Mimar Sinan Güzel Sanatlar Üniversitesi

Fen-Edebiyat Fakültesi

Matematik Bölümü

K- ORTALAMALAR KÜMELEME

Gamze DİNÇ

Danışman: Prof. Dr. Özgür MARTİN

Diploma Çalışması

Ocak, 2024

Özet

Bu çalışmada, basit anlamda kümeleme ve kümeleme yöntemlerinden biri olan K-Ortalamalar kümeleme yöntemi geniş çapta ele alınmıştır.
Kümeleme, veri madenciliği ve makine öğrenimi alanlarında yaygın olarak kullanılan bir tekniktir. Bu teknik, veri noktalarını benzer özelliklere sahip gruplara ayırmayı amaçlar. K-Ortalamalar kümeleme ise kümeleme analizinde yaygın olarak kullanılan bir yöntemdir. Bu yöntem, veri noktalarını belirli bir sayıda kümeye ayırmak için kullanılır. K-Ortalamalar gibi kümeleme yöntemlerinin gelecekte, karmaşık veri setleri ile çalışma ihtiyacının artması ve bu yöntemlerin veri analizi ile yorumlanması ve karar alma süreçlerine olan katkılarından dolayı önemi daha da artacaktır.
Bu yüzden bu çalışmaya gereksinim olup çalışmada da birinci bölümde bu işin teorisi, mantığı açıklanmış diğer bölümde Pyhton üzerinden örneklerle desteklenip değerlendirmeler yapılmış konu hakkında hiç bilgisi olmayan okuyucunun bile konu hakkında bilgilenip anlaması hedeflenip yazılmıştır.

Teşekkür

Daha önce fazla bilmediğim, ancak oldukça merak ettiğim bu konuda benimle proje çalışmayı kabul eden Özgür hocama teşekkür ederim. İstediğim konuda çalışmak, keyif alarak öğrenip geliştiğim bir süreç oluşturmuş oldu. Bu projede algoritmaların matematiğini teorisini anlamaya çalışırken başta ufkumu ve bakış açımı geliştirip genişleterek zorlandığım zamanlarda motive ederek; aynı zamanda bir mentor gibi davranarak bana akademik ve kişisel olarak katkı sağlayıp, gelecek için bilgiler vererek; bana ilerisi için yeni bir yol göstermiş oldu. Kısaca projeye başladığım zamanla şu anki ben arasında oldukça fark var. Bana katkısı oldukça fazla olan hocama net bir şekilde şunu söylemeliyim.

Sayın Özgür Hocam,

Bu projedeki deneyimin, benim için sadece bir başlangıç olmadığına, aynı zamanda bu alanda uzun bir yolculuğun temelini oluşturduğuna emin olabilirsiniz.

SAYGILARIMLA

Gamze DİNÇ
Ocak, 2024

İçindekiler

Bölüm 1

K-ORTALAMALAR KÜMELEME

1.1 Gözetimsiz Öğrenme

İstatistiksel öğrenme problemleri genellikle iki kategoriden birine dahil olur: gözetimli (denetimli) veya gözetimsiz (denetimsiz) öğrenme. Gözetimli öğrenme, her bir gözlem için x_i ölçümlerini gözlemleyip buna karşılık gelen bir y_i ilişkilendirilmiş yanıt değişkenine sahip olduğumuz durumu ifade ederken, gözetimsiz öğrenmede ise her bir gözlem için x_i ölçümlerini gözleriz, ancak bu ölçümlere karşılık gelen bir y_i ilişkilendirilmiş yanıt değişkeni bulunmaz.

Tahmin edilecek bir yanıt değişkeni olmadığından doğrusal regresyon modeli uygulamak mümkün değildir. Yanıt değişkeni olmaması (etiketsizlik) genellikle gözetimsiz öğreneme için kullanılan bir terimdir. Bir veri noktasının etiketsiz olması, bu noktanın bir hedef etikete (yanıta) sahip olmaması anlamına gelir. Bu da veri setindeki yapıları kendi başına tanımlaması ve öğrenmesini sağlar. Bunu, veri setindeki veri örneklerinin komşuluk ilişkileri, uzaklıkları, benzerlikleri gibi yapıları inceleyerek çıkarımlar yaparak yapmaya çalışır.

Gözetimsiz öğrenme; pazarlama ve müşteri segmentasyonu, tıp, finans, bilgisayar ağları ve güvenlik başta olmak üzere hayatın her alanında kullanılır.

Örnek 1. Kullanıcıların geçmiş izleme veya dinleme alışkanlıklarını analiz ederek benzer profillere sahip grupları belirleyebilir ve bu gruplara özel içerik önerileri sunabiliriz. Bu şekilde, benzer ilgi alanlarına sahip kullanıcılar arasında etkileşimi artırarak daha kişiselleştirilmiş ve tatmin edici bir kullanıcı deneyimi sağlayabiliriz. Bu bir gözetimsiz öğrenme problemidir çünkü bir veri kümesi temelinde belirgin kümeler dahil olmak üzere yapıyı keşfetmeye çalışıyoruz.

Kümeleme ve boyut azaltma (temel bileşen analizi) gibi teknikler önemli gözetimsiz öğrenme algoritmalarıdır. Kümelemenin amacı, $x_1, ..., x_n$ temelinde gözlemlerin mümkün olduğunca farklı gruplara ait olup olmadığını belirlemektir. Başka bir deyişle etiketsiz veri noktalarını benzer özelliklere sahip gruplara ayırarak bu yapıları ortaya çıkarmaktadır. Ayrıca, boyut azaltma teknikleri kullanarak veri setini daha anlaşılır ve işlenebilir bir formata getirebilir. Bu, veri setindeki önemli özellikleri vurgulayarak anlamayı artırabilir ve modelin gürültüye (dış etkenlere) duyarlılığını azaltabilir. Aslında burada da tam olarak bunları inceleyeceğiz.

1.2 Kümeleme

Kümeleme, gözetimsiz öğrenme yöntemlerinden biri olup bir veri kümesindeki alt grupları veya kümeleri bulmak için kullanılan oldukça geniş bir algoritmik modelleme tekniğidir.

Veri setimizi kümelerken amacımız benzer özelliklere sahip gözlemleri bir araya getirerek farklı gruplara bölmeye çalışmak böylece her bir grup, birbirine oldukça benzer gözlemleri içerirken, farklı gruplar arasındaki gözlemler birbirinden belirgin bir şekilde farklılık gösterecektir. İki nesne arasındaki benzerlik, genellikle bu nesneler arasındaki özelliklerin ne kadar ortak olduğuyla ölçülür. Uzaklık ise genellikle bu nesneler arasındaki farklılıkları veya özelliklerindeki mesafeyi ifade eder. İki nesne arasındaki benzerlik ne kadar düşükse, uzaklık o kadar yüksektir. Bunu somutlaştırmak için iki veya daha fazla gözlemin benzer veya farklı olması ne anlama geldiğini bir örnek üzerinden açıklamaya çalışalım.

Örnek 2. Bir mağaza, müşterilerini alışveriş alışkanlıklarına göre gruplandırmak istiyor. Bunun için alışveriş sıklığı ve harcama miktarlarına göre benzer ve farklılıklarını değerlendirebilir. Bu kümeleme sayesinde her bir grup, birbirine benzeyen müşteri profillerini içerecek. Örneğin, sık alışveriş yapanlar genellikle yüksek harcama yaparken, seyrek alışveriş yapanlar daha düşük harcama yapabilir. Bu şekilde, farklı gruplar arasındaki müşteri davranışlarında belirgin farklılıklar ortaya çıkacaktır. Bu örnek sonucunda da sık alışveriş yapanlar, orta düzeyde alışveriş yapanlar ve seyrek alışveriş yapanlar diye kümeler elde edebilir. Sık alışveriş yapanlara özel promosyonlar , orta düzeyde alışveriş yapanlara yeni ürünleri tanıtmak, ve seyrek alışveriş yapanları daha sık ziyaret etmeleri için teşvik edilebilir. Ya da başka özellikleri de dahil ederek belirli bir yaş grubundaki müşterilere özel kampanyalar düzenler veya belirli gelir seviyelerine sahip müşterilere özel indirimler sunmak gibi stratejiler geliştirebilir. Yani kümeleme yöntemlerinin benzer özelliklere sahip gözlemleri bir araya getirerek farklı gruplara böldüğünü göstermektedir. Genel olarak bu şekilde kümelerin anlamlılığını ve faydalılığını kullanabiliriz.

Kümeleme yaparken, bazı özelliklerin sağlanması gerekir.

$C_1, C_2, \ldots, C_K$ her kümedeki gözlemlerin indislerini içeren küme adları olmak üzere

$$C_1 \cup C_2 \cup \ldots \cup C_K = \bigcup_{k=1}^{K} C_k = \{1, \ldots, n\}.$$

Her bir C_k, bir alt küme oluşturur ve tüm C_k kümeleme sonuçları, 1 ile n arasındaki tüm elemanları içerir. Bu ifade, bir veri kümesini oluşturan gözlemlerin, en azından bir kümeye ait olduğunu ve bu kümelerin birleşiminin tüm gözlemleri içerdiğini temsil eder.

$$C_k \cap C_{k'} = \emptyset \quad \text{tüm } k \neq k'$$

Bu ifade, tüm farklı k ve k' çiftleri için C_k ve $C_{k'}$ kümelerinin kesişimlerinin boş küme olduğunu belirtir. Yani hiç ortak elemanları yoktur. Hiçbir gözlem birden fazla kümeye ait değildir.

Kümeleme oldukça önemli ve yararlı bir yöntem olmasının yanısıra bazı olumsuz özellikleri vardır.

- Bazı kümeleme algoritmalarında küme sayısı önceden belirlenmelidir. Doğru küme sayısını bulmak zor ve kesin değildir.

- Kümeleme sonuçları, kullanılan benzerlik metriklerine ve algoritmalara bağlı olarak hassaslık gösterebilir. Bazı durumlarda, kümeleme sonuçları istenen düzeyde net olmayabilir

- Büyük veri setleri üzerinde çalışmak ve çok boyutlu verileri kümelemek bazen zorlu olabilir. Bazı algoritmalar boyutlanabilirlik sorunlarına neden olabilir.

K- ortalamalar kümeleme, hiyerarşik kümeleme, spektral kümeleme gibi çeşitli kümeleme yöntemleri bulunmaktadır. Ayrıca benzer ve farklılıkları ölçen yöntemlerde çeşitlidir. Öklid, Manhattan, Minkowski gibi... Kümeleme yöntemlerinin kullanımı, uygulama kapsamına, veri setinin özelliklerine ve analiz yapma amacına bağlı olarak değişir. Bu nedenle, doğru kümeleme yöntemini seçmek ve sonuçları doğru bir şekilde yorumlamak önemlidir.

Biz burada birçok kümleme yöntemi olmasına karşın sadece K-ortalamalar kümelemeyi inceleyeceğiz.

1.3 K-Ortalamalar Kümeleme

K-ortalamalar kümeleme, kümeleri oluşturmak için yaygın olarak kullanılan bir kümeleme yöntemidir. Veri setini belirli bir sayıda K kümesine bölerek benzer özelliklere sahip veri noktalarını aynı grupta toplamayı amaçlar. K-ortalama kümeleme yapabilmek için önce istenen küme sayısı K'yı belirlememiz gerekmektedir. Belirlenen K sayısı, kümeleme sürecinin temelini oluşturur.

Diğer adımlara ve algoritmaya geçmeden önce işlemleri anlamak için bazı terimleri anlamaya çalışalım.

Biz gerçekleştirilen ayrıştırma sonunda elde edilen kümelerin, küme içi benzerliklerinin en yüksek seviyede ve kümeler arası benzerliklerinin ise en düşük seviyede olması gerektiğinin temel amaç olduğunu biliyoruz.

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\} \tag{1.1}$$

(1.1) bu formül, bir optimizasyon problemini aynı zamanda hedemizi ifade eder. Burada, $C_1, \ldots, C_K$ kümeleme sonuçlarıdır, yani veri kümesini K adet alt küme ya da kümeleme grubuna böldüğümüzü temsil ederler.

Her bir $W(C_k)$ ifadesi, bir kümenin içindeki veri noktalarının birbirine ne kadar benzer olduğunu ölçer. Minimizasyon probleminin genel amacı, tüm kümelerin içsel uyumluluğunu maksimize ederek veri kümesini en iyi şekilde kümelemektir. Bu şekilde, her bir kümenin belirgin ve benzer özelliklere sahip veri noktalarının olması hedeflenir.

Her bir kümenin içindeki gözlemlerin birbirine ne kadar benzer olduğunu ifade eden bir ölçümü minimize etmeye çalışırız. Biz amacımızı gerçekleştirmek için bunu uygulanabilir hale getirmeliyiz. O zaman iç-küme varyasyonunu tanımlamamız gerekiyor. Şimdi $W(C_k)$ ifadesini daha detaylı inceleyelim.

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \tag{1.2}$$

(1.2) Bu formül, küme C_k'nin içindeki gözlemlerin (veri noktalarının) birbirinden ne kadar farklı olduğunu ölçen ifadedir. Bunu elde etmek için ilk olarak C_k kümesindeki her bir eleman çifti i ve i' için, p özellik boyutu üzerinden elemanlar arasındaki farkların karesini alırız: $(x_{ij} - x_{i'j})^2$.

Bu farkların karelerini toplamak için $j = 1$ ila p arasındaki tüm özellikler için toplam yapılır.

Sonra C_k kümesindeki tüm eleman çiftlerinin bu özellikler arasındaki kare farklarının toplamının, k. küme içindeki gözlem sayısına bölerek, kümenin içindeki elemanların birbirleriyle olan benzerliğini veya uzaklığını ölçen değeri elde ederiz. Elde edilen değer ne kadar küçükse,o kadar kümenin içindeki gözlemler arasındaki benzerlik artar.

(1.1) bu ifade içerisine (1.2) ifadesini koyarak birleştirelim. Böylece amacımıza daha net bir şekilde ulaşabiliriz.

$$\min_{C_1,\ldots,C_K} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\} \tag{1.3}$$

Bu problemin amacı, K farklı küme $(C_1, C_2, \ldots, C_K)$ seçerek, kümeler arası benzerliğin düşük, küme içi benzerliğin yüksek olmasını sağlamaktır. Şimdi (1.3) problemini çözmek için bir algoritma bulmamamız lazım çünkü küme merkezi ve küme atamalarının optimize edilmesi karmaşık bir süreçtir ayrıca başlangıç küme atamalarına duyarlıdır dolayısıyla farklı başlangıç durumları ile çalışan ve en iyi sonuçları bulmaya çalışan bir algoritma ile ifade etmeye çalışmalıyız. Bu minimize etme problemi bir süreci tekrarlayan bir yaklaşımı ifade eder. Kümeleme sonuçları değişmeyene kadar tekrarlanır.

Algoritmanın başarısının nedeni, her adımda küme merkezlerinin ve gözlemlerin kümelere atanmasının belirli bir ölçü olan (1.5) geliştirmiş olması. Yani K-ortalama algoritması bir yerel optimuma ulaştığı için, başlangıçta rastgele belirlenen küme atamalarına bağlıdır. Bu nedenle, algoritmanın farklı rastgele başlangıçlarla çalıştırılması ve en iyi çözümün seçilmesi önerilir. Bu durum, algoritmanın her seferinde farklı sonuçlar üretebileceği anlamına gelir.

$$\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij} \tag{1.4}$$

olmak üzere

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2 \tag{1.5}$$

Kanıt.

$$
\begin{aligned}
&\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \\
&= \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij}^2 - 2x_{ij}x_{i'j} + x_{i'j}^2) \quad \text{(Çarpma açılımı)} \\
&= \frac{1}{|C_k|} \left(2 \sum_{i \in C_k} \sum_{j=1}^{p} x_{ij}^2 - 2 \sum_{i,i' \in C_k} \sum_{j=1}^{p} x_{ij}x_{i'j} \right) \quad \text{(Toplam sembolü dışa çıkartma)} \\
&= 2 \left(\frac{1}{|C_k|} \sum_{i \in C_k} \sum_{j=1}^{p} x_{ij}^2 - \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} x_{ij}x_{i'j} \right) \\
&= 2 \left(\sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij}^2 - \bar{x}_{kj}x_{ij} + \bar{x}_{kj}^2) \right) \quad \text{(Ortalama değeri ekleme ve çıkartma)} \\
&= 2 \sum_{i \in C_k} \sum_{j=1}^{p} (x_{ij} - \bar{x}_{kj})^2 \quad \text{(Karesel farkların toplamı)}
\end{aligned}
$$

Bu adımlar, verilen ifadenin, tanımlanan $\bar{x}_{kj} = \frac{1}{|C_k|} \sum_{i \in C_k} x_{ij}$ ifadesi kullanılarak ikinci ifadeyle eşit olduğunu gösterir. □

Böylece hedefimizi gerçekleştirmiş olduk.

1.4 Dirsek Yöntemi

K- ortalamalar kümeleme algoritmasında küme sayısını doğru belirlemeye çalışmak oldukça önemlidir. Küme sayısını önceden kesin bir şekilde belirlemek genellikle mümkün değildir. Doğru küme saysını bulmak için çeşitli yöntemler vardır ancak biz burada grafiksel inceleme şansımızın olduğu bize daha basit ve hızlı ayrıca direkt anlamlı sonuçlar sunması nedeniyle dirsek yöntemini anlatacağız.

Tanım 1. Dirsek yönteminde öncellikle noktaların her bir farklı K değerine (küme sayısına) göre küme merkezine uzaklıklarının karesinin toplamı hesaplanmaktadır. Bu işlemi her küme sayısı için uyguladıktan sonra küme içi kümelenme hatasının grafiği çizilir. Başlangıçta küme sayısı arttıkça hata genellikle hızlıca azalırken, optimal küme sayısına ulaşıldığında bu azalma hızı belirgin şekilde yavaşlayacaktır. Bu nedenle grafikte genellikle bir "dirsek" veya "kıvrım" noktası gözlemlenir. Bu nokta, küme sayısını belirlemede bir rehber olarak kullanılır, çünkü eklenen her kümenin küme içi hata üzerindeki azalma oranı bu noktadan sonra yavaşlar. Yani küme içi kareler toplamı ne kadar düşükse, küme içindeki gözlemler birbirine o kadar benzer ve kümeleme algoritması o kadar iyi performans gösterir. Yani optimal küme sayısına ulaşıldığında, küme sayısındaki küme içi kareler toplamı düşüş hızı azalır ve grafikte bir "dirsek" oluşur. Bu "dirsek" noktası bize optimal küme sayısını gösterir.

Küme içi kareler toplamının formulü

$$(WCSS) = \sum_{i=1}^{K} \sum_{j=1}^{p} \|x_{ij} - c_i\|^2 \tag{1.6}$$

Burada K küme içindeki toplam küme sayısını temsil eder. p_i ise i-inci kümenin içindeki veri noktalarının sayısını gösterir. x_{ij} ise i-inci kümenin j-inci veri noktasını temsil eder. c_i ise i-inci kümenin merkezini ifade eder.

İlk $\sum$ ifadesi, küme sayısını (K) temsil eder ve her bir küme için tekrarlanır. İkinci $\sum$ ifadesi, i-inci kümenin içindeki veri noktalarını (p_i) temsil eder ve her bir veri noktası için tekrarlanır. $\|x_{ij} - c_i\|^2$, j-inci veri noktasının i-inci küme merkezine olan uzaklığının karesini ifade eder.

Bu ifade, her bir veri noktasının ait olduğu küme merkezine olan uzaklıklarının karelerini toplar. Bu, küme içi kareler toplamını elde etmek için kullanılır ve genellikle kümeleme algoritmalarının performansını ölçmede bir ölçüt olarak kullanılır.

K-Ortalamalar kümeleme analizi sırasında en uygun küme sayısını belirlemek için oldukça fazla kullanılan bu yöntem net bir dirsek noktası belirginse ve optimal küme sayısı açıkça tanımlanabiliyorsa etkili olur. Bu nedenle, bazen farklı yöntemler de kullanılabilir. Bazen de görselde kararsız kalınan küme sayıları için deneme yanılma yaparak bizim için en iyi küme sayısını bulabiliriz. Bu yüzden farklı küme sayıları için performans ölçütlerini göz önüne alarak, kümeleme sonuçlarının doğrulanması ve uygulamanın gereksinimlerine göre bizim için en iyi küme sayısını bulmak daha iyi olacaktır.

Şimdi yaptığımız tüm K-ortalamalar kümeleme algoritması işlemlerini rastgele ve az veri sayısı ile görselleştirelim.

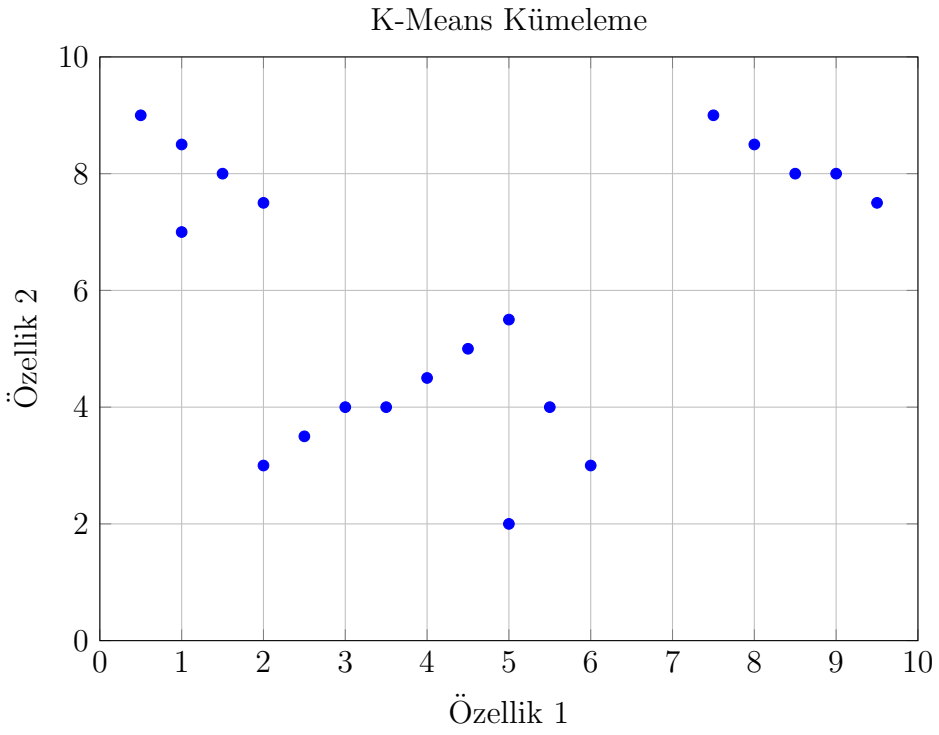
1.5 K-Ortalamalar Kümeleme Algoritması

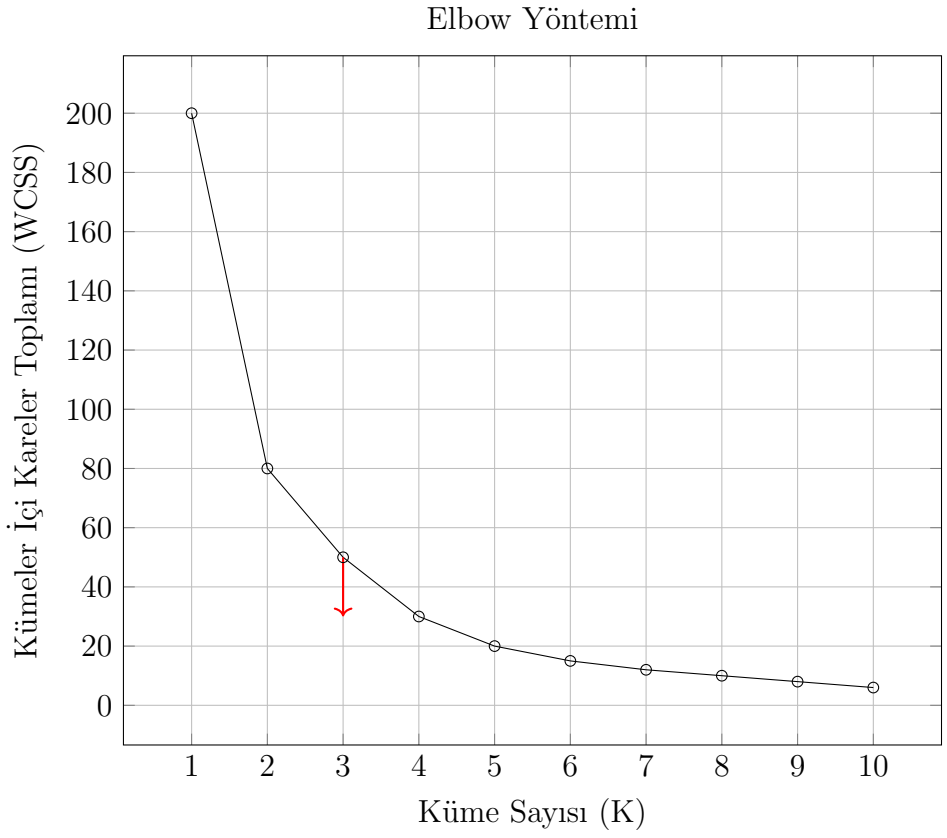
1. **Başlangıç:** Kümelerin sayısını belirle (K) ve her veri noktasını rastgele bir kümeye ata.

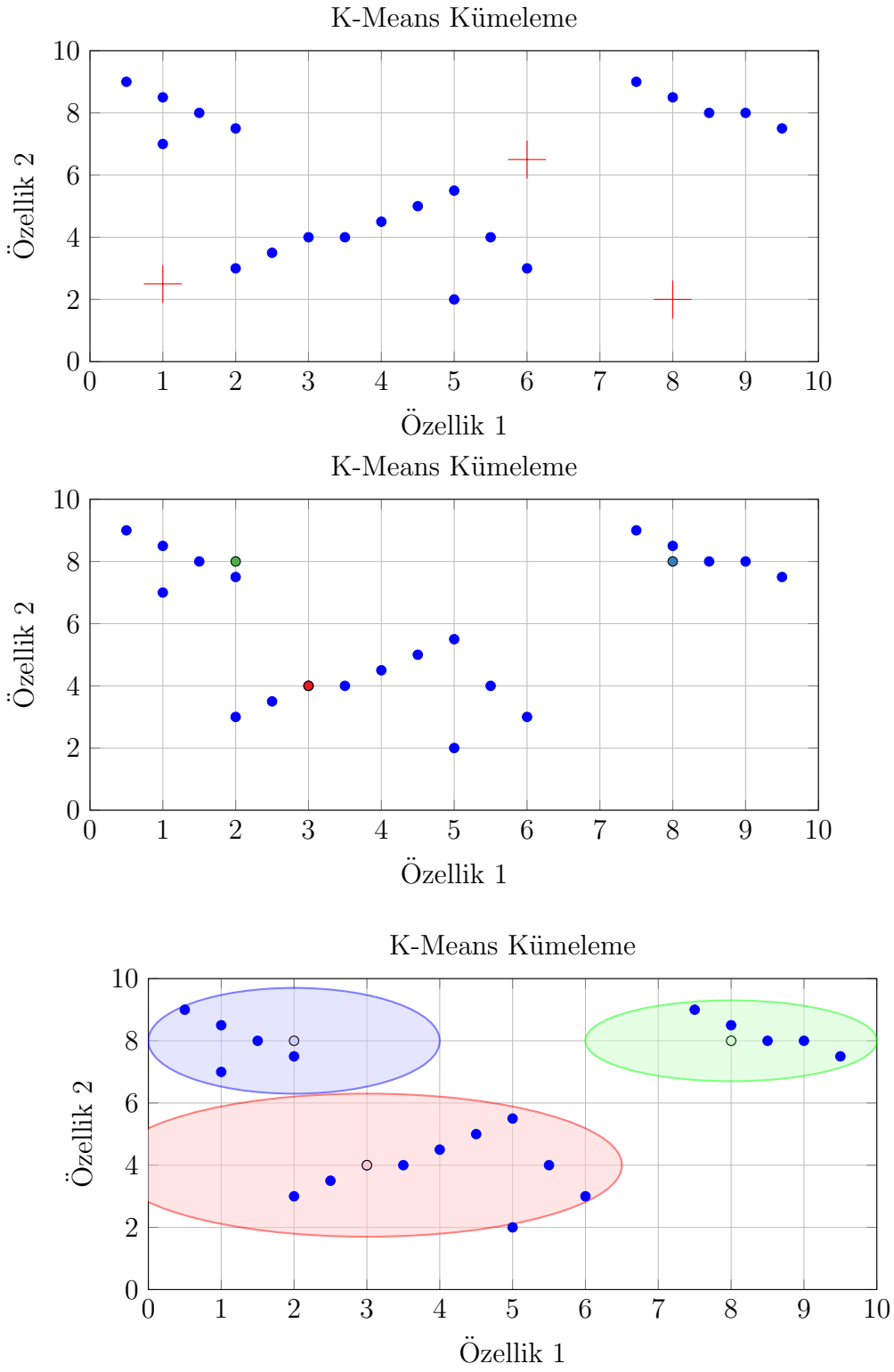
2. **Küme Merkezlerini Hesapla:** Her küme için küme merkezini, o kümedeki veri noktalarının ortalaması olarak hesapla:

3. **Yeniden Atama:** Her veri noktasını, ona en yakın küme merkezine atanacak şekilde kümelere yeniden ata.

4. **Tekrarla:** Küme merkezleri veya atamalar değişmeyene kadar 2. ve 3. adımları tekrarla.


K-Means Kümeleme
Özellik 2
Özellik 1
0
1
2
3
4
5
6
7
8
9
10


Elbow Yöntemi
Kümeler İçi Kareler Toplamı (WCSS)
Küme Sayısı (K)
0
20
40
60
80
100
120
140
160
180
200
1
2
3
4
5
6
7
8
9
10



Şekil 1.1: Son Kümeleme Sonuçları

Bölüm 2

KÜMELEME ÖRNEKLERİ VE DEĞERLENDİRMELER

2.1 Meme Kanseri Veri Seti İçin Python Örneği

Örnek 3. 30 farklı özellik üzerinden 569 kişinin verileri bulunan bir veri setini K-ortalamalar kümeleme algoritması ile kümelemeye çalışacağız ve elde ettiğimiz kümelerin anlamlılığını değerlendireceğiz.

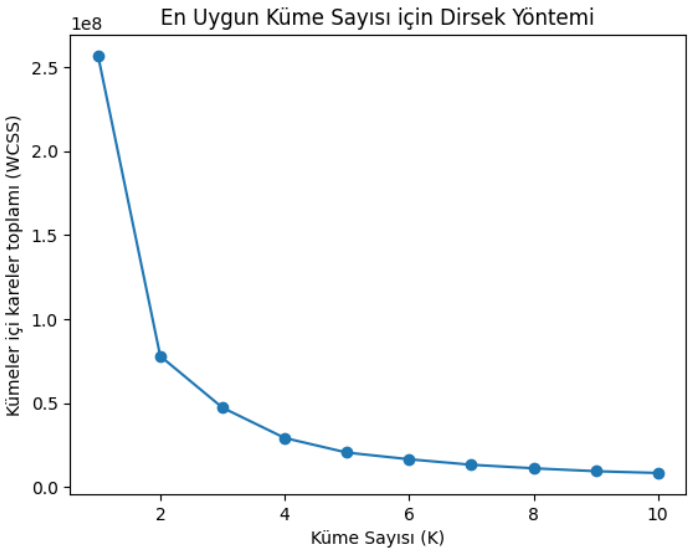
Yaptığımız kümelerin anlamlılığını ve kümelerin faydalılığını anlamak için sınıfları belli olan (etiketlenmiş) veri setini seçmemiz iyi olacaktır. Bu yüzden biz verilerimizi meme kanseri veri setinden aldık. Bu veri setinde kötü huylu sınıfta 212 örnek bulunurken, iyi huylu sınıfta 357 örnek bulunmaktadır. Buraya baktığımızda kümenin ikiye ayrılması gerektiğini görüyoruz.

Önemli K- ortalamalar kümeleme algoritmasında küme sayısı belli değildir. Ayrıca verilerimiz etiketsizdir. Bu örnekte kümelemenin ne kadar doğru bir şekilde veri setini anlamlı alt gruplara böldüğünü değerlendireceğimiz için sınıfları belli olan veri setini aldık.

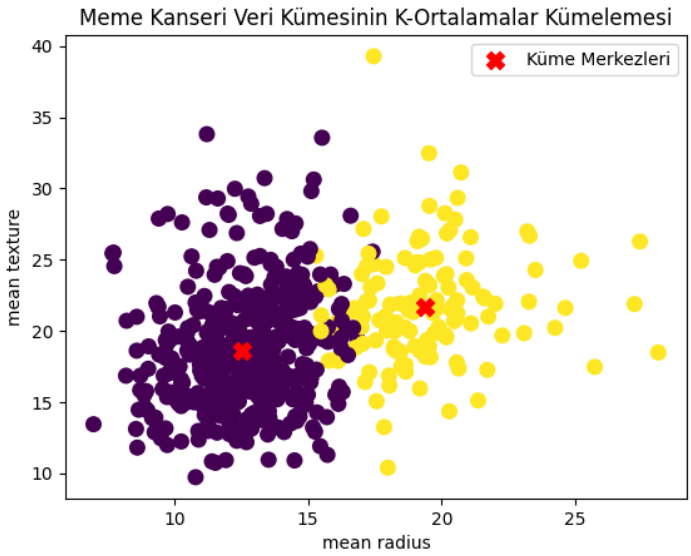
Şimdi yapacağımız tüm işlemleri adım adım açıklayalım. İlgili kodların tamamı EK-1'e konmuştur.

Adım 1: Meme kanseri veri setini yükleyip bir DataFrame oluşturduk. DataFrame veriyi düzenlemek, analiz etmek için kullanılır.

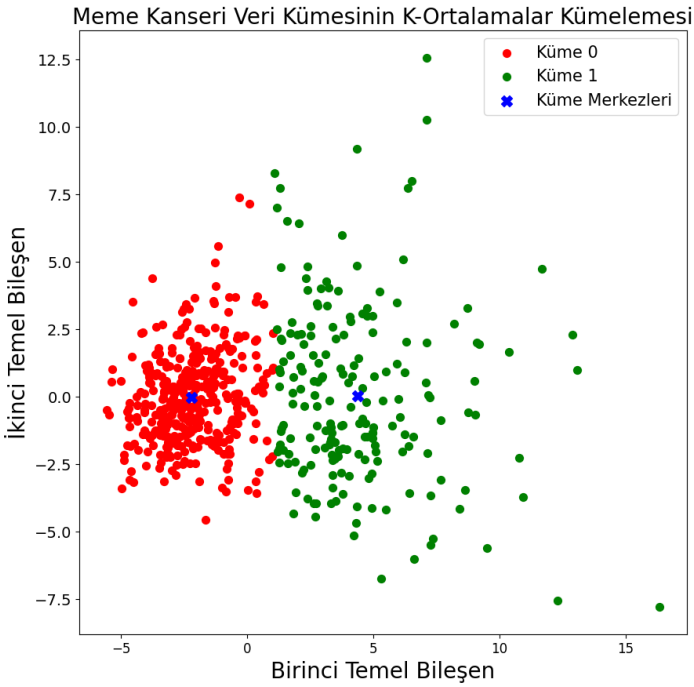
Adım 2: Küme sayısını önceden bilmediğimizden doğru küme sayısını bulmamız gerekir. Bunun için çeşitli yöntemler olsa da, biz dirsek yöntemini kullanıcağız. Meme kanseri veri seti üzerinde K-ortalamalar algoritması uygulayarak 1'den 10'a kadar küme sayıları için kümeler içi kareler toplamını hesaplatıyoruz Elde edilen değerleri grafikle görselleştiriyoruz. Ve bu grafiğe bakınca dirsek noktasını bulurken biraz dikkatli bakmamız sonucu küme sayısının 2 olacağını görebiliriz.

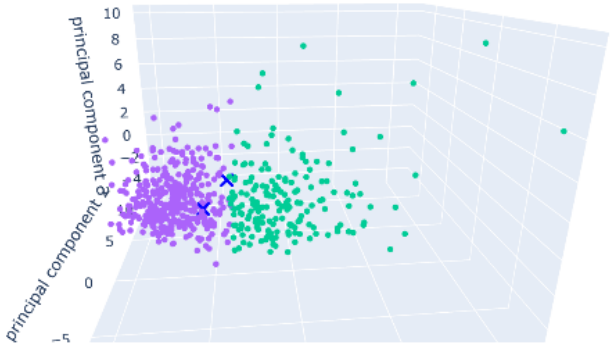


Adım 3: Küme sayımızı 2 olarak belirlediğimize göre artık K-ortalamalar kümeleme algoritmasını uygulayabiliriz. Kümelerken tüm özellikleri dahil etsekte görselleştirme işlemini 2 özellik üzerinden yaptık. Görsele bakarsak biraz öbekleşmiş iki kümeyi ve küme merkezlerini görüyoruz.



Adım 4: Veri setimizde kümeleme işlemimiz bitti. Biz kümelemenin iki boyuta ve üç boyuta indirince kümelemenin nasıl olacağını merak ediyoruz. Bunun için temel bileşen analizi uyguluyoruz ve sonrasında 2 ve 3 boyutlu olarak yaptığımız temel bileşen analizleri için tekrar K-ortalamalar kümeleme algoritmasını uyguluyoruz.





Bu işlemi neden yaptığımızı anlamak için temel bileşen analizini biraz açıklayalım. TBA, veri kümesindeki değişken sayısını azaltarak karmaşık veri yapılarını basitleştirir. Özellikle çok sayıda değişken içeren veri setlerinde, analiz ve yorumlama sürecini daha anlaşılır hale getirebilir. Bu adımda genel olarak TBA yapılarak kümelerimizin nasıl etkilediğini inceleyeceğiz. Burada Kanser ve sonra ki örnek için Spotify verilerini TBA ile 2 ve 3 boyuta indirip tekrar K-Ortalama kümeleme algoritması ile kümeleme yapacağız. TBA ile boyut azaltmanın, veriyi daha anlaşılır hale getirerek analiz ve yorumlamayı kolaylaştırdığını görebiliriz. Ancak, unutmamalıyız ki: TBA ile boyut azaltmanın, kümeleme sonuçları ile doğrusal bir ilişkisi yoktur. Boyut azaltma ve kümeleme, farklı amaçlara hizmet eden analiz yöntemleridir.

Adım 5: Yaptığımız K- ortalamalar kümeleme algoritması sonucu esas kümeleme, 2 temel bileşen için kümeleme ve 3 boyut için kümelemedeki değerleri hata

matrisi ve hata oranı başarı sonuçlarına göre daha iyi analiz edebilmek için tablo yaparak görselleştiriyoruz.

-	Esas Kümeleme	TBA 2 boyut	TBA 3 boyut
Hata Oranı	0.14586	0.0931	0.0896
Hatalı kötü huylu sınıf oranı	0.38679	0.0448	0.1745
Hatalı iyi huylu sınıf oranı	0.0028	0.1745	0.0392
Küme 0 toplam veri sayısı	438 veri	378 veri	380 veri
Küme 0 doğru ve yanlış v. s.	356 D, 82 Y	341 D, 37 Y	343 D, 37 Y
Küme 1 toplamveri sayısı	131 veri	191 veri	189 veri
Küme 1 doğru ve yanlış v. s.	130 D, 1 Y	175 D, 16 Y	175 D, 14 Y

Tablo 2.1: K-ortalamalar kümeleme sonucu yaptığımız kümelemelerin değerleri

Notlar:

$$\text{Hata Oranı } = \frac{\text{Toplam Yanlış Sınıflandırmalar}}{\text{Toplam Veri Sayısı}}$$

– "D" doğru, "Y" yanlış sınıflandırma sayısını "v. s." veri sayısını temsil eder.

Adım 6: Meme kanseri veri seti için yaptığımız tüm K- ortalamalar kümeleme sonuçlarını değerlendiriyoruz. *Tablo 2.1* bakarsak meme kanseri veri seti için yaptığımız kümelemelerin oldukça başarılı bir sonuç verdiğini görmüş oluruz. 3 boyutlu TBA ise aralarından verileri en çok açıklayandır. Bu örnek özelinde temel bileşen analizinin, meme kanseri verilerini daha anlaşılır hale getirerek doğru teşhisler konusunda önemli bir araç olduğunu çıkarabiliriz.

2.2 Spotify Kümeleme Örneği

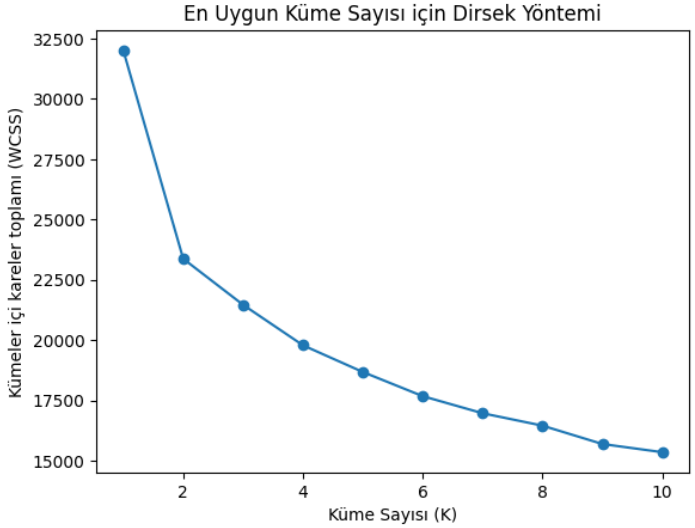
Örnek 4. Başlangıçta, Kaggle'dan aldığımız Spotify veri setimiz 14 bin örnek ve 21 özellik içeriyordu. Biz kümeleme işlemini daha açıklayıcı hale getirmek amacıyla veri sayısını 2000'e indiriyoruz. Bu yeni veri setimizde her biri 21 özellik içeren toplam 2000 örnek bulunmaktadır. Bu verileri K-ortalamalar kümeleme algoritması ile kümelemeye çalışacağız ve elde ettiğimiz kümelerin anlamlılığını değerlendireceğiz.

Bu örnekte, verileri indirirken etiketlenmiş müzik türlerini kümelemeye çalışmak istediğimiz için en zıt 2 müzik türü olarak yeni veri setini oluşturduk. Etikelenmiş verilerimiz olan 2000 örneğin 1000'i akustik müzik türü diğer 1000'i ise black metal müzik türlerinden oluşmaktadır. Yeni veri seti, müzik türlerini daha belirgin bir şekilde ayırt etmeye olanak tanıyacak şekilde düzenlenmiştir. Bakalım kümelerimiz bunu ne kadar doğru bir şekilde ayırt edebilecek.

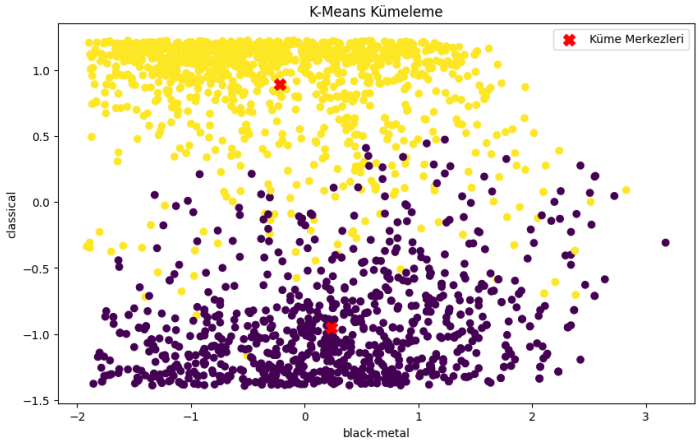
Tekrardan hatırlatma yapmam gerekirse biz biliyoruz ki K- ortalamalar kümeleme algoritmasında küme sayısı belli değildir. Ayrıca verilerimiz etiketsizdir. Bu örnekte

kümelemenin ne kadar doğru bir şekilde veri setini anlamlı alt gruplara böldüğünü değerlendirmek için akustik müzik ve black metal müziği olmak üzere iki belirgin müzik türü seçilmiştir.

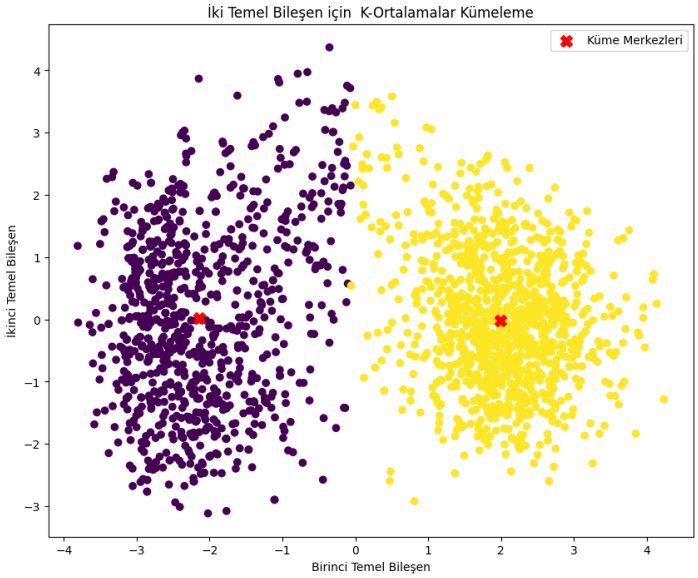
Adım 1: Veri setimize dirsek yöntemi uygulayarak bizim için en iyi küme sayısının hangisi olacağına karar vermeye çalışıyoruz. Görselde açıkca görüyoruz ki dirsek noktası 2 olduğundan küme sayımız iki olmalıdır.

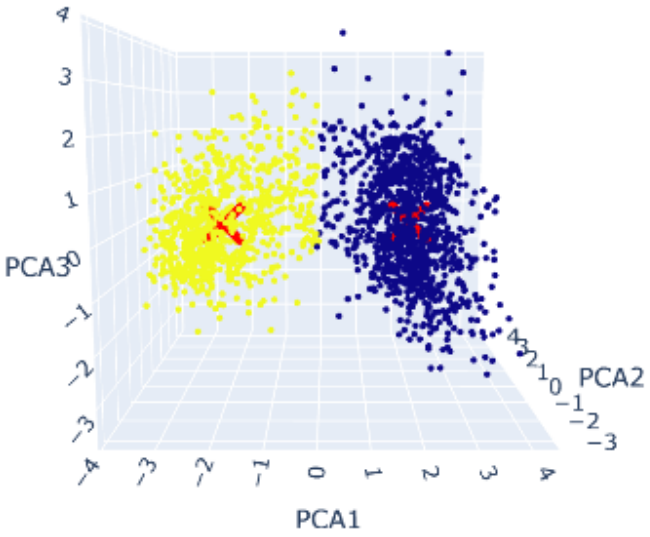


Adım 2: K sayısını bulduğumuza göre artık kümeleme yapabiliriz.



Adım 3: Temel bileşen analizi ile veri setimizi önce iki boyuta indirip sonra 3 boyuta indirip tekrar kümeleme yaptık. Görsellere baktığımızda net bir şekilde kümelerimizin ayrıldığını görüyoruz buradan başarılı bir kümeleme işlemi yaptığımızı sezebiliriz.





Adım 4: Spotify veri setimizde 1000 verinin akustik 1000 verinin black-metal sınıfına dahil olduğunu biliyorduk. Bunu ilk tüm özellikleri kullanarak yaptığımız K-ortalamalar algoritması ile daha sonra 2 boyuta indirip daha sonra 3 boyuta indirerek yaptığımız K- ortalamalar kümesini bir tablo ile hata oranı ve diğer genel özelliklerin doğruluklarını kıyaslamak için tabloya aktarıyoruz böylece yorumlamamız daha kolay olacaktır.

Notlar:

$$\text{Hata Oranı} = \frac{\text{Toplam Yanlış Sınıflandırmalar}}{\text{Toplam Veri Sayısı}}$$

- "D" doğru, "Y" yanlış sınıflandırma sayısını "v. s." veri sayısını temsil eder.

-	Esas Kümeleme	TBA 2 boyut	TBA 3 boyut
Hata Oranı	0.0425	0.0435	0.0435
Hatalı black-metal sınıf oranı	0.026	0.026	0.026
Hatalı classical sınıf oranı	0.059	0.061	0.061
Küme 0 toplam veri sayısı	1033 veri	1035 veri	1035 veri
Küme 0 doğru ve yanlış v. s.	974 D, 59 Y	974 D, 61 Y	974 D, 61 Y
Küme 1 toplam veri sayısı	967 veri	965 veri	965 veri
Küme 1 doğru ve yanlış v. s.	941 D, 26 Y	939 D, 26 Y	939 D, 26 Y

Tablo 2.2: Spotify veri setinde yaptığımız kümelemeler sonucu hata tablosu

Adım 5: Genel olarak bir değerlendirme yaparsak. En çok başarılı olan kümelemenin esas kümeleme olduğunu görüyoruz. Oldukça başarılıdır çünkü hata oranı düşük ve doğru sınıflandırma oranları yüksektir. TBA kullanarak veriyi 2 boyuta indirdiğimizde, hata oranı hafifçe artmış ancak bu artış bu veri sayısı için fazla önemli olmayabilir. TBA ile 3 boyuta indirdiğimizde hata oranı aynı kaldığını görüyoruz yani bunu yapmak bizim veri setimiz için gerkesizdi. Buradan yüksek boyutlu veriyi düşük boyutlara indirirken, doğru boyut seçimi önemli olduğunu çünkü gereksiz boyut indirme performansı olumsuz etkileyebileceğini ve gereksiz maliyet oluşturabileceğini söyleyebiliriz

►Oldukça başarılı bir şekilde yaptığımız iki farklı alanı içeren iki örnek için net bir şekilde TBA'nın K-ortalamalar kümeleme ile doğrusal bir ilişkisi olmadığını söyleyebiliriz.

► Meme kanseri verilerini doğru bir şekilde kümelemek, erken tanı, daha etkili tedavi stratejileri ve risk değerlendirmeleri sağlayarak hastaların kişiselleştirilmiş tedavilere erişimini artırabilir. Spotify verilerini doğru bir şekilde kümelemek günlük hayatta müzik platformları için öneri sistemleri geliştirebilir, Konser veya etkinlik organizasyonlarında hedef kitle belirlemede kullanabilir, reklamcılık stratejilerini optimize edebilir.

Ek A

Ek 1

A.1 Ek 1: Kanser Verisi Kodları

kanser_kortalamalarkümeleme

January 25, 2024

[28]:
```
from sklearn.datasets import load_breast_cancer
breast = load_breast_cancer()
breast_data = breast.data
breast_data.shape
breast_labels = breast.target
breast_labels.shape
import numpy as np
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data,labels],axis=1)
final_breast_data.shape
labels = np.reshape(breast_labels,(569,1))
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data,labels],axis=1)
final_breast_data.shape
import pandas as pd
breast_dataset = pd.DataFrame(final_breast_data)
features = breast.feature_names

features_labels = np.append(features,'label')
breast_dataset.columns = features_labels
breast_dataset.head()
```

[28]:

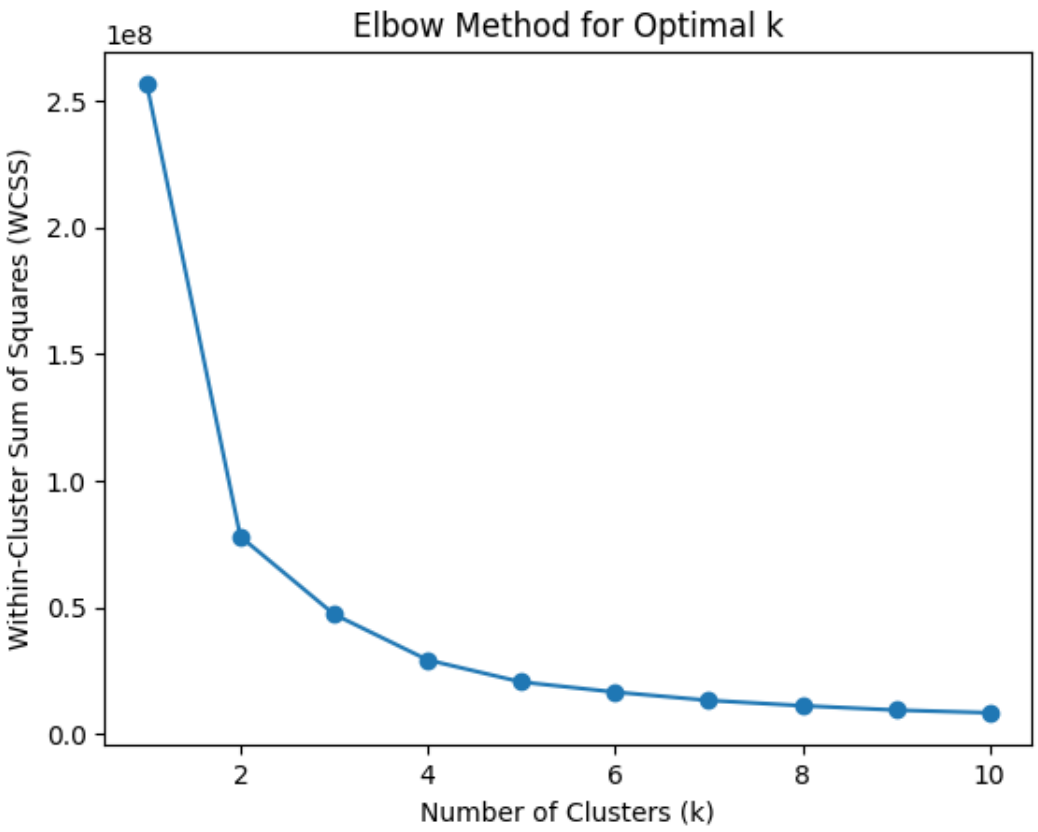
	mean radius	mean texture	mean perimeter	mean area	mean smoothness	\
0	17.99	10.38	122.80	1001.0	0.11840	
1	20.57	17.77	132.90	1326.0	0.08474	
2	19.69	21.25	130.00	1203.0	0.10960	
3	11.42	20.38	77.58	386.1	0.14250	
4	20.29	14.34	135.10	1297.0	0.10030	

	mean compactness	mean concavity	mean concave points	mean symmetry	\
0	0.27760	0.3001	0.14710	0.2419	
1	0.07864	0.0869	0.07017	0.1812	
2	0.15990	0.1974	0.12790	0.2069	
3	0.28390	0.2414	0.10520	0.2597	
4	0.13280	0.1980	0.10430	0.1809	

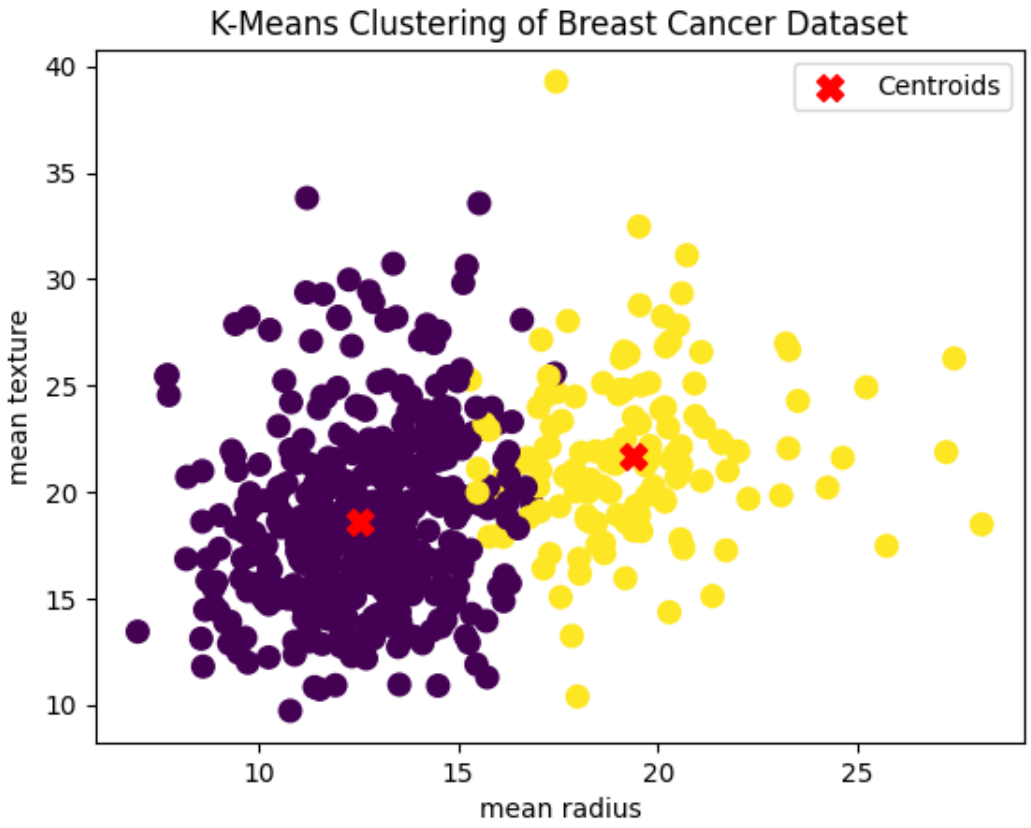
	mean fractal dimension	…	worst texture	worst perimeter	worst area	\

```
0                 0.07871  …          17.33           184.60      2019.0
1                 0.05667  …          23.41           158.80      1956.0
2                 0.05999  …          25.53           152.50      1709.0
3                 0.09744  …          26.50            98.87       567.7
4                 0.05883  …          16.67           152.20      1575.0

   worst smoothness  worst compactness  worst concavity  worst concave points  \
0            0.1622             0.6656           0.7119                0.2654
1            0.1238             0.1866           0.2416                0.1860
2            0.1444             0.4245           0.4504                0.2430
3            0.2098             0.8663           0.6869                0.2575
4            0.1374             0.2050           0.4000                0.1625

   worst symmetry  worst fractal dimension  label
0          0.4601                  0.11890    0.0
1          0.2750                  0.08902    0.0
2          0.3613                  0.08758    0.0
3          0.6638                  0.17300    0.0
4          0.2364                  0.07678    0.0

[5 rows x 31 columns]
```

[29]:
```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
wcss = []
X = breast_dataset.drop('label', axis=1)
kume_sayisi_listesi = range(1, 11)
for i in kume_sayisi_listesi :
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300,
                    n_init = 10, random_state = 0)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss, marker='o')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.title('Elbow Method for Optimal k')
plt.show()
```



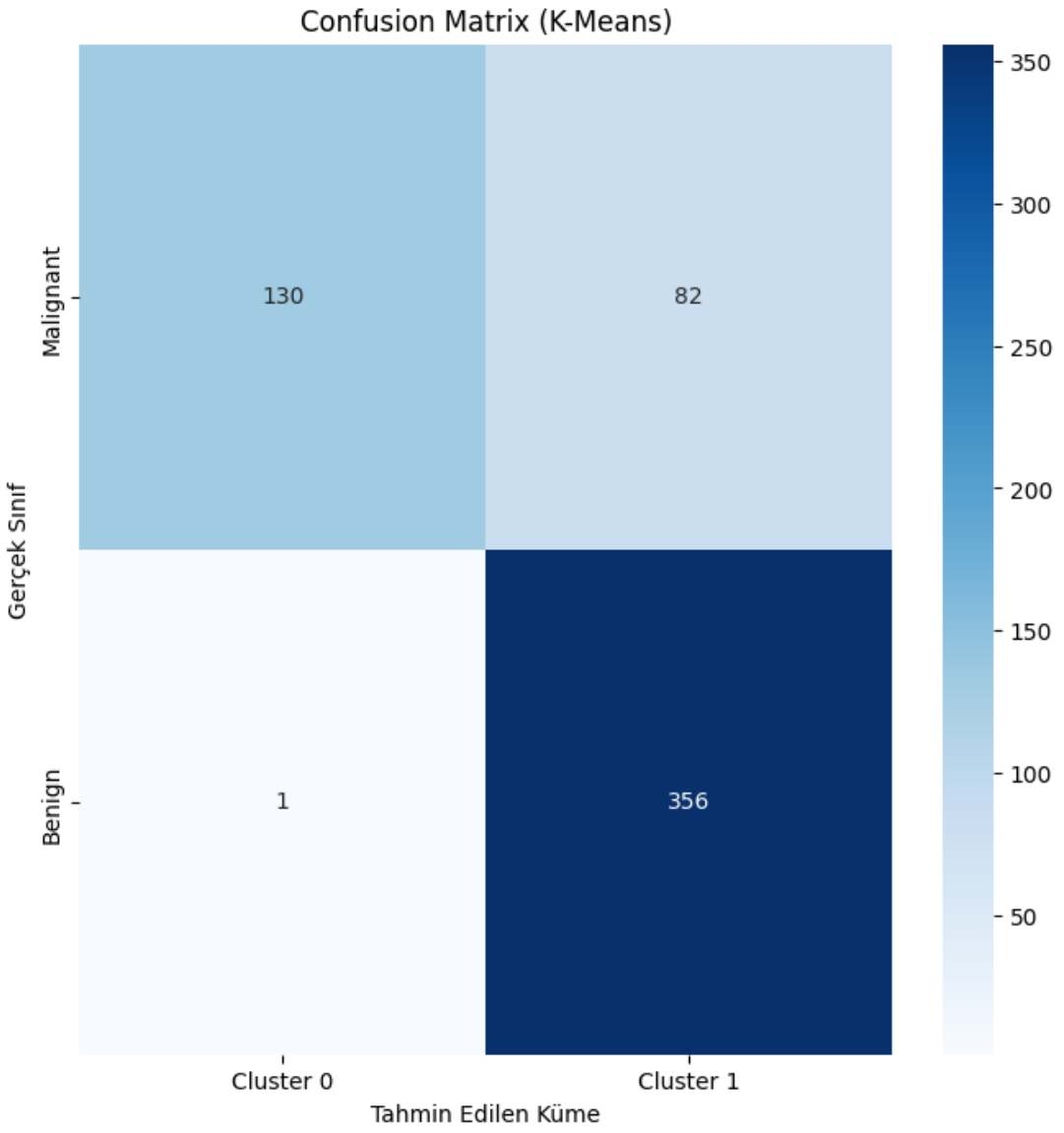
[30]:
```
kmeans = KMeans(n_clusters = 2, init = 'k-means++', max_iter = 300, n_init =␣
 ↪10, random_state = 0)
y_kmeans = kmeans.fit_predict(X)
cluster_labels = kmeans.labels_
plt.scatter(breast_dataset.iloc[:, 0], breast_dataset.iloc[:, 1],␣
 ↪c=cluster_labels, cmap='viridis', s=70)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
 ↪c='red', marker='X', s=100, label='Centroids')
plt.xlabel(breast.feature_names[0])
plt.ylabel(breast.feature_names[1])
plt.title('K-Means Clustering of Breast Cancer Dataset')
plt.legend()
plt.show()
```



[31]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

predicted_labels = 1 - cluster_labels

true_labels = breast_labels

misclassified_malignant = sum((true_labels == 0) & (predicted_labels == 1))
misclassified_benign = sum((true_labels == 1) & (predicted_labels == 0))

num_malignant = sum(true_labels == 0)
num_benign = sum(true_labels == 1)
error_rate = (misclassified_malignant + misclassified_benign) / len(true_labels)

misclassified_malignant_rate = misclassified_malignant / num_malignant
misclassified_benign_rate = misclassified_benign / num_benign

print(f"Hata Oranı: {error_rate}")
```

```
print(f"Kötü Huylu Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_malignant_rate}")
print(f"Iyi Huylu Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_benign_rate}")

if error_rate < 0.05:
    print("Model oldukça başarılı.")
elif 0.05 <= error_rate < 0.9:
    print("Model kabul edilebilir bir başarı gösteriyor.")
else:
    print("Modelin performansı düşük, iyileştirme yapılması gerekebilir.")

conf_matrix = confusion_matrix(true_labels, predicted_labels)

plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=['Cluster 0', 'Cluster 1'], yticklabels=['Malignant', 'Benign'])
plt.xlabel('Tahmin Edilen Küme')
plt.ylabel('Gerçek Sınıf')
plt.title('Confusion Matrix (K-Means)')
plt.show()


accuracy = accuracy_score(true_labels, predicted_labels)
print(f'Doğruluk (Accuracy): {accuracy:.4f}')

breast_dataset['kmeans_cluster'] = kmeans.labels_


cluster_counts = breast_dataset['kmeans_cluster'].value_counts().sort_index()

correctly_classified_samples = (breast_dataset['label'] ==␣
 ↪breast_dataset['kmeans_cluster']).sum()


for cluster, count in cluster_counts.items():
    print(f"Cluster {cluster}: {count} eleman")

total_samples = len(breast_dataset)
print(f"Toplam eleman Sayısı: {total_samples}")
```

```
Hata Oranı: 0.14586994727592267
Kötü Huylu Sınıfında Hatalı Sınıflandırma Oranı: 0.3867924528301887
Iyi Huylu Sınıfında Hatalı Sınıflandırma Oranı: 0.0028011204481792717
Model kabul edilebilir bir başarı gösteriyor.
```




```
Doğruluk (Accuracy): 0.8541
Cluster 0: 438 eleman
Cluster 1: 131 eleman
Toplam eleman Sayısı: 569
```

[32]:
```
breast_dataset['kmeans_cluster'] = kmeans.labels_


cluster_counts = breast_dataset['kmeans_cluster'].value_counts().sort_index()

for cluster, count in cluster_counts.items():
```

```
    print(f"Cluster {cluster}: {count} örnek")

total_samples = len(breast_dataset)
print(f"Toplam Örnek Sayısı: {total_samples}")
```

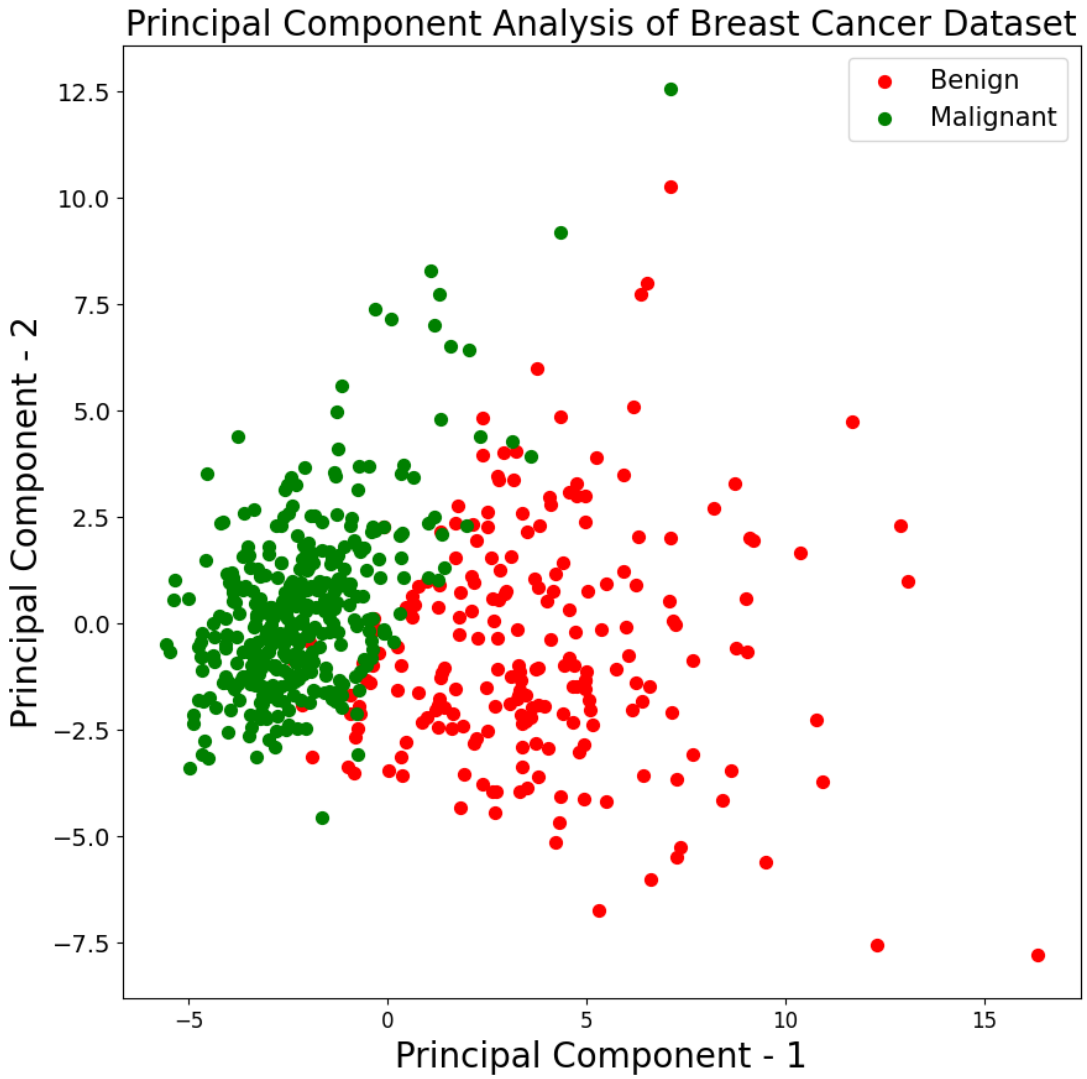
```
Cluster 0: 438 örnek
Cluster 1: 131 örnek
Toplam Örnek Sayısı: 569
```

[33]:
```
from sklearn.datasets import load_breast_cancer
breast = load_breast_cancer()
breast_data = breast.data
breast_data.shape
breast_labels = breast.target
breast_labels.shape
import numpy as np
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data,labels],axis=1)
final_breast_data.shape
labels = np.reshape(breast_labels,(569,1))
labels = np.reshape(breast_labels,(569,1))
final_breast_data = np.concatenate([breast_data,labels],axis=1)
final_breast_data.shape
import pandas as pd
breast_dataset = pd.DataFrame(final_breast_data)
features = breast.feature_names
features
features_labels = np.append(features,'label')
breast_dataset.columns = features_labels
breast_dataset.head()
from sklearn.cluster import KMeans
import pandas as pd
import matplotlib.pyplot as plt
breast_dataset['label'].replace(0, 'Benign',inplace=True)
breast_dataset['label'].replace(1, 'Malignant',inplace=True)
breast_dataset.tail()
from sklearn.preprocessing import StandardScaler
x = breast_dataset.loc[:, features].values
x = StandardScaler().fit_transform(x)
x.shape
np.mean(x),np.std(x)
feat_cols = ['feature'+str(i) for i in range(x.shape[1])]
normalised_breast = pd.DataFrame(x,columns=feat_cols)
normalised_breast.tail()
from sklearn.decomposition import PCA
pca_breast = PCA(n_components=2)
principalComponents_breast = pca_breast.fit_transform(x)
```

```
principal_breast_Df = pd.DataFrame(data = principalComponents_breast
             , columns = ['principal component 1', 'principal component 2'])
principal_breast_Df.tail()
print('Explained variation per principal component: {}'.format(pca_breast.
 ↪explained_variance_ratio_))

plt.figure()
plt.figure(figsize=(10,10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('Principal Component - 1',fontsize=20)
plt.ylabel('Principal Component - 2',fontsize=20)
plt.title("Principal Component Analysis of Breast Cancer Dataset",fontsize=20)
targets = ['Benign', 'Malignant']
colors = ['r', 'g']
for target, color in zip(targets,colors):
    indicesToKeep = breast_dataset['label'] == target
    plt.scatter(principal_breast_Df.loc[indicesToKeep, 'principal component 1']
               , principal_breast_Df.loc[indicesToKeep, 'principal component␣
 ↪2'], c = color, s = 50)

plt.legend(targets,prop={'size': 15})
```

```
Explained variation per principal component: [0.44272026 0.18971182]
```

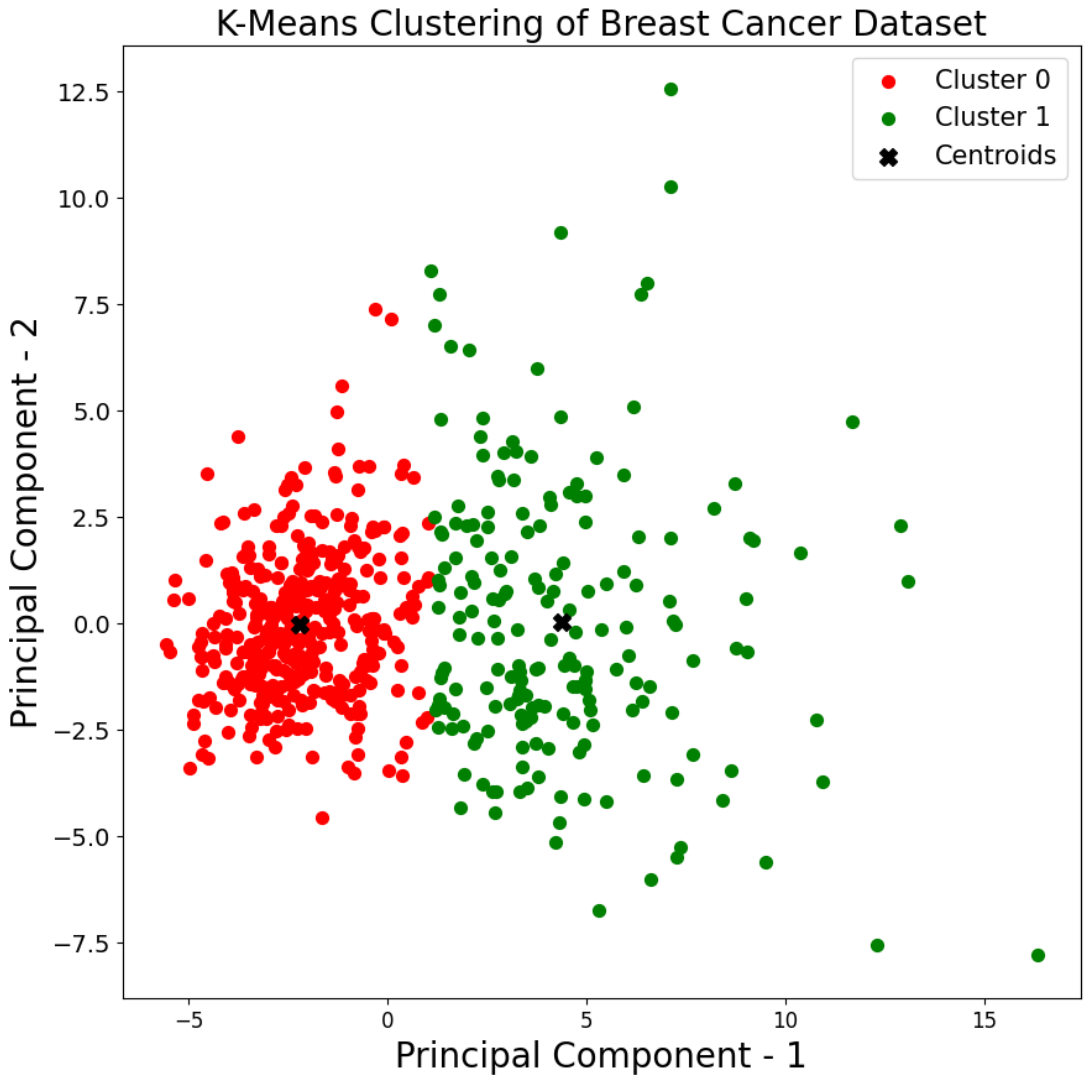
[33]:
```
<matplotlib.legend.Legend at 0x7da6ab3e8490>
```



[34]:
```
pca_data = principal_breast_Df[['principal component 1', 'principal component␣
↪2']]


n_clusters = 2
kmeans = KMeans(n_clusters=n_clusters, random_state=0)
kmeans.fit(pca_data)

cluster_labels = kmeans.labels_

principal_breast_Df['cluster'] = cluster_labels
```

```
plt.figure(figsize=(10, 10))
plt.xticks(fontsize=12)
plt.yticks(fontsize=14)
plt.xlabel('Principal Component - 1', fontsize=20)
plt.ylabel('Principal Component - 2', fontsize=20)
plt.title("K-Means Clustering of Breast Cancer Dataset", fontsize=20)


indices_cluster_0 = principal_breast_Df['cluster'] == 0
plt.scatter(principal_breast_Df.loc[indices_cluster_0, 'principal component 1'],
            principal_breast_Df.loc[indices_cluster_0, 'principal component␣
 ↪2'], c='r', s=50, label='Cluster 0')


indices_cluster_1 = principal_breast_Df['cluster'] == 1
plt.scatter(principal_breast_Df.loc[indices_cluster_1, 'principal component 1'],
            principal_breast_Df.loc[indices_cluster_1, 'principal component␣
 ↪2'], c='g', s=50, label='Cluster 1')


plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
 ↪c='black', marker='X', s=100, label='Centroids')

plt.legend(prop={'size': 15})
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
```



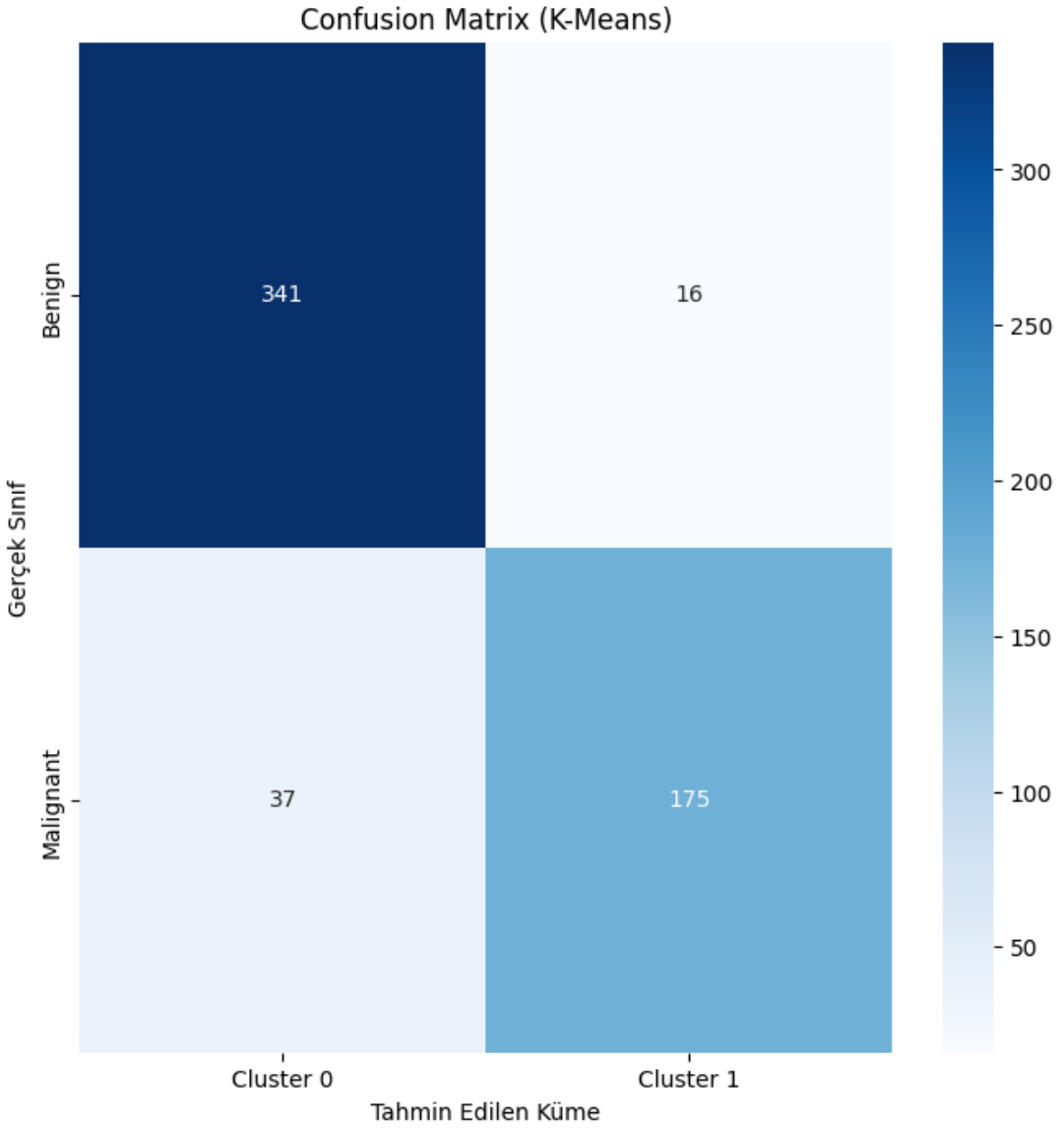
[36]:
```
breast_dataset['kmeans_cluster'] = kmeans.labels_

cluster_counts = breast_dataset['kmeans_cluster'].value_counts().sort_index()

for cluster, count in cluster_counts.items():
    print(f"Cluster {cluster}: {count} eleman")

total_samples = len(breast_dataset)
print(f"Toplam eleman Sayısı: {total_samples}")
```

```
Cluster 0: 378 eleman
Cluster 1: 191 eleman
Toplam eleman Sayısı: 569
```

[35]:
```
predicted_labels = principal_breast_Df['cluster']

# Gerçek etiketler
true_labels = 1- breast_labels

# Hatalı sınıflandırılan örnek sayısı
misclassified_malignant = sum((true_labels == 0) & (predicted_labels == 1))
misclassified_benign = sum((true_labels == 1) & (predicted_labels == 0))

# Hata oranları
error_rate = (misclassified_malignant + misclassified_benign) / len(true_labels)
misclassified_malignant_rate = misclassified_malignant / sum(true_labels == 0)
misclassified_benign_rate = misclassified_benign / sum(true_labels == 1)

print(f"Hata Oranı: {error_rate:.4f}")
print(f"Kötü Huylu Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_malignant_rate:.4f}")
print(f"Iyi Huylu Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_benign_rate:.4f}")

if error_rate < 0.05:
    print("Model oldukça başarılı.")
elif 0.05 <= error_rate < 0.1:
    print("Model kabul edilebilir bir başarı gösteriyor.")
else:
    print("Modelin performansı düşük, iyileştirme yapılması gerekebilir.")

# Karmaşıklık matrisini görselleştirme
conf_matrix = confusion_matrix(true_labels, predicted_labels)
plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=[f'Cluster {i}' for i in range(n_clusters)],
            yticklabels=['Benign', 'Malignant'])
plt.xlabel('Tahmin Edilen Küme')
plt.ylabel('Gerçek Sınıf')
plt.title('Confusion Matrix (K-Means)')
plt.show()

# Doğruluk (Accuracy) hesaplama
accuracy = accuracy_score(true_labels, predicted_labels)
print(f'Doğruluk (Accuracy): {accuracy:.4f}')

# Küme sayılarını görselleştirme
cluster_counts = principal_breast_Df['cluster'].value_counts().sort_index()
for cluster, count in cluster_counts.items():
    print(f"Cluster {cluster}: {count} eleman")
```

```
total_samples = len(principal_breast_Df)
print(f"Toplam eleman Sayısı: {total_samples}")
```

```
Hata Oranı: 0.0931
Kötü Huylu Sınıfında Hatalı Sınıflandırma Oranı: 0.0448
Iyi Huylu Sınıfında Hatalı Sınıflandırma Oranı: 0.1745
Model kabul edilebilir bir başarı gösteriyor.
```




```
Doğruluk (Accuracy): 0.9069
Cluster 0: 378 eleman
Cluster 1: 191 eleman
```

```
Toplam eleman Sayısı: 569
```

[37]:
```
from sklearn.decomposition import PCA
pca_breast = PCA(n_components=3)

principalComponents_breast = pca_breast.fit_transform(x)
principal_breast_Df = pd.DataFrame(data = principalComponents_breast
             , columns = ['principal component 1', 'principal component␣
 ↪2','principal component 3'])
principal_breast_Df.tail()
```

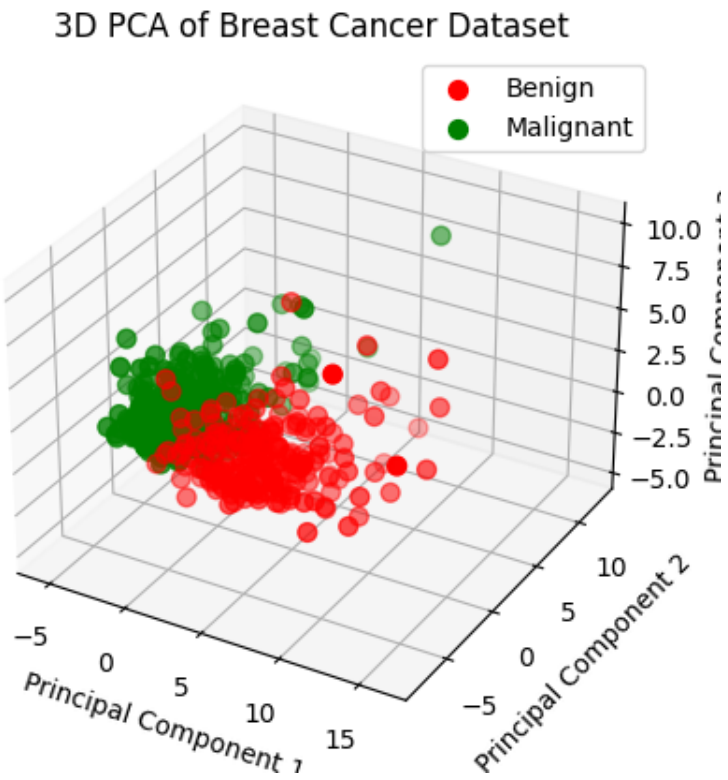
[37]:

	principal component 1	principal component 2	principal component 3
564	6.439315	-3.576817	2.459485
565	3.793382	-3.584048	2.088476
566	1.256179	-1.902297	0.562730
567	10.374794	1.672009	-1.877020
568	-5.475243	-0.670637	1.490447

[38]:
```
print('Explained variation per principal component: {}'.format(pca_breast.
 ↪explained_variance_ratio_))
```

```
Explained variation per principal component: [0.44272026 0.18971182 0.09393163]
```

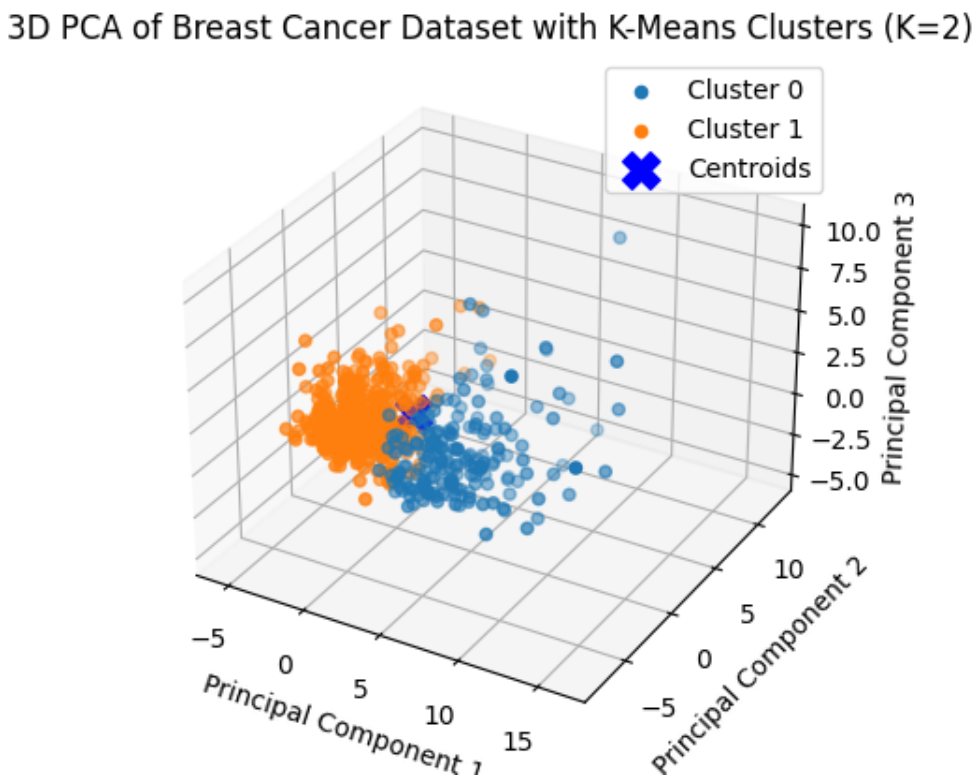
[39]:
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import pandas as pd
from sklearn.decomposition import PCA
fig = plt.figure()
ax = fig.add_subplot(projection='3d')
colors = {'Benign': 'r', 'Malignant': 'g'}
for target, color in colors.items():
    indicesToKeep = breast_dataset['label'] == target
    ax.scatter(principal_breast_Df.loc[indicesToKeep, 'principal component 1'],
               principal_breast_Df.loc[indicesToKeep, 'principal component 2'],
               principal_breast_Df.loc[indicesToKeep, 'principal component 3'],
               c=color, s=50, label=target)


ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
ax.set_title("3D PCA of Breast Cancer Dataset")

ax.legend()

plt.show()
```



[40]:
```
import plotly.express as px
import numpy as np

selected_columns = ["principal component 1", "principal component 2",␣
 ↪"principal component 3"]


target_labels = breast_dataset.label


principal_breast_Df["target_column"] = target_labels
colors = {'Benign': 'red', 'Malignant': 'green'}
fig = px.scatter_3d(principal_breast_Df, x=selected_columns[0],␣
 ↪y=selected_columns[1], z=selected_columns[2],
                    color="target_column", color_discrete_map=colors)

fig.update_traces(marker=dict(size=3))

fig.show()
```

[41]:
```
optimal_k = 2


kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(x)


breast_dataset['kmeans_cluster'] = kmeans.labels_

plt.show()

fig = plt.figure()
ax = fig.add_subplot(projection='3d')

for cluster in range(optimal_k):
    cluster_indices = breast_dataset['kmeans_cluster'] == cluster
    ax.scatter(principal_breast_Df.loc[cluster_indices, 'principal component␣
 ↪1'],
               principal_breast_Df.loc[cluster_indices, 'principal component␣
 ↪2'],
               principal_breast_Df.loc[cluster_indices, 'principal component␣
 ↪3'],
               label=f'Cluster {cluster}')


ax.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], kmeans.
 ↪cluster_centers_[:, 2],
           c='blue', marker='X', s=200, label='Centroids')

ax.set_xlabel('Principal Component 1')
ax.set_ylabel('Principal Component 2')
ax.set_zlabel('Principal Component 3')
ax.set_title(f'3D PCA of Breast Cancer Dataset with K-Means Clusters␣
 ↪(K={optimal_k})')
ax.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
```



[42]:
```
import plotly.express as px
import numpy as np


kmeans = KMeans(n_clusters=optimal_k, random_state=42)
kmeans.fit(x)
cluster_centers = kmeans.cluster_centers_


breast_dataset['kmeans_cluster'] = kmeans.labels_

cluster_centers = kmeans.cluster_centers_

fig = px.scatter_3d(principal_breast_Df, x='principal component 1',␣
↪y='principal component 2', z='principal component 3',
                    color=breast_dataset['kmeans_cluster'].astype(str),␣
↪color_discrete_map=colors,
                    title='3D PCA of Breast Cancer Dataset with K-Means␣
↪Clusters (K=2)')
import plotly.graph_objects as go
```

```
fig.add_trace(go.Scatter3d(
    x=cluster_centers[:, 0],
    y=cluster_centers[:, 1],
    z=cluster_centers[:, 2],
    mode='markers',
    marker=dict(color='blue', symbol='x', size=15),
    name='Cluster Centers'
))



fig.update_traces(marker=dict(size=3))
fig.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the
value of `n_init` explicitly to suppress the warning
```

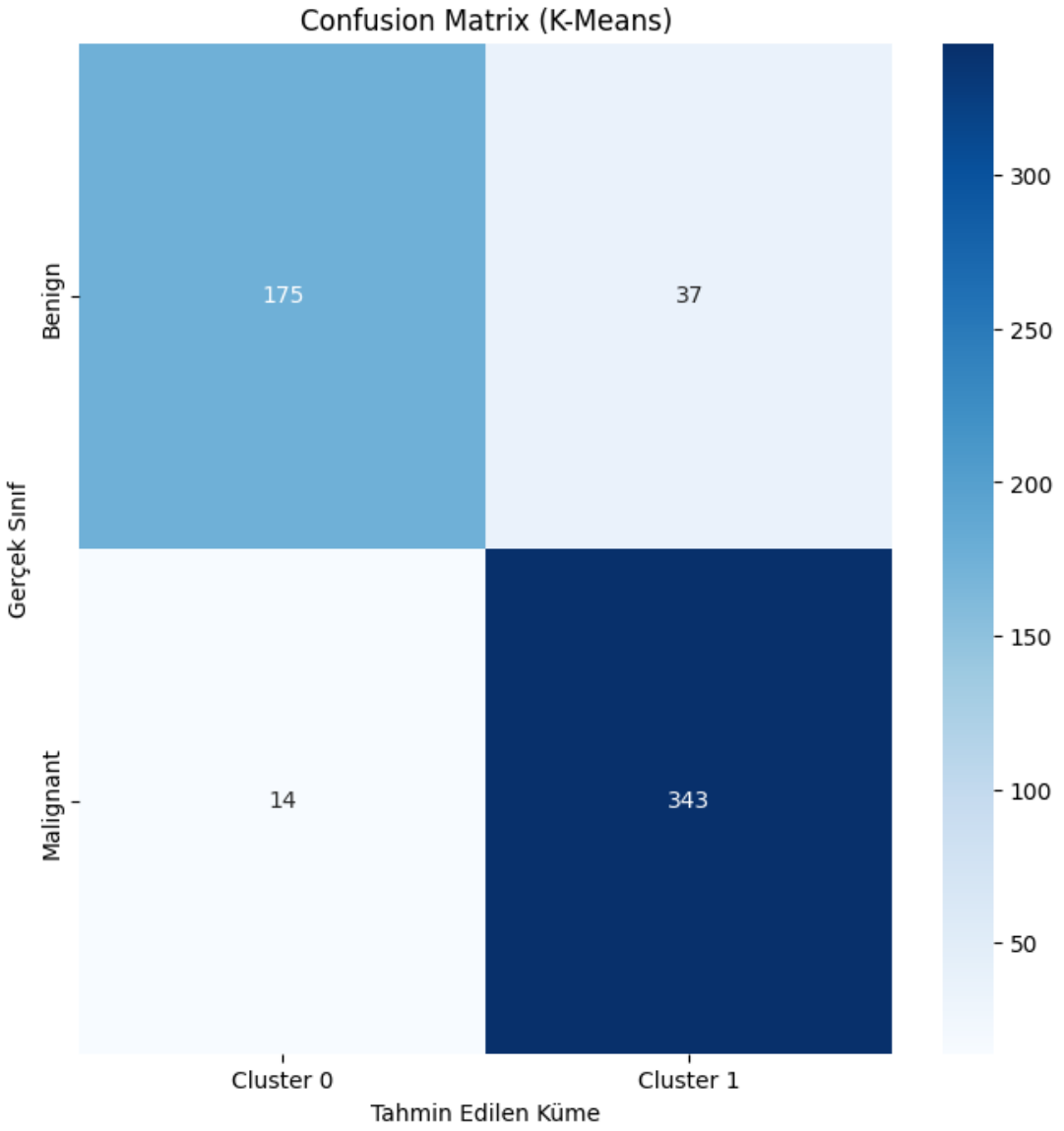
[43]:
```
breast_dataset['kmeans_cluster'] = kmeans.labels_

cluster_counts = breast_dataset['kmeans_cluster'].value_counts().sort_index()

for cluster, count in cluster_counts.items():
    print(f"Cluster {cluster}: {count} eleman")

total_samples = len(breast_dataset)
print(f"Toplam eleman Sayısı: {total_samples}")
```

```
Cluster 0: 189 eleman
Cluster 1: 380 eleman
Toplam eleman Sayısı: 569
```

[44]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt


predicted_labels = breast_dataset['kmeans_cluster']


true_labels = breast_labels


misclassified_malignant = sum((true_labels == 0) & (predicted_labels == 1))
```

```
misclassified_benign = sum((true_labels == 1) & (predicted_labels == 0))


error_rate = (misclassified_malignant + misclassified_benign) / len(true_labels)


misclassified_malignant_rate = misclassified_malignant / num_malignant
misclassified_benign_rate = misclassified_benign / num_benign


print(f"Hata Oranı: {error_rate:.4f}")
print(f"Kötü Huylu Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_malignant_rate:.4f}")
print(f"Iyi Huylu Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_benign_rate:.4f}")


if error_rate < 0.05:
    print("Model oldukça başarılı.")
elif 0.05 <= error_rate < 0.1:
    print("Model kabul edilebilir bir başarı gösteriyor.")
else:
    print("Modelin performansı düşük, iyileştirme yapılması gerekebilir.")


conf_matrix = confusion_matrix(true_labels, predicted_labels)

plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
            xticklabels=[f'Cluster {i}' for i in range(optimal_k)],
            yticklabels=['Benign', 'Malignant'])
plt.xlabel('Tahmin Edilen Küme')
plt.ylabel('Gerçek Sınıf')
plt.title('Confusion Matrix (K-Means)')
plt.show()


accuracy = accuracy_score(true_labels, predicted_labels)
print(f'Doğruluk (Accuracy): {accuracy:.4f}')


cluster_counts = breast_dataset['kmeans_cluster'].value_counts().sort_index()
for cluster, count in cluster_counts.items():
    print(f"Cluster {cluster}: {count} eleman")

total_samples = len(breast_dataset)
print(f"Toplam eleman Sayısı: {total_samples}")
```

```
Hata Oranı: 0.0896
Kötü Huylu Sınıfında Hatalı Sınıflandırma Oranı: 0.1745
Iyi Huylu Sınıfında Hatalı Sınıflandırma Oranı: 0.0392
Model kabul edilebilir bir başarı gösteriyor.
```




```
Doğruluk (Accuracy): 0.9104
Cluster 0: 189 eleman
Cluster 1: 380 eleman
Toplam eleman Sayısı: 569
```

A.2 Ek 1: Spotify Verisi Kodları

spotify

January 25, 2024

[ ]:
```
import pandas as pd

df = pd.read_csv("/content/drive/MyDrive/train.csv")

df.head()
```

[ ]:
```
   Unnamed: 0                track_id                 artists  \
0           0  5SuOikwiRyPMVoIQDJUgSV             Gen Hoshino
1           1  4qPNDBW1i3p13qLCt0Ki3A            Ben Woodward
2           2  1iJBSr7s7jYXzM8EGcbK5b  Ingrid Michaelson;ZAYN
3           3  6lfxq3CG4xtTiEg7opyCyx            Kina Grannis
4           4  5vjLSffimiIP26QG5WcN2K        Chord Overstreet

                                          album_name  \
0                                             Comedy
1                                   Ghost (Acoustic)
2                                     To Begin Again
3  Crazy Rich Asians (Original Motion Picture Sou…
4                                          Hold On

                  track_name  popularity  duration_ms  explicit  \
0                     Comedy          73       230666     False
1           Ghost - Acoustic          55       149610     False
2             To Begin Again          57       210826     False
3  Can't Help Falling In Love          71       201933     False
4                    Hold On          82       198853     False

   danceability  energy  …  loudness  mode  speechiness  acousticness  \
0         0.676  0.4610  …    -6.746     0       0.1430        0.0322
1         0.420  0.1660  …   -17.235     1       0.0763        0.9240
2         0.438  0.3590  …    -9.734     1       0.0557        0.2100
3         0.266  0.0596  …   -18.515     1       0.0363        0.9050
4         0.618  0.4430  …    -9.681     1       0.0526        0.4690

   instrumentalness  liveness  valence    tempo  time_signature  track_genre
0          0.000001    0.3580    0.715   87.917               4     acoustic
1          0.000006    0.1010    0.267   77.489               4     acoustic
```

```
2          0.000000    0.1170    0.120   76.332               4     acoustic
3          0.000071    0.1320    0.143  181.740               3     acoustic
4          0.000000    0.0829    0.167  119.949               4     acoustic

[5 rows x 21 columns]
```

[ ]:
```
unique_values = df['track_genre'].unique()
print(unique_values)
```

```
['acoustic' 'afrobeat' 'alt-rock' 'alternative' 'ambient' 'anime'
 'black-metal' 'bluegrass' 'blues' 'brazil' 'breakbeat' 'british'
 'cantopop' 'chicago-house' 'children' 'chill' 'classical' 'club' 'comedy'
 'country' 'dance' 'dancehall' 'death-metal' 'deep-house' 'detroit-techno'
 'disco' 'disney' 'drum-and-bass' 'dub' 'dubstep' 'edm' 'electro'
 'electronic' 'emo' 'folk' 'forro' 'french' 'funk' 'garage' 'german'
 'gospel' 'goth' 'grindcore' 'groove' 'grunge' 'guitar' 'happy'
 'hard-rock' 'hardcore' 'hardstyle' 'heavy-metal' 'hip-hop' 'honky-tonk'
 'house' 'idm' 'indian' 'indie-pop' 'indie' 'industrial' 'iranian'
 'j-dance' 'j-idol' 'j-pop' 'j-rock' 'jazz' 'k-pop' 'kids' 'latin'
 'latino' 'malay' 'mandopop' 'metal' 'metalcore' 'minimal-techno' 'mpb'
 'new-age' 'opera' 'pagode' 'party' 'piano' 'pop-film' 'pop' 'power-pop'
 'progressive-house' 'psych-rock' 'punk-rock' 'punk' 'r-n-b' 'reggae'
 'reggaeton' 'rock-n-roll' 'rock' 'rockabilly' 'romance' 'sad' 'salsa'
 'samba' 'sertanejo' 'show-tunes' 'singer-songwriter' 'ska' 'sleep'
 'songwriter' 'soul' 'spanish' 'study' 'swedish' 'synth-pop' 'tango'
 'techno' 'trance' 'trip-hop' 'turkish' 'world-music']
```

[ ]:
```
df = df.loc[(df.track_genre == 'black-metal') | (df.track_genre == 'classical')␣
↪]
```

[ ]:
```
df
```

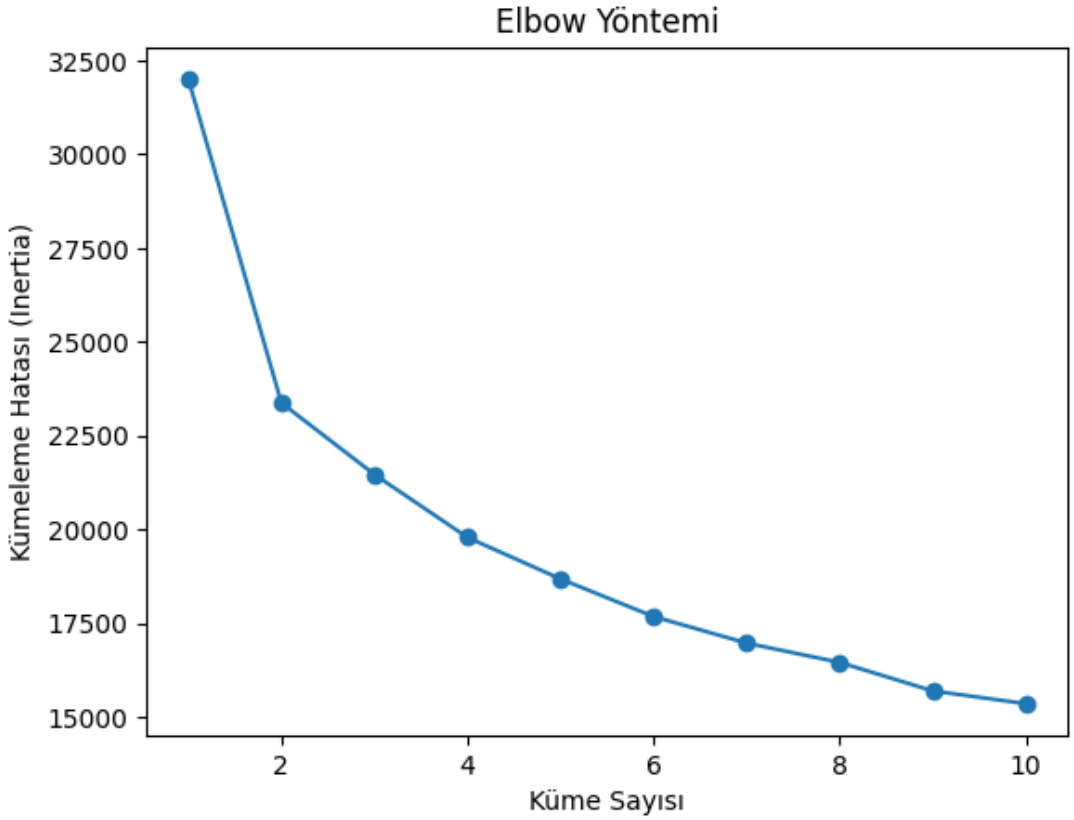
[ ]:
```
       Unnamed: 0                track_id  \
6000         6000  0mJUxFpEI1eAOIIfnNoZ4G
6001         6001  7v9HNMlAe2UBaEhvaCk5wX
6002         6002  3wBHF6evf55iEzyMtReJSH
6003         6003  5FBToB2y0ie4fq3WjfsFFE
6004         6004  6WuqJLVZcyJklg7lIozAO8
...           ...                     ...
16995       16995  5nRAp6NQOoOFdk3CJY1WrL
16996       16996  4TujvPqDwwcewGVv1TgLLI
16997       16997  6OtqRo9wsVAtYhpDjWB7Di
16998       16998  23LveHXJT1mbjaxaqeWB5Y
16999       16999  5NmcuvXdnuAqcfx2zJe5Bp

                                                artists  \
6000                                    Cradle Of Filth
```

```
6001                                    Make Them Suffer
6002                                            Behemoth
6003                                             Sadness
6004                                      Cradle Of Filth
…                                                     …
16995  Wolfgang Amadeus Mozart;Concerto Köln;Werner E…
16996                 Wolfgang Amadeus Mozart;Erik Smith
16997             Wolfgang Amadeus Mozart;Danielle Laval
16998       Wolfgang Amadeus Mozart;Francesco Piemontesi
16999  Wolfgang Amadeus Mozart;Wiener Mozart Ensemble…

                        album_name  \
6000       Lovecraft & Witch Hearts
6001                     Doomswitch
6002    I Loved You at Your Darkest
6003              I Want to Be There
6004   Nymphetamine Special Edition
…                              …
16995     Mozart - All Day Classics
16996     Mozart - All Day Classics
16997     Mozart - All Day Classics
16998     Mozart - All Day Classics
16999     Mozart - All Day Classics

                                              track_name  popularity  \
6000                                Hallowed Be Thy Name          51
6001                                          Doomswitch          58
6002                                           Bartzabel          47
6003                                I Want to Be with You          41
6004                                    Nymphetamine Fix          54
…                                                     …           …
16995              Six German Dances, K.571: No. 4 in G           6
16996                          Prelude in G Major, K.15g          15
16997  12 Variations on ‘Je suis Lindor’ from ‘Le Bar…          15
16998          8 Menuette, K.315a: Menuetto 5 in F Major           7
16999     Six Contredanses K. 462: No. 2 in E Flat Major           6

       duration_ms  explicit  danceability  energy  …  loudness  mode  \
6000        430733     False         0.428  0.9720  …    -1.998     0
6001        275205      True         0.250  0.9520  …    -4.059     0
6002        301285     False         0.468  0.9130  …    -5.670     1
6003        355474     False         0.144  0.4660  …    -6.715     1
6004        302360     False         0.462  0.9050  …    -3.825     0
…              …         …             …       …  …         …   …
16995        96133     False         0.570  0.1770  …   -18.339     1
16996        63360     False         0.193  0.0844  …   -22.463     1
16997        61800     False         0.519  0.1990  …   -22.841     1
```

```
16998        97906     False         0.422  0.0238  …   -27.829     1
16999        98160     False         0.642  0.0801  …   -20.228     1

       speechiness  acousticness  instrumentalness  liveness  valence  \
6000        0.0666      0.000072            0.0736    0.0787   0.2250
6001        0.2090      0.000024            0.0135    0.0892   0.1260
6002        0.0720      0.026500            0.0769    0.0787   0.1870
6003        0.0522      0.686000            0.4230    0.0764   0.0737
6004        0.0438      0.000660            0.0402    0.0839   0.2530
…              …             …                 …         …        …
16995       0.0488      0.963000            0.9170    0.1560   0.5150
16996       0.0494      0.685000            0.0000    0.1130   0.2370
16997       0.0334      0.995000            0.9390    0.1110   0.5810
16998       0.0413      0.994000            0.9090    0.1110   0.7170
16999       0.0501      0.909000            0.5910    0.2410   0.6020

         tempo  time_signature  track_genre
6000   118.306               4  black-metal
6001   143.022               3  black-metal
6002   109.003               4  black-metal
6003    87.290               4  black-metal
6004   122.925               3  black-metal
…          …               …            …
16995   96.075               1    classical
16996  182.218               3    classical
16997   92.138               4    classical
16998  155.189               4    classical
16999  121.991               4    classical

[2000 rows x 21 columns]
```

[ ]:
```python
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt


k_values = range(1, 11)
inertia_values = []

for k in k_values:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    inertia_values.append(kmeans.inertia_)

plt.plot(k_values, inertia_values, marker='o')
plt.title('Elbow Yöntemi')
plt.xlabel('Küme Sayısı')
plt.ylabel('Kümeleme Hatası (Inertia)')
```

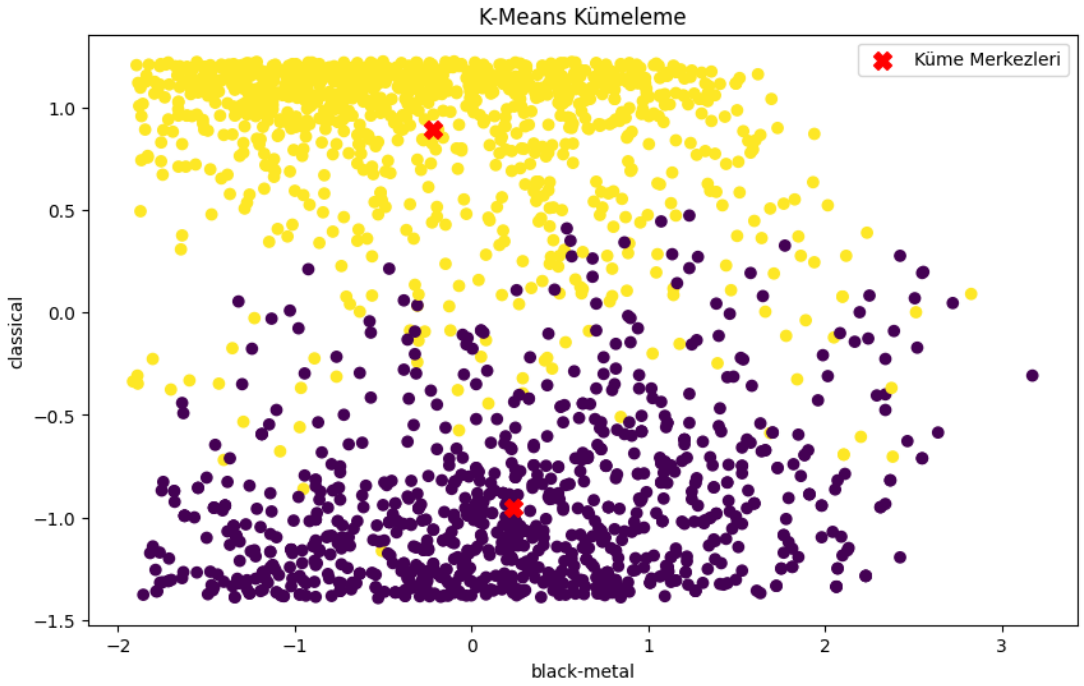
```
plt.show()
```



[ ]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt


df_subset = df.loc[(df['track_genre'] == 'black-metal') | (df['track_genre'] ==␣
 ↪'classical')]


numerical_features = ["danceability", "energy", "acousticness", "popularity",␣
 ↪"duration_ms",
                      "explicit", "key", "loudness", "mode", "speechiness",
                      "acousticness", "instrumentalness", "liveness",␣
 ↪"valence",
                      "tempo", "time_signature"]
X = df_subset[numerical_features]
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X_scaled)

df_subset['cluster'] = kmeans.labels_

plt.figure(figsize=(10, 6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=kmeans.labels_, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
 ↪s=100, c='red', marker='X', label='Küme Merkezleri')
plt.xlabel('black-metal')
plt.ylabel('classical')
plt.title('K-Means Kümeleme')
plt.legend()
plt.show()
```

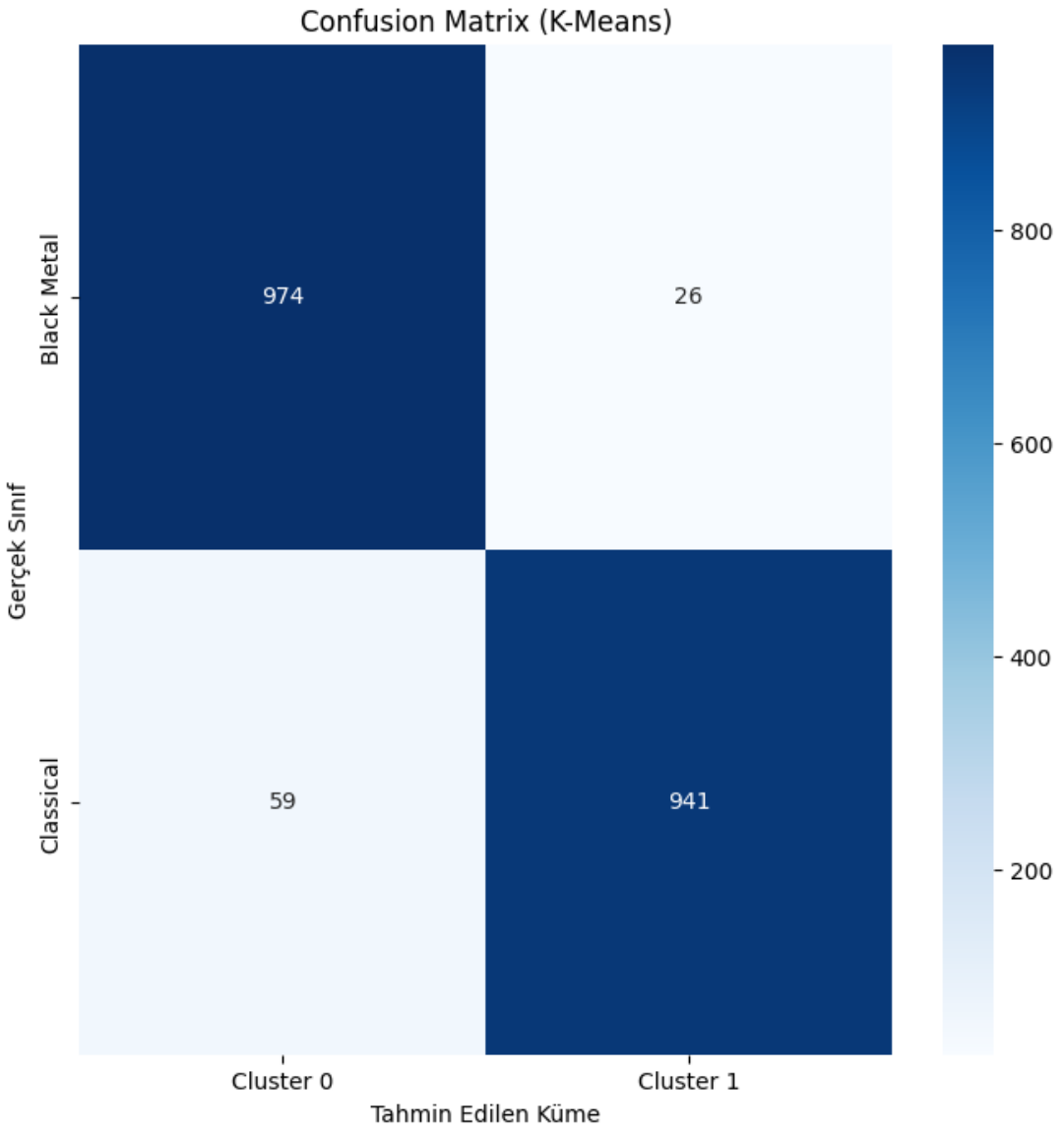
```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```



[ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt


true_labels = df_subset['track_genre'].map({'black-metal': 0, 'classical': 1})

kmeans_labels = kmeans.labels_


predicted_labels = 1 - kmeans_labels


misclassified_black_metal = sum((true_labels == 0) & (predicted_labels == 1))
misclassified_classical = sum((true_labels == 1) & (predicted_labels == 0))

error_rate = (misclassified_black_metal + misclassified_classical) /␣
 ↪len(true_labels)
misclassified_black_metal_rate = misclassified_black_metal /␣
 ↪len(df_subset[df_subset['track_genre'] == 'black-metal'])
misclassified_classical_rate = misclassified_classical /␣
 ↪len(df_subset[df_subset['track_genre'] == 'classical'])

print(f"Hata Oranı: {error_rate}")
print(f"Black Metal Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_black_metal_rate}")
print(f"Classical Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_classical_rate}")


if error_rate < 0.05:
    print("Model oldukça başarılı.")
elif 0.05 <= error_rate < 0.9:
    print("Model kabul edilebilir bir başarı gösteriyor.")
else:
    print("Modelin performansı düşük, iyileştirme yapılması gerekebilir.")


conf_matrix = confusion_matrix(true_labels, predicted_labels)

plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=['Cluster 0', 'Cluster 1'], yticklabels=['Black Metal',␣
 ↪'Classical'])
plt.xlabel('Tahmin Edilen Küme')
plt.ylabel('Gerçek Sınıf')
plt.title('Confusion Matrix (K-Means)')
```

```
plt.show()

accuracy = accuracy_score(true_labels, predicted_labels)
print(f'Doğruluk (Accuracy): {accuracy:.4f}')

eleman_sayilari = df_subset['cluster'].value_counts()

for cluster_label, eleman_sayisi in eleman_sayilari.items():
    print(f"Küme {cluster_label}: {eleman_sayisi} eleman")
    cluster_counts = df_subset['cluster'].value_counts()
```

```
Hata Oranı: 0.0425
Black Metal Sınıfında Hatalı Sınıflandırma Oranı: 0.026
Classical Sınıfında Hatalı Sınıflandırma Oranı: 0.059
Model oldukça başarılı.
```



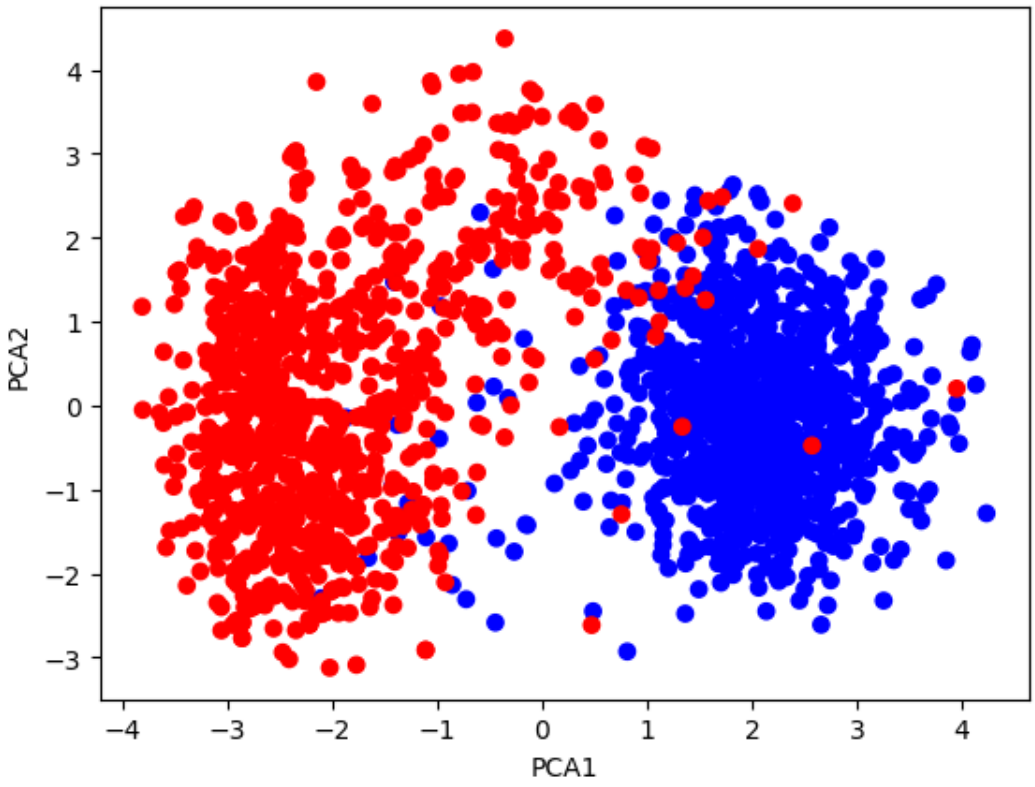

```
Doğruluk (Accuracy): 0.9575
Küme 1: 1033 eleman
Küme 0: 967 eleman
```

[ ]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X_scaled)
pca.transform(X_scaled)
scores_pca = pca.transform(X_scaled)
```

[ ]:
```
df_pca = pd.DataFrame(data=scores_pca, columns=['PCA1', 'PCA2'])
df_pca.head()
```

[ ]:
```
       PCA1      PCA2
0  2.565895  0.967508
1  4.082024  0.637770
2  1.951746  1.250685
3  0.119568 -0.927523
4  2.144233  1.238227
```

[ ]:
```
import numpy as np

pca = PCA(n_components=2)
pca.fit(X_scaled)


scores_pca = pca.transform(X_scaled)
df_pca = pd.DataFrame(data=scores_pca, columns=['PCA1', 'PCA2'])

plt.scatter(df_pca['PCA1'], df_pca['PCA2'], c=np.where(df_subset['track_genre']␣
 ↪== 'classical', 'red', 'blue'))
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.show()
```



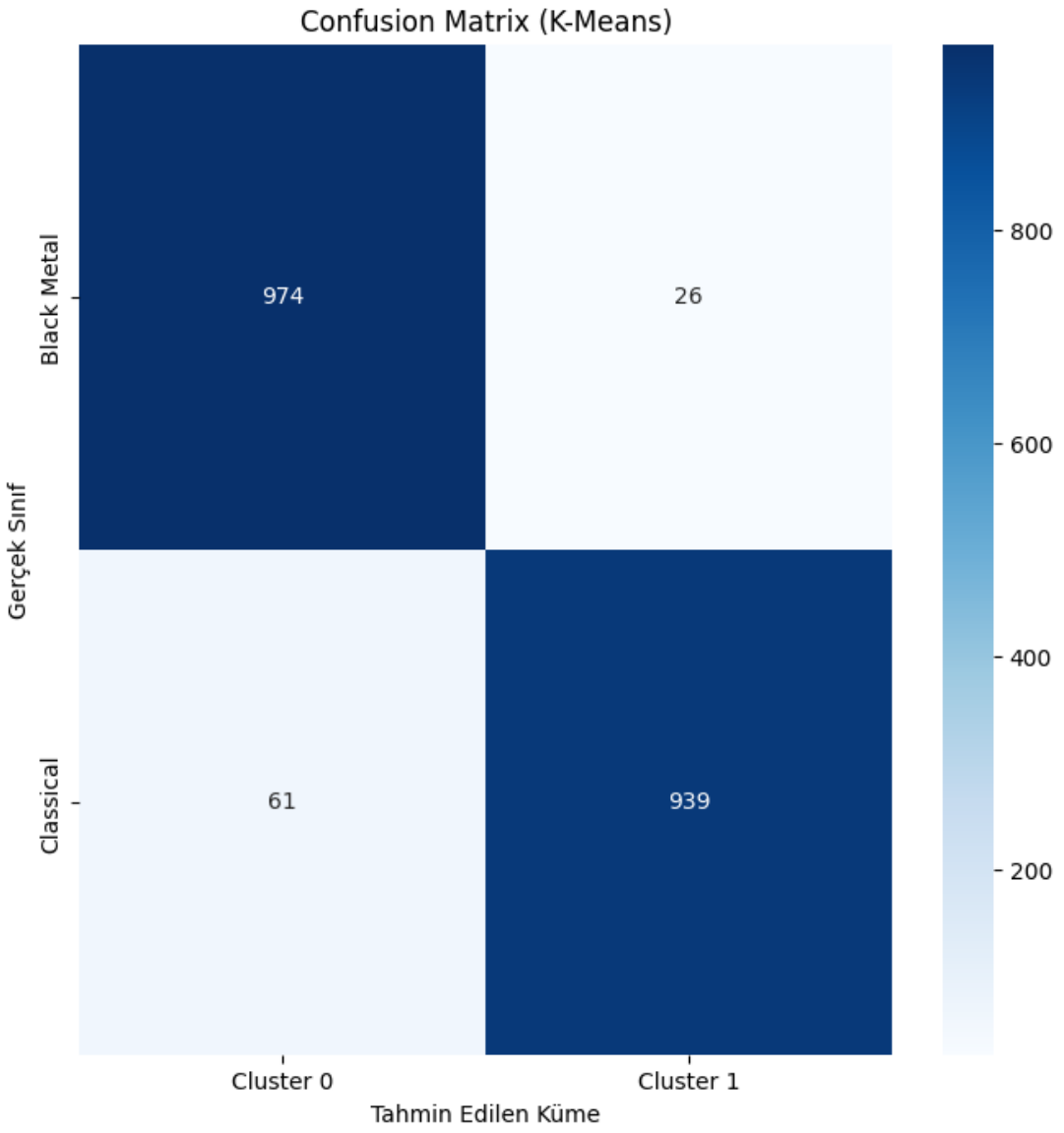
[ ]:
```
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(scores_pca)


df_pca['cluster'] = kmeans.labels_

plt.figure(figsize=(10, 8))
plt.scatter(df_pca['PCA1'], df_pca['PCA2'], c=df_pca['cluster'], cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1],␣
 ↪s=100, c='red', marker='X', label='Küme Merkezleri')
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title('PCA ve K-Means Kümeleme')
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870:
FutureWarning: The default value of `n_init` will change from 10 to 'auto' in
1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

[ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt


true_labels = df_subset['track_genre'].map({'black-metal': 0, 'classical': 1})

kmeans_labels = kmeans.labels_


predicted_labels = 1 - kmeans_labels

misclassified_black_metal = sum((true_labels == 0) & (predicted_labels == 1))
misclassified_classical = sum((true_labels == 1) & (predicted_labels == 0))

error_rate = (misclassified_black_metal + misclassified_classical) /␣
 ↪len(true_labels)
```

```
misclassified_black_metal_rate = misclassified_black_metal /␣
 ↪len(df_subset[df_subset['track_genre'] == 'black-metal'])
misclassified_classical_rate = misclassified_classical /␣
 ↪len(df_subset[df_subset['track_genre'] == 'classical'])

print(f"Hata Oranı: {error_rate}")
print(f"Black Metal Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_black_metal_rate}")
print(f"Classical Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_classical_rate}")


if error_rate < 0.05:
    print("Model oldukça başarılı.")
elif 0.05 <= error_rate < 0.9:
    print("Model kabul edilebilir bir başarı gösteriyor.")
else:
    print("Modelin performansı düşük, iyileştirme yapılması gerekebilir.")


conf_matrix = confusion_matrix(true_labels, predicted_labels)

plt.figure(figsize=(8, 8))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=['Cluster 0', 'Cluster 1'], yticklabels=['Black Metal',␣
 ↪'Classical'])
plt.xlabel('Tahmin Edilen Küme')
plt.ylabel('Gerçek Sınıf')
plt.title('Confusion Matrix (K-Means)')
plt.show()


accuracy = accuracy_score(true_labels, predicted_labels)
print(f'Doğruluk (Accuracy): {accuracy:.4f}')
cluster_counts = df_pca['cluster'].value_counts()

for cluster, count in cluster_counts.items():
    print(f"Cluster {cluster}: {count} eleman")
```

```
Hata Oranı: 0.0435
Black Metal Sınıfında Hatalı Sınıflandırma Oranı: 0.026
Classical Sınıfında Hatalı Sınıflandırma Oranı: 0.061
Model oldukça başarılı.
```




```
Doğruluk (Accuracy): 0.9565
Cluster 1: 1035 eleman
Cluster 0: 965 eleman
```

[ ]:
```
explained_variance = pca.explained_variance_ratio_
print("Toplam varyansın açıklanma oranları:", explained_variance)
```

```
Toplam varyansın açıklanma oranları: [0.30493646 0.10506512]
```

[ ]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=3)
```

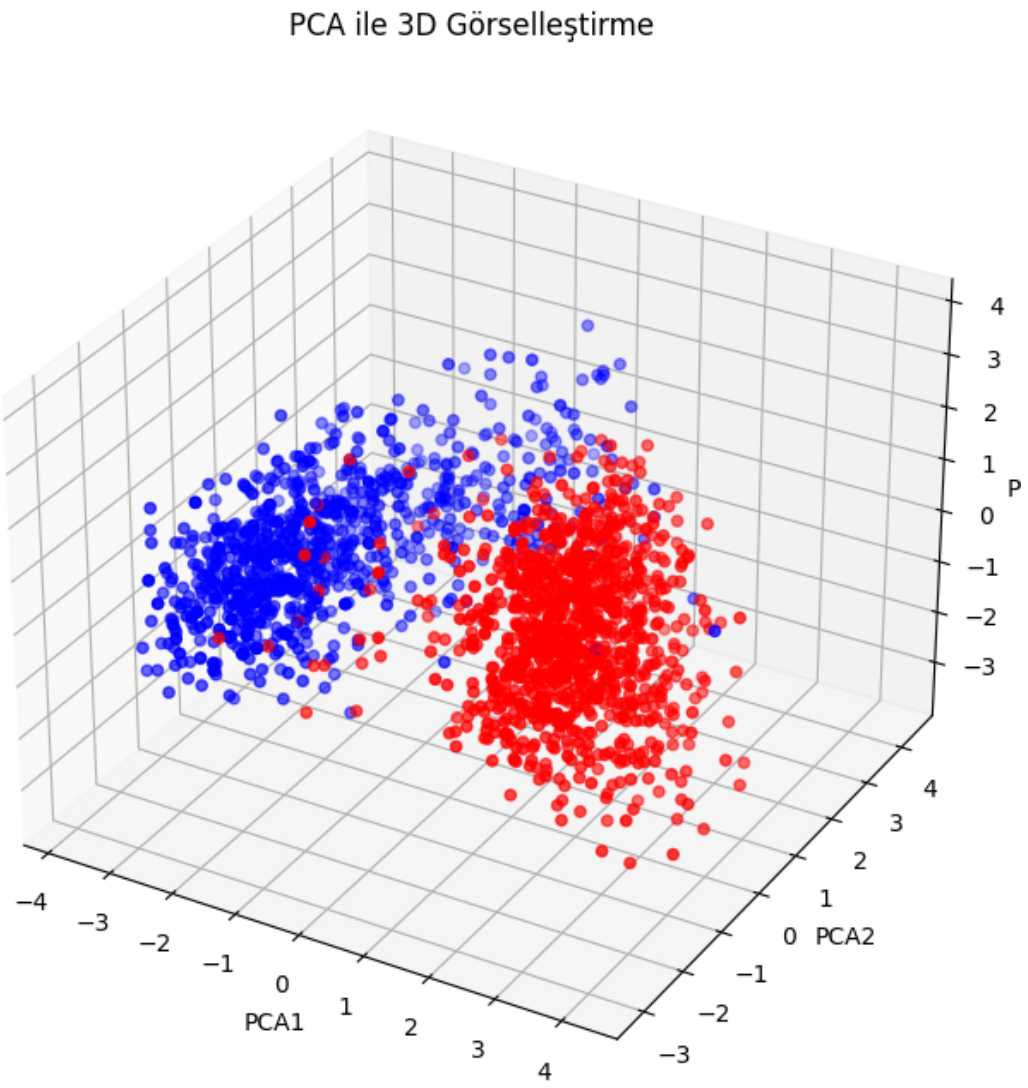
```
pca.fit(X_scaled)
pca.transform(X_scaled)
scores_pca = pca.transform(X_scaled)
```

[ ]:
```
df_pca = pd.DataFrame(data=scores_pca, columns=['PCA1', 'PCA2','PCA3'])
df_pca.head()
```

[ ]:

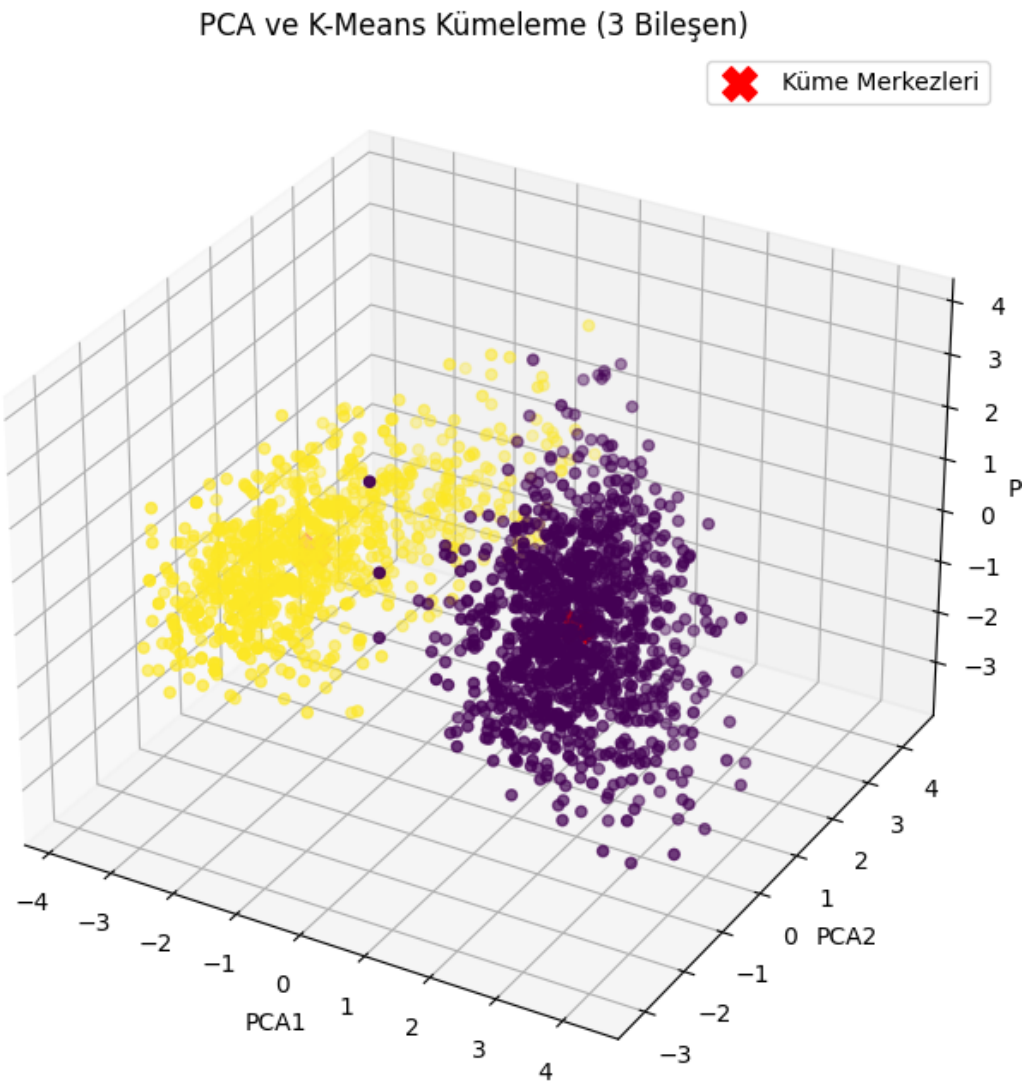
	PCA1	PCA2	PCA3
0	2.565896	0.967537	1.642433
1	4.082026	0.637953	0.500871
2	1.951751	1.251110	0.569481
3	0.119563	-0.927974	0.452209
4	2.144235	1.238423	1.630150

[ ]:
```
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)


fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')
ax.scatter(X_pca[:, 0], X_pca[:, 1], X_pca[:, 2], c=df_subset['track_genre'].
 ↪map({'black-metal': 'red', 'classical': 'blue'}), marker='o')
ax.set_xlabel('PCA1')
ax.set_ylabel('PCA2')
ax.set_zlabel('PCA3')
ax.set_title('PCA ile 3D Görselleştirme')

plt.show()
```



[ ]:
```
import plotly.express as px


fig = px.scatter_3d(df_subset, x=X_pca[:, 0], y=X_pca[:, 1], z=X_pca[:, 2],
                    color='track_genre', color_discrete_map={'black-metal':␣
 ↪'red', 'classical': 'blue'},
                    title='3D PCA Görseli',
                    labels={'x': 'PCA1', 'y': 'PCA2', 'z': 'PCA3'})
fig.update_traces(marker=dict(size=3))

fig.show()
```

[ ]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)


kmeans_pca = KMeans(n_clusters=2, random_state=42, n_init=10)
kmeans_pca.fit(X_pca)


df_subset = df_subset.copy()
df_subset['cluster_pca'] = kmeans_pca.labels_


fig = plt.figure(figsize=(12, 8))
ax = fig.add_subplot(111, projection='3d')

ax.scatter(X_pca[:, 0], X_pca[:, 1], X_pca[:, 2], c=kmeans_pca.labels_,␣
 ↪cmap='viridis', marker='o')
ax.scatter(kmeans_pca.cluster_centers_[:, 0], kmeans_pca.cluster_centers_[:,␣
 ↪1], kmeans_pca.cluster_centers_[:, 2],
           s=200, c='red', marker='X', label='Küme Merkezleri')

ax.set_xlabel('PCA1')
ax.set_ylabel('PCA2')
ax.set_zlabel('PCA3')
ax.set_title('PCA ve K-Means Kümeleme (3 Bileşen)')

plt.legend()
plt.show()
```



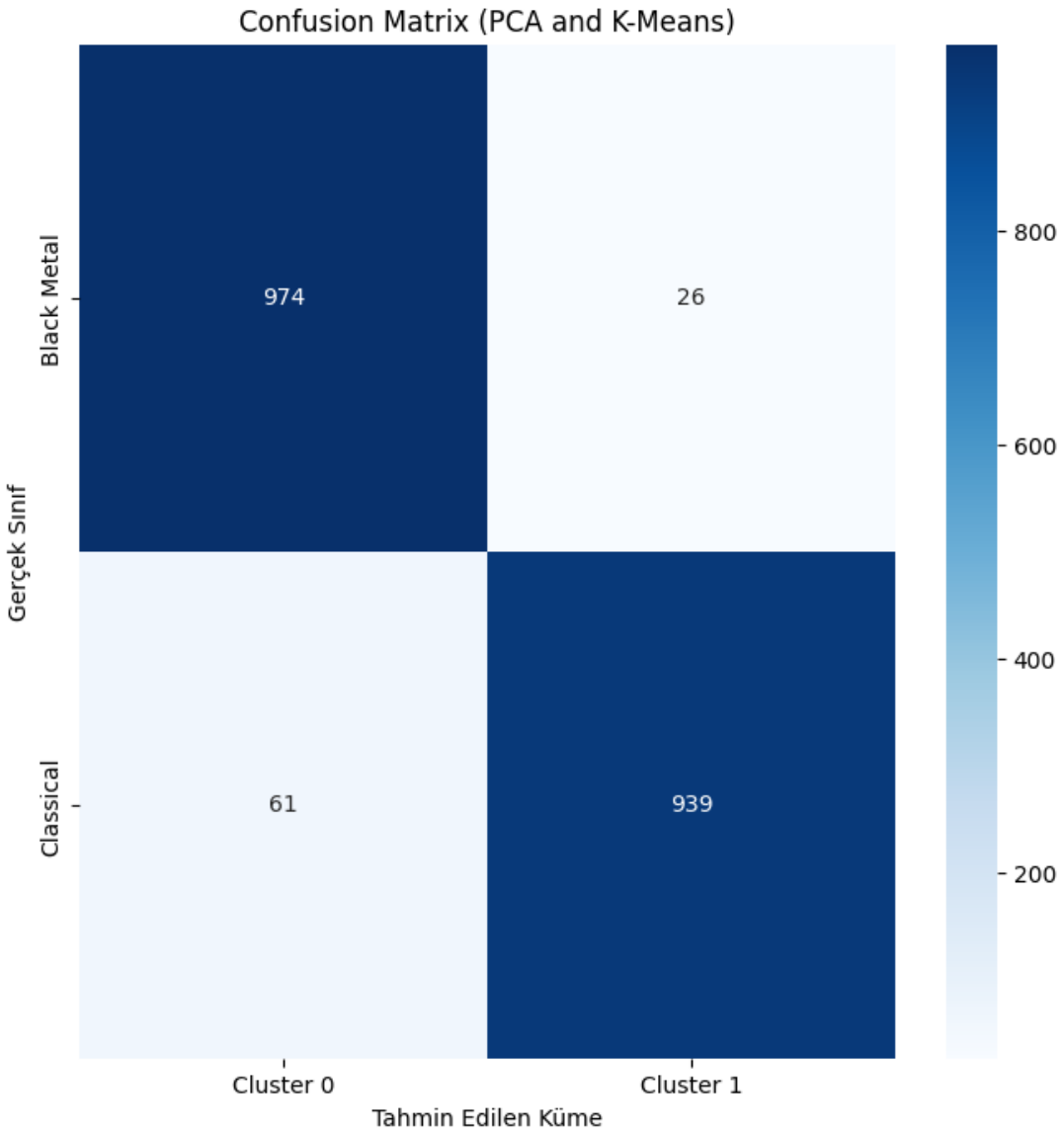
[ ]:
```
import plotly.express as px
import plotly.graph_objects as go


fig = px.scatter_3d(df_subset, x=X_pca[:, 0], y=X_pca[:, 1], z=X_pca[:, 2],
                    color='cluster_pca', color_discrete_map={0: 'red', 1:␣
 ↪'blue'},
                    title='3D PCA ve K-Means Kümeleme Görseli',
                    labels={'x': 'PCA1', 'y': 'PCA2', 'z': 'PCA3'})
fig.update_traces(marker=dict(size=3))
```

```
fig.add_trace(go.Scatter3d(x=kmeans_pca.cluster_centers_[:, 0],
                           y=kmeans_pca.cluster_centers_[:, 1],
                           z=kmeans_pca.cluster_centers_[:, 2],
                           mode='markers',
                           marker=dict(size=10, color='red', symbol='x'),
                           name='Küme Merkezleri'))

fig.show()
```

[ ]:
```
from sklearn.metrics import accuracy_score, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

true_labels = df_subset['track_genre'].map({'black-metal': 0, 'classical': 1})

kmeans_labels = kmeans.labels_


pred_labels_pca = df_subset['cluster_pca'].values

predicted_labels = 1 - kmeans_labels

misclassified_black_metal = sum((true_labels == 0) & (predicted_labels == 1))
misclassified_classical = sum((true_labels == 1) & (predicted_labels == 0))

error_rate = (misclassified_black_metal + misclassified_classical) /␣
 ↪len(true_labels)
misclassified_black_metal_rate = misclassified_black_metal /␣
 ↪len(df_subset[df_subset['track_genre'] == 'black-metal'])
misclassified_classical_rate = misclassified_classical /␣
 ↪len(df_subset[df_subset['track_genre'] == 'classical'])

print(f"Hata Oranı: {error_rate}")
print(f"Black Metal Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_black_metal_rate}")
print(f"Classical Sınıfında Hatalı Sınıflandırma Oranı:␣
 ↪{misclassified_classical_rate}")

accuracy = accuracy_score(true_labels, predicted_labels)
print(f'Doğruluk (Accuracy): {accuracy:.4f}')


conf_matrix_pca = confusion_matrix(true_labels, pred_labels_pca)

plt.figure(figsize=(8, 8))
```

```
sns.heatmap(conf_matrix_pca, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=['Cluster 0', 'Cluster 1'], yticklabels=['Black Metal',␣
 ↪'Classical'])
plt.xlabel('Tahmin Edilen Küme')
plt.ylabel('Gerçek Sınıf')
plt.title('Confusion Matrix (PCA and K-Means)')
plt.show()

error_rate_pca = (conf_matrix_pca[0, 1] + conf_matrix_pca[1, 0]) /␣
 ↪len(true_labels)
print(f'Hata Oranı (PCA ve K-Means): {error_rate_pca:.4f}')

cluster_counts = df_subset['cluster_pca'].value_counts().sort_index()
for cluster, count in cluster_counts.items():
    print(f"Cluster {cluster}: {count} eleman")
```

```
Hata Oranı: 0.0435
Black Metal Sınıfında Hatalı Sınıflandırma Oranı: 0.026
Classical Sınıfında Hatalı Sınıflandırma Oranı: 0.061
Doğruluk (Accuracy): 0.9565
```




```
Hata Oranı (PCA ve K-Means): 0.0435
Cluster 0: 1035 eleman
Cluster 1: 965 eleman
```

Kaynakça

[1] Gareth James, Daniela Witten, Trevor Hastie, ve Robert Tibshirani, *An Introduction to Statistical Learning*, Springer, 2013.

[2] DataCamp.(2020) "Principal Component Analysis in Python." Erişim: `https://www.datacamp.com/tutorial/principal-component-analysis-in-python`

Bu çalışmada kullanılan kanser verileri, DataCamp'ın "Principal Component Analysis in Python" tutorial'ından alınmıştır Veriler ve daha fazla bilgi için DataCamp'ın "Principal Component Analysis in Python" tutorial'ını ziyaret edebilirsiniz:
DataCamp: Principal Component Analysis in Python

[3] Maharshi Pandya. "Spotify Tracks Dataset". Hugging Face Datasets. URL: `https://huggingface.co/datasets/maharshipandya/spotify-tracks-dataset`

[4] Google Colab. "Google Colaboratory".`https://colab.research.google.com/`