SELÇUK BİLİŞİM TEKNOLOJİLERİ TOPLULUĞU

YAPAY ZEKA EĞİTİMİ DERS-5

SELÇUK ÜNİVERSİTESİ

Denetimsiz Öğrenme Nedir?

- Denetimsiz öğrenme, modeli denetlemenize gerek olmayan bir makine öğrenme tekniğidir. Bunun yerine, modelin bilgileri keşfetmek için kendi başına çalışmasına izin vermeniz gerekir. Esas olarak etiketlenmemiş verilerle ilgilenir.

- Denetimsiz öğrenme algoritmaları, denetimli öğrenmeye kıyasla daha karmaşık işleme görevleri gerçekleştirmenizi sağlar. Bununla birlikte, denetimsiz öğrenme diğer doğal öğrenme yöntemlerine kıyasla daha öngörülemez olabilir.

Bir bebeğin ve onun aile köpeğinin vakasını ele alalım.

Bu köpeği tanıyor ve tanıyor. Birkaç hafta sonra bir aile dostu bir köpeği getirir ve bebekle oynamaya çalışır.

- Bebek bu köpeği daha önce görmedi. Ancak birçok özelliği (2 kulak, göz, 4 ayak üzerinde yürüme) evcil köpeği gibi tanır. Yeni hayvanı bir köpek olarak tanımlar. Bu denetimsiz öğrenmedir, burada öğretilmiyorsunuz, ancak verilerden öğreniyorsunuz (bu durumda bir köpek hakkındaki veriler.) **Bu gözetimli öğrenme olsaydı, aile dostu bebeğe köpek olduğunu söylerdi.**

Neden Denetimsiz Öğrenme?

- Denetimsiz Öğrenmeyi kullanmanın başlıca nedenleri şunlardır:
- Denetimsiz makine öğrenimi, verilerdeki bilinmeyen her türlü paterni bulur.
- Denetimsiz yöntemler, kategorizasyon için yararlı olabilecek özellikleri bulmanıza yardımcı olur.
- Gerçek zamanlı olarak gerçekleşir, böylece tüm girdi verileri öğrencilerin varlığında analiz edilecek ve etiketlenecektir.
- Bir bilgisayardan etiketsiz veri almak, manuel müdahale gerektiren etiketli verilere göre daha kolaydır.

Denetimsiz Öğrenme Türleri

Kümeleme

sample

Cluster/group

Denetimsiz öğrenme söz konusu olduğunda kümelenme önemli bir kavramdır. Genel olarak, kategorize edilmemiş verilerden oluşan bir koleksiyonda bir yapı veya model bulma ile ilgilenir.

Kümeleme algoritmaları verilerinizi işler ve verilerde varsa doğal kümeleri (grupları) bulur. Ayrıca algoritmalarınızın kaç kümeyi tanımlaması gerektiğini de değiştirebilirsiniz. Bu grupların ayrıntı düzeyini ayarlamanıza olanak tanır.

K-NN(*K-Nearest Neighbor)*

K-NN *(K-Nearest Neighbor)* algoritması en basit ve en çok kullanılan sınıflandırma algoritmasından biridir. K-NN **non-parametric (** parametrik olmayan **)**, **lazy** (tembel) bir öğrenme algoritmasıdır. *lazy* kavramını anlamaya çalışırsak *eager learning* aksine lazy learning'in bir eğitim aşaması yoktur. Eğitim verilerini öğrenmez, bunun yerine eğitim veri kümesini *"ezberler"*. Bir tahmin yapmak istediğimizde, tüm veri setinde en yakın komşuları arar.

Algoritmanın çalışmasında bir ***K*** değeri belirlenir. Bu K değerinin anlamı bakılacak eleman sayısıdır. Bir değer geldiğinde en yakın K kadar eleman alınarak gelen değer arasındaki uzaklık hesaplanır. Uzaklık hesaplama işleminde genelde **Öklid fonksiyonu** kullanılır. Öklid fonksiyonuna alternatif
olarak **Manhattan**, **Minkowski** ve **Hamming** fonksiyonl arıda kullanılabilir. Uzaklık hesaplandıktan sonra sıralanır ve gelen değer uygun olan sınıfa atanır.

```python
import pandas as pd


# csv dosyamızı okuduk.
data = pd.read_csv('Iris.csv')
```

```
# Bağımlı Değişkeni ( species) bir değişkene atadık
species = data.iloc[:,-1:].values


# Veri kümemizi test ve train şekinde bölüyoruz
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(data.iloc[:,1:-1],species,test_size=0.33,random_state=0)
```

```python
# KNeighborsClassifier sınıfını import ettik
from sklearn.neighbors import KNeighborsClassifier

# KNeighborsClassifier sınıfından bir nesne ürettik
# n_neighbors : K değeridir. Bakılacak eleman sayısıdır. Default değeri 5'tir.
# metric : Değerler arasında uzaklık hesaplama formülüdür.
# p : Alternatif olarak p parametreside verilir. p değerini 2 vererek uzaklık hesaplama formülünü
# minkowski yerine öklid olarak değiştirebilirsiniz.
knn = KNeighborsClassifier(n_neighbors=5,metric='minkowski')
```

```python
# Makineyi eğitiyoruz
knn.fit(x_train,y_train.ravel())


# Test veri kümemizi verdik ve iris türü tahmin etmesini sağladık
result = knn.predict(x_test)
```

```python
# Başarı Oranı
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, result)

print(accuracy)
```

Hiyerarşik Kümeleme Tekniği

- **Hiyerarşik küme analizi** veya **HCA** olarak da adlandırılan , yukarıdan aşağıya baskın sıralama içeren kümeler oluşturmayı içeren denetimsiz bir kümeleme algoritmasıdır.
- Örneğin: Sabit diskimizdeki tüm dosya ve klasörler bir hiyerarşide düzenlenmiştir.

- Algoritma benzer nesneleri ***kümeler*** adı verilen gruplara ***ayırır*** . Son nokta kümeleri ya da grupların bir dizi , her küme birbirinden kümeden farklı olan, ve her bir küme içinde geniş bir şekilde birbirine benzemektedir.

Bu kümeleme tekniği iki türe ayrılır:

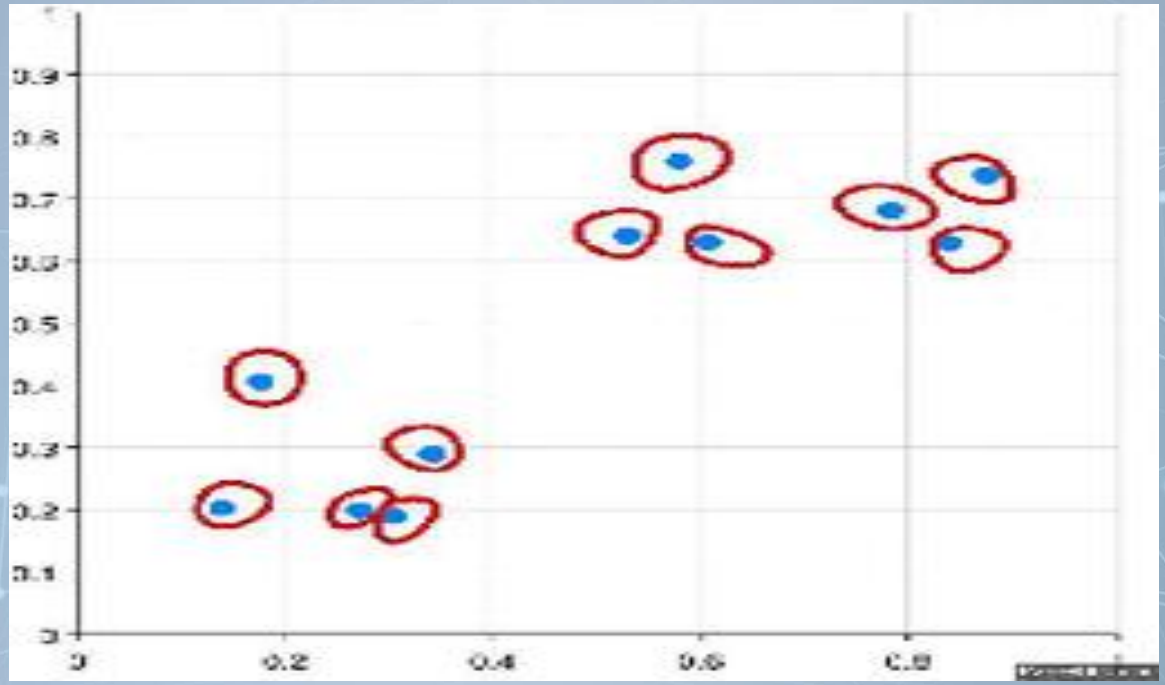
1. Aglomeratif Hiyerarşik Kümeleme
2. Bölünmüş Hiyerarşik Kümeleme

- **Aglomeratif Hiyerarşik Kümeleme**

Aglomeratif Hiyerarşik Kümeleme, kümelenmelerdeki nesneleri benzerliklerine göre gruplandırmak için kullanılan en yaygın hiyerarşik kümeleme türüdür. AGNES (Aglomerasyonlu Yuvalama) olarak da bilinir. Bu "aşağıdan yukarıya " bir yaklaşımdır: ***her gözlem kendi kümesinde başlar ve küme çiftleri hiyerarşide yukarı doğru ilerledikçe birleştirilir.***

Nasıl çalışır?

- Her veri noktasını tek noktalı bir küme haline getirme → N kümeleri oluşturur
- En yakın iki veri noktasını alın ve bir küme haline getirin → N-1 kümeleri oluşturur
- En yakın iki kümeyi alın ve bir küme haline getirin → N-2 kümeleri oluşturur.
- Yalnızca bir kümeyle bırakılana kadar 3. adımı yineleyin.


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.2
0.4
0.6
0.8

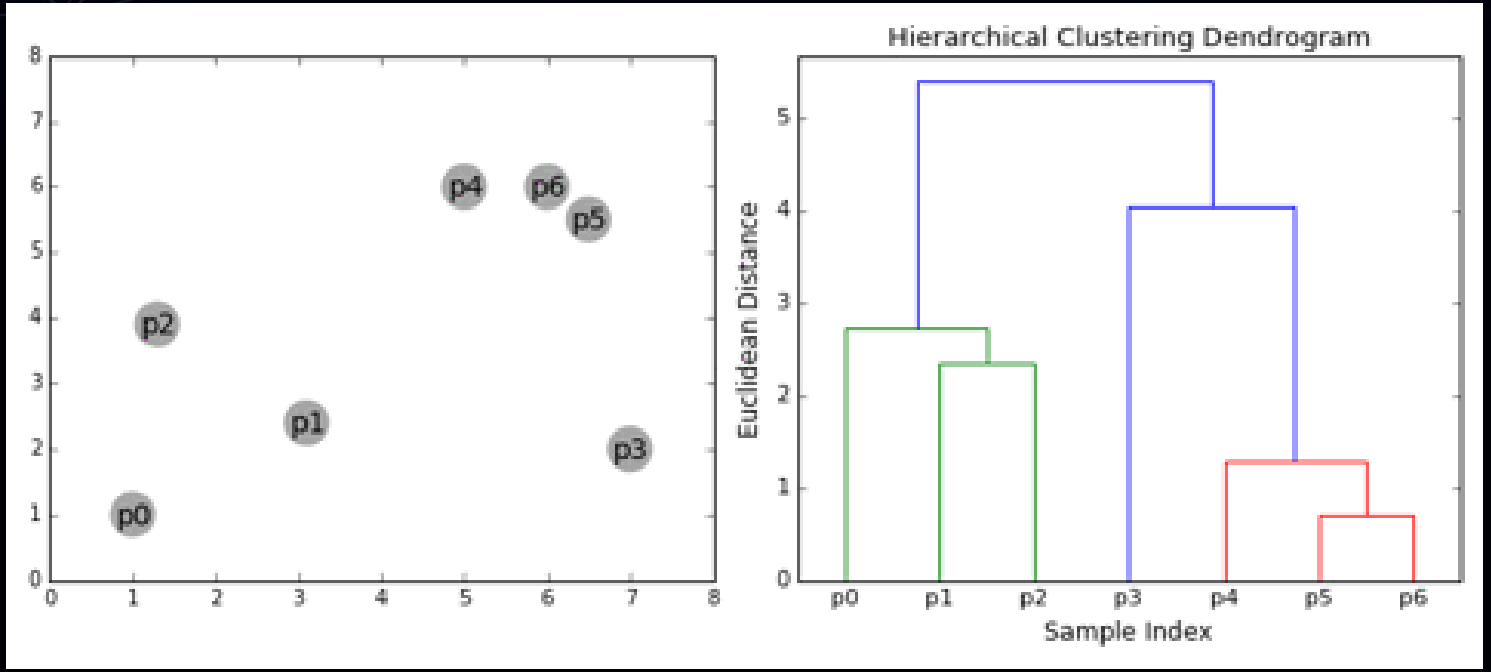
Kümeleme kurallarına karar vermek için kümeler arasındaki mesafeyi ölçmenin birkaç yolu vardır ve bunlara genellikle Bağlantı Yöntemleri denir. Ortak bağlantı yöntemlerinden bazıları şunlardır:

- **Tam bağlantı** : İki küme arasındaki mesafe, her kümedeki iki nokta arasındaki en *uzun* mesafe olarak tanımlanır .
- **Tek bağlantı** : İki küme arasındaki mesafe, her kümedeki iki nokta arasındaki en *kısa* mesafe olarak tanımlanır . Bu bağlantı, veri kümenizdeki, sonunda birleştirileceklerinden aykırı olabilecek yüksek değerleri tespit etmek için kullanılabilir.

- **Ortalama bağlantı** : iki küme arasındaki mesafe, bir kümedeki her nokta ile diğer kümedeki her nokta arasındaki ortalama mesafe olarak tanımlanır.
- **Centroid bağlantısı:** küme 1'in sentroidini ve küme 2'nin sentroidini bulur ve daha sonra birleştirmeden önce ikisi arasındaki mesafeyi hesaplar.

Dendrogram nedir?

- Dendrogram, farklı veri kümeleri arasındaki hiyerarşik ilişkileri gösteren bir ağaç diyagramı türüdür.
- Bir Dendrogram hiyerarşik kümeleme algoritmasının belleğini içerir, bu nedenle sadece Dendrogram'a bakarak kümenin nasıl oluştuğunu söyleyebilirsiniz.


p0
p1
p2
p3
p4
p5
p6
Hierarchical Clustering Dendrogram
Euclidean Distance
Sample Index