**SVM** ve **KNN**: Gözetimli (supervised) algoritmalardır. Eğitimde gerçek etiketleri bilerek öğrenirler. Kolayca **accuracy**, **precision**, **recall**, **f1-score** gibi metrikler ve **confusion matrix** hesaplanır.

**K-Means**: Gözetimsiz (unsupervised) kümeleme algoritmasıdır. Gerçek etiketlerden **haberi yoktur**. Eğer elinizde yine de “gerçek sınıf/küme” bilgisi varsa, K-Means’in **cluster** etiketlerini gerçek **label** ile eşleştirip bir çeşit performans değerlendirmesi yapabilirsiniz.

Küçük veri setlerinde (8-10 örnek gibi) genellikle çok yüksek doğruluk alabilirsiniz. Daha büyük ve gerçekçi veri setlerinde **train/test ayrımı**, **cross-validation**, **ölçeklendirme (scaling)**, **parametre optimizasyonu** (C, gamma, k vb.) gibi konulara özellikle dikkat etmek gerekir.



import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.svm import SVC

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

# --- 1) Veri oluşturma ---

data = {

'x1': [1.0, 2.0, 1.5, 2.5, 4.0, 5.0, 4.5, 6.0],

'x2': [2.0, 2.0, 1.8, 1.5, 5.0, 6.0, 4.8, 7.0],

'class': [0, 0, 0, 0, 1, 1, 1, 1]

}

df = pd.DataFrame(data)

X = df[['x1','x2']].values

y = df['class'].values

# --- 2) Train/Test Ayrımı ---

# Verimiz az olduğu için test\_size'ı küçük tutalım (örnek olarak).

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.25, random\_state=42

)

# --- 3) SVM Modeli Eğitimi ---

svm\_model = SVC(kernel='linear') # 'rbf', 'poly' vb. de olabilir

svm\_model.fit(X\_train, y\_train)

# --- 4) Tahmin ve Performans ---

y\_pred = svm\_model.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

acc = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

print("\nAccuracy:", acc)

print("\nClassification Report:\n", report)



import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import confusion\_matrix, classification\_report, accuracy\_score

# --- 1) Veri oluşturma ---

data = {

'x1': [1.0, 2.0, 1.5, 2.5, 4.0, 5.0, 4.5, 6.0],

'x2': [2.0, 2.0, 1.8, 1.5, 5.0, 6.0, 4.8, 7.0],

'class': [0, 0, 0, 0, 1, 1, 1, 1]

}

df = pd.DataFrame(data)

X = df[['x1','x2']].values

y = df['class'].values

# --- 2) Train/Test Ayrımı ---

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.25, random\_state=42

)

# --- 3) KNN Modeli (K=3) ---

knn\_model = KNeighborsClassifier(n\_neighbors=3)

knn\_model.fit(X\_train, y\_train)

# --- 4) Tahmin ve Performans ---

y\_pred = knn\_model.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

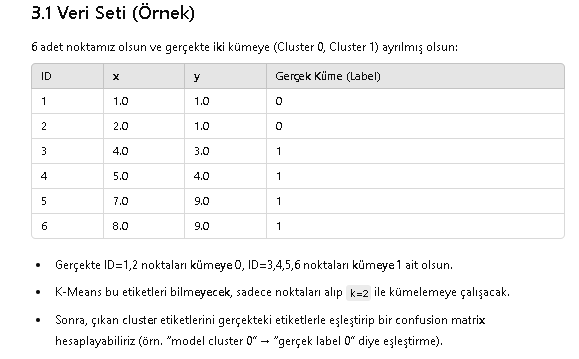
acc = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

print("Confusion Matrix:\n", cm)

print("\nAccuracy:", acc)

print("\nClassification Report:\n", report)



import numpy as np

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.metrics import confusion\_matrix, accuracy\_score

# --- 1) Veri oluşturma ---

data = {

'x': [1.0, 2.0, 4.0, 5.0, 7.0, 8.0],

'y': [1.0, 1.0, 3.0, 4.0, 9.0, 9.0],

'true\_label': [0, 0, 1, 1, 1, 1] # Gerçek kümeler (bilmiyor gibi davranacağız)

}

df = pd.DataFrame(data)

X = df[['x','y']].values

true\_labels = df['true\_label'].values

# --- 2) K-Means Eğitimi (k=2) ---

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans.fit(X)

# kmeans.labels\_ --> Her noktaya atanan küme etiketi (0 veya 1)

pred\_clusters = kmeans.labels\_

print("K-Means Tahmini Küme Etiketleri:", pred\_clusters)

print("Gerçek Küme Etiketleri :", true\_labels)

# --- 3) Eşleştirme ve Performans ---

# K-Means'te hangi cluster'ın gerçekte hangi sınıfa denk geldiğini bulmak için:

# Örneğin cluster "0" gerçekte "0" mıdır, yoksa "1" midir?

# Basit bir yol: her cluster'ın çoğunluğuna bakmak. Burada 2 cluster var.

# Manuel bir eşleştirme veya bir fonksiyon yardımıyla.

import numpy as np

import pandas as pd

from sklearn.cluster import KMeans

from sklearn.metrics import confusion\_matrix, accuracy\_score

# --- 1) Veri oluşturma ---

data = {

'x': [1.0, 2.0, 4.0, 5.0, 7.0, 8.0],

'y': [1.0, 1.0, 3.0, 4.0, 9.0, 9.0],

'true\_label': [0, 0, 1, 1, 1, 1] # Gerçek kümeler (bilmiyor gibi davranacağız)

}

df = pd.DataFrame(data)

X = df[['x','y']].values

true\_labels = df['true\_label'].values

# --- 2) K-Means Eğitimi (k=2) ---

kmeans = KMeans(n\_clusters=2, random\_state=42)

kmeans.fit(X)

# kmeans.labels\_ --> Her noktaya atanan küme etiketi (0 veya 1)

pred\_clusters = kmeans.labels\_

print("K-Means Tahmini Küme Etiketleri:", pred\_clusters)

print("Gerçek Küme Etiketleri :", true\_labels)

# --- 3) Eşleştirme ve Performans ---

# K-Means'te hangi cluster'ın gerçekte hangi sınıfa denk geldiğini bulmak için:

# Örneğin cluster "0" gerçekte "0" mıdır, yoksa "1" midir?

# Basit bir yol: her cluster'ın çoğunluğuna bakmak. Burada 2 cluster var.

# Manuel bir eşleştirme veya bir fonksiyon yardımıyla.

# Örnek ve basit yöntem:

# cluster0\_indices = pred\_clusters == 0 => bu noktaların çoğunluk "true\_label" ı ne?

# cluster1\_indices = pred\_clusters == 1 => ...

# Biz elle bakacağız.

# Aşağıda otomatik bir yaklaşım:

import numpy as np

def cluster\_to\_label\_map(true\_lbls, cluster\_lbls):

"""

Her cluster (0/1) için, oradaki gerçek etiketlerin çoğunluk sınıfını bulur.

Ardından cluster -> gerçek etiket eşlemesi döner.

"""

from collections import Counter

mapping = {}

clusters = np.unique(cluster\_lbls)

for c in clusters:

# Bu cluster'a ait indexleri bul

idx = np.where(cluster\_lbls == c)[0]

# Gerçek etiketlerin dağılımı

majority\_label = Counter(true\_lbls[idx]).most\_common(1)[0][0]

mapping[c] = majority\_label

return mapping

mapping\_dict = cluster\_to\_label\_map(true\_labels, pred\_clusters)

# Her noktaya "Tahmini etiket" atayalım:

pred\_labels\_mapped = np.array([mapping\_dict[c] for c in pred\_clusters])

# Şimdi confusion matrix ve accuracy'yi hesaplayabiliriz

cm = confusion\_matrix(true\_labels, pred\_labels\_mapped)

acc = accuracy\_score(true\_labels, pred\_labels\_mapped)

print("\nMapping (Küme -> Label):", mapping\_dict)

print("Tahmin Edilen Etiketler (Map Sonrası):", pred\_labels\_mapped)

print("\nConfusion Matrix:\n", cm)

print("\nAccuracy (Küme -> Label Eşlemesi ile):", acc)