**Kümeleme ve K-Means Algoritması**

**Clustering (Kümeleme) Nedir?**

Denetimsiz öğrenme yöntemlerinden biridir ve benzer özelliklere sahip verileri gruplamak amacıyla kullanılır. Bu yöntem verileri etiketler veya önceden belirlenmiş kategoriler olmadan, doğal bir şekilde kümelere ayırır.

**Denetimsiz Öğrenme**

Makine öğrenme modellerinden biridir. Modelin veriler üzerinde etiketli (doğru yanıtlar) bilgiye ihtiyaç duymadan öğrenmesini sağlar. Yani, denetimsiz öğrenmede, eğitim verisinde etiketler (yani, doğru sınıf etiketleri veya yanıtlar) bulunmaz. Model, sadece verilerdeki desenleri, yapıları veya gizli ilişkileri keşfederek sonuçlar çıkarır.

**Kullanım Alanları:**

1- **Müşteri Segmentasyonu**: Müşterinin davranışlarını analiz ederek benzer özellik sahibi müşterileri gruplamak için kullanılabilir.

2- **Görüntü Sıkıştırma**: Görüntüdeki renklerin yoğunluğunu analiz ederek benzer renkleri gruplamak ve renk sayısını azaltmak için kullanılabilir.

3- **Genetik Veri Analizi**: Genetik benzerliklerine göre DNA dizilimlerine göre ayırmak için kullanılabilir.

4- **Belge-Metin Kümelemesi**: Haber makalelerinin otomatik kategorilere ayrılması (spor, ekonomi, politika vb.) ve arama motoru optimizasyonu gibi alanlarda kullanılabilir.

5- **Tıbbi Veri Analizi**: Tıbbi verileri kümelere ayırarak hastalık teşhisi veya hasta grupları oluşturmak için kullanılabilir. Benzer semptomlara sahip hasta gruplarının bulunması veya belirli test sonuçlarına göre hasta-sağlıklı olarak sınıflandırılması yapılabilir.

6- **Sosyal Ağ Analizi**: Sosyal ağlardaki kullanıcıların davranışlarını inceleyerek benzer profilleri gruplamak için kullanılabilir.

7- **Coğrafi Veri Analizi ve Haritalama**: Coğrafi lokasyonlara göre gruplandırma yaparak veriyi anlamlandırmak için kullanılabilir ve bu amaçla şehir planlaması, doğal afet risk bölgelerinin tespiti için kullanılabilir.

8- **Anomali Tespiti** : Kümelerden sapma gösteren (farklı davranış sergileyen) verileri tespit etmek için kullanılabilir.

9- **E-Ticaret ve Ürün Öneri Sistemleri**: Ürünleri veya kullanıcıları benzerliklerine göre gruplandırmak için kullanılır ve kişiselleştirilmiş ürün önerileri oluşturma ve kullanıcıların alışveriş alışkanlıklarına göre segmentasyon gibi olaylarda da etkili biçimde kullanılabilir.

**K-Means Algoritması Nedir?**

Bir veri setindeki örnekleri benzerliklerine göre gruplara ayırmak, yani kümeler oluşturmaktır. Algoritmanın adı, her kümenin merkezi olarak küme ortalamasını (mean) kullanmasından gelir.

Algoritma için önemli terimler:

‘Centroid’ = Verilerin ortasında yer alan, kümenin ortalama konumunu temsil eden noktadır. = Merkez Noktası

‘Voronoi Hücreleri’ = Bir düzlemi farklı bölgeler olarak bölmek için kullanılan yöntemidir. = Centroid etkisi altındaki bölgeler

**K-Means Algoritması Nasıl Çalışır?**

**1- Küme Sayısının Belirlenmesi**:

* Önceden bir **k** değeri (kaç küme oluşturulacağı) belirlenir.

**2- Başlangıç Küme Merkezlerinin (Centroid) Rastgele Seçilmesi**:

* İlk aşamada, veri uzayında k adet küme merkezi rastgele seçilir.

**3- Verilerin Kümelere Atanması**:

* Her bir veri noktası, kendisine en yakın küme merkezine atanır.
* Burada, genelde **Öklid Uzaklığı** gibi bir mesafe metriği kullanılır.

ekran görüntüsü, diyagram, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

**4- Küme Merkezlerinin Güncellenmesi**:

* Her küme için, o kümeye ait veri noktalarının aritmetik ortalaması alınır ve yeni küme merkezi olarak atanır.

**5- Adımların Tekrarı**:

* Veri noktalarının kümelere atanması ve küme merkezlerinin güncellenmesi işlemleri, belirli bir durma kriteri (örneğin, küme merkezleri artık değişmiyorsa veya belirli bir iterasyon sayısına ulaşıldıysa) sağlanana kadar devam eder.

**Kod İle K-Means Algoritmaları**

Kütüphaneler

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

Algoritmanın Genel Kod Yapısı

class KMeansManual:  
 def \_\_init\_\_(self, n\_clusters=4, max\_iter=300, tol=1e-4, random\_state=None):  
 self.n\_clusters = n\_clusters  
 self.max\_iter = max\_iter  
 self.tol = tol  
 self.random\_state = random\_state  
  
 def fit(self, X):  
 # Başlangıç küme merkezlerini rastgele seç  
 if self.random\_state:  
 np.random.seed(self.random\_state)  
 initial\_indices = np.random.choice(X.shape[0], self.n\_clusters, replace=False)  
 centroids = X[initial\_indices]  
  
 for i in range(self.max\_iter):  
 # 1. Adım: Her noktayı en yakın küme merkezine ata  
 labels = self.\_assign\_clusters(X, centroids)  
  
 # 2. Adım: Yeni küme merkezlerini hesapla  
 new\_centroids = np.array([self.\_compute\_centroid(X[labels == j]) for j in range(self.n\_clusters)])  
  
 # Küme merkezleri arasında değişim  
 shift = self.\_calculate\_shift(new\_centroids, centroids)  
  
 # Eğer değişim belirtilen toleransın altına düşerse, algoritmayı durdur  
 if shift < self.tol:  
 print(f"Converged after {i+1} iterations.")  
 break  
  
 centroids = new\_centroids  
  
 self.centroids\_ = centroids  
 self.labels\_ = labels  
  
 def \_assign\_clusters(self, X, centroids):  
 # Öklid mesafesini manuel olarak hesapla ve her nokta için en yakın küme merkezini ata  
 labels = []  
 for point in X:  
 distances = [self.\_euclidean\_distance(point, centroid) for centroid in centroids]  
 labels.append(np.argmin(distances))  
 return np.array(labels)  
  
 def \_compute\_centroid(self, points):  
 # Verilen noktalar için merkez (ortalama) hesapla  
 return np.mean(points, axis=0)  
  
 def \_euclidean\_distance(self, point1, point2):  
 # İki nokta arasındaki Öklid mesafesini manuel hesaplama  
 return np.sqrt(np.sum((point1 - point2) \*\* 2))  
  
 def \_calculate\_shift(self, new\_centroids, centroids):  
 # Merkezler arasındaki mesafeyi hesapla (değişim)  
 shifts = [self.\_euclidean\_distance(new, old) for new, old in zip(new\_centroids, centroids)]  
 return np.sum(shifts)

Grafiği Çizme Kod Yapısı

def load\_and\_cluster\_data(csv\_path, feature\_columns, n\_clusters=4, random\_state=111):  
 # CSV dosyasını oku  
 df = pd.read\_csv(csv\_path, sep=';') # Specify the semicolon delimiter  
  
 # Seçilen özellikleri al  
 X = df[feature\_columns].values  
  
 # KMeans algoritmasını uygula  
 algorithm = KMeansManual(n\_clusters=n\_clusters, random\_state=random\_state)  
 algorithm.fit(X)  
  
 # Sonuçları görselleştir  
 plt.figure(figsize=(10, 8))  
 for i in range(n\_clusters):  
 plt.scatter(X[algorithm.labels\_ == i, 0],  
 X[algorithm.labels\_ == i, 1],  
 label=f"Cluster {i+1}",  
 s=30)  
  
 plt.scatter(algorithm.centroids\_[:, 0],  
 algorithm.centroids\_[:, 1],  
 c='black',  
 marker='x',  
 s=200,  
 label='Centroids')  
  
 plt.title("Manual KMeans Clustering")  
 plt.xlabel(feature\_columns[0])  
 plt.ylabel(feature\_columns[1])  
 plt.legend()  
 plt.grid(True)  
 plt.show()  
  
 return algorithm

Kodun Çalıştırılması

load\_and\_cluster\_data('/content/S1\_.csv', ['X','Y'])

metin, ekran görüntüsü, yazı tipi, sayı, numara içeren bir resim

Açıklama otomatik olarak oluşturulduÖrnek Verimiz

metin, harita, diyagram, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturulduSonuç çıktımız