Kümeleme

Kümeleme algoritmaları, veri noktalarını belirli ölçütler doğrultusunda gruplara ayırmak için kullanılan makine öğrenmesi ve veri analizi teknikleridir. Bu algoritmalar, veri setindeki benzerlikleri ve farklılıkları tanımlayarak verilerin doğal gruplarına bölünmesine yardımcı olur. İşte bazı popüler kümeleme algoritmaları ve nasıl çalıştıkları hakkında detaylı bilgiler:

1. K-Means Kümeleme

K-Means kümeleme, belirli bir veri kümesini K sayıda küme veya grup olarak böler. Algoritma, küme merkezlerini rastgele başlatır ve her veri noktasını en yakın küme merkezine atar. Ardından, her veri noktası için yeni küme merkezleri hesaplanır ve bu işlem küme merkezleri sabitlenene kadar tekrar eder. K-Means algoritması genellikle hızlı ve ölçeklenebilir olmasıyla tercih edilir, ancak küme sayısını önceden belirtmek gereklidir.

2. Hiyerarşik Kümeleme

Hiyerarşik kümeleme, aglomeratif (birleştirici) veya bölücü (bölücü) olmak üzere iki temel yaklaşımdan biriyle çalışır. Aglomeratif yöntemde, her veri noktası ayrı bir küme olarak başlar ve benzer kümeleme kriterlerine dayalı olarak en yakın küme veya veri noktaları birleştirilir. Bölücü yöntemde ise tüm veri noktaları tek bir küme olarak başlar ve daha sonra benzerlik kriterlerine göre kümeleme işlemi yapılır. Hiyerarşik kümeleme, veri setinde farklı düzeylerde küme yapısını görselleştirmek için dendrogramlar kullanır.

3. DBSCAN (Yoğunluk Temelli Kümeleme)

DBSCAN, veri setinde yoğun bölgeleri tanımlamak ve düşük yoğunluklu bölgeleri ayırmak için kullanılan bir yoğunluk temelli kümeleme algoritmasıdır. Algoritma, veri noktalarını çekirdek noktaları, sınırlayıcı noktaları ve gürültü noktaları olarak sınıflandırır. Veri noktalarını birbirine yakınlığı ve belirli bir yoğunluk eşiğini temel alarak kümeleme işlemi yapar.

4. K-Medoids Kümeleme

K-Medoids kümeleme, K-Means'e benzer ancak küme merkezlerinin veri noktalarından seçilmesine dayanır. Bu algoritma, küme içi değişkenlikleri minimize etmeye çalışır ve küme merkezi olarak veri noktalarını seçer. K-Medoids, gürültüye karşı daha dayanıklı olabilir ve medoid (küme merkezi olarak seçilen bir veri noktası) seçimiyle daha sağlam kümeleme sonuçları verir.

5. Gaussian Mixture Model (GMM)

Gaussian Mixture Model (GMM), veri noktalarını Gauss dağılımı ile temsil edilen bileşenlere ayırarak kümeleme yapar. Her bir bileşen, bir merkez, bir varyans ve ağırlıklarla temsil edilir. GMM, veri setindeki her noktanın birden fazla kümeye ait olabileceği belirsizliği ele alabilir ve yumuşak (soft) kümeleme yapabilir.

6. OPTICS (Sıralama Temelli Kümeleme)

OPTICS, veri noktalarını birbirine olan uzaklıkları ve yoğunluklarına dayanarak sıralı bir küme yapısı oluşturan bir kümeleme algoritmasıdır. OPTICS, DBSCAN'den farklı olarak, kümeleme yaparken küme sayısını önceden belirtmeye gerek duymaz ve veri setindeki küme yapılarını daha esnek bir şekilde keşfeder.

Kümeleme Algoritmalarının Seçimi

Hangi kümeleme algoritmasının kullanılacağı, veri setinin özelliklerine, veri noktalarının dağılımına, küme sayısının bilinip bilinmemesine ve kümeleme sonuçlarından beklenen özelliklere bağlı olarak değişir. Veri analizi ve makine öğrenmesinde doğru kümeleme algoritmasını seçmek, veri setinizin yapısal özelliklerini anlamak ve belirli bir uygulama bağlamında doğru çözümü sağlamak için önemlidir.

Örnek

K-Means, veriyi K kümesine bölen ve her kümeyi bir merkez noktasına göre tanımlayan yaygın bir kümeleme algoritmasıdır. Bu örnekte, `scikit-learn` kütüphanesini kullanarak K-Means algoritmasını uygulayacağız.

Öncelikle gerekli kütüphaneleri yükleyelim:

```
pip install numpy matplotlib scikit-learn
```

K-Means Kümeleme Örneği

Aşağıda, `make_blobs` fonksiyonunu kullanarak oluşturulan örnek veri seti üzerinde K-Means kümeleme algoritmasını uygulayan bir Python kodu bulunmaktadır:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans

# Örnek veri seti oluşturma
np.random.seed(42)
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# K-Means modelinin eğitilmesi
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

# Küme merkezlerinin alınması
centers = kmeans.cluster_centers_

# Sonuçların görselleştirilmesi
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75, marker='X')
plt.title('K-Means Kümeleme')
plt.xlabel('Özellik 1')
plt.ylabel('Özellik 2')
plt.show()
```

Açıklamalar

1. **Kütüphanelerin İmport Edilmesi:**
   - `numpy` kütüphanesi matematiksel işlemler için kullanılır.
   - `matplotlib.pyplot` kütüphanesi veri görselleştirme için kullanılır.
   - `sklearn.datasets` kütüphanesi örnek veri seti oluşturmak için kullanılır.
   - `sklearn.cluster` kütüphanesi K-Means algoritmasını uygulamak için kullanılır.
2. **Örnek Veri Seti Oluşturma:**
   - `make_blobs` fonksiyonu, belirli sayıda merkez etrafında dağılmış veri noktaları oluşturur.
   - Bu örnekte, 4 merkez (küme) etrafında toplanan 300 veri noktası oluşturulur.
3. **K-Means Modelinin Eğitilmesi:**
   - `KMeans` sınıfı, belirli sayıda küme (n_clusters) ile bir K-Means modeli oluşturur.
   - `fit` fonksiyonu, modeli veri seti üzerinde eğitir.
   - `predict` fonksiyonu, her veri noktası için hangi kümeye ait olduğunu tahmin eder.
4. **Küme Merkezlerinin Alınması:**
   - `cluster_centers_` özelliği, her kümenin merkez noktasını döner.
5. **Sonuçların Görselleştirilmesi:**
   - `plt.scatter` fonksiyonu, veri noktalarını ve küme merkezlerini görselleştirmek için kullanılır.
   - Veri noktaları, tahmin edilen kümelerine göre renklendirilir.
   - Küme merkezleri kırmızı 'X' işareti ile gösterilir.

Bu kod, K-Means algoritması kullanarak basit bir kümeleme örneğini göstermektedir. Bu temel yapı, diğer kümeleme problemlerine de uyarlanabilir.