Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

VERİ MADENCİLİĞİ (DATA MİNİNG)

Dr. Öğr. Üyesi Alper Talha Karadeniz

2025-2026

Hafta 6

Sınıflandırma Algoritmaları

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

01

Karar Ağaçları,
KNN

02

Naive Bayes

03

Doğruluk, F1,
ROC-AUC

04

Uygulama

Sınıflandırma Algoritmaları

Sınıflandırma, makine öğrenmesinde verileri önceden belirlenmiş kategorilere ayırma sürecidir ve genellikle etiketli verilerle çalışır. Bu süreçte model, eğitim verisinden öğrendiği kuralları kullanarak yeni verilerin hangi sınıfa ait olduğunu tahmin eder. Karar ağaçları, K-En Yakın Komşu (KNN), Naive Bayes, Destek Vektör Makineleri (SVM), lojistik regresyon, random forest ve yapay sinir ağları en yaygın kullanılan sınıflandırma algoritmalarıdır. Her algoritmanın çalışma prensibi farklıdır; bazıları istatistiksel olasılıkları kullanırken, bazıları veriyi dallara ayırır veya çok sayıda modeli bir araya getirir. Sınıflandırma, spam filtreleme, hastalık teşhisi ve görüntü tanıma gibi pek çok alanda yaygın olarak kullanılmaktadır.

Sınıflandırma Teknikleri ve Algoritmaları

- Karar Ağaçları (Decision Trees)
- K-En Yakın Komşu (K-Nearest Neighbors, KNN)
- Naive Bayes
- Destek Vektör Makineleri (Support Vector Machines, SVM)
- Lojistik Regresyon (Logistic Regression)
- Random Forest
- Yapay Sinir Ağları (Artificial Neural Networks, ANN)
- Gradient Boosting (XGBoost, LightGBM, CatBoost vb.)
- k-Ensemble Yöntemleri (Bagging, Boosting, Stacking)

Karar Ağaçları Nedir?

- Akış şemasına benzeyen yapıdadır, ancak ağaç yapısının ters çevrilmiş hali olarak düşünülebilir.
- Yaygın kullanılan öngörü yöntemlerinden biridir ve hem sınıflandırma hem regresyon için kullanılabilir.
- Her bir nitelik (özellik) karar noktası olarak belirlenir.
- Ağaç yaprakları tahmin edilen sınıf etiketini veya değeri temsil eder.
- Kökten başlayarak özelliklere göre dallara ayrılır, her dal bir karar kuralını gösterir.
- Bölme işlemleri Gini katsayısı, Entropi veya Bilgi Kazancı gibi ölçütlerle yapılır.
- Yorumlaması kolaydır ve görselleştirilebilir.
- Çok derinleştiğinde overfitting riski vardır.
- Sınıfı bilinmeyen yeni örneğin özellikleri karar ağacında test edilerek sınıfı bulunur.

Karar Ağacı Çıkarımı?

Ağacı Oluşturma (Tree Construction):

- Bu aşamada veri, özelliklerine göre dallara ayrılır.
- Her düğümde, veriyi en iyi ayıran özellik seçilir.
- Bölme ölçütleri olarak genellikle Gini Katsayısı, Entropi / Bilgi Kazancı veya Kare Hatalar Toplamı (MSE) kullanılır.
- Bu işlem, tüm alt gruplar homojen bir sınıfa ulaşana veya başka ayırma yapılamayana kadar devam eder.

Budama (Pruning) veya Ağacı Basitleştirme:

- Ağacın aşırı dallanarak overfitting yapmasını önlemek için uygulanır.
- Gereksiz dallar kesilir veya bazı düğümler birleştirilir.
- Amaç, hem eğitim verisine uygun hem de genelleme kabiliyeti yüksek bir ağaç elde etmektir.

Entropi Temelleri

- Belirsizliğin ölçütüdür.
- 0-1 arasında değer alır.
- Eğer örnekelrin tamamı aynı sınıfta ise Entropi 0 olur.
- Örnekler belirlenen sınıflar arasında eşit dağılmış ise Entropi 1 olur.
- Formülü:

$$Ent(A) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Ent_B(A) = \sum_{j=1}^{v} \frac{|A_j|}{|A|} Ent(A_j)$$

$$Gain = Ent(A) - Ent_A(B)$$

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

Entropi Temelli Algoritmalar

ID3

- Genel entropi hesaplanır.
- Her özniteliğin entropisi ayrı ayrı hesaplanır.
- Hesaplanan entropi değerleri genel entropi değerinden çıkartılarak hangi özniteliğin entropi değerini en çok azalttığı belirlenir (kazanç).
- Entropiyi en çok azaltan öznitelik en çok kazanç sağlayandır.

C4.5

- C4.5 algoritması ile sayısal değerlere sahip niteliklerin de karar ağaçları oluşturulabilir.
- Sayısal değerler için en büyük bilgi kazancını sağlayacak biçimde bir eşik değeri belirlenir.
- Nitelik değerler sıralanır eşik değer ile nitelik değeri iki parçaya ayrılır.
- Eşik değeri dzinin orta noktası olabilir.

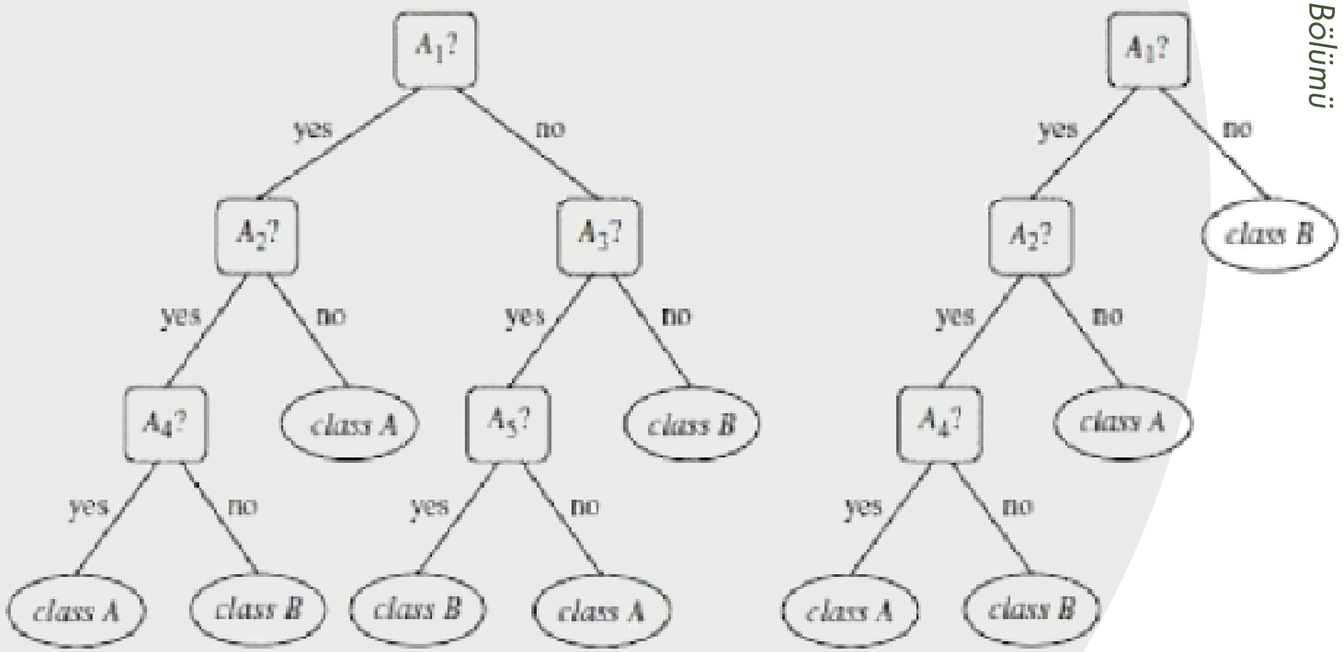
Çalışma Sorusu

ID3 algoritmasını kullanarak karar ağacı oluşturunuz?

HAVA	ISI	NEM	RÜZGAR	OYUN
güneşli	sıcak	yüksek	hafif	hayır
güneşli	sıcak	yüksek	kuvvetli	hayır
bulutlu	sıcak	yüksek	hafif	evet
yağmurlu	ılık	yüksek	hafif	evet
yağmurlu	soğuk	normal	hafif	hayır
yağmurlu	soğuk	normal	kuvvetli	evet
bulutlu	soğuk	normal	kuvvetli	hayır
güneşli	ılık	yüksek	hafif	evet
güneşli	soğuk	normal	hafif	evet
yağmurlu	ılık	normal	hafif	evet
güneşli	ılık	normal	kuvvetli	evet
bulutlu	ılık	yüksek	kuvvetli	evet
bulutlu	sıcak	normal	hafif	evet
yağmurlu	ılık	yüksek	kuvvetli	hayır

Karar Ağaçlarının Budanması

- Amaç, karmaşık olmayan ağaçlar oluşturmaktır.
- Ağacın budanması, bütün bir alt ağacın yerine bir yaprak düğümünün yerleştirilmesiyle yapılır.
- Yerleştirme ancak bir alt ağaçtaki beklenen hata tek yapraktakinden daha büyükse yapılır.
- Alt ağacın yerine yaprak yerleştirmekle, algoritma "öngörülü hata oranının" azaltmayı ve sınıflandırma modelinin kalitesini arttırmayı amaçlar.



KNN Algoritması

KNN Nedir?

KNN, sınıflandırma ve regresyon problemlerinde kullanılan basit ve etkili bir makine öğrenmesi algoritmasıdır. Algoritma, yeni bir verinin sınıfını veya değerini tahmin etmek için en yakın K komşusuna bakar.

Temel Özellikleri

- **Parametrik olmayan bir yöntemdir:** Verinin dağılımı hakkında ön bilgi gerektirmez.
- **Temel fikir:** Benzer veriler, birbirine yakın konumda bulunur.
- **K değeri:** Kaç komşunun dikkate alınacağını belirler; küçük K hassas ama gürültüye duyarlı, büyük K daha genelleştirilmiş sonuç verir.
- **Mesafe ölçümleri:** En yaygın olarak Euclidean, Manhattan veya Minkowski mesafeleri kullanılır.

KNN Algoritması

Çalışma Adımları

- Tahmin yapılacak veri noktasını belirle.
- Eğitim veri setindeki tüm noktalar ile bu veri arasındaki mesafeyi hesapla.
- En yakın K komşuyu seç.
- Sınıflandırmada çoğunluk oyuna, regresyonda ise ortalama değere bakarak tahmini yap.

Avantajları

- Basit ve kolay anlaşılır.
- Parametrik olmadığı için veri dağılımına duyarlı değildir.

Dezavantajları

- Büyük veri setlerinde yavaş çalışır.
- Özelliklerin ölçeklendirilmesine dikkat edilmezse performans düşer.

Örnek

Elimizde aşağıdaki eğitim verisi var:

Nokta	X	Y	Sınıf
A	1	2	0
B	2	3	0
C	3	3	1
D	6	5	1

Bir test noktamız var: $P = (3, 2)$

KNN algoritması ile $k = 3$ olduğunda P noktasının sınıfını belirleyelim.

Örnek

Adım 1: Uzaklıkları hesapla

KNN'de genellikle **Öklidyen mesafe** kullanılır:

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Mesafeler:

1. $d(P, A) = \sqrt{(3-1)^2 + (2-2)^2} = \sqrt{2^2 + 0^2} = \sqrt{4} = 2$
2. $d(P, B) = \sqrt{(3-2)^2 + (2-3)^2} = \sqrt{1^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} \approx 1.41$
3. $d(P, C) = \sqrt{(3-3)^2 + (2-3)^2} = \sqrt{0^2 + (-1)^2} = \sqrt{1} = 1$
4. $d(P, D) = \sqrt{(3-6)^2 + (2-5)^2} = \sqrt{(-3)^2 + (-3)^2} = \sqrt{9+9} = \sqrt{18} \approx 4.24$

Adım 2: En yakın 3 komşuyu seç

Mesafelere göre sıralama:

1. C → 1
2. B → 1.41
3. A → 2
4. D → 4.24

En yakın 3 komşu: **C, B, A**

Adım 3: Oy çokluğu ile sınıf belirleme

Komşuların sınıfları:

- C → 1
- B → 0
- A → 0

Oylama:

- Sınıf 0: 2 kişi
- Sınıf 1: 1 kişi

✅ Sonuç: P noktasının sınıfı **0**

Naive Bayes

Naive Bayes, olasılık temelli bir sınıflandırma algoritmasıdır. “Naive” yani “naif” olarak adlandırılmasının nedeni, tüm özelliklerin birbirinden bağımsız olduğu varsayımıdır. Gerçek hayatta bu varsayım her zaman doğru olmasa da, pratikte oldukça başarılı sonuçlar verebilir. Algoritma, özellikle metin sınıflandırma, spam tespiti ve duygu analizi gibi yüksek boyutlu veri setlerinde yaygın olarak kullanılmaktadır.

Naive Bayes’in en büyük avantajı, basit yapısına rağmen hızlı ve etkili sonuçlar vermesidir. Özellikle çok boyutlu veri setlerinde bile az sayıda örnekle tatmin edici performans gösterebilir. Ayrıca hesaplama maliyeti oldukça düşüktür. Bununla birlikte, özellikler arasındaki bağımsızlık varsayımı bazı durumlarda modelin doğruluğunu düşürebilir. Bu nedenle, sürekli değişkenlerde doğru türde Naive Bayes modeli (Gaussian, Multinomial, Bernoulli gibi) seçmek önemlidir.

Naive Bayes Çalışma Mantığı

1- Verileri inceleme:

- Eğitim verilerindeki her sınıfın (kategori) kaç kez geçtiğini sayar.
- Bu sayılar, gelecekteki tahminlerde temel oranları belirlemek için kullanılır.

2- Özelliklerin dağılımını çıkarma:

- Her özelliğin, her sınıfta ne kadar sık geçtiğini hesaplar.
- Örneğin: “Evet” sınıfında renk kırmızı kaç kez görülmüş, “Hayır” sınıfında kaç kez görülmüş gibi.

3- Yeni veri geldiğinde karşılaştırma:

- Yeni verinin özellikleri tek tek alınır ve her sınıfa ait olma ihtimali hesaplanır.
- Bu aşamada, daha önce elde edilen sayılar ve oranlar kullanılır.

4- Bağımsızlık varsayımı ile hesaplama:

- Özelliklerin birbirinden bağımsız olduğu kabul edilir.
- Böylece her özelliğin etkisi ayrı ayrı değerlendirilip çarpılarak birleştirilir.

5- En olası sınıfı seçme:

- Her sınıf için elde edilen sonuçlar karşılaştırılır.
- En yüksek değere sahip olan sınıf, tahmin sonucu olarak seçilir.

6- Sonucun yorumlanması:

- Tahmin edilen sınıf, yeni verinin ait olduğu kategori olarak döndürülür.

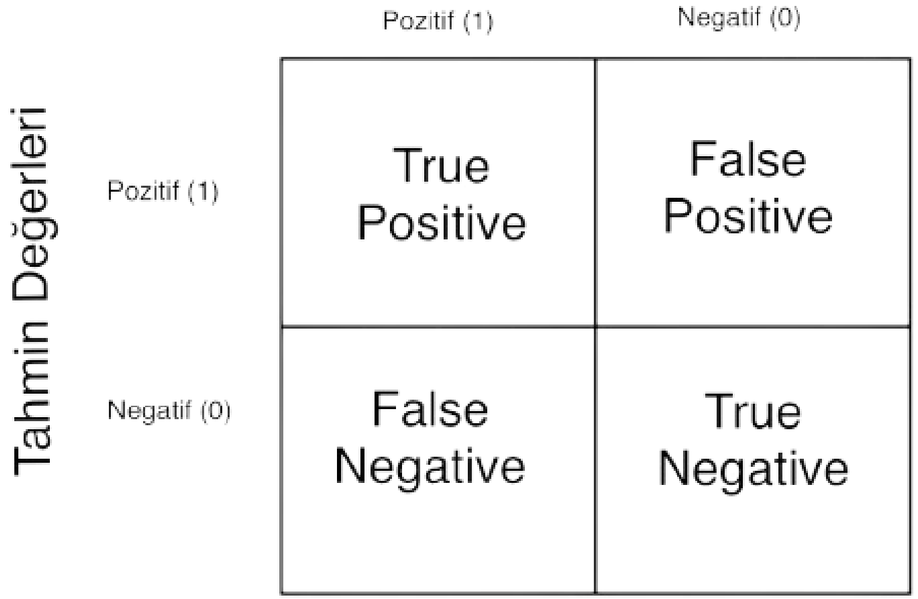
Performans Metrikleri

Modelin tahminlerinin ne kadar doğru ve güvenilir olduğunu ölçmek için kullanılır. Bu metrikler, modelin hem pozitif hem de negatif sınıflarda ne kadar başarılı olduğunu anlamamıza yardımcı olur.

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

Confusion Matrix (Karmaşıklık Matrisi)

Karmaşıklık matrisi, bir sınıflandırma modelinin performansını değerlendirmek için kullanılan bir tablodur. Gerçek pozitifler, negatifler, yanlış pozitifler ve negatifler gibi farklı hata türlerin gösterirler.



- **TP (True Positive):** Gerçekten pozitif olanı model pozitif tahmin etti.
- **TN (True Negative):** Gerçekten negatif olanı model negatif tahmin etti.
- **FP (False Positive):** Gerçek negatifken model pozitif tahmin etti (yalancı alarm).
- **FN (False Negative):** Gerçek pozitifken model negatif tahmin etti (kaçırma).

Örnek

100 e-posta üzerinden spam tespiti yapıyoruz. Modelin tahminleri ve gerçek değerler şöyle olsun:

- Gerçek spam olan e-postalar: 55
- Gerçek spam olmayan e-postalar: 45
- Modelin tahminleri:
  - TP = 50 (gerçek spam, model spam dedi)
  - FN = 5 (gerçek spam, model spam demedi)
  - FP = 10 (gerçek değil ama model spam dedi)
  - TN = 35 (gerçek değil ve model de spam demedi)

2×2 Confusion Matrix

	Tahmin: Spam	Tahmin: Spam Değil
Gerçek: Spam	TP = 50	FN = 5
Gerçek: Spam Değil	FP = 10	TN = 35

Doğruluk (Accuracy)

Doğruluk, sınıflandırma modellerinde en temel performans ölçütlerinden biridir.
Formül olarak "doğru tahmin sayısı / toplam tahmin sayısı" şeklinde hesaplanır.
Yani, modelin tüm veri üzerinde ne kadar doğru tahmin yaptığını gösterir.
Örnek:
Eğer 100 örnekten 90'ını doğru tahmin etmişsen, doğruluğun %90'dır.

Avantajları:

- Anlaşılması ve hesaplanması çok kolaydır.
- Dengesiz olmayan veri setlerinde güvenilir sonuçlar verir.
- Genel performans hakkında hızlı fikir verir.

Dezavantajları:

- Dengesiz veri setlerinde yanıltıcı olabilir. Örneğin, bir veri setinde %95'i "olumsuz" sınıfa aitse, model hep "olumsuz" derse doğruluk %95 olur ama aslında model iyi çalışmıyordur.

Duyarlılık / Kesinlik (Precision)

Modelin pozitif tahmin yaptığı durumlarda, bu tahminlerin ne kadarının gerçekte doğru (pozitif) olduğunu gösterir. Precision yüksekse, model pozitif sınıfı tahmin ederken çoğunlukla haklı çıkıyor.
Örneğin bir hastalık testinde, gerçekten hasta olan kişilerin ne kadarının test tarafından doğru şekilde tespit edildiğini gösterir.

Geri Çağırma (Recall- Sensitivity)

Gerçek pozitiflerin içinden, modelin pozitif olarak yakalayabildiği paydır. Recall yüksekse, gerçekte pozitif olanları kaçırmıyor (FN düşük).
Örneğin sahte banknot tespitinde, gerçekten sahte olan banknotların ne kadarının sistem tarafından sahte olarak sınıflandırıldığını gösterir.

F1 Skoru

F1 skoru, Precision (kesinlik) ve Recall (duyarlılık) değerlerinin harmonik ortalamasıdır.

Amaç, yanlış pozitif (FP) ve yanlış negatifleri (FN) dengeli bir şekilde değerlendirmektir.

Neden önemli?

- Sadece doğruluk oranına bakmak yanıltıcı olabilir. F1, özellikle dengesiz veri setlerinde daha adil bir ölçüm sağlar.
- Mesela kanser tespiti gibi bir durumda "yanlış negatif" çok tehlikelidir, bu yüzden recall ve precision birlikte değerlendirilir.

Avantajları:

- Yanlış sınıflandırmaları göz önünde bulundurur.
- Hem precision hem recall'u tek bir değerde toplar.

Dezavantajları:

- Precision veya recall'dan birinin çok düşük olması F1 skorunu da düşürür.

$$\text{Accuracy (Acc)} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Sensitivity (Se)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{Specificty (Sp)} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

$$\text{Precision (Prec)} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Fscore (F} - \text{Sc)} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

Örnek

Confusion Matrix

	Frenquette	Bilecik	Chandler	Fernette	Fernor	Hardley	Howard	Kaman1	Kaplan86	Lara	Maya1	Mithland	Oguzlar77	Pedro	Sebin	Sen	Serr	Yalova3
renquette	457	0	4	2	0	0	0	2	1	0	0	15	0	0	0	3	4	0
Bilecik	2	355	1	0	0	0	0	0	0	0	0	0	0	4	0	6	3	7
Chandler	1	0	307	2	2	1	0	3	0	3	2	0	0	0	0	1	1	1
Fernette	4	1	1	324	6	1	0	1	2	6	0	1	0	2	0	0	2	0
Fernor	1	0	3	8	366	0	7	8	1	7	1	3	0	0	0	0	0	1
Hardley	0	6	10	1	1	342	0	0	1	0	0	1	0	7	2	2	1	1
Howard	0	1	1	0	12	0	277	19	4	13	3	0	0	1	0	0	1	0
Kaman1	0	0	2	6	7	1	7	350	7	5	1	0	0	0	0	1	1	1
Kaplan86	0	0	2	1	0	0	1	3	173	2	147	0	0	0	0	0	1	0
Lara	0	0	4	9	8	0	5	5	5	202	8	0	0	0	0	0	0	3
Maya1	0	0	2	0	1	0	0	1	169	0	117	0	0	0	0	0	0	1
Mithland	10	0	2	1	1	0	0	0	0	0	0	393	0	0	0	2	9	0
Oguzlar77	0	1	0	0	0	1	0	0	0	1	1	1	211	0	3	7	1	5
Pedro	0	2	3	3	2	0	1	0	0	1	6	1	0	280	1	2	0	1
Sebin	0	1	0	0	0	4	0	0	0	0	0	0	6	2	323	4	3	1
Sen	1	8	0	0	0	0	0	0	0	0	0	5	0	0	0	505	0	9
Serr	2	10	0	0	1	1	0	0	0	0	1	24	0	1	2	2	394	3
Yalova3	0	7	0	0	0	0	0	0	0	0	0	0	1	0	0	14	6	399

TP - FN - FP - TN bulunması

	Tahmin: Pozitif	Tahmin: Negatif
Gerçek: Pozitif	TP = 50	FN = 5
Gerçek: Negatif	FP = 10	TN = 35

- TP (True Positive) = 50
- FN (False Negative) = 5
- FP (False Positive) = 10
- TN (True Negative) = 35

Bu matrisin satırlarında verinin **gerçek etiketi**, sütunlarında ise **modelin tahmini** bulunur.

> **Toplam örnek sayısı** = TP + FN + FP + TN = 50 + 5 + 10 + 35 = 100

Örnek

Doğruluk

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Accuracy} = \frac{50 + 35}{50 + 35 + 10 + 5} = \frac{85}{100} = 0.85$$

Duyarlılık

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Precision} = \frac{50}{50 + 10} = \frac{50}{60} = 0.83\overline{3}$$

Geri Çağırma

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{Recall} = \frac{50}{50 + 5} = \frac{50}{55} = 0.90\overline{90}$$

Sınıflandırma Başarısının Değerlendirilmesi

ROC (Receiver Operating Characteristic) Eğrisi

- ROC, sınıflandırma modelinin farklı eşik (threshold) değerlerinde doğru pozitif oranı (TPR) ile yanlış pozitif oranı (FPR) arasındaki ilişkiyi gösteren bir grafiktir.
- X ekseni: Yanlış Pozitif Oranı (FPR) → Yanlış alarm verme olasılığı
- Y ekseni: Doğru Pozitif Oranı (TPR) → Gerçek pozitifleri doğru yakalama oranı
- ROC'un amacı:
- Farklı karar sınırları (0.1, 0.5, 0.9 gibi) denendiğinde modelin performansının nasıl değiştiğini görselleştirmek.
- Eğri ne kadar sol üst köşeye yakınsa, model o kadar iyidir.

AUC (Area Under Curve)

- AUC, ROC eğrisinin altında kalan alanın büyüklüğünü ifade eden bir sayıdır.
- Yani ROC eğrisini özetleyerek tek bir 0 ile 1 arasında sayı verir.
- AUC = 1: Mükemmel model (tüm pozitif ve negatifleri %100 ayırt eder)
- AUC = 0.5: Rastgele tahmin yapan model
- AUC < 0.5: Yanlış tahmin yapma eğiliminde model (sınıfları ters çevirmek gerekir)
- Avantajı: ROC grafiğine tek bakışta karar vermek zor olabilir ama AUC sayısal bir değer vererek karşılaştırmayı kolaylaştırır.

UYGULAMA

KNN Modeli

Kod Parçası

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# 1. Veri setini yükle
iris = load_iris()
X = iris.data
y = iris.target

# 2. Eğitim ve test verisine ayır
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 3. KNN Modeli
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

print("KNN Modeli")
print("Doğruluk:", accuracy_score(y_test, y_pred_knn))
print(classification_report(y_test, y_pred_knn))
y_pred_dt = dt.predict(X_test)
```

Kod Çıktısı

```
KNN Modeli
Doğruluk: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

UYGULAMA

Karar Ağacı

Kod Parçası

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# 1. Veri setini yükle
iris = load_iris()
X = iris.data
y = iris.target

# 2. Eğitim ve test verisine ayır
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# 3. KNN Modeli
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred_knn = knn.predict(X_test)

print("\nKarar Ağacı Modeli")
print("Doğruluk:", accuracy_score(y_test, y_pred_dt))
print(classification_report(y_test, y_pred_dt))
```

Kod Çıktısı

```
Karar Ağacı Modeli
Doğruluk: 1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00        13

    accuracy                           1.00        45
   macro avg       1.00      1.00      1.00        45
weighted avg       1.00      1.00      1.00        45
```

Samsun Üniversitesi
Yazılım Mühendisliği Bölümü

VERİ MADENCİLİĞİ (DATA MİNİNG)

Dr. Öğr. Üyesi Alper Talha Karadeniz

2025-2026