

BİL 460-VERİ MADENCİLİĞİ PROJESİ

MUSIC GENRES CLASSIFICATION AND CLUSTERING

ŞEVVAL YOĞURTCUOĞLU 100043533

ÖZGE ÇOKÇA 200002350

İÇİNDEKİLER

► GİRİŞ(INTRODUCTION):

1. MÜZİK NEDİR? TÜRLERİ NELERDİR?
2. NEDEN MÜZİK SINIFLANDIRMASI YAPILMALIDIR?
3. GERÇEK HAYATTA PROBLEMİN GÖRÜLDÜĞÜ ALANLAR
4. PROBLEM ÇÖZÜMÜ ADIMLARI

► DENEYSEL KURULUM(EXPERIMENTAL SETUP):

1. VERİNİN ANLATIMI

VERİNİN İŞLEMSİZ HAM HALİ,KARŞILAŞILAN ZORLUKLAR,SINIF SAYISI,OBJE SAYISI,ÖZNİTELİK SAYISI

2. KULLANILAN YÖNTEMLER

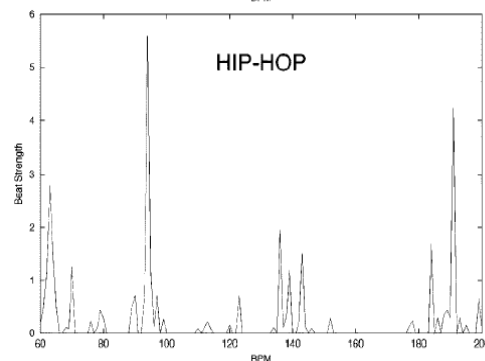
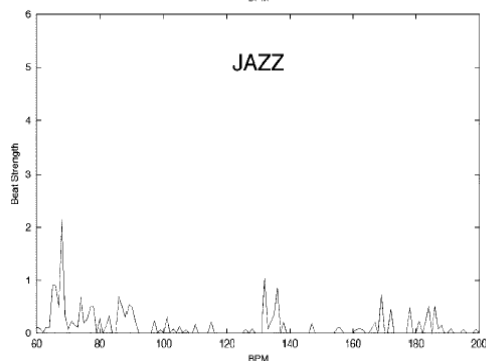
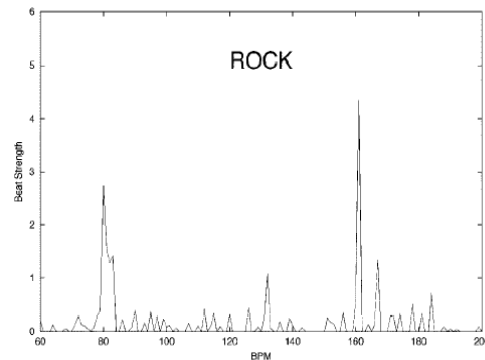
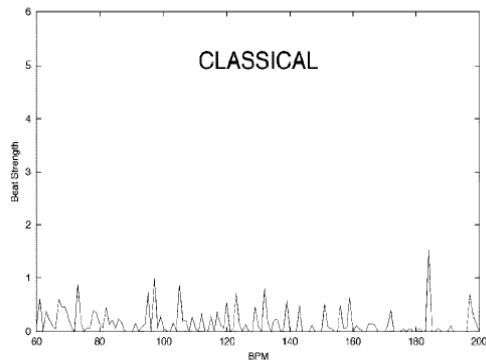
3. BULUNAN SONUÇLAR

► SONUÇ(CONCLUSION)

GİRİŞ (INTRODUCTION)

MÜZİK NEDİR? TÜRLERİ NELERDİR?

- Müzik, sesin biçim ve anlamlı titreşimler kazanmış halidir.



Audio classification hierarchy.

NEDEN MÜZİK SINIFLANDIRMASI YAPILMALIDIR?

Müzik hizmetleri(örneğin; Spotify, Fizy, vb.) , İnsanların en sevdikleri müzikleri dinleyebilmeleri için birincil araç haline geldi. Ancak aynı zamanda, müzik hizmetleri kullanıcıların zevklerine uygun müzik türleri aramaya çalışırken en uygununu bulmakta zorluk çekmiş ve zaman kaybı yaşamıştır. Bu nedenle, müzik hizmetleri kişiselleştirilmiş tavsiyelere izin vermek için müziği kategorize etme yollarını araştırmıştır.

Kullandığımız bu yöntem, ham ses bilgisinin belirli bir şarkıdaki doğrudan analizini içerir ve ham verileri çeşitli metrikler üzerinde sayısallaştırır. Bugün, GTZAN Tür Koleksiyonu veri setini kullanarak G. Tzanetakis ve P. Cook tarafından derlenen verileri inceleyeceğiz. Amacımız bu veri setine bakarak müziklerden oluşan ses sinyallerinin müzik türü olarak('Hip-Hop' 'Rock' vb.) olarak sınıflandırılmasıdır.

Bunu yaparken, verilerimizi nasıl temizlediğimizi, keşfedici veri görselleştirmeyi nasıl yaptığımızı, karar ağaçları ve çeşitli bazı basit makine öğrenme algoritmalarıyla verilerimize özellik çıkarımı nasıl yaptığımızı göstereceğiz.

GERÇEK HAYATTA PROBLEMİN GÖRÜLDÜĞÜ ALANLAR

Günümüzde müzik hizmetleri(Spotify, Shazam, Soundcloud), müşterilerine tavsiyelerde bulunabilmek için müzik sınıflandırması kullanmaktadırlar. Müzik türlerini belirlemek, bu yöndeki ilk adımdır.

Makine öğrenimi tekniklerinin, geniş veri havuzundan trendleri ve kalıpları çıkarmakta oldukça başarılı olduğu kanıtlanmıştır. Aynı ilkeler müzik analizinde de uygulanmaktadır.

PROBLEM ÇÖZÜMÜ ADIMLARI:

Bu projede Python'da bir ses/müzik sinyalinin nasıl analiz edileceğini inceleyeceğiz.

Daha sonra müzikleri farklı türlerde sınıflandırmak için öğrenilen becerileri kullandık.

Verilerimize;

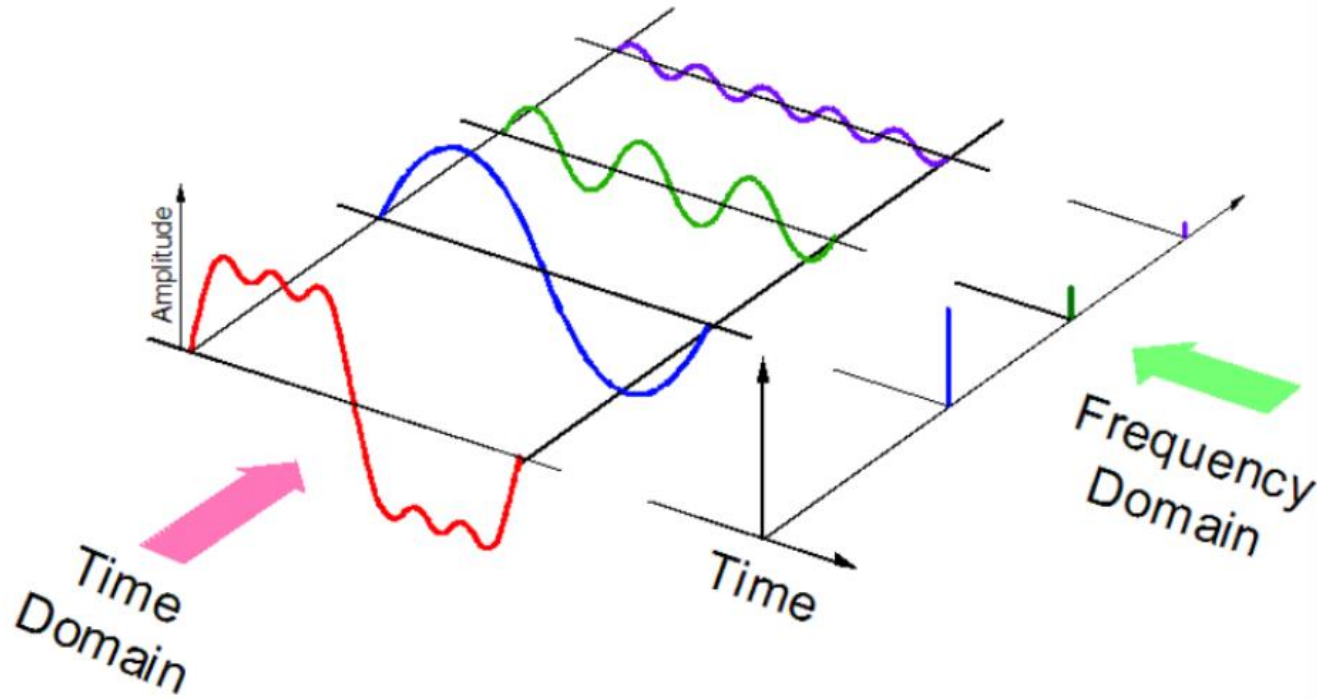
1. Önışlem (Pre-processing)
2. Öznitelik temsili (Feature Representation)
3. Öznitelik seçimi (Feature Selection)
4. Öznitelik çıkarımı (Feature Extraction)

Uygulayarak data setimizi oluşturduk.

Daha sonra öğrenme algoritmaları(Learning Methods) kullanarak müziklerimizi sınıflandırdık ve kümeledik. Performans Değerlendirmesi(Performance Evaluation) yaparak hangi öğrenme algoritmasının böyle bir veri setinde daha iyi performans gösterdiğini inceledik.


DENEYSEL KURULUM (EXPERIMENTAL SETUP)

AUDIO PROCESSING



► Audio, frekans, bant genişliği, desibel vb. gibi parametrelere sahip bir ses sinyali şeklinde temsil edilir.

Tipik bir ses sinyali, genlik ve zamanın bir fonksiyonu olarak ifade edilebilir.



Bu sesler, bilgisayarın bunları okumasını ve analiz etmesini mümkün kılan birçok formatta mevcuttur.

Bazı örnekler;

- mp3 format

- WMA(Windows media audio) format

- Wav(Waveform Audio File) format

Python, librosa ve pyAudio gibi ses işlemesi için kütüphanelere sahiptir.

Librosa:

Müzik ve ses analizi için kullanılan bir python paketidir.

Librosa Kütüphanesinin Yüklenmesi

```
!pip install librosa
```

DATASET

- ▶ 30 saniye uzunluğunda 1000 adet audio track dan oluşmaktadır.
- ▶ 10 farklı tür içermektedir. (**blues, classical, country, disco, hip-hop, jazz, reggae, rock, metal ve pop**)
- ▶ Her tür 100 ses dosyasından oluşmaktadır.

blues	2.12.2019 15:00	Dosya klasörü
classical	2.12.2019 15:00	Dosya klasörü
country	2.12.2019 15:00	Dosya klasörü
disco	2.12.2019 14:59	Dosya klasörü
hiphop	2.12.2019 14:59	Dosya klasörü
img_data	2.12.2019 15:23	Dosya klasörü
jazz	2.12.2019 14:59	Dosya klasörü
metal	2.12.2019 14:59	Dosya klasörü
pop	2.12.2019 14:59	Dosya klasörü
reggae	2.12.2019 14:59	Dosya klasörü
rock	2.12.2019 14:59	Dosya klasörü

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
blues.00009.wav
blues.00010.wav
blues.00011.wav
blues.00012.wav
blues.00013.wav
blues.00014.wav
blues.00015.wav
blues.00016.wav

```
In [13]: data1
Out[13]:
```

	filename	chroma_stft	rmse	...	mfcc19	mfcc20	label
0	blues.00000.wav	0.349943	0.130225	...	-2.300208	1.219928	blues
1	blues.00001.wav	0.340983	0.095918	...	-0.287431	0.531573	blues
2	blues.00002.wav	0.363603	0.175573	...	-3.433434	-2.226821	blues
3	blues.00003.wav	0.404779	0.141191	...	-0.619690	-3.408233	blues
4	blues.00004.wav	0.308590	0.091563	...	-4.409333	-11.703781	blues
...
995	rock.00095.wav	0.351991	0.079469	...	-6.717573	-1.189238	rock
996	rock.00096.wav	0.398653	0.076452	...	-7.459579	-2.802677	rock
997	rock.00097.wav	0.432103	0.081617	...	-12.594178	-2.107003	rock
998	rock.00098.wav	0.362349	0.083888	...	-5.043121	-3.585596	rock
999	rock.00099.wav	0.358195	0.054461	...	-2.022035	1.158525	rock

[1000 rows x 28 columns]

data - DataFrame

Index	chroma_stft	rmse	spectral_centroid	spectral_bandwidth	rolloff	zero_crossing_rate	mfcc1	mfcc2	mfcc3	mfcc4	mfcc5	mfcc6	mfcc7	mfcc8	mfcc9	mfcc10	mfcc11	mfcc12	mfcc13
0	0.349943	0.130225	1784.42	2002.65	3806.49	0.0830664	-113.597	121.557	-19.1588	42.351	-6.37646	18.6189	-13.6979	15.3446	-12.2853	10.9805	-8.32432	8.81067	-3.66737
1	0.340983	0.0959184	1529.84	2038.62	3548.82	0.0560443	-207.557	124.007	8.93056	35.8747	2.91604	21.5237	-8.5547	23.3587	-10.1036	11.9037	-5.56039	5.3768	-2.23912
2	0.363603	0.175573	1552.48	1747.17	3040.51	0.0763007	-90.7544	140.46	-29.11	31.689	-13.987	25.7548	-13.6496	11.6293	-11.7806	9.70644	-13.1231	5.78926	-8.90522
3	0.404779	0.141191	1070.12	1596.33	2185.03	0.0333089	-199.431	150.099	5.64759	26.8719	1.75446	14.2383	-4.83088	9.29797	-0.757741	8.14901	-3.19631	6.08768	-2.47642
4	0.30859	0.0915632	1835.49	1748.36	3580.95	0.1015	-160.266	126.199	-35.6054	22.1533	-32.4893	10.8645	-23.3579	0.503117	-11.8058	1.2068	-13.0838	-2.80638	-6.93412
5	0.302346	0.103468	1831.94	1729.48	3480.94	0.0940399	-177.869	118.197	-17.5507	30.7586	-21.7427	11.9038	-20.7342	3.1806	-8.58348	-0.936488	-11.7763	-2.42061	-9.33936

IPython console

Console 1/A

```
In [8]: runfile('C:/Users/ozgec/Desktop/genres/untitled0.py', wdir='C:/Users/ozgec/Desktop/genres')
filename
chroma_stft
rmse
spectral_centroid
spectral_bandwidth
rolloff
zero_crossing_rate
mfcc1
mfcc2
mfcc3
mfcc4
mfcc5
mfcc6
mfcc7
mfcc8
mfcc9
mfcc10
mfcc11
mfcc12
mfcc13
mfcc14
mfcc15
mfcc16
mfcc17
mfcc18
mfcc19
mfcc20
label
```

Permissions: RW | End-of-lines: CRLF | Encoding: UTF-8 | Line: 13 | Column: 15 | Memory: 38 %

SES DOSYASININ YÜKLENMESİ

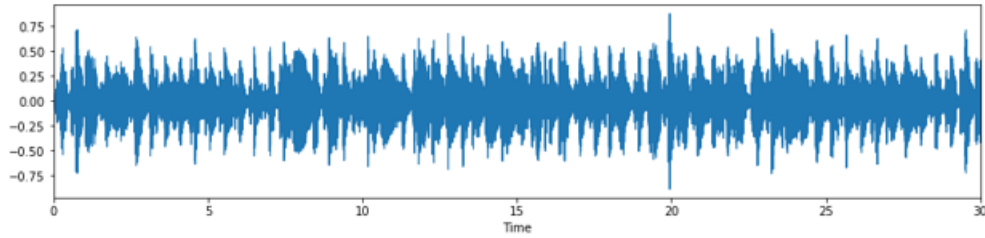
Name	Type	Size	Value
ses	str	1	classical.00000.wav
sr	int	1	22050
y	float32	(661794,)	[-0.02008057 -0.01748657 0.00418091 ... 0.01934814 0.027771 0.031 ...

y: ses zaman serisi

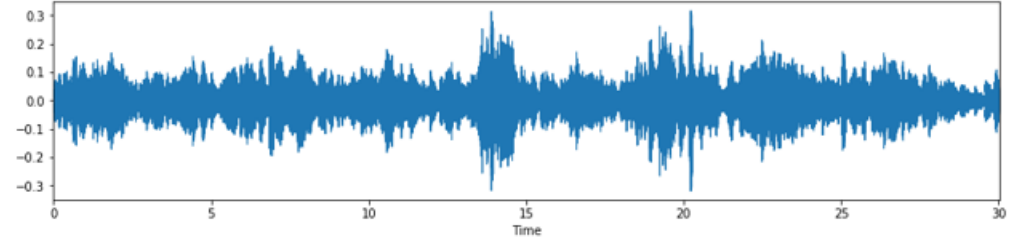
sr: ses frekansı

SESİN GÖRÜNTÜLENMESİ

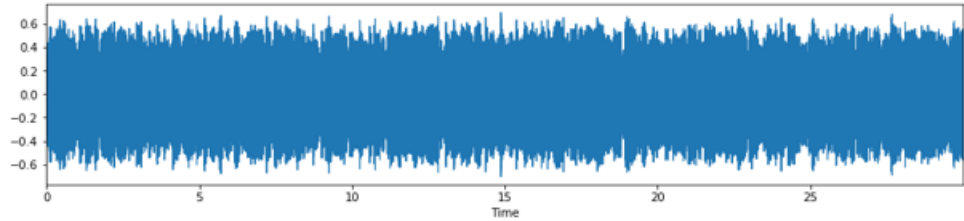
BLUES



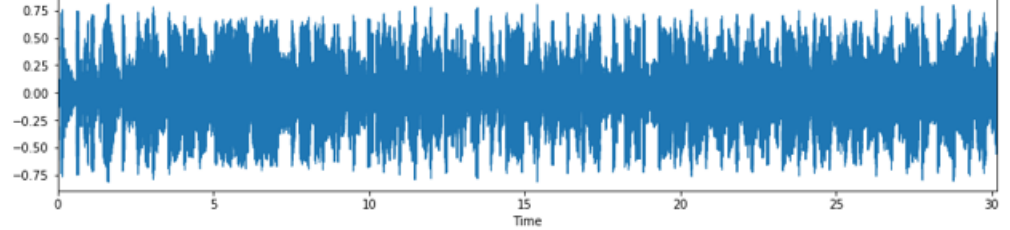
CLASSICAL

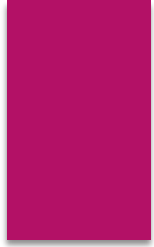


COUNTRY

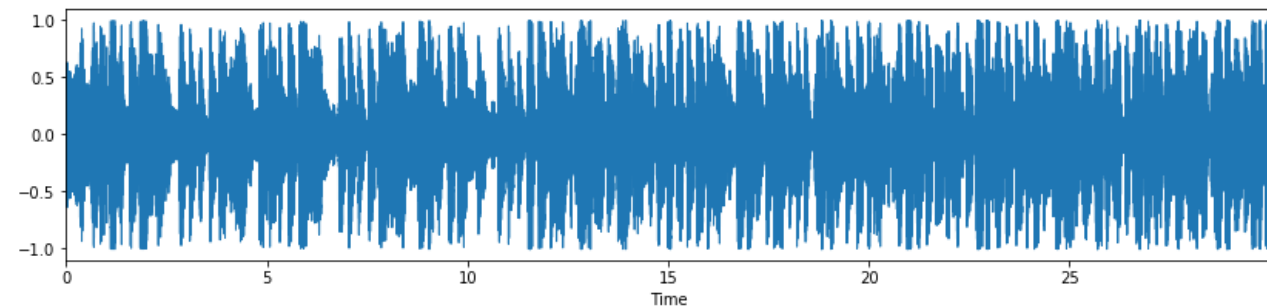


DISCO

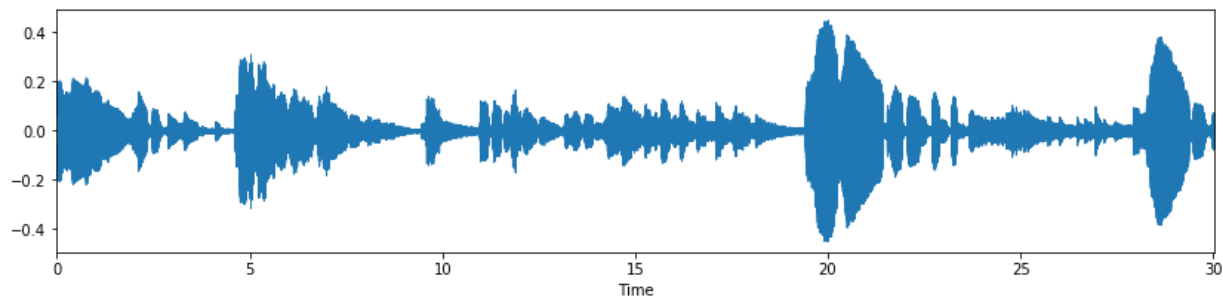




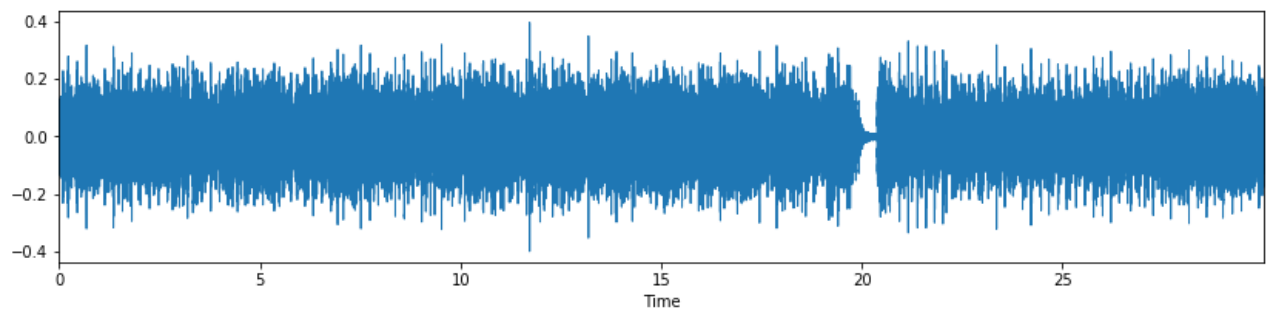
HIP-HOP



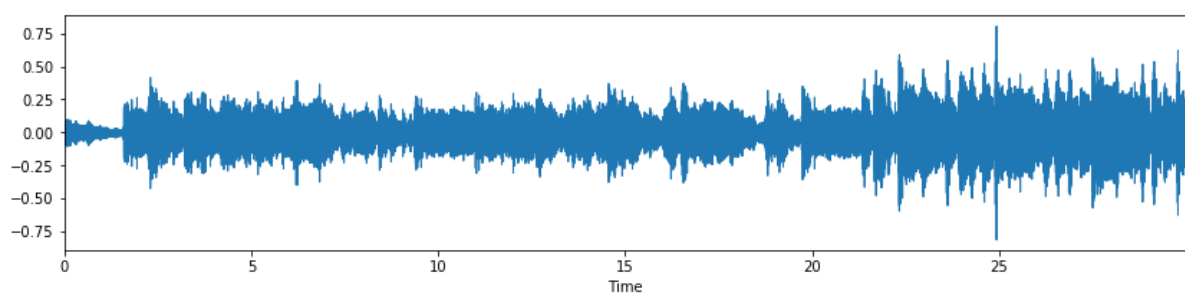
JAZZ



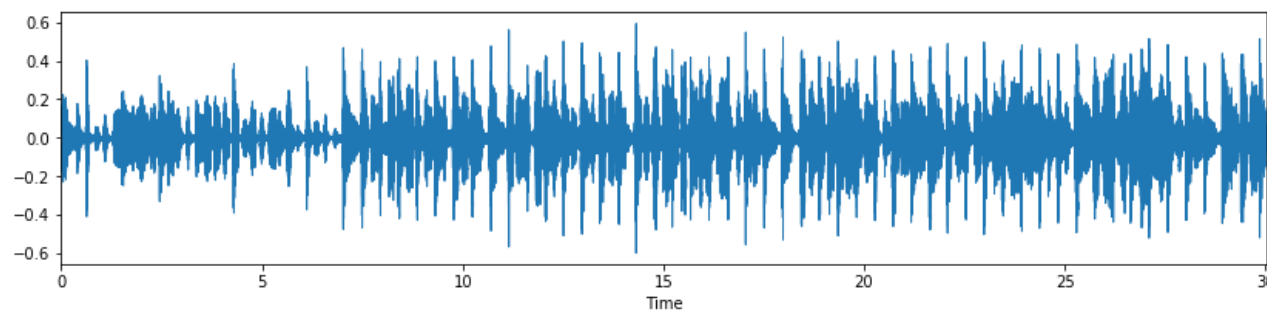
METAL



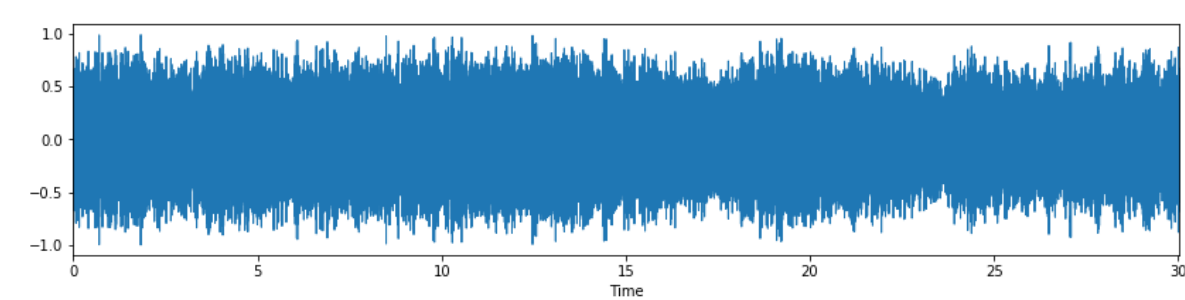
POP



RAGGE



ROCK

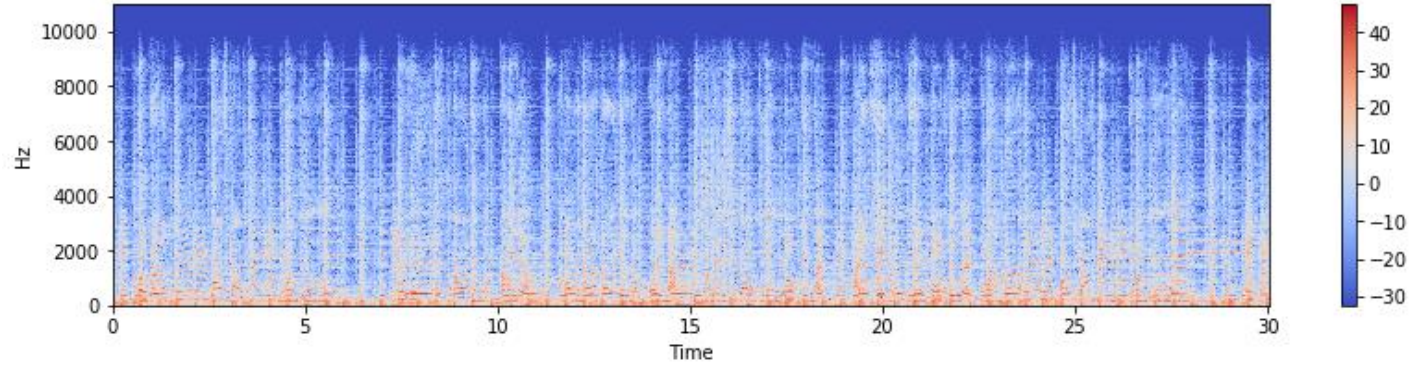


SPEKTROGRAM

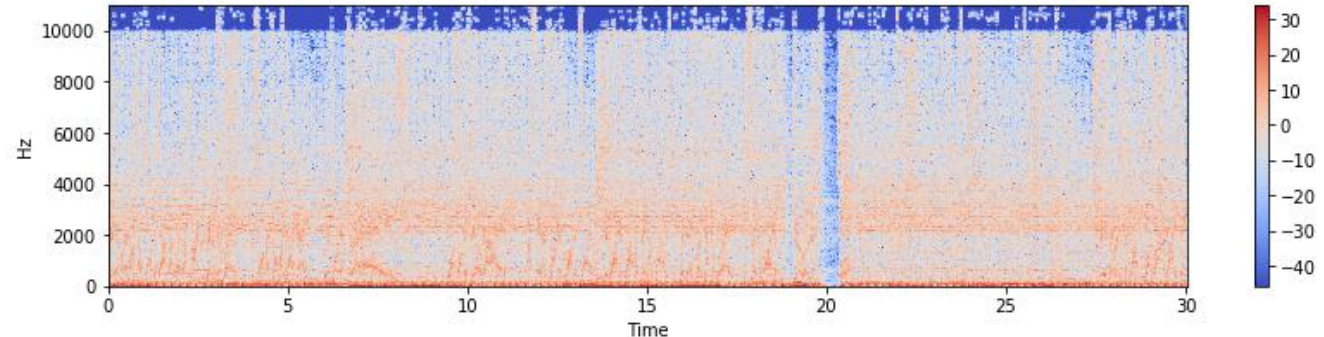
- Spektrogram, belirli bir dalga formunda bulunan çeşitli frekanslarda bir sinyalin sinyal gücünü veya yüksekliğini temsil eden görseldir. Aynı zamanda enerji seviyelerini zaman içinde nasıl değiştiğini de gösterir.

Dikey eksen frekansları
(0-10 kHz) ve yatay eksen
audio nun süresini gösterir.

BLUES



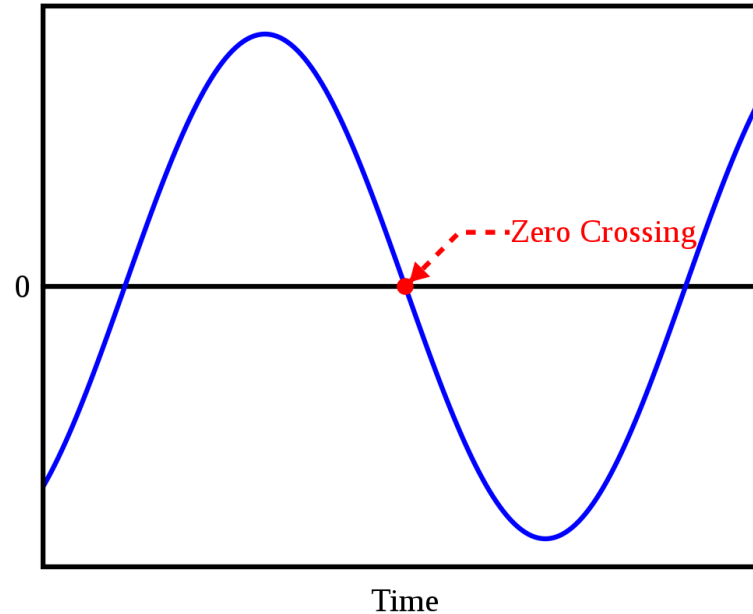
METAL

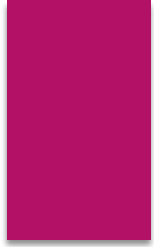


ÖZNİTELİK ÇIKARIMI(FEATURE EXTRACTION)

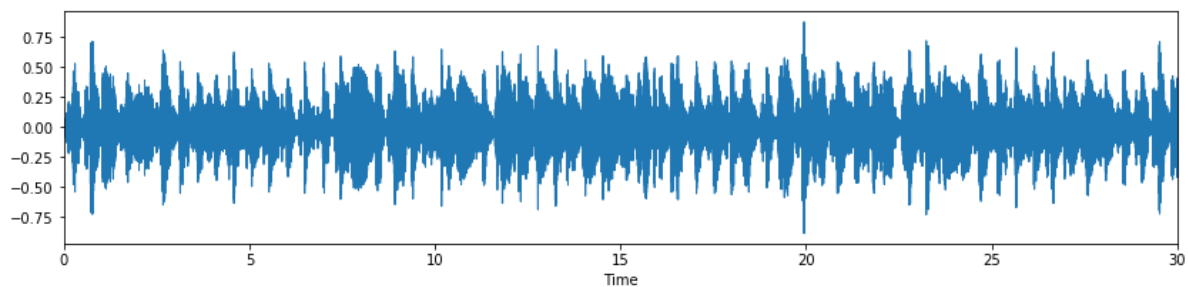
► Zero Crossing Rate

Zero crossing rate, bir sinyal boyunca işaret değiştirme oranıdır, yani sinyalin pozitifden negatife geçişi veya geri dönme hızıdır. Bu özellik hem konuşma tanıma hem de müzik bilgisi alımında yoğun bir şekilde kullanılmıştır. Metal ve Rock gibi yüksek seslere sahip sesler için genellikle daha yüksek değerlere sahiptir.

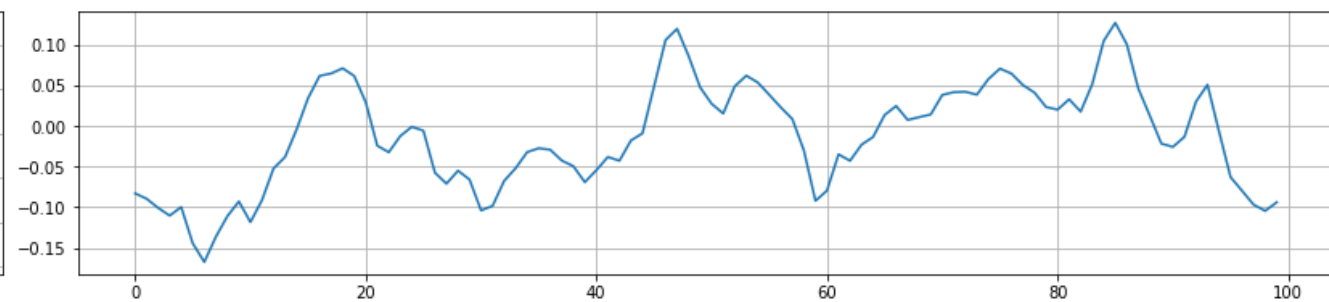
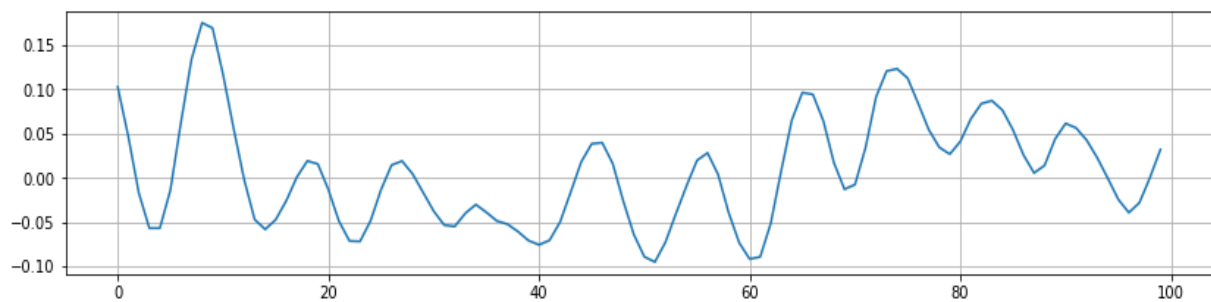
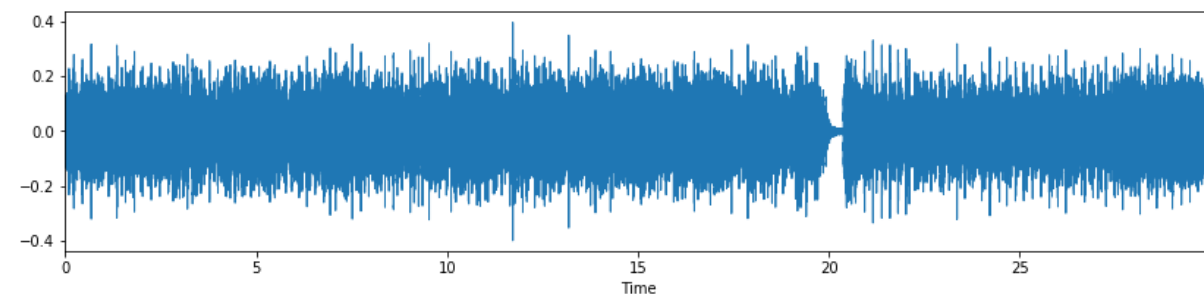




► BLUES



► METAL



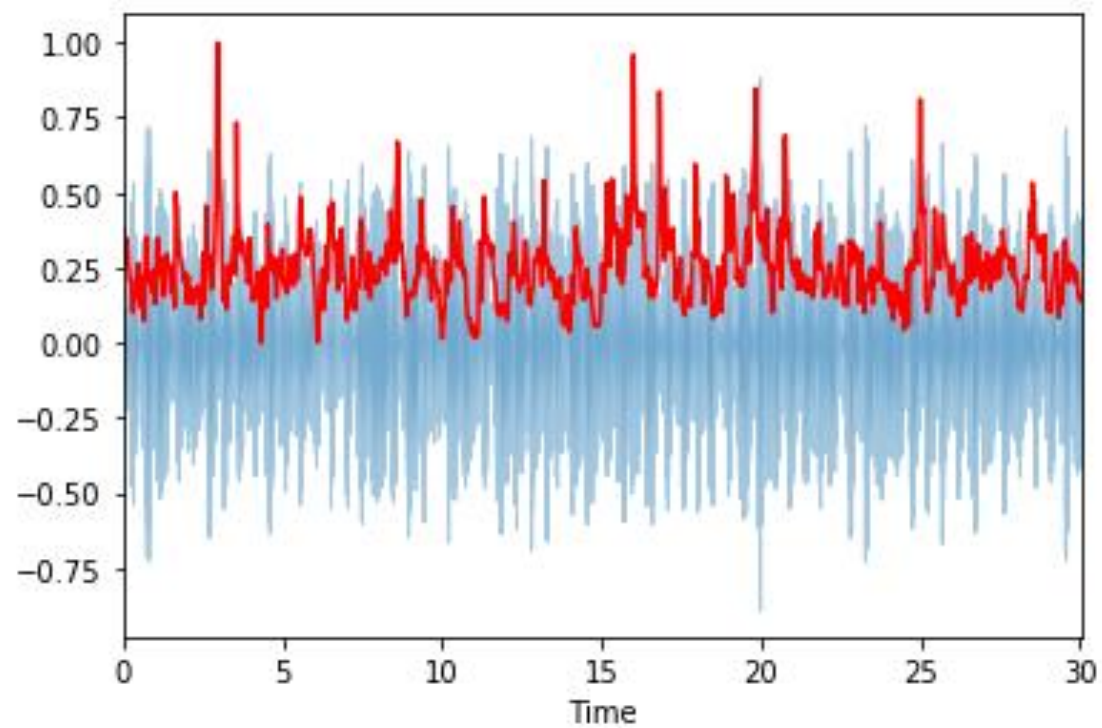
SPECTRAL CENTROID

- Spektrumun kütle merkezinin nerede olduğunu gösterir.

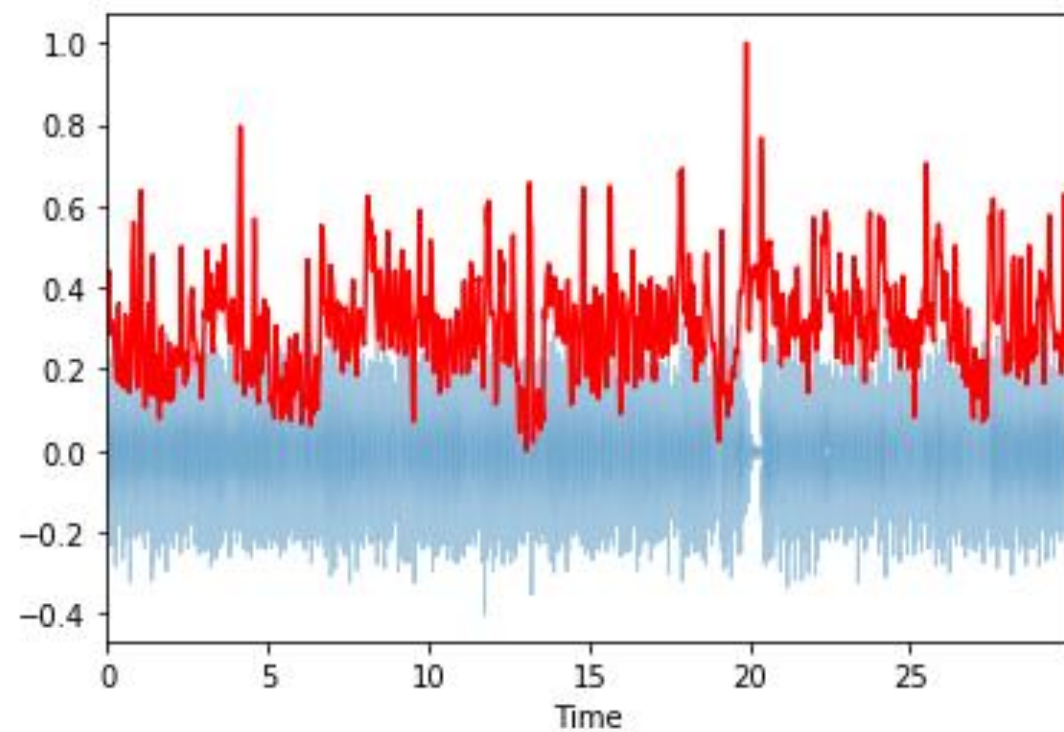
Bir ses için “kütle merkezinin” nerede bulunduğunu ve seste bulunan frekansların ağırlıklı ortalaması olarak hesaplandığını gösterir.

Biri blues tarzında diğeri metale ait olan iki şarkı düşünün. Şimdi, uzunluğu boyunca aynı olan blues tarz şarkısına kıyasla, metal şarkının sonuna doğru daha fazla frekansı var. Bu nedenle, blues şarkısı için spektral centroid, spektrumunun ortasına yakın bir yere uzanırken, metal bir şarkı için sonuna kadar olacaktır.

► BLUES



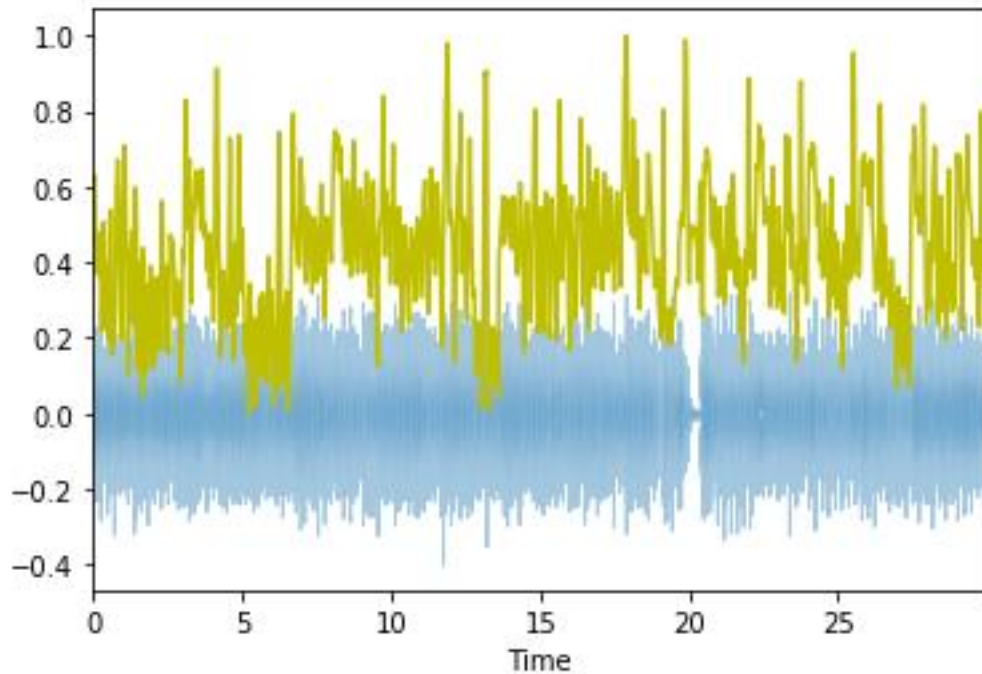
► METAL



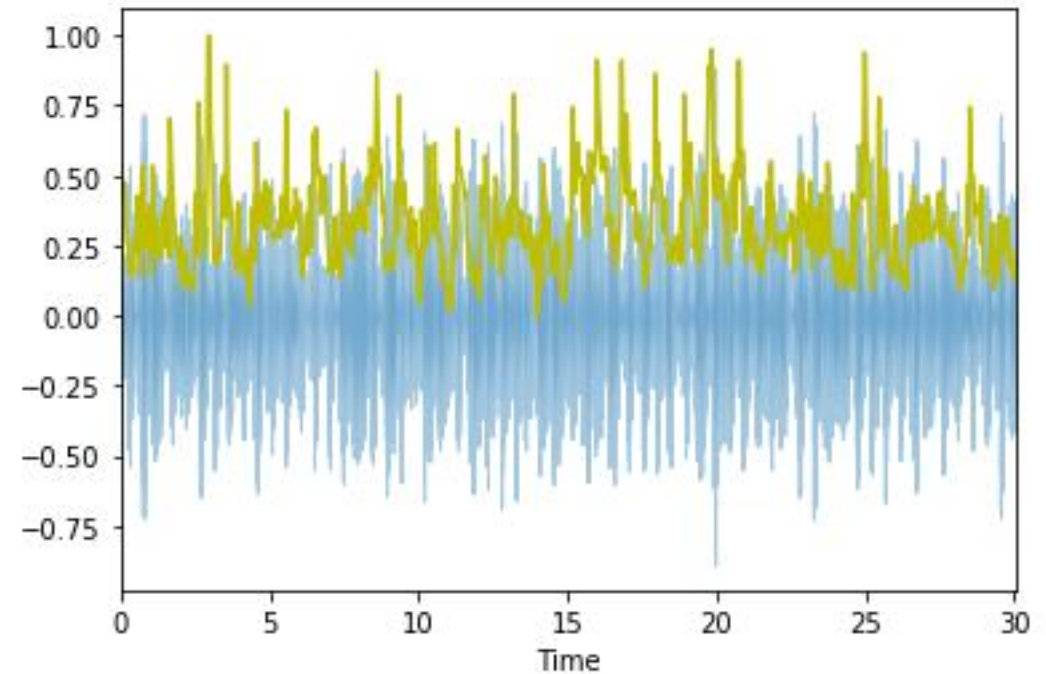
SPECTRAL ROLLOFF

- Sinyal şeklinin ölçüsüdür. Toplam spektral enerjinin belirli bir yüzdesidir.

► METAL



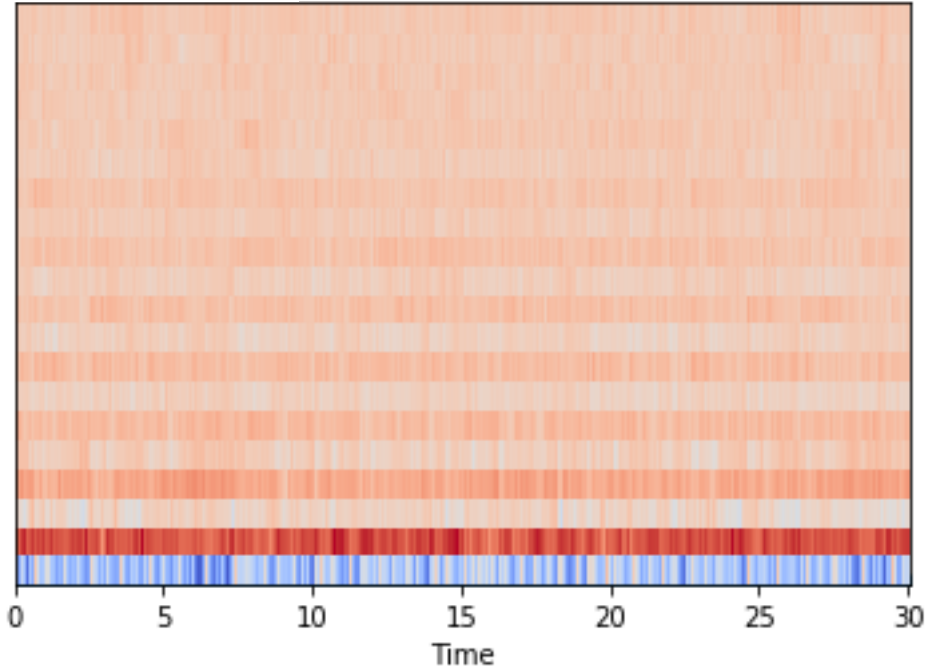
► BLUES



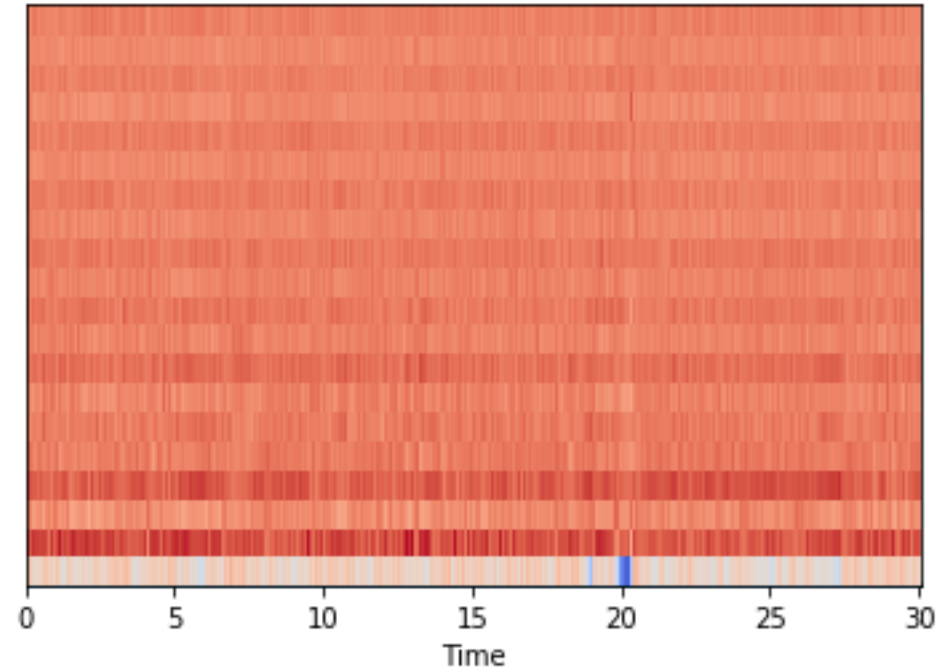
MEL-FREKANS KEPSTRAL KATSAYILARI (MEL-FREQUENCY CEPSTRAL COEFFICIENTS)

- Mel frekans ölçeği, insan kulağının ses frekanslarındaki değişimi algılayışını gösteren bir ölçektir.

► BLUES



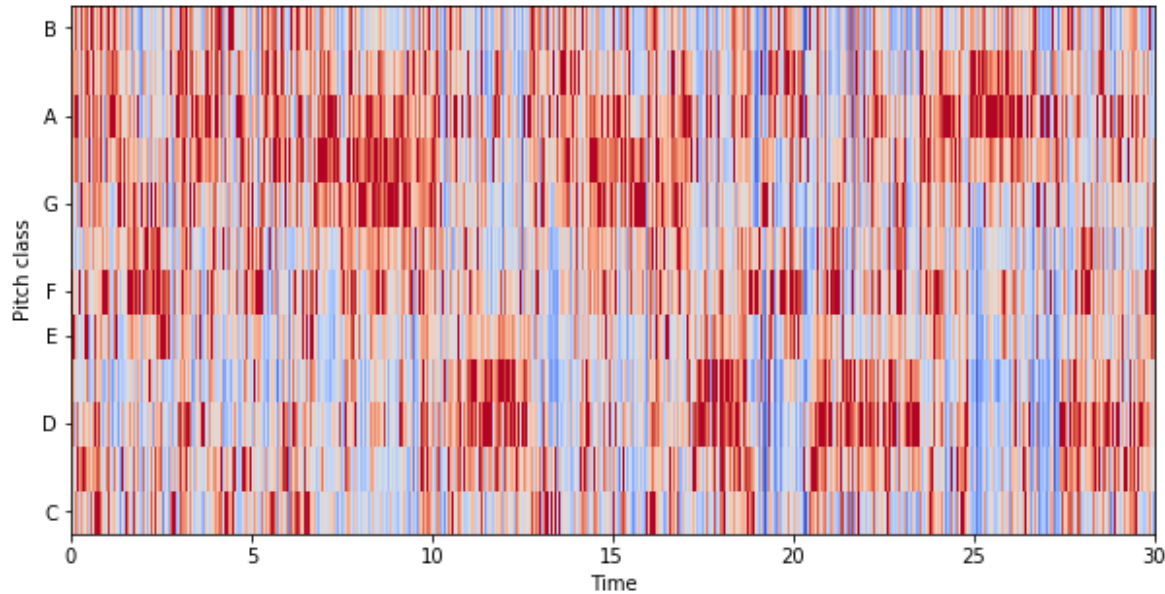
► METAL



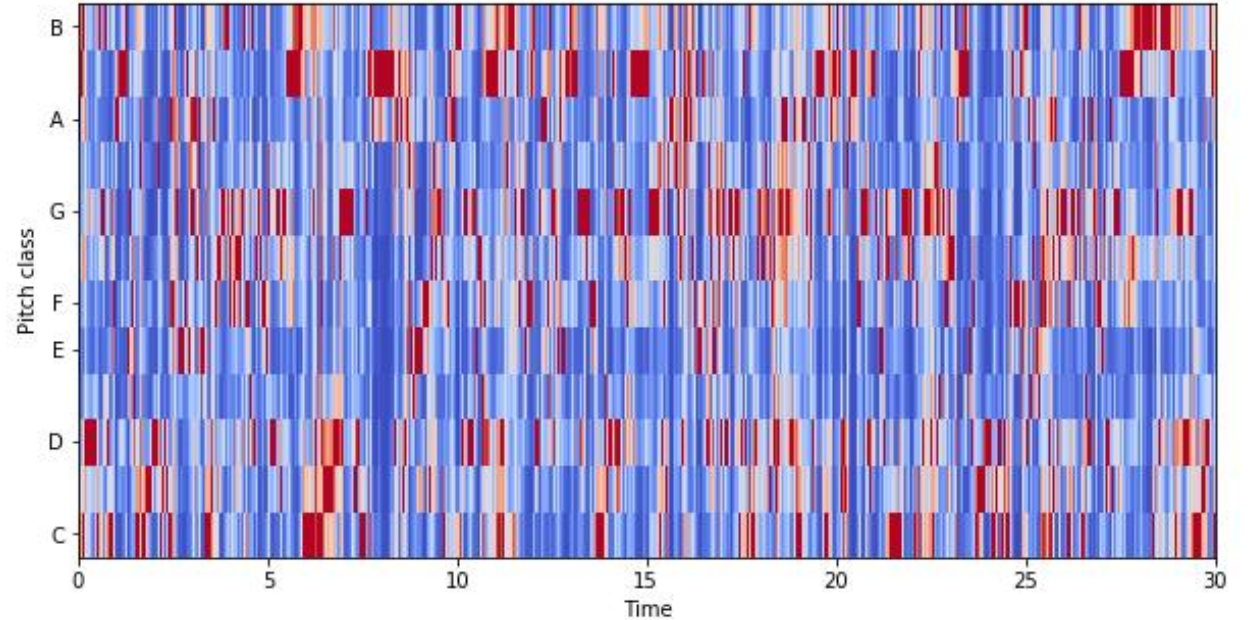
CHROMA FREKANSI

- Spektrum müzikal oktavının 12 farklı yarı tonunu(chroma) temsil eden 12 parçanın belirtildiği ses için güçlü bir sunumudur.

► METAL



► BLUES



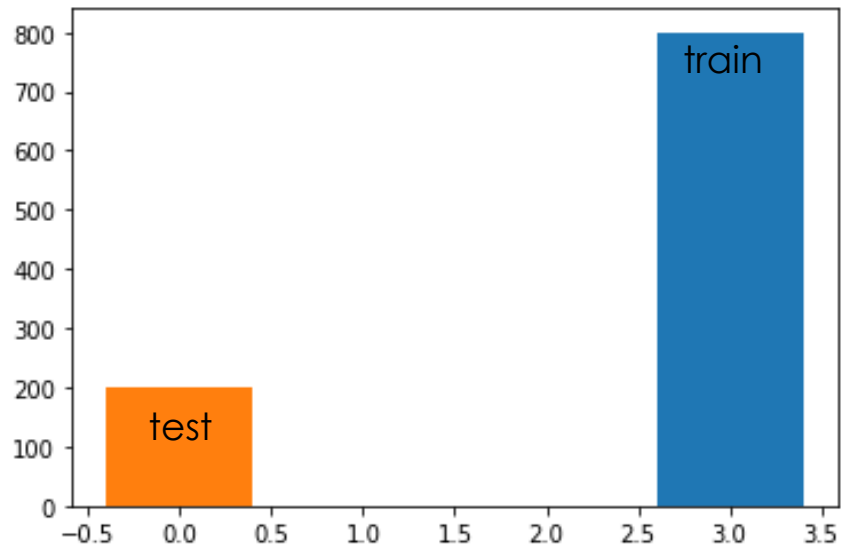
NORMALİZASYON

dat - NumPy array

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.482814	0.518921	0.458298	0.471773	0.468583	0.467969	0.552693	0.594453	0.470073	0.558559	0.474731	0.555705	0.42876
1	0.471198	0.463596	0.420631	0.479011	0.45125	0.399545	0.453395	0.602734	0.607224	0.517397	0.555266	0.581692	0.48356
2	0.500524	0.592054	0.423982	0.420354	0.417058	0.450837	0.576833	0.65836	0.421485	0.490795	0.408772	0.619543	0.42927
3	0.553908	0.536607	0.352614	0.389998	0.359513	0.341977	0.461983	0.69095	0.591194	0.460179	0.545199	0.516516	0.52323
4	0.429201	0.456572	0.465855	0.420595	0.453411	0.514645	0.503373	0.610145	0.38977	0.430189	0.248418	0.486333	0.32583
5	0.421106	0.47577	0.465329	0.416796	0.446684	0.495755	0.48477	0.583092	0.477925	0.484881	0.341555	0.495631	0.35378
6	0.406795	0.537582	0.410162	0.348253	0.400585	0.44255	0.471792	0.624001	0.386163	0.49921	0.626255	0.383634	0.35242

CLASSIFICATION

- Test ve Train seti oluşturuldu.



x_test - DataFrame

Index	chroma_stft	rmse	spectral_centroid	spectral_kurtosis
993	0.349251	0.255938	0.354512	0.37669
859	0.415242	0.308496	0.543904	0.69649
298	0.201061	0.43519	0.221307	0.30357
553	0.284591	0.317774	0.319489	0.36901
672	0.555726	0.481461	0.458006	0.48013
971	0.574451	0.396483	0.376533	0.50444
27	0.237782	0.230416	0.212559	0.22859
231	0.287662	0.243392	0.383757	0.49380
306	0.61442	0.475483	0.589533	0.60981
706	0.412465	0.225869	0.402467	0.67997
496	0.514381	0.268889	0.30592	0.34981
558	0.306207	0.191769	0.252594	0.26130
784	0.482608	0.443489	0.696755	0.82283

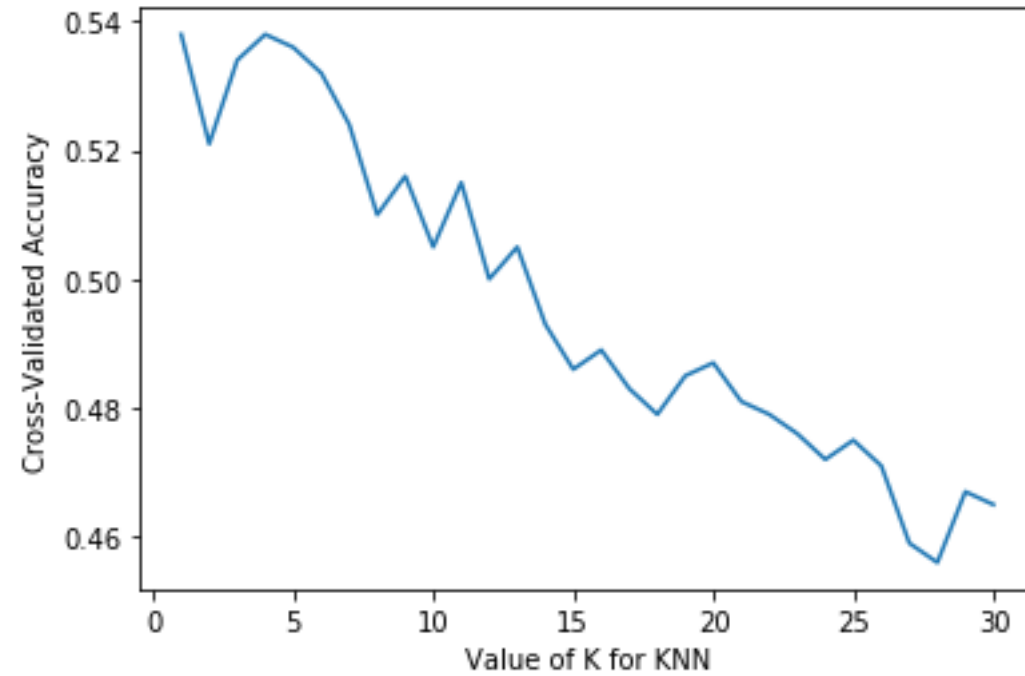
x_train - DataFrame

Index	chroma_stft	rmse	spectral_centroid	spectral_kurtosis
687	0.701346	0.667255	0.716469	0.6881
500	0.351339	0.105649	0.241809	0.3271
332	0.399954	0.269563	0.485826	0.5941
979	0.382789	0.283844	0.408472	0.4631
817	0.516644	0.342366	0.319196	0.4111
620	0.659004	0.285336	0.540367	0.5541
814	0.453999	0.277212	0.296602	0.4451
516	0.15812	0.140873	0.231748	0.3211
518	0.369459	0.119632	0.36821	0.4701
940	0.49682	0.350363	0.631369	0.7191
113	0.122678	0.0673886	0.211697	0.2131
612	0.880944	0.277661	0.577907	0.4921
37	0.156256	0.162627	0.159978	0.3001

► OPTIMAL K-VALUE

k_scores - List (30 elements)

Index	Type	Size	Value
1	float64	1	0.521
2	float64	1	0.534
3	float64	1	0.5379999999999999
4	float64	1	0.536
5	float64	1	0.532
6	float64	1	0.5239999999999999
7	float64	1	0.51
8	float64	1	0.516
9	float64	1	0.505
10	float64	1	0.515
11	float64	1	0.5



► K-NN PERFORMANS DEĞERLENDİRMESİ

scores - NumPy array

	0
0	0.35
1	0.56
2	0.66
3	0.53
4	0.53
5	0.51
6	0.6
7	0.56
8	0.5
9	0.52

IPython console

Console 1/A

knn:	precision	recall	f1-score	support
0	0.65	0.87	0.74	15
1	0.65	1.00	0.79	11
2	0.62	0.56	0.59	27
3	0.50	0.55	0.52	22
4	0.58	0.61	0.60	23
5	0.71	0.56	0.63	18
6	0.89	0.80	0.84	20
7	0.71	0.62	0.67	24
8	0.50	0.53	0.52	15
9	0.55	0.48	0.51	25
accuracy			0.63	200
macro avg	0.64	0.66	0.64	200
weighted avg	0.64	0.63	0.63	200

IPython console History log

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 88 Column: 1 Memory: 64 %

cross-validation accuary değerleri

RANDOM FOREST

```
IPython console
Console 1/A x

In [739]: print("Random Forest:")
...: print(classification_report(y_test,y_pred))
Random Forest:
              precision    recall  f1-score   support

     0       0.62         0.87         0.72         15
     1       0.52         1.00         0.69         11
     2       0.56         0.56         0.56         27
     3       0.44         0.55         0.49         22
     4       0.56         0.61         0.58         23
     5       0.91         0.56         0.69         18
     6       0.89         0.80         0.84         20
     7       0.74         0.58         0.65         24
     8       0.50         0.47         0.48         15
     9       0.65         0.44         0.52         25

 accuracy          0.61         200
  macro avg       0.64         0.64         0.62         200
 weighted avg     0.64         0.61         0.61         200
```

IPython console History log

Permissions: RW End-of-lines: CR LF Encoding: UTF-8 Line: 156 Column: 1 Memory: 69 %

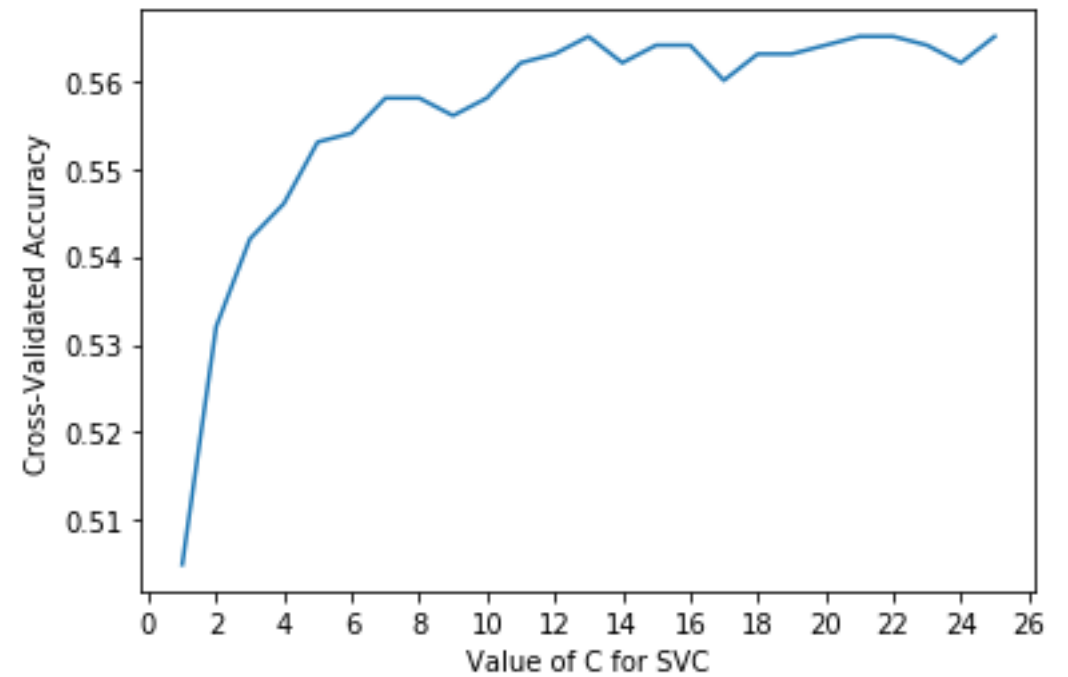
SVM

► PARAMETER TUNING

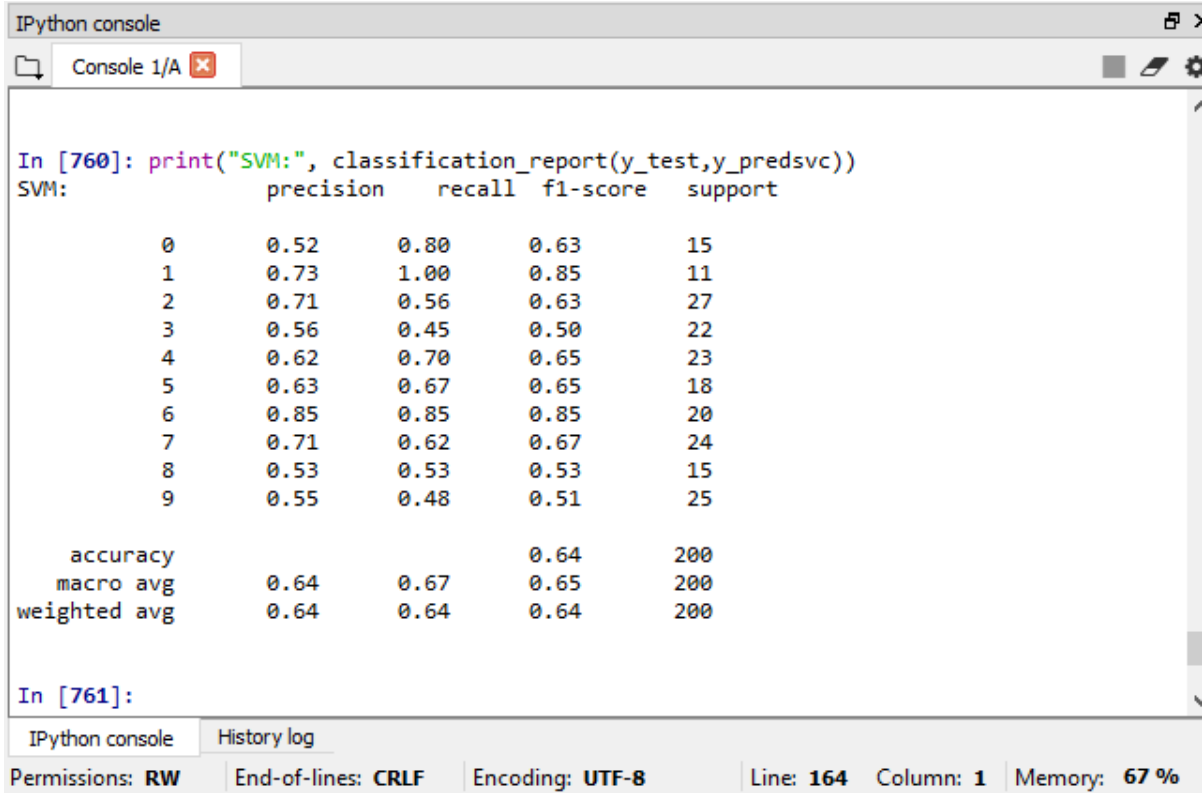
result_kernel - List (4 elements)

Index	Type	Size	Value
0	float64	1	0.59 linear
1	float64	1	0.46 sigmoid
2	float64	1	0.35 poly
3	float64	1	0.53 rbf

Save and Close Close



► SVM PERFORMANS DEĞERLENDİRMESİ



```
In [760]: print("SVM:", classification_report(y_test,y_predsvc))
SVM:
```

	precision	recall	f1-score	support
0	0.52	0.80	0.63	15
1	0.73	1.00	0.85	11
2	0.71	0.56	0.63	27
3	0.56	0.45	0.50	22
4	0.62	0.70	0.65	23
5	0.63	0.67	0.65	18
6	0.85	0.85	0.85	20
7	0.71	0.62	0.67	24
8	0.53	0.53	0.53	15
9	0.55	0.48	0.51	25
accuracy			0.64	200
macro avg	0.64	0.67	0.65	200
weighted avg	0.64	0.64	0.64	200

```
In [761]:
```

IPython console History log

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 164 Column: 1 Memory: 67 %

NAIVE BAYES

```
In [768]:  
print("\n", "GaussianNB:", nb.score(x_test, y_test), "\n", "MultinomialNB:", clf1.score(x_test,  
y_test), "\n", "ComplementNB:", clf2.score(x_test, y_test))
```

```
GaussianNB: 0.41  
MultinomialNB: 0.355  
ComplementNB: 0.285
```

```
In [769]:
```

IPython console History log

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 178 Column: 1 Memory: 66 %

IPython console

Console 1/A

GaussianNB

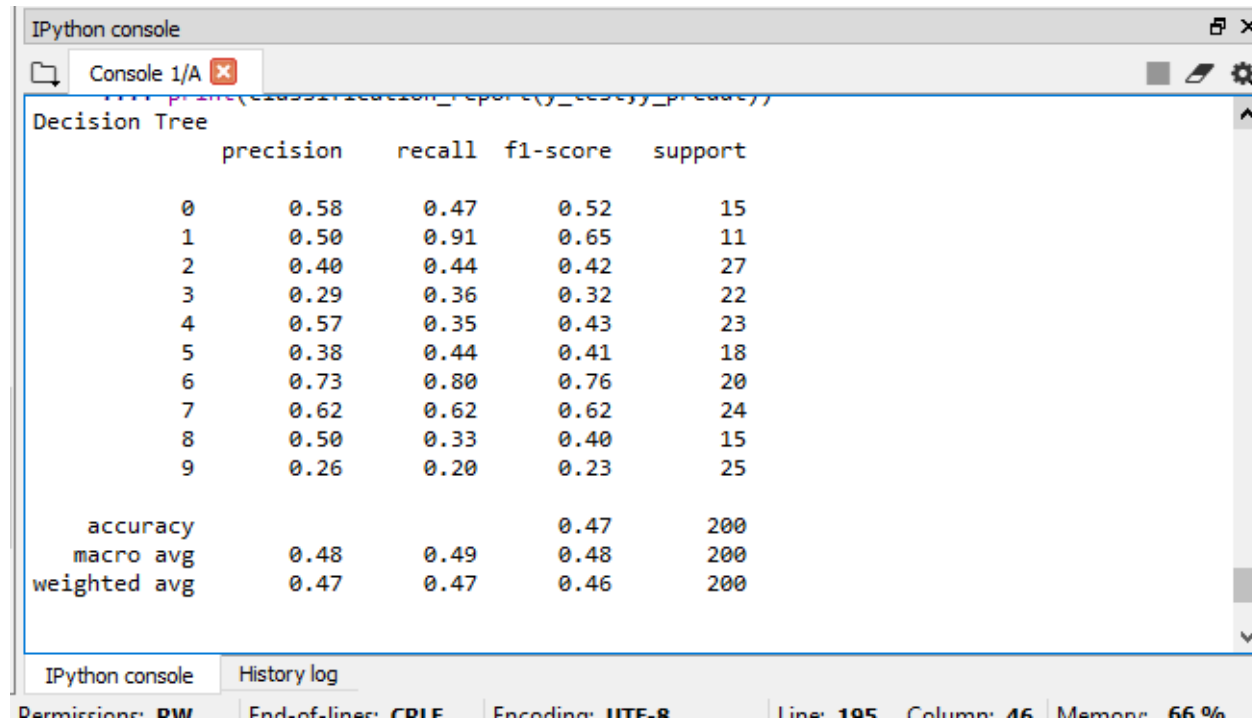
	precision	recall	f1-score	support
0	0.27	0.40	0.32	15
1	0.62	0.91	0.74	11
2	0.56	0.33	0.42	27
3	0.40	0.09	0.15	22
4	0.42	0.22	0.29	23
5	0.25	0.17	0.20	18
6	0.33	0.80	0.47	20
7	0.50	0.79	0.61	24
8	0.39	0.47	0.42	15
9	0.38	0.20	0.26	25
accuracy			0.41	200
macro avg	0.41	0.44	0.39	200
weighted avg	0.42	0.41	0.37	200

In [774]:

IPython console History log

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 178 Column: 61 Memory: 67 %

DECISION TREE

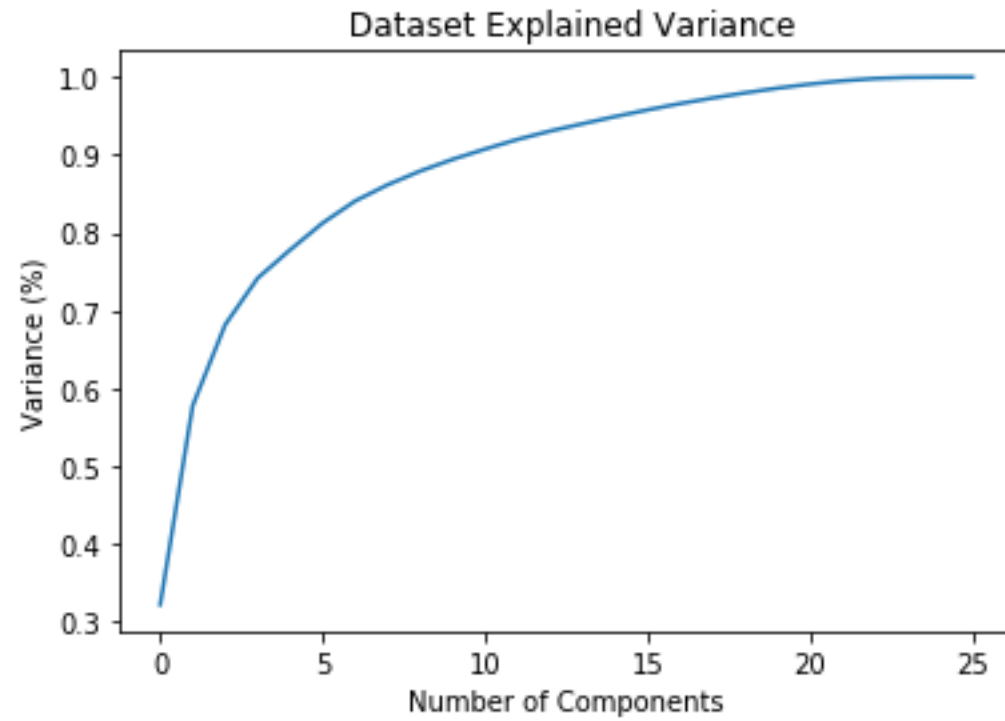


The screenshot shows an IPython console window with a tab labeled 'Console 1/A'. The output is a classification report for a Decision Tree model. The report includes a table with columns for precision, recall, f1-score, and support for each class (0-9), followed by summary statistics for accuracy, macro average, and weighted average. The status bar at the bottom indicates 'Permissions: RW', 'End-of-line: CRLF', 'Encoding: UTF-8', 'Line: 105', 'Column: 46', and 'Memory: 66%'.

	precision	recall	f1-score	support
0	0.58	0.47	0.52	15
1	0.50	0.91	0.65	11
2	0.40	0.44	0.42	27
3	0.29	0.36	0.32	22
4	0.57	0.35	0.43	23
5	0.38	0.44	0.41	18
6	0.73	0.80	0.76	20
7	0.62	0.62	0.62	24
8	0.50	0.33	0.40	15
9	0.26	0.20	0.23	25
accuracy			0.47	200
macro avg	0.48	0.49	0.48	200
weighted avg	0.47	0.47	0.46	200

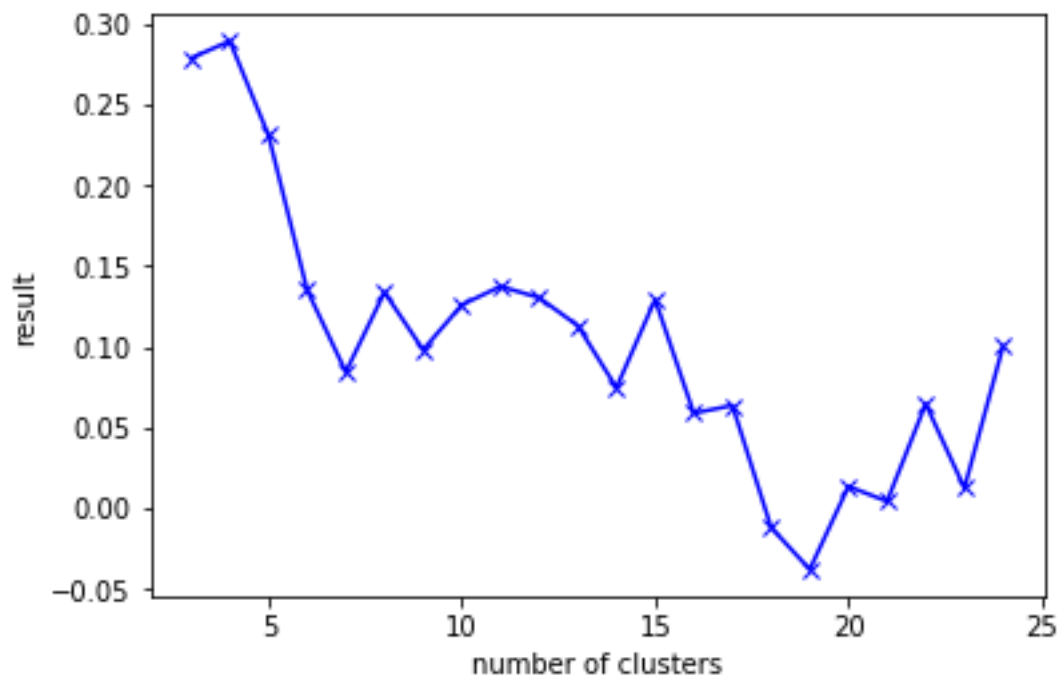
IPython console History log
Permissions: RW End-of-line: CRLF Encoding: UTF-8 Line: 105 Column: 46 Memory: 66%

CLUSTERING



Choosing the Number of Components in a Principal Component Analysis

K-MEANS



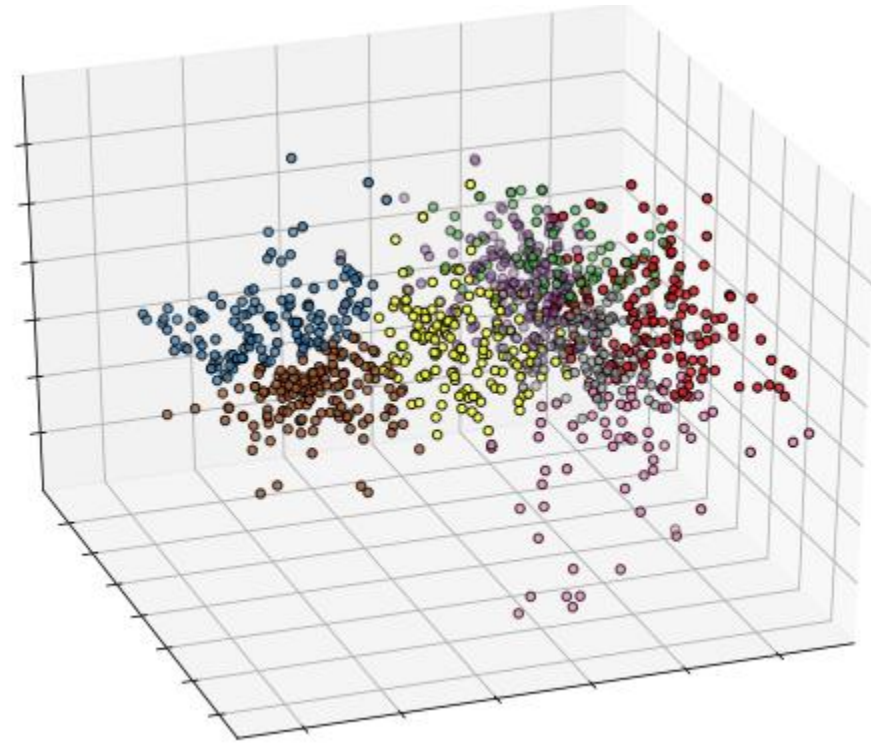
SILHOUETTE-SCORE

preds - NumPy array

	0
106	6
107	6
108	6
109	6
110	6
111	6
112	6
113	6
114	6
115	6
116	6
117	6
118	6

principalComponents - NumPy array

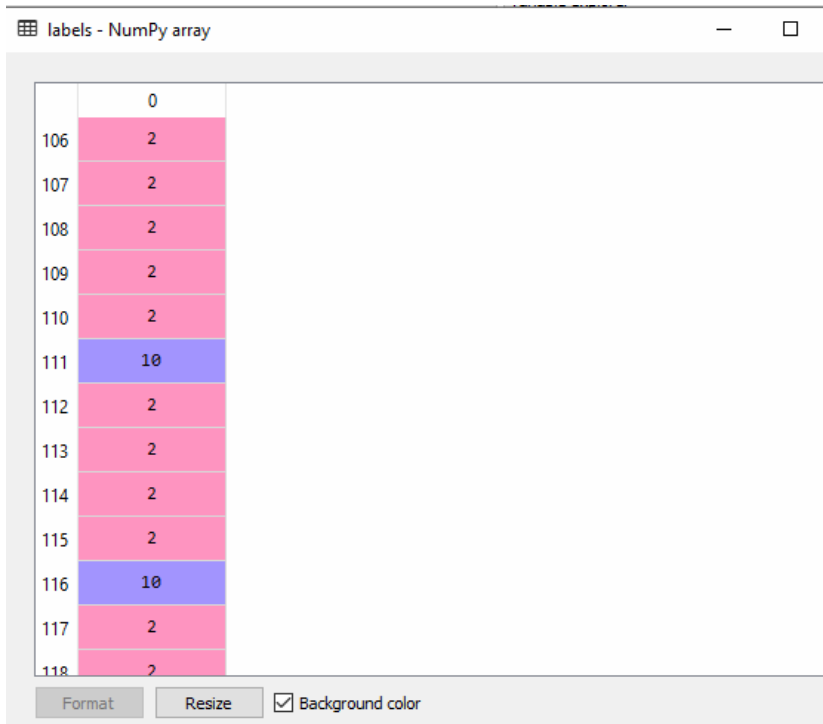
	0	1	2
0	0.126546	-0.12477	-0.0887447
1	0.0112461	-0.148989	-0.156406
2	0.173827	-0.256221	0.0902809
3	-0.149286	-0.234138	-0.131486
4	-0.0235505	-0.367581	0.312839
5	-0.122518	-0.316093	0.244586
6	-0.256677	-0.39946	0.347246
7	-0.185886	-0.423366	0.385923
8	0.229448	-0.140867	-0.0725341
9	-0.128604	-0.216886	-0.0212242
10	-0.208152	-0.276293	-0.0161113
11	-0.257064	-0.262707	-0.0126226
12	-0.146561	-0.333586	0.0896949



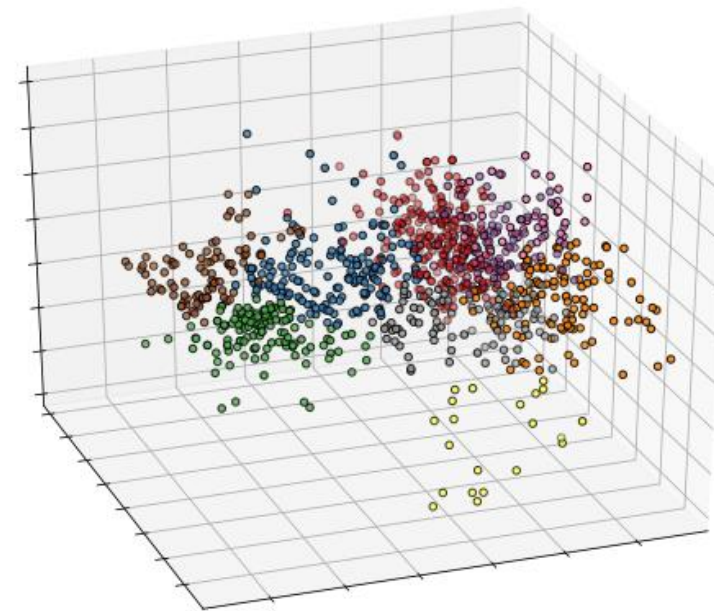
K-MEANS

HIEARARCHIAL CLUSTERING

► Agglomerative Clustering



distance_threshold=22



distance_threshold=2.4

SONUÇ (CONCLUSION)

MODELS

Algorithm Comparison

