Şarkı Özelliklerinden Yararlanarak Popülarite Tahmini

*Spotify kitlesi baz alınarak

Harun Serkan Metin
201101034

Bu çalışmada: popülarite kavramının şarkı özellikleriyle alakalı olup olmadığı, başlıca yapay zekâ ve makine öğrenmesi tekniklerinden yararlanılarak tahmin edilmeye çalışılmıştır. Uygun hale getirilen verilerden; sınıflandırmak için birden fazla algoritma, yapay sinir ağları, derin öğrenme gibi yöntemler kullanarak popülerlik tahmini yapılmıştır ve sonuçlar karşılaştırılmıştır.

I. GİRİŞ

İnsan hayatında müziğin her zaman bir yeri olmuştur. Bilgi teknolojilerinde yüksek hızda yaşanan gelişmeler ve internet kullanımının çok yaygın hale gelmesiyle çeşitli platformlardan dinlen içeriklerin çeşitliliği ve hacmi de artmıştır. Dinlenen bu müzik içerikleri ne kadar fazla insan tarafından dinlenirse o kadar popüler olmuştur. Müzik dağıtım şirketleri (Spotify, Apple Music, Deezer) yayımlayacakları içeriğin birçok verisini tutarlar. Bu şirketler, dinleyici kitle arasında aldığı etkileşime göre, popüler olan parçaları daha çok insanlara göstermek gibi bir politika sergilerler. Benim bu çalışmadaki motivasyonum, üretilen sanat eserleri, daha müzik pazarına çıkmadan dinleyici kitle arasında popülaritesini tahmin etmek. Eğer tahmin konusunda yeterli başarı elde edilirse yayımlanacak şarkının özellikleri uygun seçilerek kitle arasında popülerlik yakalanabilir.

Spotify API sini kullanarak 228160 şarkıyı ".csv" dosyası şeklinde kaydedilmiştir. Bu verilerin 18 tane özniteliği bulunmaktadır (3. Başlıkta detaylıca bahsedilmiştir). Öznitelikleri anlamlı hale getirerek 0-100 arasında değerleri olan popülarite özelliğini sınıflara ayırmıştır. Birden fazla sınıflandırma yöntemi kullanarak sonuçları:

II. VERİ SETİ, VERİ ÖZELLİKLERİ

Google E-Tablolara yüklenen veri setime bu bağlantıdan ulaşabilirsiniz:
https://docs.google.com/spreadsheets/d/1fDCu4KlQoNbab1gISGQCBzJ8gxmtBXN89ZPxdqOeSoE/edit#gid=210231712

genre:	Şarkının türü (pop, rock, caz …)
artist_name:	Sanatçı ismi
track_name:	Şarkının ismi
track_id:	Şarkının Spotify içindeki ID'si. Her bir veri özgündür
popularity:	0 ile 100 arasında popülerlik derecesi
acousticness:	0,0 ile 1,0 arasında şarkının akustik olup olmadığına dair ölçüt.
danceability:	0,0 ile 1,0 arasında şarkının dans etmeye olan uygunluğudur.
duration_ms :	Parçanın milisaniye cinsinden süresi.
energy:	0,0 ile 1,0 arasında Yoğunluk ve aktivitenin algısal ölçüsüdür.
instrumentalness:	0,0 ile 1,0 arasında parçanın be kadar vokal içerdiğinin ölçütüdür.
key:	Parçanın nota olarak hangi tonda olduğunu belirtir.
mode:	Parçanın Majör mü Minör mü olduğunu belirtir
Liveness:	0,0 ile 1,0 arasında Kaydın içinde izleyicinin varlığının ölçütüdür. (Konser kaydı>= 0,8 Stüdyo Kaydı<=0,5 gibi)
loudness:	-60 ile 0 arasında Şarkının desibel (dB) cinsinden ses yüksekliği.
Tempo:	Dakikadaki vuruş sayısı (BPM) bir parçanın genel tahmini temposu
time_signature:	Ritim tipi
Valence:	Parçanın müzikal pozitifliği tanımlayan 0,0 ile 1,0 arasında bir ölçü. 1,0'e yakın parçalar daha olumlu (mutlu, neşeli), 0,0'a yakın parçalar daha olumsuz (örneğin üzgün, depresif, kızgın).
speechiness:	Bir parçada konuşulan kelimelerin varlığını algılar. Kayıt gibi konuşma ne kadar özel olursa (örneğin Talk Show, sesli kitap, şiir), öznitelik değeri 1.0'a o kadar yakındır.

III. ÖNİŞLEME AŞAMALARI

Veri setimin küçük bir örneği **Resim 1.1**'de belirtilmiştir.Veri setinde "NULL" değerlerin var olup olmadığını kontrol ettim ve her değerin dolu olduğunu gördüm (**Resim 1.2**). 'duration_ms' sütununda şarkı süresini milisaniye cinsinden pek bir anlam ifade etmediğinden o değerleri saniye cinsine çevirdim. (**Resim 1.3**)

	genre	artist_name	track_name	track_id	popularity	acousticness	danceability	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence	duration
0	Opera	Giuseppe Verdi	Stiffelio, Act III: Ei fugge! ... Lina, pensai c...	7EsKYeHtTc4H4xWiTqSVZA	21	0.986	0.313	0.23100	0.000431	C#	0.0964	-14.287	Major	0.0547	86.001	4/4	0.0886	490.867
1	Opera	Giacomo Puccini	Madama Butterfly / Act 1: ... E soffitto e pareti	7MfmRBvqaW0I6UTxXnad8p	18	0.972	0.360	0.20100	0.028000	D#	0.1330	-19.794	Major	0.0581	131.798	4/4	0.3690	176.797
2	Opera	Giacomo Puccini	Turandot / Act 2: Gloria, gloria, o vincitore	7pBo1GDhlysyUMFXiDVoON	10	0.935	0.168	0.47000	0.020400	C	0.3630	-8.415	Major	0.0383	75.126	3/4	0.0696	266.184
3	Opera	Giuseppe Verdi	Rigoletto, Act IV: Venti scudi hai tu detto?	02mvYZX5aKNzdqEo6jF20m	17	0.961	0.250	0.00605	0.000000	D	0.1200	-33.440	Major	0.0480	76.493	4/4	0.0380	288.573
4	Opera	Giuseppe Verdi	Don Carlo / Act 4: "Ella giammai m'amò!"	03TW0jwGMGhUabAjOpB1T9	19	0.985	0.142	0.05800	0.146000	D	0.0969	-23.625	Major	0.0493	172.935	4/4	0.0382	629.760

Resim 1.1

```
(228159, 18)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 228159 entries, 0 to 228158
Data columns (total 18 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   genre             228159 non-null  object
 1   artist_name       228159 non-null  object
 2   track_name        228159 non-null  object
 3   track_id          228159 non-null  object
 4   popularity        228159 non-null  int64
 5   acousticness      228159 non-null  float64
 6   danceability      228159 non-null  float64
 7   energy            228159 non-null  float64
 8   instrumentalness  228159 non-null  float64
 9   key               228159 non-null  object
 10  liveness          228159 non-null  float64
 11  loudness          228159 non-null  float64
 12  mode              228159 non-null  object
 13  speechiness       228159 non-null  float64
 14  tempo             228159 non-null  float64
 15  time_signature    228159 non-null  object
 16  valence           228159 non-null  float64
 17  duration          228159 non-null  float64
dtypes: float64(10), int64(1), object(7)
memory usage: 31.3+ MB
```

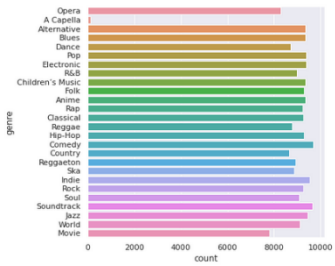
Resim 1.2

duration_ms	duration_m
5.866720e+05	2.115050
2.300512e+05	1.636667
1.265261e+05	
3.344000e+03	3.027333
1.750930e+05	
2.148930e+05	2.948450
2.638670e+05	
5.621218e+06	2.718000

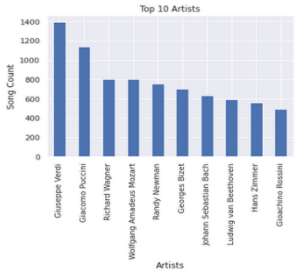
Resim 1.3

A. Sınıfsal (Nominal) Veriler

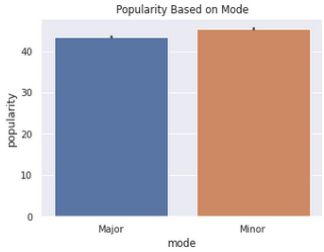
Genre (Resim 2.1), artist_name (Resim 2.2), track_name, track_id, mode (Resim 2.3), key (Resim 2.4), time_signature



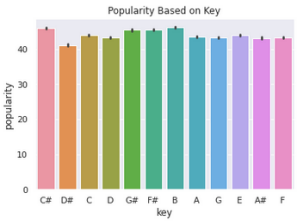
Resim 2.1



Resim 2.2



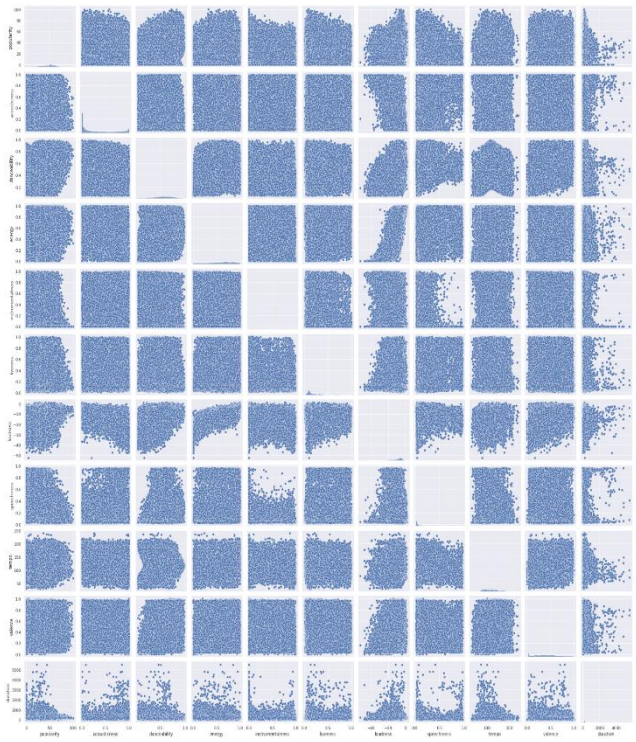
Resim 2.3



Resim 2.4

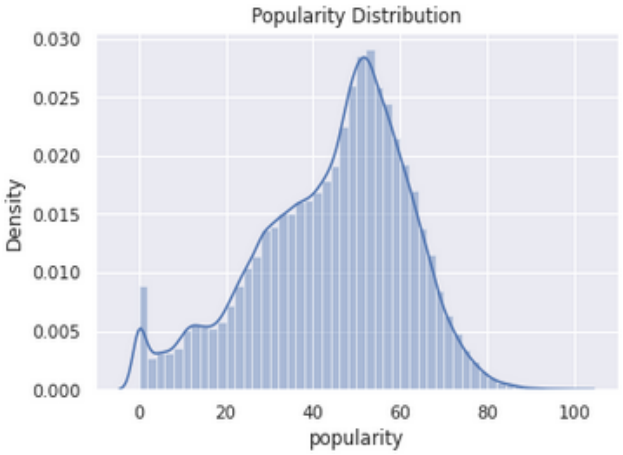
B. Sayısal(Numeric) Veriler

Popularity, acousticness, danceability, energy, instrumentalness, liveness, loudness, speechiness, tempo, valence, duration



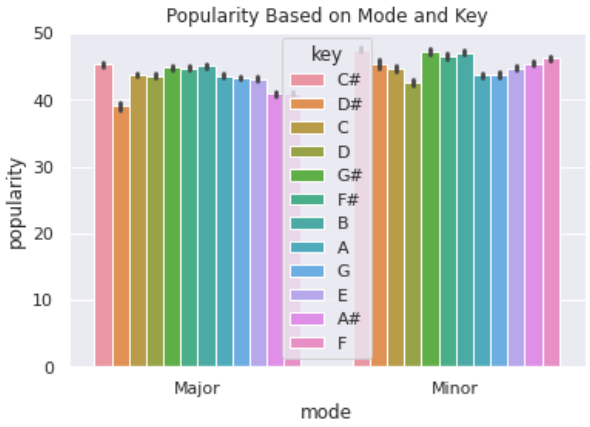
Resim 3.1

Resim 3.1'de görüldüğü üzere veriler arasında belirgin bir lineer ilişki gözükmemekte.
Popülaritenin Veri seti üzerinde dağılımını inceleyerek daha fazla bilgi sahibi olabiliriz



Resim 3.2

Resim 3.2'de görüldüğü gibi çoğu şarkının popülerliği 30-70 arasında dağılmış. Yani her popülerlik değeri için eşit verimiz yok bu durum bizi problem çıkartabilir.



Resim 4.1

Batı müziğinde 12 nota vardır. Bununla birlikte, her bir notanın bir minör ve bir majör modu vardır. Yandaki grafikte (Resim 4.1), aynı nota için popülerliğin farklı tonaliteler arasında nasıl farklılık gösterdiği ifade edilmiştir. (0 minör mod, 1 majör mod).

Örneğin, D# minördeki bir melodi, D# majördeki bir melodiden daha popüler olma eğilimindedir

Nominal verilerin popülerlik ile aralarındaki korelasyonu incelemek ve bu verileri modellerde kullanabilmek için vektöreler cinsinden ifade etmek gerekecektir.

```
['A Capella' 'Alternative' 'Anime' 'Blues' 'Children's Music' 'Classical'
 'Comedy' 'Country' 'Dance' 'Electronic' 'Folk' 'Hip-Hop' 'Indie' 'Jazz'
 'Movie' 'Opera' 'Pop' 'R&B' 'Rap' 'Reggae' 'Reggaeton' 'Rock' 'Ska'
 'Soul' 'Soundtrack' 'World']
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7
 1.8 1.9 2.  2.1 2.2 2.3 2.4 2.5]
```

Her bir tür sayılar cinsinden ifade edildi

```
['A' 'A#' 'B' 'C' 'C#' 'D' 'D#' 'E' 'F' 'F#' 'G' 'G#']
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.  1.1]
```

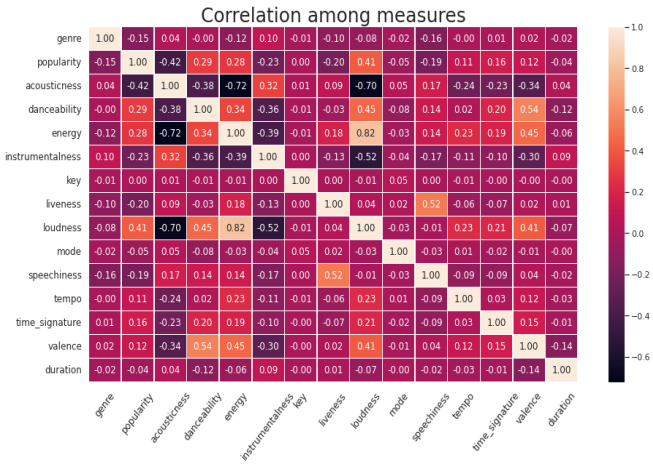
Herbir nota sayılar cinsinden ifade edildi

```
['0/4' '1/4' '3/4' '4/4' '5/4']
[0.  0.1 0.2 0.3 0.4]
```

Her bir ritim sayılar cinsinden ifade edildi

```
df.loc[df["mode"] == 'Major', "mode"] = 1
df.loc[df["mode"] == 'Minor', "mode"] = 0
```

Her bir mode sayılar cinsinden ifade edildi



Resim 5.1

Yukardaki görselde (Resim 5.1) korelasyon matrisi verilmiştir. Bu matris ile ilgili gözlemler:

- "Energy" ve "loudness" en yüksek pozitif korelasyona sahiptir
- "Energy" ve "acousticness" yüksek oranda ters ilişkiye sahiptir (negatif korelasyona).
- Ne yazık ki, bağımlı değişkenimiz "popülerlik" diğer değişkenler arasında çok zayıf korelasyon değerleri fark ediyoruz. Aldığımız en iyi değer, "acousticness" ve "popülerlik" arasındaki korelasyon "-0.42" dir.

<u>İyi Korelasyona Sahip Olan öznitelikler</u>

genre

acousticness

danceability

energy

instrumentalness

liveness

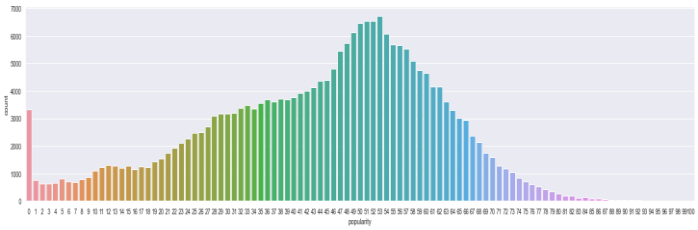
loudness

speechiness

tempo

time_signature

valence



Resim 6.1

Popülerlik dağılımı homojen olmadığı (Resim 6.1) için Standardizasyon yaparak dağılımı daha optimal bir seviyeye çekiyorum.

Standardizasyon Formülü:

$$x_{new} = (x_i - x) / s$$

- xi: Veri kümesindeki i. değer
- x: Sample ortalama
- s: Sample standart sapması

IV. KULLANILAN YAPAY ÖĞRENME METOTLARI

K Neighbors Regressor:

KNN, Denetimli Öğrenmede sınıflandırma ve regresyon için kullanılan algoritmalardan biridir. En basit makine öğrenmesi algoritması olarak kabul edilir. Diğer Denetimli Öğrenme algoritmalarının aksine, eğitim aşamasına yoktur. Eğitim ve test hemen hemen aynı şeydir. Tembel bir öğrenme türüdür. Bu nedenle, KNN, geniş veri setini işlemek için ideal bir algoritma değildir.

Komşularının çoğunluk oyuyla sınıflandırılır; bu olay, bir mesafe fonksiyonuyla ölçülen en yakın komşuları arasında en yakın olan sınıfa atanır. K = 3 ise, En yakın 3 komşusunun sınıfına atanır.

K için en uygun değeri seçmek için önce verileri incelemek gerekir. Büyük bir K değeri, genel noise'u düşürdüğü için daha hassastır, ancak garantisi yoktur. Çapraz Doğrulama, iyi bir K değerini belirlemek için bağımsız bir veri kümesi kullanır. Çoğu veri kümesi için optimal K, 3-10 arasında olmuştur.

Linear Regression:

Bağımlı değişken ile bağımsız değişken arasındaki ilişkiyi tahmin etmek için kullanılan yöntemdir. Daha spesifik olarak, bağımlı değişkenin diğer değişkenlerin değişimine göre nasıl değiştiğine odaklanır. Ayrıca değişkenler arasındaki gelecekteki ilişkinin modellenmesine de yardımcı olur.

Regresyon analizi, **doğrusal, doğrusal olmayan ve çoklu doğrusal** olmak üzere çeşitli türlerden oluşur. Ancak en kullanışlı regresyon türleri, basit doğrusal ve çoklu doğrusal regresyondur.

$$Y = MX + b$$

- Y, regresyon denkleminin bağımlı değişkenidir.
- M, regresyon denkleminin eğimidir.
- X, regresyon denkleminin bağımlı değişkenidir.
- b denklemin sabitidir.

Decision Tree Regressor:

Karar ağacı, sınıflandırma problemlerinde çoğunlukla kullanılan bir denetimli öğrenme algoritmasıdır. Öğrenilen ağaçlar insan okunabilirliğini artırmak için if-then-else olarak temsil edilirler. Karar ağacı, bir ağaç yapısı biçiminde sınıflandırma veya regresyon modelleri için kullanılabilir. Bir veri kümesini sürekli olarak daha küçük alt kümelere bölerken, aynı zamanda karar ağacı kademeli olarak geliştirilir. Hangi kısıta göre bölme işlemi yapacağımız Entropy'e göre belirlenmektedir. Bu ağacın bir karar düğümü, iki veya daha fazla dallara sahiptir. Yaprak düğüm bir sınıflandırma veya kararı temsil eder. Bir ağaçtaki en üstteki karar düğümü, kök düğüm olarak adlandırılan en iyi belirleyiciye karşılık gelir (Informatiom Gain en fazla olan). Karar ağaçları hem kategorik hem de sayısal verileri işleyebilir.

Random Forest Regressor:

Random Forest, hiper parametre kestirimi yapılmadan da iyi sonuçlar vermesi hem regresyon hem de sınıflandırma problemlerine uygulanabilir olmasından dolayı popüler makine öğrenmesi modellerinden biridir. Random forest regresyon birden fazla karar ağacını kullanarak daha uyumlu modeller üreterek isabetli tahminlerde bulunmaya yarar. Fakat geleneksel yöntemlerden biri olan karar ağaçlarının en büyük problemlerinden biri "Overfitting"dir. Random forest modeli bu problemi çözmek için hem veri setinden hem de öznitelik setinden rassal olarak farklı alt kümeler seçiyor ve bunları eğitiyor. Bu yöntemle birden fazla ve değişik yapılarda karar ağacı oluşturuluyor ve her bir karar ağacı bireysel olarak tahminde bulunuyor. Tahminlerinin ortalamasını seçiyoruz.

Gradient Boosting Regressor:

Gradient Boosting'de öncelikli olarak ilk yaprak (initial leaf) oluşturulur. Sonrasında tahmin hataları göz önüne alınarak yeni ağaçlar oluşturulur. GBR ile her ağaç bir önce eğitilen ağacın eğitim verisindeki etiketleri tahmin etmekteki hatası üzerinde (on the residuals) eğitilir, yani önceki ağacın hataları bir nevi yeni etiketler haline gelir. Art arda kendini geliştiren bu ağaçlar bütünü daha fazla gelişme kaydedilemeyinceye kadar kendini geliştirmeye devam eder.

Support Vector Regressor:

Destek Vektör Regresyon Adından da anlaşılacağı gibi hem doğrusal hem de doğrusal olmayan regresyonları destekleyen bir regresyon algoritmasıdır. SVR, SVM'nin ayrık kategorik etiketleri tahmin etmek için kullanılan bir sınıflandırıcı olması yönüyle SVM'den farklıdır, SVR ise sürekli sıralı değişkenleri tahmin etmek için kullanılan bir regresördür.
Destek Vektör Regresyonu (SVR), sınıflandırma için SVM ile aynı prensipleri kullanır, sadece birkaç küçük farklılık vardır. Her şeyden önce, çıktı gerçek bir sayı olduğundan, sonsuz olasılıklara sahip olan eldeki bilgiyi tahmin etmek çok zor hale gelir. Gerileme durumunda, problemden zaten talep etmiş olacak olan SVM'ye yaklaşık olarak bir tolerans değeri ayarlanır. Ana fikir her zaman aynıdır: hatayı en aza indirmek, marjı en üst düzeye çıkaran hiper düzlemi net olarak belirlemeye çalışmak, hatanın bir kısmının tolere edildiğini akılda tutmak.

V. MODELLERİN KULLANILMASI VE HATA KARŞILAŞTIRMASI

Veri kümesinin rastgele bir şekilde %80 alıyorum ve aldığım bu bölümün
%30 unu test için %70 ini öğrenme için kullanıyorum

Regresyon için alınan sonuçlar:

	Model	R2_score	RMSEscore	Cross_val_score
0	KNeighborsRegressor	23.350166	15.250035	22.031882
1	LinearRegression	30.930857	14.476290	30.395029
2	DecisionTreeRegressor	40.465806	13.439976	39.957908
3	RandomForestRegressor	69.809932	9.570779	69.985865
4	GradientBoostingRegressor	67.984790	9.855835	67.872594
5	SVR	17.176699	15.852272	16.743151

Görüldüğü üzere en iyi sonuç %70 e yakın bir R2 score ile Random Forest Regressor ile elde edildi. Verilerim lineer olarak iyi ayrışamadığı için Basit Regresyon modellerinde iyi sonuçlar elde edemedim.

Başka bir yol olarak, bu problemi Sınıflandırma problemine çevirerek ne kadar doğru tahminler yapabiliriz buna bakmak istedim.
Popülarite değerlerine farklı eşik değerleri uygulayarak 2 sınıflı (Resim 7.1), 4 sınıflı (Resim 7.2), ve 6 sınıflı (Resim 7.3) veri setleri yarattım.

```
dfBinaryClass.loc[dfBinaryClass['popularity'] < 57, 'popularity'] = 0
dfBinaryClass.loc[dfBinaryClass['popularity'] >= 57, 'popularity'] = 1
```

Resim 7.1

```
df4Class.loc[(df['popularity'] >= 0) & (df4Class['popularity'] < 25), 'popularity'] = 0
df4Class.loc[(df['popularity'] >= 25) & (df4Class['popularity'] <50), 'popularity'] = 1
df4Class.loc[(df['popularity'] >= 50) & (df4Class['popularity'] <75), 'popularity'] = 2
df4Class.loc[(df['popularity'] >= 75) & (df4Class['popularity'] < 100), 'popularity'] = 3
```

Resim 7.2

```
df6Class.loc[(df6Class['popularity'] >= 0) & (df6Class['popularity'] < 20), 'popularity'] = 0
df6Class.loc[(df6Class['popularity'] >= 20) & (df6Class['popularity'] <28 ), 'popularity'] = 1
df6Class.loc[(df6Class['popularity'] >= 28) & (df6Class['popularity'] <45), 'popularity'] = 2
df6Class.loc[(df6Class['popularity'] >= 45) & (df6Class['popularity'] < 65), 'popularity'] = 3
df6Class.loc[(df6Class['popularity'] >= 65) & (df6Class['popularity'] < 80), 'popularity'] = 4
df6Class.loc[(df6Class['popularity'] >=80) & (df6Class['popularity'] < 100), 'popularity'] = 5
```

Resim 7.3

Bu veri setlerini aşağıdaki algoritmalarda Sınıflandırmayı denedim:

- KNeighbors Classifier,
- Logistic Regression,
- Decision Tree Classifier,
- Random Forest Classifier,
- Multi Layer Perceptron Classifier,
- Ligth Gradient Booster Classifier

Klasifikasyon için alınan sonuçlar:

Resim 8.1, 8.2, 8.3'te görüldüğü üzere en iyi sonuçlar

```
-------------------------
Binary Classified Result:
-------------------------
```

	Model	Accuracy	Precision	Recall	F1 score
0	KNeighbors Classifier	77.348929	75.949306	77.348929	76.401232
1	Logistic Regression	74.932888	68.998562	74.932888	66.099897
2	Decision Tree Classifier	82.312497	82.400937	82.312497	82.355439
3	Random Forest Classifier	88.842930	88.547577	88.842930	88.496173
4	Multi Layer Perceptron Classifier	77.987180	75.753520	77.987180	74.598760
5	Ligth Gradient Booster Classifier	81.882430	80.767557	81.882430	80.515423

Resim 8.1

Binary Sınıflandırma için %89 e yakın bir Accuracy ile Random Forest Classifier ile elde edildi.

```
-------------------------
4 Part Classified Result:
-------------------------
```

	Model	Accuracy	Precision	Recall	F1 score
0	KNeighbors Classifier	61.669315	61.121472	61.669315	61.219436
1	Logistic Regression	58.546540	57.841721	58.546540	56.740742
2	Decision Tree Classifier	70.177505	70.393320	70.177505	70.280925
3	Random Forest Classifier	80.641538	80.998694	80.641538	80.156735
4	Multi Layer Perceptron Classifier	61.598093	60.806102	61.598093	59.928231
5	Ligth Gradient Booster Classifier	67.772421	67.557151	67.772421	67.638753

Resim 8.2

4 Part Sınıflandırma için %80 e yakın bir Accuracy ile Random Forest Classifier ile elde edildi.

```
-------------------------
6 Part Classified Result:
-------------------------
```

	Model	Accuracy	Precision	Recall	F1 score
0	KNeighbors Classifier	54.385580	52.244597	54.385580	52.808279
1	Logistic Regression	49.709637	39.596126	49.709637	43.221313
2	Decision Tree Classifier	58.562976	58.866455	58.562976	58.707325
3	Random Forest Classifier	70.774667	70.380594	70.774667	68.649614
4	Multi Layer Perceptron Classifier	45.800690	20.977032	45.800690	28.774942
5	Ligth Gradient Booster Classifier	59.212184	57.159280	59.212184	57.889569

Resim 8.2

6 Part Sınıflandırma için %71 e yakın bir Accuracy ile Random Forest Classifier ile elde edildi.

6. Sonuçlar (conclusions)

Bu çalışmada, bir şarkının hedef kitledeki popülaritesi, şarkı özelliklerinden yararlanarak tahmin edilmeye çalışılmıştır. Normalizasyonu yapılıp uygun hale getirilen veriler popülerlik tahmini yapmak için regresyon ve klasifikasyon problemlerine çevrilmiştir. Birden fazla yapay öğrenme algoritması, yapay sinir ağları, derin öğrenme yöntemleri kullanarak popülerlik tahmini yapılmıştır ve sonuçlar karşılaştırılmıştır.

Sonuç olarak bu veri seti için en iyi çalışan yapay öğrenme modeli Random Forest Modeli olduğu ulaşılan bulgulardan rahatlıkla çıkartılabilir.

Çalışma sonucunda, homojen dağılmayan verilerden modelin öğrenilmesinin çok güç olduğu ve verilerin normalizasyonunun ne kadar önemli olduğunu öğrendim. Eşit şekilde populerlik verim olsaydı model eğitimi daha kolay ve tahminler daha isabetli olurdu. Ayrıca enhanced şekilde deep learning methodları çalıştırsaydım (yani birden fazla derin öğrenme algoritmasını birbirine entegre etseydim) daha kuvvetli tahminler sağlayacağımı düşünmekteyim.

7. Ekler

Dataset (SpotifyFeatures.csv):

https://docs.google.com/spreadsheets/d/1fDCu4KlQoNbab1gISGQCBzJ8gxmtBXN89ZPxdqOeSoE/edit#gid=210231712

Sunum Videosu:

https://youtu.be/N-SKDFoLLPk

Python Kodu (BIL470Project.ipynb):

https://drive.google.com/file/d/1tVSAoOSHiley5s5iJLfvN1fkGjKsOZod/view?usp=sharing