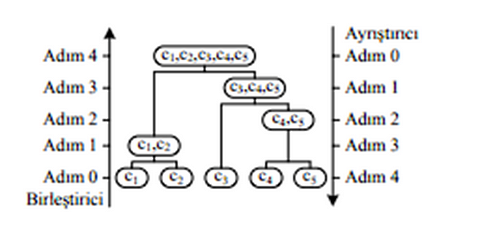
**Veri madenciliği ve makine öğrenmesinin önemli bir yönü olarak, verileri benzer özelliklere sahip gruplara ayırma sürecidir. “Denetimsiz Öğrenme” yöntemlerindendir. Yani kümeleme algoritmaları verileri sınıflandırmak için etiketli veriye ihtiyaç duymaz.Kümeleme, verideki yapıları ortaya çıkarmaya çalışır ve verinin içinde doğal olarak bulunan benzerliklere dayanarak veri noktalarını kümelere ayırır.**

**Hiyerarşik Kümeleme - Hiyerarşik Olmayan Kümeleme**

***Hiyerarşik kümeleme yöntemleri*, kümelerin bir ana küme olarak ele alınması ve sonra aşamalı olarak içerdiği alt kümelere ayrılması veya ayrı ayrı ele alınan kümelerin aşamalı olarak bir küme biçiminde birleştirilmesi esasına dayanır. [1]**

**Bu yöntemle veriyi daha küçük alt kümelere ayırma işlemini iteratif bir şekilde yapar ve iki ana türü vardır:**

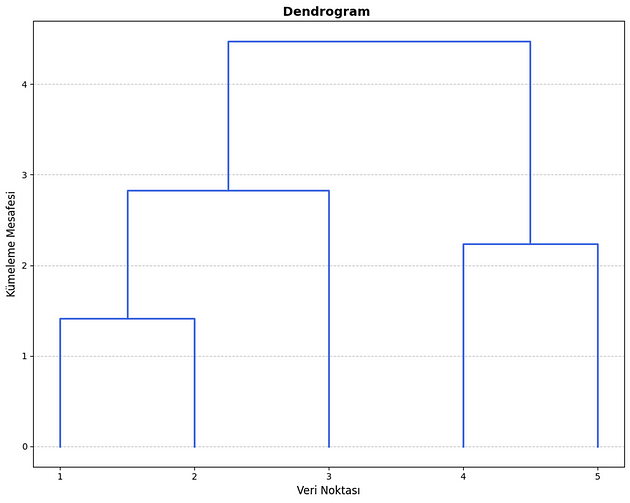
1. **Aşağıdan Yukarı / Birleştirici (Agglomerative): Küçük kümeler (başlangıçta her veri bir küme olarak kabul edilir) birleştirilerek daha büyük kümeler oluşturulur. Bu yöntem genellikle daha yaygındır.**
2. **Yukarıdan Aşağıya / Ayrıştırıcı (Divisive): Bütün veri tek bir küme olarak kabul edilir ve daha sonra bu küme alt kümelere ayrılır.**

****

**[2]**

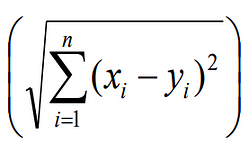
***Hiyerarşik olmayan kümeleme*,veriyi kümeler oluştururken hiyerarşik bir yapıyı takip etmeyen bir yöntemdir. Yani verinin tek bir düzeyde kümelendiği ve sıralı bir yapı oluşturmadığı bir tekniktir. “K-Means algoritması” hiyerarşik olmayan kümelemenin bir örneğidir. Bu tip kümeleme algoritmalarında kümeler genellikle başlangıçta önceden belirlenen bir sayıda kümeye ayrılır.**

**! Dendrogram : Veriler arasındaki hiyerarşik ilişkileri görselleştiren bir ağaç yapısıdır.**

****

**[3] Örnek Dendrogram Gösterimi**

**Bu veri noktalarını kümeleme için genellikle uzaklık noktaları hesaplanır. Bu hesaplamalar için en bilineni “Öklid” olmak üzere “Manhattan” ve “Minkowski” uzaklığı formülleri kullanılır.**

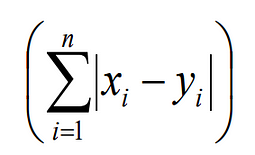
****

**[4] Öklid Uzaklığı Formülü**

**yazı tipi, beyaz, grafik, tipografi içeren bir resim

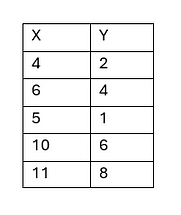
Açıklama otomatik olarak oluşturuldu**

**[5] Minkowski Uzaklığı Formülü**

****

**[6] Manhattan Uzaklığı Formülü**

**Örnek Uygulama**

****

**[7] Örnek Veri Seti**

**Veriyi Öklit Uzaklığı ile birbirleriyle olan uzaklıkları hesaplanır. Bütün hesaplamalar yapılmalı.**

****

**d(2,1) = 2.83**

****

**d(3,1) = 1.41**

****

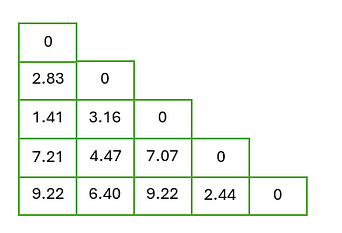
**d(4,1) = 7.21**

**yazı tipi, metin, beyaz, çizgi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

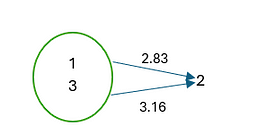
**d(5,1) = 9.21**

**Buna göre tüm hesaplamalar yapınca oluşan tablo aşağıda görülebilir.**

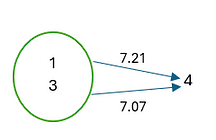
****

**[8] En küçüğü seçilmelidir. Bu da d(3,1) noktadır.**

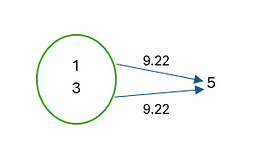
1. **1. ve 3. noktalar birbirlerine en yakın olduğu için bu noktalar seçilir.**
2. **Bu noktaların diğer noktalar ile olan en kısa uzaklığı seçmeliyiz.**

****

**1–3 oluşan noktanın 2 noktasına en yakın uzaklığı “2.83” olarak diğer işleme geçmeliyiz.**

****

**7.07 seçilmeli**

****

**9.22 seçilmeli**

**Bu oluşan yeni uzaklıklara göre tablo oluşturulmalı.**

**metin, diyagram, çizgi, yazı tipi içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**[9] 2.Aşamada oluşan tablolar.**

**! Gözden kaçırılmamalı:**

* **Burada en kısa mesafe seçilirken aralarındaki uzaklık her zaman ilk tabloya göre hesaplanıyor. Oluşan yeni tablolar bizim için hamlemizi kolaylaştıran kısım.**
* **Eğer daha önceden kümelenmiş veriler arasında uzaklık hesabı yapılıyorsa tüm noktalar göz önüne alınarak hesaplama yapılmalıdır.**

**çizgi, diyagram, daire içeren bir resim

Açıklama otomatik olarak oluşturuldu**

**Örneğin 3–4 arası 7.07 en küçük olduğu için bu adım seçilmeli.**

**Kod İle Kümeleme**

***Kütüphaneler***

**import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt**

***Veri noktaları arasındaki Öklid Uzaklığı hesaplayan ve bu mesafeye göre en yakın iki kümeyi birleştirmek için kullanan fonksiyonlarımız***

**def euclidean\_distance(p1, p2):  
 return np.sqrt(np.sum((p1 - p2) \*\* 2))  
  
def hierarchical\_clustering(data):  
   
 data\_array = data.values   
   
 n = len(data\_array)  
 clusters = {i: [i] for i in range(n)}   
 distances = np.zeros((n, n))   
  
   
 for i in range(n):  
 for j in range(i + 1, n):  
   
 distances[i, j] = euclidean\_distance(data\_array[i], data\_array[j])   
 distances[j, i] = distances[i, j]  
  
 linkage\_matrix = []  
  
 while len(clusters) > 1:  
   
 min\_dist = float('inf')  
 cluster\_pair = (None, None)  
 keys = list(clusters.keys())  
  
 for i in range(len(keys)):  
 for j in range(i + 1, len(keys)):  
 c1, c2 = keys[i], keys[j]  
 dist = min(distances[p1, p2] for p1 in clusters[c1] for p2 in clusters[c2])  
 if dist < min\_dist:  
 min\_dist = dist  
 cluster\_pair = (c1, c2)  
  
   
 c1, c2 = cluster\_pair  
 new\_cluster = clusters[c1] + clusters[c2]  
 new\_cluster\_id = max(clusters.keys()) + 1  
 clusters[new\_cluster\_id] = new\_cluster  
 del clusters[c1]  
 del clusters[c2]  
  
   
 linkage\_matrix.append([min(c1, c2), max(c1, c2), min\_dist, len(new\_cluster)])  
  
 return np.array(linkage\_matrix)  
  
linkage\_matrix = hierarchical\_clustering(data)**

***Dendogram çizdiren fonksiyonumuz***

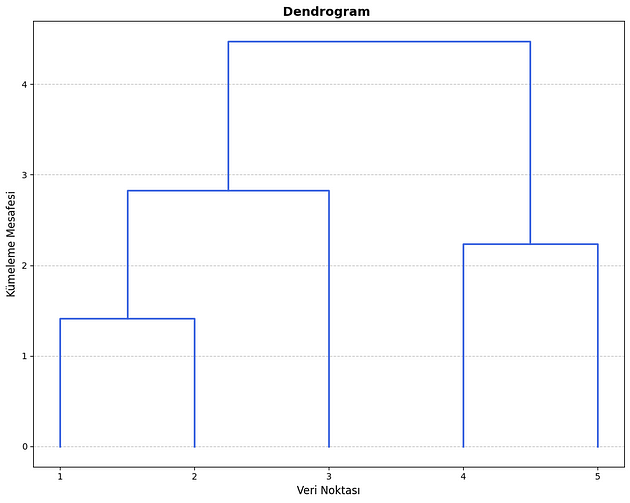
**def plot\_dendrogram(linkage\_matrix, labels):  
 plt.figure(figsize=(10, 8))  
 current\_positions = {i: (i, 0) for i in range(len(labels))} # Her kümenin (x, y) koordinatları  
  
 for i, (c1, c2, dist, \_) in enumerate(linkage\_matrix):  
 # Alt kümelerin pozisyonlarını al  
 x1, y1 = current\_positions[c1]  
 x2, y2 = current\_positions[c2]  
 y = dist # Birleşme yüksekliği  
  
 # Alt kümeler sıfıra inmesin, doğru yüksekliği kullan  
 plt.plot([x1, x1, x2, x2], [y1, y, y, y2], c='royalblue', lw=2)  
  
 # Yeni kümenin pozisyonunu güncelle  
 current\_positions[len(labels) + i] = ((x1 + x2) / 2, y)  
  
 # Ekseni ve tasarımı ayarla  
 plt.xticks(range(len(labels)), labels, fontsize=10)  
 plt.title("Dendrogram ", fontsize=14, fontweight='bold')  
 plt.xlabel("Veri Noktası", fontsize=12)  
 plt.ylabel("Kümeleme Mesafesi", fontsize=12)  
 plt.grid(axis='y', linestyle='--', alpha=0.7)  
 plt.tight\_layout()  
 plt.show()  
  
# Dendrogram'ı çiz  
plot\_dendrogram(linkage\_matrix, labels=np.arange(1, len(data) + 1))  
  
# Kümeleme için desired\_clusters değerini ayarlayın (2'yi örnek olarak kullanıyoruz)  
desired\_clusters = 2  
  
# Kümelemeyi gerçekleştirin  
linkage\_matrix = hierarchical\_clustering(data)**

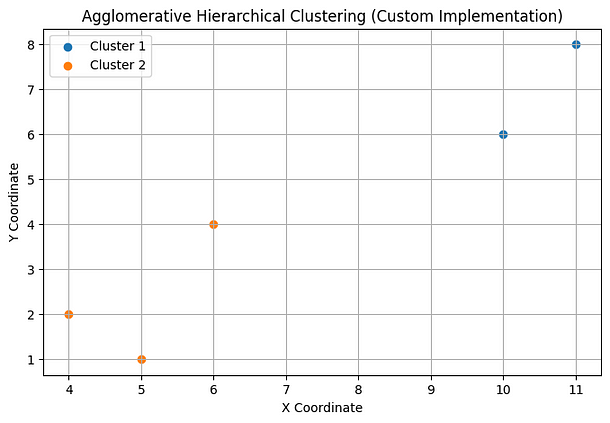
**! Hazır kütüphane içinde de dendrogram çizdiren fonksiyon var**

**import numpy as np  
from scipy.cluster.hierarchy import dendrogram, linkage  
import matplotlib.pyplot as plt  
# 1. Veri setini oluştur  
data = np.array([  
 [4, 2],  
 [11, 8],  
 [6, 4],  
 [5, 1],  
 [10, 6]  
])  
  
# 2. Bağlantı matrisi oluştur (linkage)  
# 'ward' yöntemiyle hiyerarşik kümeleme yapılır  
linked = linkage(data, method='ward')  
  
# 3. Dendrogram çizimi  
plt.figure(figsize=(8, 6))  
dendrogram(linked,  
 orientation='top', # Yukarıdan aşağıya çizim  
 labels=np.arange(1, len(data)+1), # Her bir veri noktası için etiket  
 distance\_sort='descending',  
 show\_leaf\_counts=True) # Yapraklardaki örnek sayısını göster  
  
plt.title("Dendrogram")  
plt.xlabel("Veri Noktası")  
plt.ylabel("Kümeleme Mesafesi")  
plt.show()**

***Diğer kümeleme gösterimi***

**# Scipy'nin fcluster fonksiyonunu kullanarak etiketleri alın  
from scipy.cluster.hierarchy import fcluster  
labels = fcluster(linkage\_matrix, t=desired\_clusters, criterion='maxclust')  
  
plt.figure(figsize=(8, 5))  
plt.title("Agglomerative Hierarchical Clustering (Custom Implementation)")  
for label in np.unique(labels):  
 group = data[labels == label]  
 # Access DataFrame columns using their names or .iloc  
 plt.scatter(group.iloc[:, 0], group.iloc[:, 1], label=f'Cluster {label}')   
  
plt.xlabel("X Coordinate")  
plt.ylabel("Y Coordinate")  
plt.legend()  
plt.grid(True)  
plt.show()  
  
print(f"Cluster labels:\n{labels}")**

****

****

**Kaynakça**

**[1] Özkan, Y., “Kümeleme”, Veri Madenciliği Yöntemleri, Papatya Yayıncılık, İstanbul, 131- 148, 2008.**

**[2] Yeşilbudak, M., Kahraman, H. T., & Karacan, H. (2010). *Veri madenciliğinde nesne yönelimli birleştirici hiyerarşik kümeleme modeli*.**

**[4]-[5]-[6] Taşcı, E., & Onan, A. (t.y.). K-En yakın komşu algoritması parametrelerinin sınıflandırma performansı üzerine etkisinin incelenmesi. *Ege Üniversitesi*, *Celal Bayar Üniversitesi*.**