**Karar Ağaçları**

Veri setindeki özelliklerin ve hedef değişkenin (etiketlerin veya sınıfların) ilişkisini öğrenmek için kullanılır. Örneğin, bir sınıflandırma görevinde, bir karar ağacı belirli bir özellik değerine göre farklı sınıflara ayırmak için kararlar alır.

Karar ağaçları, veri setindeki özelliklerin değerlerini kullanarak her düğümde bir soru sorar ve bu sorulara göre veriyi farklı dallara ayırır. Her aşamada, en iyi bölünmeyi yapacak özellik ve değer seçilir. Bu süreç, veri setindeki desenleri ve ilişkileri öğrenirken ağacın derinliği arttıkça devam eder.

metin, diyagram, çizgi, ekran görüntüsü içeren bir resim

Açıklama otomatik olarak oluşturulduKarar ağaçları, veri setini sınıflara ayırmak için basit "evet/hayır" veya "doğru/yanlış" sorularıyla çalışır. Örneğin, bir özellik değeri belirli bir eşiği aşıyor mu, bir kategorik değişken belirli bir sınıfa mı ait gibi sorularla veriyi böler.

Temel olarak, bir karar ağacı yapısı şu adımları izler:

1. **Kök düğüm (root node)**: Karar ağacının en üst düzeyinde bulunur. Tüm veri seti bu düğümde bulunur ve ağaç oluşturulmaya başlanır.
2. **Karar düğümleri (decision nodes)**: Kök düğümden başlayarak, her düğüm belirli bir özelliği test eder ve veriyi bu özelliğin değerlerine göre farklı dallara böler.
3. **Yaprak düğümleri (leaf nodes)**: Karar ağacının en alt seviyesinde bulunur. Yaprak düğümleri homojen bir sınıf veya sonuç içerir. Bu düğümlerde artık veri bölünemez, çünkü bir sınıfa ait olduğu kesindir.

metin, ekran görüntüsü, yazı tipi, daire içeren bir resim

Açıklama otomatik olarak oluşturuldu

Karar ağaçları sk-learn kullanarak adapte edilebilir.

1. Adım : Kütüphaneler dahil edilir.

from sklearn.tree import DecisionTreeClassifier

1. DecisionTreeClassifier sınıfından nesne oluşturulur.

# Karar ağacı modelini oluşturma ve eğitme

clf = DecisionTreeClassifier(random\_state=42)

clf.fit(X\_train, y\_train)

# Test seti üzerinde tahmin yapma

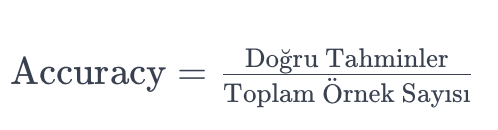
predictions = clf.predict(X\_test)

1. Sonuçları değerlendirme

Accuracy (Doğruluk), sınıflandırma problemlerinde kullanılan temel bir değerlendirme metriğidir.

**Doğru Tahminler:** Modelin doğru bir şekilde sınıflandırdığı örneklerin sayısı.

**Toplam Örnek Sayısı:** Tüm test veya doğrulama setindeki örneklerin toplam sayısı.



Sklearn aracılığıyla dahil edilebilir.

Kod:

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

# Model performansını değerlendirme

accuracy = accuracy\_score(y\_test, predictions)

print("Doğruluk (Accuracy):", accuracy)

# Sınıflandırma raporu

print("\nSınıflandırma Raporu:")

print(classification\_report(y\_test, predictions))

**TF-IDF (Terim Frekansı - Ters Belge Frekansı)**

TF-IDF (Terim Frekansı - Ters Belge Frekansı), metinsel belgelerdeki kelimelerin önemini ölçmek için kullanılan istatistiksel bir yöntemdir.

**TfidfVectorizer** sınıfı, metinsel verileri sayısal vektörlere dönüştürmek için kullanılan bir **scikit-learn** sınıfıdır. **analyzer='char'** parametresi, **TfidfVectorizer**'ın karakter düzeyinde vektörize etme işlemini yapmasını sağlar.

**analyzer='char'**, metinsel verileri karakter bazında ele alır ve her bir belgedeki karakterleri dikkate alarak vektörleştirme yapar. Yani, her karakterin bir öznitelik olarak ele alınarak metindeki karakter sıklığına göre bir vektör oluşturur. Bu, kelimenin yapısı yerine metindeki karakterlerin dağılımını dikkate alır.

Örneğin, "Ahmet" kelimesi **analyzer='char'** ile işlendiğinde, "A", "h", "m", "e", "t" gibi karakterlerin kullanım sıklıklarına göre vektörize edilir. Bu yöntem, özellikle kısa metinler veya kelime bazında anlamsal yapı yerine karakter bazında yapıyı dikkate almak istendiğinde kullanışlı olabilir.

**TfidfVectorizer**'ın **analyzer** parametresinin farklı değerleri vardır ve kullanımına bağlı olarak **'word'** (kelime bazında), **'char'** (karakter bazında) veya kullanıcı tarafından belirlenen özel bir fonksiyon kullanılabilir.

Özetle, **analyzer='char'** parametresi, **TfidfVectorizer**'ın metinsel verileri karakter bazında vektörlere dönüştürmesini sağlar ve her karakterin kullanım sıklığını temsil eden öznitelikler oluşturur. Bu, metinlerdeki karakter düzenini dikkate alarak metinsel verileri sayısal olarak temsil etmek için kullanılabilir.

from sklearn.feature\_extraction.text import TfidfVectorizer

# TF-IDF vektörleri oluşturalım

vectorizer = TfidfVectorizer(analyzer='char', ngram\_range=(1, 3))

X\_train\_vect = vectorizer.fit\_transform(X\_train)

X\_test\_vect = vectorizer.transform(X\_test)