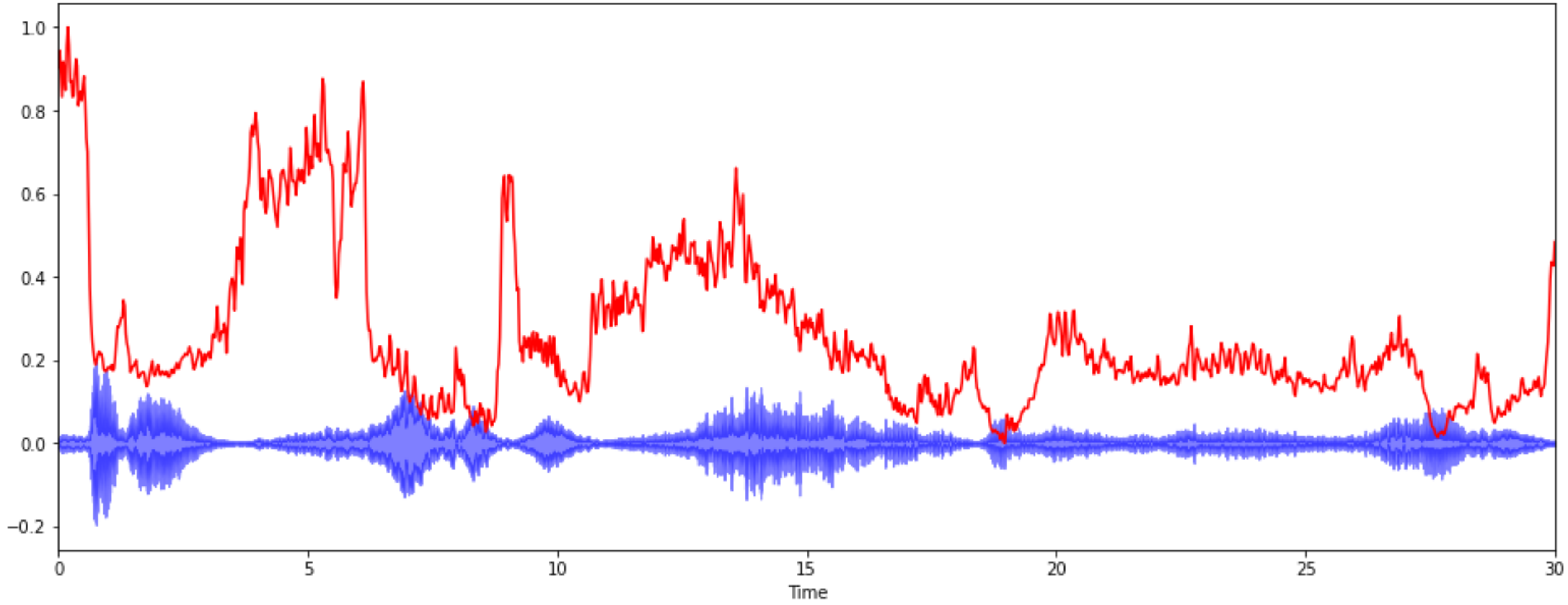
Tempo : 곡의 빠르기

Zero crossing rate : 파형이 t축을 지나는 비율

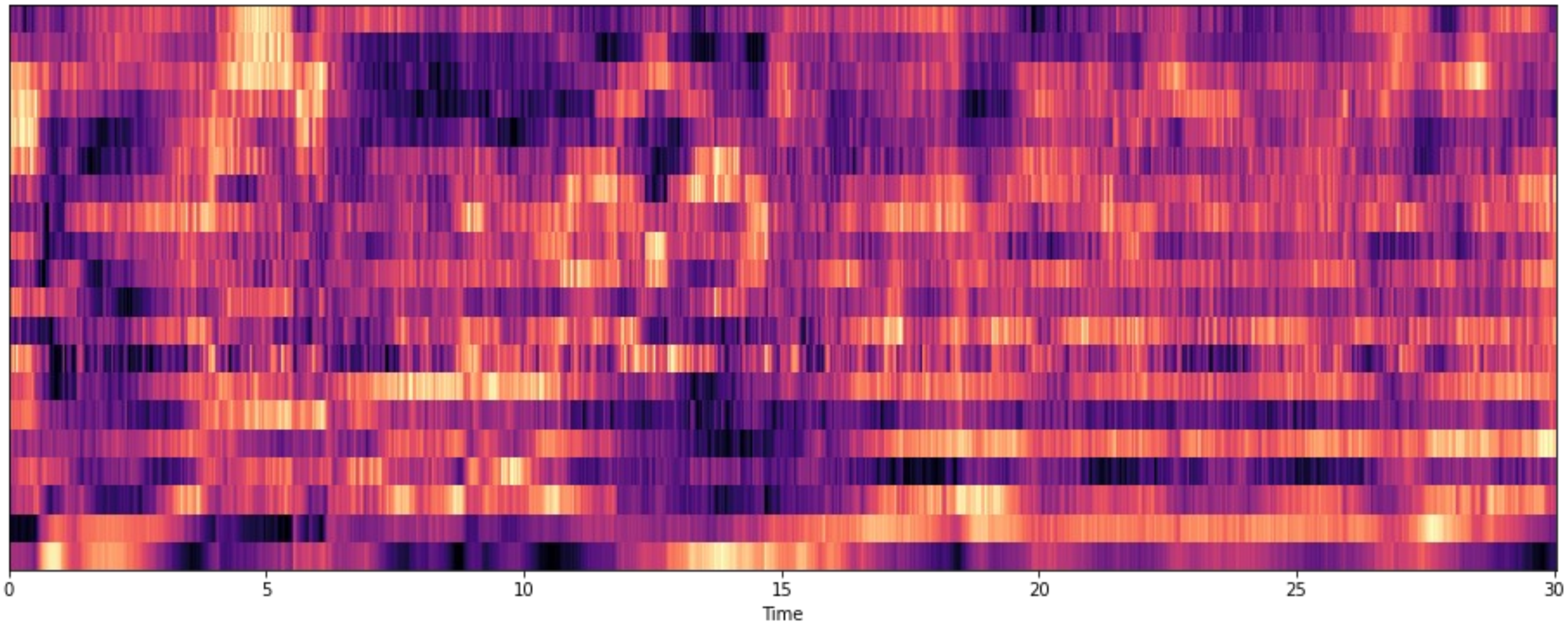


Spectral centroid : 소리의 무게 중심(주파수의 가중 평균)

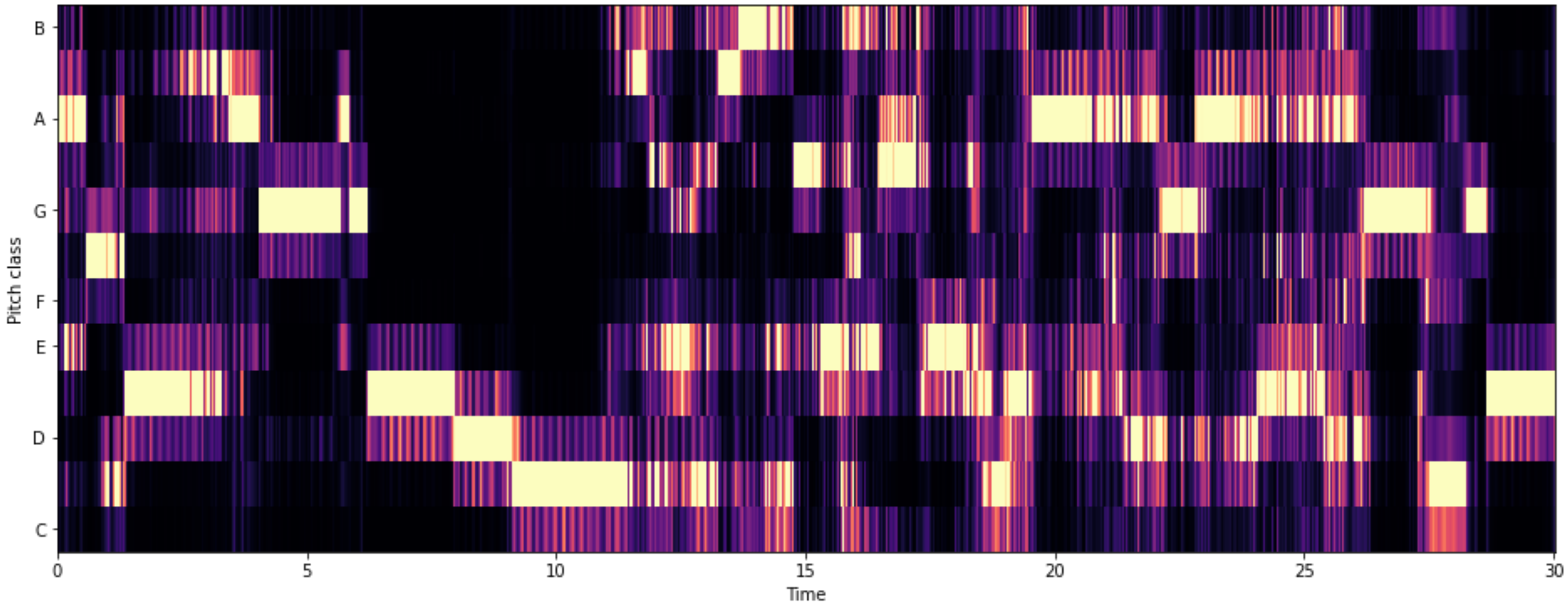
Spectral roll off : 총 Spectral Energy에서 85%이하 집중 정도



MFCCs : 사람의 청각 인식을 모방하여 추출한 음성 정보



Chroma frequency : 주파수 분포를 12음계로 대응시킨 데이터



wav file $\longrightarrow$ Function $\longrightarrow$ csv data

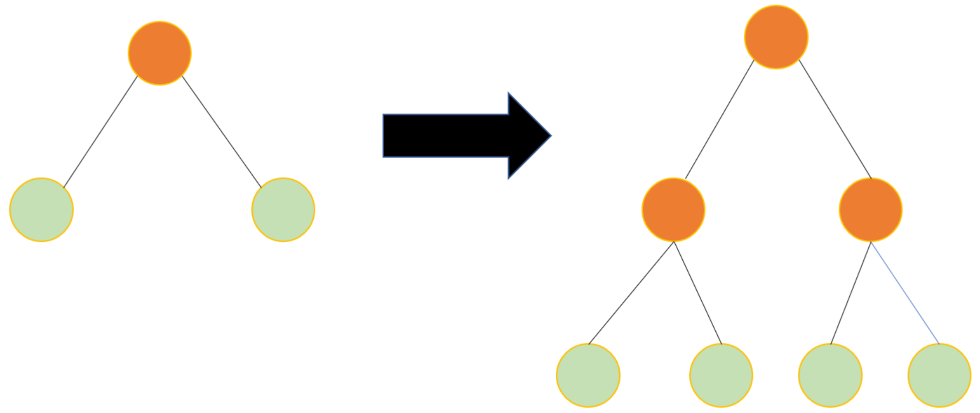
csv data

	filename	length	chroma_st	chroma_st	rms_mean	rms_var	spectral_ce	spectral_ce	spectral_b	spectral_b	rolloff_me	rolloff_var	zero_cross	zero_cross	harmony_r	harmony_v	perceptr_r	perceptr_v	tempo	mfcc1_me	mfcc1_var
1	filename	length	chroma_st	chroma_st	rms_mean	rms_var	spectral_ce	spectral_ce	spectral_b	spectral_b	rolloff_me	rolloff_var	zero_cross	zero_cross	harmony_r	harmony_v	perceptr_r	perceptr_v	tempo	mfcc1_me	mfcc1_var
2	blues.0000	66149	0.335406	0.091048	0.130405	0.003521	1773.065	167541.6	1972.744	117335.8	3714.56	1080790	0.081851	0.000558	-7.85E-05	0.008354	-6.82E-05	0.005535	129.1992	-118.628	2440.287
3	blues.0000	66149	0.343065	0.086147	0.112699	0.00145	1816.694	90525.69	2010.052	65671.88	3869.682	672244.8	0.087173	0.00103	-9.89E-05	0.00495	-0.0001	0.004854	123.0469	-125.591	2038.344
4	blues.0000	66149	0.346815	0.092243	0.132003	0.00462	1788.54	111407.4	2084.565	75124.92	3997.639	790712.7	0.071383	0.000425	-6.56E-05	0.012476	6.51E-05	0.004357	123.0469	-132.442	3798.532
5	blues.0000	66149	0.363639	0.086856	0.132565	0.002448	1655.289	111952.3	1960.04	82913.64	3568.3	921652.4	0.069426	0.000304	-1.38E-05	0.008318	1.83E-05	0.005927	123.0469	-118.231	2508.781
6	blues.0000	66149	0.335579	0.088129	0.143289	0.001701	1630.656	79667.27	1948.504	60204.02	3469.993	610211.1	0.070095	0.000289	4.05E-05	0.009634	-0.00011	0.005833	123.0469	-105.968	2118.92
7	blues.0000	66149	0.37667	0.089702	0.132618	0.003583	1994.915	211700.6	2152.768	74263.87	4371.986	1067105	0.09242	0.001174	-3.95E-05	0.008208	-0.00025	0.005501	129.1992	-100.753	1920.36
8	blues.0000	66149	0.379909	0.088827	0.130335	0.003166	1962.15	177443.1	2146.503	98020.54	4325.027	1172363	0.089573	0.001087	-5.29E-05	0.007008	0.000214	0.007208	123.0469	-101.773	3051.967
9	blues.0000	66149	0.33188	0.092119	0.1406	0.002546	1701.891	35678.13	1979.388	36670.73	3625.28	317461.7	0.070936	0.000231	1.90E-05	0.009667	-8.97E-05	0.005745	123.0469	-109.165	1993.801
10	blues.0000	66149	0.347877	0.094209	0.13313	0.002538	1746.474	138073.9	1887.62	117069.9	3586.935	1057633	0.087669	0.000822	-8.14E-05	0.007613	-5.87E-05	0.007005	123.0469	-113.373	1945.647
11	blues.0000	66149	0.358061	0.082957	0.115312	0.001846	1763.949	61493.42	1874.196	51944.92	3505.523	445157.9	0.101777	0.000713	-0.00012	0.006161	0.000166	0.004469	123.0469	-125.533	3038.313
12	blues.0000	66149	0.402401	0.09034	0.093024	0.003876	1279.182	406513.8	1921.306	196573.4	2890.253	3380315	0.039566	0.001595	2.50E-05	0.00569	-4.99E-05	0.002913	103.3594	-266.122	15292.04
13	blues.0000	66149	0.345507	0.091037	0.094656	0.001495	1513.764	214768.8	2091.433	121877	3479.186	2144222	0.049992	0.000628	8.26E-06	0.004774	-1.55E-05	0.00259	67.99959	-202.107	5132.732
14	blues.0000	66149	0.338119	0.083682	0.097776	0.001386	1308.87	154209.1	1508.009	111306.1	2557.068	817418.1	0.077302	0.001931	0.000215	0.005176	-0.00029	0.002935	107.666	-197.394	4885.877
15	blues.0000	66149	0.330751	0.093936	0.085365	0.002641	1479.53	646868.4	1906.454	357705.4	3227.993	4172253	0.055886	0.001882	0.000156	0.004099	-5.84E-05	0.00326	112.3471	-228.195	9476.916
16	blues.0000	66149	0.348027	0.100096	0.088437	0.002204	1729.64	568827.5	2150.738	153824.5	4099.673	2497105	0.065302	0.001956	0.000195	0.004171	-0.00017	0.002758	161.499	-205.457	6792.198
17	blues.0000	66149	0.329435	0.104149	0.093201	0.001929	1492.089	228951.6	2189.639	151468.9	3682.592	2907021	0.037804	0.000173	0.000184	0.004165	-0.00021	0.003391	184.5703	-223.847	5531.173
18	blues.0000	66149	0.343666	0.096098	0.089783	0.002173	1571.42	257630.1	2183.513	158620.7	3832.248	3048531	0.05181	0.000641	0.000166	0.003767	-0.00019	0.003592	107.666	-219.99	6966.423
19	blues.0000	66149	0.327732	0.099648	0.110633	0.003306	1877.638	510311.4	2326.81	135085.8	4509.136	3124113	0.070602	0.003136	0.00022	0.006645	-0.0003	0.003701	69.83742	-170.457	4736.644
20	blues.0000	66149	0.329428	0.090254	0.111013	0.001072	1665.6	239973.6	2168.032	107476.1	3973.042	2468428	0.059135	0.000545	0.00016	0.007322	-5.87E-05	0.002429	103.3594	-166.231	3048.577
21	blues.0000	66149	0.292987	0.082145	0.094924	0.003031	1373.438	210606.4	1929.319	165808.6	3187.908	2096863	0.047119	0.000403	1.97E-05	0.005569	-0.00018	0.002877	66.25601	-218.009	10085.36
22	blues.0000	66149	0.366614	0.090509	0.152206	0.002467	1676.978	433761.8	1794.856	56792.98	3297.562	1078959	0.077986	0.004532	1.15E-06	0.010506	-2.08E-06	0.006715	161.499	-114.294	3255.78
23	blues.0000	66149	0.384461	0.084645	0.167405	0.001362	1337.449	69939.4	1658.057	50969.92	2662.498	475920.8	0.059285	0.000243	-1.23E-05	0.01258	-0.00013	0.007017	161.499	-104.854	2786.482
24	blues.0000	66149	0.379043	0.086581	0.176237	0.001924	1590.858	153760.8	1779.645	79475.88	3222.941	923048.5	0.074309	0.000508	6.77E-06	0.015579	-0.00014	0.007247	161.499	-83.0871	2493.011

XGBoost : 앙상블의 부스팅 기법의 한 종류

- Loss -> training data
- Level-wise 방식 모델 학습

균형 트리 분할(Level Wise)



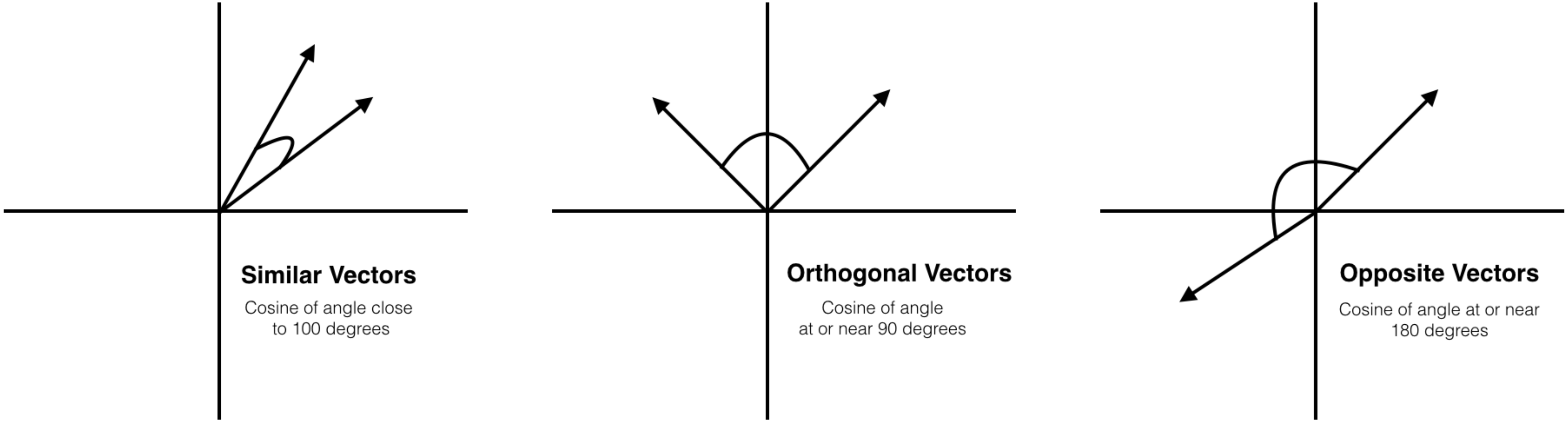
https://blog.naver.com/PostView.naver?blogId=fbfbf1&logNo=222487957675&categoryNo=34&parentCategoryNo=37&viewDate=&currentPage=1&postListTopCurrentPage=1&from=postView

TRAIN 80

TEST 20

가장 높은 accuracy : 0.88

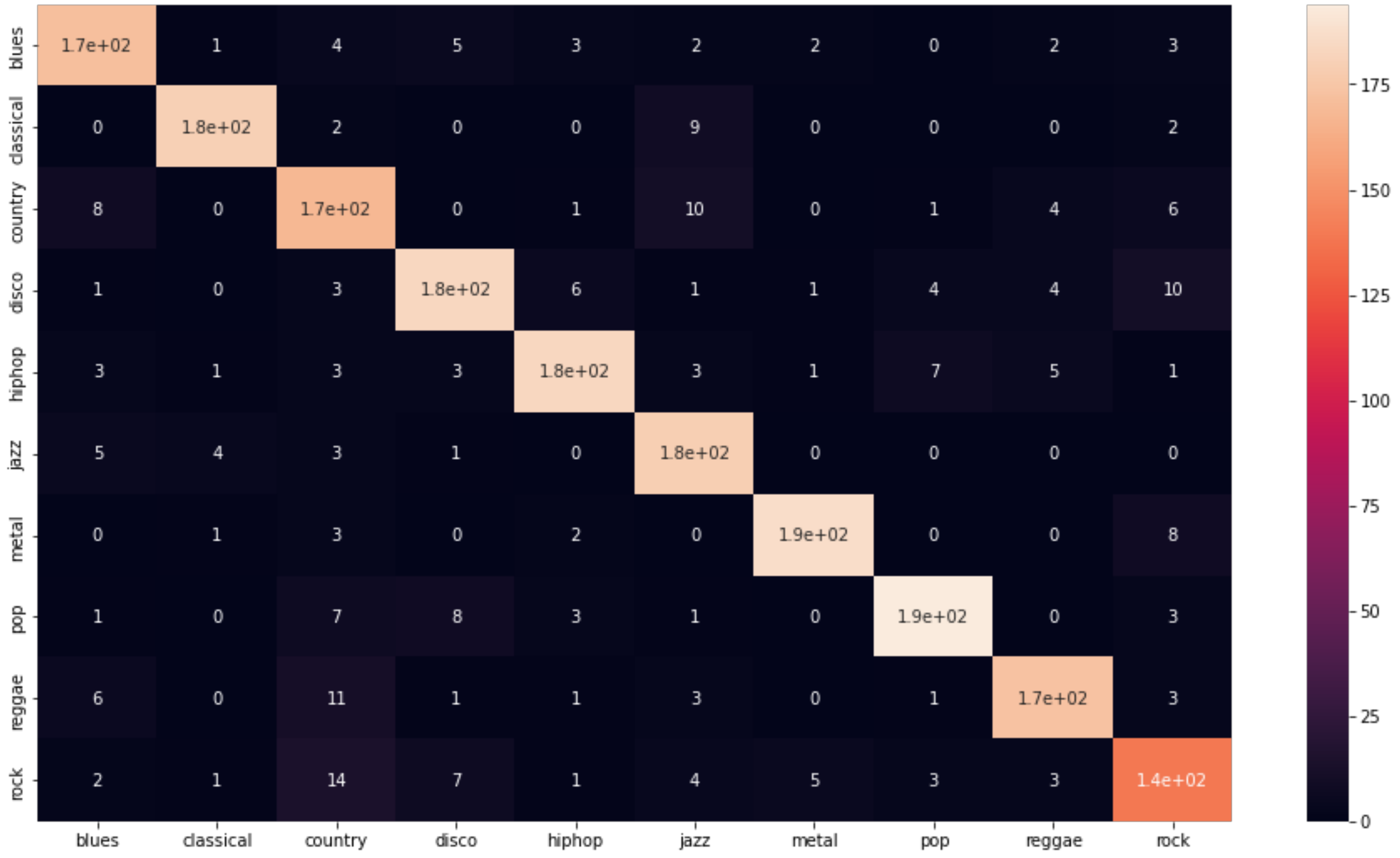
Cosine Similarity



https://needjarvis.tistory.com/665

Data 바탕으로 곡 추천

3. 코드 실행 결과



blues
classical
country
disco
hiphop
jazz
metal
pop
reggae
rock
1.7e+02 1 4 5 3 2 2 0 2 3
0 1.8e+02 2 0 0 9 0 0 0 2
8 0 1.7e+02 0 1 10 0 1 4 6
1 0 3 1.8e+02 6 1 1 4 4 10
3 1 3 3 1.8e+02 3 1 7 5 1
5 4 3 1 0 1.8e+02 0 0 0 0
0 1 3 0 2 0 1.9e+02 0 0 8
1 0 7 8 3 1 0 1.9e+02 0 3
6 0 11 1 1 3 0 1 1.7e+02 3
2 1 14 7 1 4 5 3 3 1.4e+02
175
150
125
100
75
50
25
0

(NEW)새로운 음원 데이터 입력

```
[27] import soundfile as sf
     def trim_audio_data(audio_file, save_file):
         sr = 48000
         sec = 30

         y, sr = librosa.load(audio_file, sr=sr)

         ny = y[:sr*sec]

         sf.write(save_file + '.wav', ny, sr, format='WAV', endian='LITTLE', subtype='PCM_16')
```

```
[28] audio_file = '/content/drive/MyDrive/BMX.wav'
     save_file = '/content/drive/MyDrive/BMX_30_secs'
     trim_audio_data(audio_file, save_file)
```

(NEW)새로운 음원 데이터 전처리 및 Feature Vector 생성

```
y, sr = librosa.load('/content/drive/MyDrive/BMX_30_secs.wav')

print(y)
print(len(y))
print('Sampling rate (Hz): %d' % sr)
print('Audio length (seconds): %.2f' % (len(y) / sr))

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

```
[ 0.          0.          0.          ...  0.07019664  0.04121341
 -0.10021631]
661500
Sampling rate (Hz): 22050
Audio length (seconds): 30.00
```

```
y_name = "BMX_30_secs.wav"

#chroma_stft data
y_chr_stft = librosa.feature.chroma_stft(y, sr=sr, hop_length=512)
y_chr_stft_mean = np.mean(y_chr_stft)
y_chr_stft_var= np.var(y_chr_stft)

#rms data
y_rms = librosa.feature.rms(y=y)
y_rms_mean = np.mean(y_rms)
y_rms_var = np.var(y_rms)

#spectral_centroid data
y_spec_cent = librosa.feature.spectral_centroid(y, sr=sr)[0]
y_spec_cent_mean = np.mean(y_spec_cent)
y_spec_cent_var = np.var(y_spec_cent)

#spectral_bandwidth
y_spec_bw = librosa.feature.spectral_bandwidth(y=y, sr=sr)
y_spec_bw_mean = np.mean(y_spec_bw)
y_spec_bw_var = np.var(y_spec_bw)

#rolloff
y_rolloff = librosa.feature.spectral_rolloff(y, sr=sr)[0]
y_rolloff_mean = np.mean(y_rolloff)
y_rolloff_var = np.var(y_rolloff)

#zero_crossing_rate
y_zero_cro_rate = librosa.feature.zero_crossing_rate(y)
y_zero_cro_rate_mean = np.mean(y_zero_cro_rate)
y_zero_cro_rate_var = np.var(y_zero_cro_rate)

#harmony
y_harm, y_perc = librosa.effects.hpss(y)
y_harm_mean = np.mean(y_harm)
y_harm_var = np.var(y_harm)

#tempo
y_tempo, _ = librosa.beat.beat_track(y, sr=sr)

y_hat = np.array([y_chr_stft_mean, y_chr_stft_var, y_rms_mean, y_rms_var, y_spec_cent_mean, y_spec_cent_var,
        y_spec_bw_mean, y_spec_bw_var, y_rolloff_mean, y_rolloff_var, y_zero_cro_rate_mean, y_zero_cro_rate_var,
        y_harm_mean, y_harm_var, y_tempo])

#mfcc
for i in range(1,21):
  y_mfccn = librosa.feature.mfcc(y=y, sr=sr, n_mfcc=i)
  y_hat = np.append(y_hat, np.mean(y_mfccn))
  y_hat = np.append(y_hat, np.var(y_mfccn))
#  print('mfcc',i,': ',y_mfcc)
```

A2 | BMX_30_secs.wav

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1		chroma_st	chroma_st	rms_mean	rms_var	spectral_ce	spectral_ce	spectral_ba	spectral_ba	rolloff_me	rolloff_var	zero_cross	zero_cross	harmony_r	harmony_v	tempo	mfcc1_me	mfcc1_var	mfcc2_me	mfcc2_var	mfcc3_me
2	BMX_30_s	0.415614	0.079995	0.105926	0.001747	2788.625	696893.7	2071.637	199988.6	4699.438	1956364	0.185401	0.007095	1.15E-05	0.008006	89.10291	-110.076	12402.77	-23.9384	14289.49	-36.1634
3	blues.0000	0.350088	0.088757	0.130228	0.002827	1784.166	129774.1	2002.449	85882.76	3805.84	901505.4	0.083045	0.000767	-4.53E-05	0.008172	123.0469	-113.571	2564.208	121.5718	295.9138	-19.1681
4	blues.0000	0.340914	0.09498	0.095948	0.002373	1530.177	375850.1	2039.037	213843.8	3550.522	2977893	0.05604	0.001448	0.00014	0.005099	67.99959	-207.502	7764.555	123.9913	560.2599	8.955127
5	blues.0000	0.363637	0.085275	0.17557	0.002746	1552.812	156467.6	1747.702	76254.19	3042.26	784034.5	0.076291	0.001007	2.11E-06	0.016342	161.499	-90.7226	3319.045	140.4463	508.765	-29.0939
6	blues.0000	0.404785	0.093999	0.141093	0.006346	1070.107	184355.9	1596.413	166441.5	2184.746	1493194	0.033309	0.000423	4.58E-07	0.019054	63.02401	-199.544	5507.517	150.0909	456.5054	5.662678
7	blues.0000	0.308526	0.087841	0.091529	0.002303	1835.004	343399.9	1748.172	88445.21	3579.758	1572978	0.101461	0.001954	-1.76E-05	0.004814	135.9992	-160.338	5195.292	126.2196	853.7847	-35.5878
8	blues.0000	0.302456	0.087532	0.103494	0.003981	1831.994	1030482	1729.653	201910.5	3481.518	3274440	0.094042	0.006233	1.96E-07	0.008083	69.83742	-177.774	7307.417	118.2055	3195.213	-17.5659
9	blues.0000	0.291328	0.093981	0.141874	0.008803	1459.366	437859.4	1389.009	185023.2	2795.611	1621442	0.073052	0.001909	-9.67E-06	0.016923	71.77734	-190.052	9656.535	130.2891	1932.796	-36.3695
10	blues.0000	0.307955	0.092903	0.131822	0.005531	1451.667	449568.2	1577.271	168211.9	2954.837	1629130	0.061442	0.001849	1.94E-05	0.013223	92.28516	-179.347	6573.922	136.469	1479.249	-26.6731
11	blues.0000	0.408879	0.086512	0.142416	0.001507	1719.369	163282.8	2031.74	105542.7	3782.316	1262917	0.064025	0.000731	2.26E-06	0.012702	83.35433	-121.364	2622.195	122.5067	414.5059	-14.7382
12	blues.0000	0.27395	0.092316	0.081314	0.004347	1817.151	298236.1	1973.773	114070.1	3943.491	1367638	0.079175	0.001378	-1.66E-06	0.007221	80.74951	-213.24	6133.722	115.1656	798.2787	-11.6934
13	blues.0001	0.303993	0.094703	0.142865	0.0092	1409.955	205666.8	1512.31	145008.8	2765.901	1427844	0.062959	0.000442	-0.00011	0.021768	161.499	-173.792	8144.119	137.1867	1083.69	-23.724
14	blues.0001	0.367152	0.102329	0.065741	0.0025	1352.66	512135.1	1756.78	220626.8	2880.778	2933408	0.043931	0.001896	8.72E-06	0.005012	161.499	-287.924	5931.473	124.0188	996.1501	5.469892
15	blues.0001	0.269391	0.093902	0.119046	0.004053	1360.641	257065.4	1567.565	45236.51	2738.639	952898.9	0.069095	0.002381	-2.90E-06	0.01253	184.5703	-207.223	4575.482	132.8306	666.2899	-15.4288
16	blues.0001	0.264713	0.091176	0.11317	0.004189	1324.293	171475	1828.039	78278.57	2710.269	1074400	0.051402	0.001255	-1.60E-05	0.012681	107.666	-209.826	1969.792	124.4632	541.6416	10.34245
17	blues.0001	0.329036	0.108111	0.067055	0.002717	1171.848	147278.5	1705.519	96584.44	2344.246	877596.6	0.045037	0.00156	-0.00012	0.004717	151.9991	-305.655	8194.195	113.0863	683.2161	12.11443
18	blues.0001	0.26969	0.094989	0.080624	0.002829	1420.365	199959	1731.164	81354.26	2929.964	700099	0.063701	0.001941	-1.26E-05	0.004725	99.38401	-238.516	4378.724	120.1102	728.7023	-4.22936
19	blues.0001	0.304186	0.098493	0.081472	0.002641	1454.964	152317.1	1825.798	116774.7	3009.461	1010620	0.061344	0.001419	-2.35E-05	0.004507	198.768	-233.759	4963.079	120.6751	562.0911	-3.17481
20	blues.0001	0.302124	0.093346	0.093306	0.002151	1088.844	231640.8	1410.984	69795.44	2134.993	1081720	0.048326	0.000845	-1.19E-05	0.006776	172.2656	-245.699	5194.902	140.4911	800.3101	-2.78975
21	blues.0001	0.269932	0.097658	0.079706	0.002956	1537.588	193102.1	2054.53	145776.7	3495.515	1827900	0.056142	0.001722	-4.58E-05	0.006934	83.35433	-252.493	2874.313	104.9371	603.6761	15.33231
22	blues.0001	0.257325	0.095963	0.09766	0.002575	1195.47	249522.9	1481.319	46970.54	2235.265	1208029	0.058886	0.001798	-2.59E-05	0.006598	123.0469	-236.657	5578.514	138.4093	536.373	-4.91291
23	blues.0002	0.302612	0.094329	0.075474	0.002499	1390.302	102976.5	1911.315	88916.8	3006.172	945949.5	0.052586	0.000616	-5.77E-06	0.004552	135.9992	-230.358	2988.245	127.0783	397.4443	7.123446
24	blues.0002	0.3211	0.103292	0.101132	0.003228	1046.55	291480.3	1481.058	70086.83	1830.331	1349215	0.047907	0.002331	-6.30E-06	0.008324	99.38401	-264.723	6342.384	138.8685	1001.582	11.19406

[33]
```
df_30_hat_new = pd.concat([df_y,df_30_hat])
df_30_hat_new.to_csv("Data/new_features.csv", mode='w')
df_30_scaled = sklearn.preprocessing.scale(df_30_hat_new)
```

[34]
```
from sklearn.metrics.pairwise import cosine_similarity

similarity = cosine_similarity(df_30_hat_new)

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

sim_df.head()
```

	BMX_30_secs.wav	blues.00000.wav	blues.00001.wav	blues.00002.wav	blues.00003.wav	blues.00004.wav	blues.00005.wav	blues.00006.wav	blues.00007.wav	blues.00008.wav	...	rock.00090.wav	rock.00091.wav	rock.
BMX_30_secs.wav	1.000000	0.980327	0.976410	0.989508	0.976127	0.991053	0.998547	0.996810	0.997301	0.977293	...	0.990468	0.991925	
blues.00000.wav	0.980327	1.000000	0.999581	0.998553	0.999663	0.996658	0.986370	0.992658	0.992093	0.999832	...	0.944840	0.996902	
blues.00001.wav	0.976410	0.999581	1.000000	0.997174	0.999233	0.995890	0.983952	0.989796	0.989373	0.999926	...	0.939337	0.995740	
blues.00002.wav	0.989508	0.998553	0.997174	1.000000	0.997164	0.999029	0.993551	0.997683	0.997402	0.997600	...	0.960986	0.999447	
blues.00003.wav	0.976127	0.999663	0.999233	0.997164	1.000000	0.994265	0.982280	0.990205	0.989358	0.999603	...	0.937559	0.994677	

5 rows × 1001 columns

```
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('BMX_30_secs.wav')
```

	BMX_30_secs.wav
country.00046.wav	0.999901
disco.00047.wav	0.999880
metal.00096.wav	0.999855
classical.00037.wav	0.999848

✓ 0초 오전 4:32에 완료됨

외부 파일

추천 5곡

4. 참고 자료 목록

- Oramas S, Barbieri F, Nieto O, Serra X. Multimodal deep learning for music genre classification. Transactions of the International Society for Music Information Retrieval. 2018
- 고려대학교 Data Mining Quality Analysis 세미나 자료
- Multimodal Deep Learning. Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, Andrew Y. Ng. 2011
- https://angeloyeo.github.io/2020/10/02/RBM.html
- A Comparative Study on Content-Based Music Genre Classification, Tao Li (2003)
- Deep Learning for Music, A Huang (2016)
- Deep Learning for Audio-Based Music Classification and Tagging, J Nam (2018)
- AUDIO-BASED MUSIC CLASSIFICATION WITH A PRETRAINED CONVOLUTIONAL NETWORK, S Dieleman (2011)
- https://scikit-learn.org/stable/
- https://numpy.org/
- https://xgboost.readthedocs.io/en/stable/