ERCİYES ÜNİVERSİTESİ

MÜHENDİSLİK FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROGRAMMING LABORATORY NOTLARI

10. HAFTA NOTU

(Yapay Zeka ve Makine Öğrenmesi Uygulaması)

Bu not dersin 10. haftasında yapacağınız uygulamaya hazırlanmanızı sağlayacak olup temel kavramları anlatan bilgileri içermektedir.

DERSİN ÖĞRETİM ÜYESİ

Prof. Dr. Bahriye AKAY

2026

KAYSERİ

1. YAPAY ZEKA VE MAKİNE ÖĞRENMESİ HAKKINDA TEMEL BİLGİLER

1.1. Klasik Programlama ve Makine Öğrenmesi

Yapay zeka, bilgisayarın insan zekası gerektiren işleri (algılama, akıl yürütme, öğrenme, karar verme) yerine getirmesini sağlayan tekniklerin geneline verilen addır. Spam filtreleme, yüz tanıma, makine çevirisi, tıbbi görüntü analizi gibi uygulamalar günlük hayatta sıkça karşılaşılan örneklerdir. Bu deneyin konusu olan makine öğrenmesi de yapay zekanın en aktif alt dallarından biridir.

Klasik programlamada problemi çözecek kuralları yazılımcı tanımlar: verilen girdiye, önceden yazılmış if-else blokları, döngüler ve hesaplamalar uygulanır ve bir çıktı üretilir. Bu yaklaşımda programın yetenek sınırı, yazılımcının düşünebildiği senaryo sayısı kadardır. Makine öğrenmesinde ise yaklaşım terstir: bilgisayara çok sayıda örnek ile bu örneklere ait doğru cevaplar verilir, ondan da kuralları kendisinin çıkarması beklenir. Kural çıkarma sürecine eğitim, eğitim sonunda elde edilen yapıya ise model denir. Bu sayede, yazılımcının açıkça düşünüp kodlayamayacağı kadar karmaşık örüntüler de öğrenilebilir.

> **Klasik Programlama**
> Veri + Kurallar → Cevap
> Kuralları yazılımcı yazar.
> Karmaşıklık yazılımcının düşünebildiği kadardır.

> **Makine Öğrenmesi**
> Veri + Cevap → Kurallar (Model)
> Kuralları model verilerden öğrenir.
> Karmaşıklık veri çeşitliliği ile artar.

1.2. Makine Öğrenmesinin Temel Türleri

Makine öğrenmesi, kullanılan veri ve öğrenme yaklaşımına göre üç ana kategoriye ayrılır. Denetimli öğrenme türünde modele hem girdi örnekleri hem de bu örneklere ait doğru cevaplar verilir; model, girdi-cevap eşleşmelerinden kuralları çıkarır. Bu deneydeki uygulama da bu kategoriye girmektedir. Denetimsiz öğrenmede veri etiketsizdir; modelin görevi, veri içerisindeki gizli desenleri (kümelenmeler, anomaliler vb.) ortaya çıkarmaktır. Pekiştirmeli öğrenmede ise model bir ortam ile etkileşime girerek deneme-yanılma yoluyla, aldığı ödül ve cezalardan öğrenir; oyun yapay zekaları ve robotik kontrol uygulamaları bu kategoride yer alır.

1.3. Sınıflandırma ve Regresyon

Denetimli öğrenmenin iki temel görevi vardır: sınıflandırma ve regresyon. Sınıflandırmada amaç, bir örneği önceden tanımlı sonlu sayıdaki kategoriden birine atamaktır. "Bu e-posta spam mı, değil mi?" sorusu iki sınıflı (binary), "görüntüdeki rakam hangisi?" sorusu ise on sınıflı çok-sınıflı (multi-class) bir sınıflandırma problemidir. Regresyonda ise sürekli bir sayısal değer tahmin edilir: ev fiyatı, hava sıcaklığı, kullanıcının izleyeceği film sayısı bu tür problemlere örnek verilebilir. Bu deneyde sınıflandırma türünden bir problem ele alınacaktır.

1.4. Öznitelik ve Etiket

Bir modelin eğitilebilmesi için elimizde örnek-cevap çiftlerinden oluşan bir veri seti bulunmalıdır. Her örnek iki bileşenden meydana gelir: modele girdi olarak verilen sayısal değerler — yani öznitelikler (features) — ve modelin tahmin etmeye çalışacağı doğru cevap, yani etiket (label). Konvansiyon gereği öznitelikler büyük X harfiyle gösterilir, çünkü genellikle iki boyutlu bir matristir (her satır bir örnek, her sütun bir öznitelik). Etiketler ise küçük y harfiyle gösterilir; her örneğin tek bir etiketi olduğundan y bir vektördür.

Görüntü tabanlı bir sınıflandırma probleminde X çoğunlukla piksel değerlerinden oluşan bir matris, y ise her görüntünün gerçek sınıfını gösteren bir vektördür. Eğitim sonunda modelin, daha önce görmediği bir X örneği için doğru y'yi tahmin edebilmesi beklenir.

1.5. Eğitim ve Test Veri Ayrımı

Bir modelin başarısı, daha önce hiç görmediği veriler üzerindeki performansıyla ölçülmelidir. Eğitilirken kullanılan örneklerle aynı örnekler üzerinde sınanan bir model, kuralları öğrenmek yerine ezberlemiş olabilir; bu durumda elde edilen başarı yanıltıcıdır. Sınava aynı soruyu sokmak buna iyi bir benzetmedir. Bu nedenle veri seti iki alt kümeye ayrılır: eğitim seti modelin parametrelerini öğrendiği veridir ve genellikle veri setinin %70-%80'ini kaplar; test seti ise yalnızca son değerlendirme için kullanılan, eğitim sırasında modele kesinlikle gösterilmeyen veridir ve geri kalan kısmı oluşturur.

Bu ayrım yapılırken, sonuçların tekrarlanabilir olması için bir tohum (random_state) sabitlenir; aynı tohum aynı bölünmeyi garanti eder. Sınıflandırma problemlerinde her sınıftan dengeli sayıda örnek alınması (stratified split) da yaygın bir uygulamadır.

> Test verisi, eğitim sürecinin hiçbir aşamasında modele gösterilmemelidir. Test verisinin eğitime sızması (data leakage) modelin gerçek dışı yüksek başarı göstermesine, ardından gerçek kullanım koşullarında başarısız olmasına yol açar. Bu nedenle veri ayrımı işin ilk adımı olmalı, test verisine yalnızca son değerlendirmede dokunulmalıdır.

1.6. Modelin Değerlendirilmesi

Bir sınıflandırma modelinin başarısını ölçmek için en yaygın iki ölçüt accuracy (doğruluk) ve confusion matrix'tir (karışıklık matrisi). Accuracy basitçe doğru tahmin sayısının toplam tahmin sayısına oranıdır; örneğin 360 test örneğinin 320'sini doğru tahmin eden bir modelin accuracy'si yaklaşık %88,9'dur. Hesaplaması kolaydır ve sezgiseldir, fakat sınıflar dengesiz olduğunda yanıltıcı sonuçlar verebilir.

> **Önemli Not**
>
> Sınıf dağılımı dengesizse — örneğin %95 normal, %5 hastalık örneği içeren bir veri setinde — hiçbir şey yapmadan tüm örnekleri çoğunluk sınıfa atayan bir model bile %95 accuracy elde eder. Bu yüzden accuracy tek başına değil, confusion matrix ile birlikte yorumlanmalıdır.

Confusion matrix, modelin hangi sınıfı hangisiyle karıştırdığını gösteren iki boyutlu bir tablodur ve N sınıflı bir problemde N×N boyutundadır. Satırlar gerçek sınıfları, sütunlar tahmin edilen sınıfları gösterir. Köşegen üzerindeki değerler

doğru tahminleri, köşegen dışındaki değerler ise hangi sınıfın hangi sınıfla karıştırıldığını ortaya koyar. Bu görünürlük özellikle birbirine yapısal veya görsel olarak benzeyen sınıfların analizinde değerlidir; toplam doğruluk korunsa bile zayıf noktaların nerede olduğunu görmeyi sağlar.

1.7. Tekrarlanabilirlik

Makine öğrenmesi süreçlerinin pek çok adımında rastgelelik vardır: veri setinin bölünmesi, başlangıç parametrelerinin atanması, bazı algoritmaların iç işleyişi gibi. Bu durum, aynı kodun iki farklı çalıştırmasında farklı sonuçlar üretmesine yol açabilir. Bilimsel bir deneyin gözlemlenebilir ve doğrulanabilir olabilmesi için sonuçların tekrarlanabilir olması gereklidir. scikit-learn'de rastgele süreçler kullanılırken random_state parametresine sabit bir değer atanması, aynı tohumun aynı sonuçları üretmesini sağlar. Bu deneyde de tüm rastgele bileşenler için sabit bir random_state kullanılması beklenmektedir.

1.8. İş Akışının Özeti

Bu bölümde anlatılan kavramlar, denetimli sınıflandırma için tipik iş akışını oluşturur. Bu akış aşağıdaki adımlardan oluşur ve deneyde aynı sıra ile takip edilecektir.

Adım	Açıklama
1. Yükle	Veri seti yüklenir; X ve y elde edilir.
2. Ayır	Eğitim ve test alt kümelerine ayrılır; random_state sabitlenir.
3. Eğit	Seçilen model eğitim verisi üzerinde eğitilir (fit).
4. Tahmin Et	Eğitilmiş model test verisi üzerinde tahmin yapar (predict).
5. Değerlendir	Tahminler accuracy ve confusion matrix ile karşılaştırılır.
6. Yorumla	Hangi sınıflar doğru, hangileri karıştırılmış incelenir.

2. DENEYDE KULLANILACAK KÜTÜPHANELER VE FONKSİYONLAR HAKKINDA KISA BİLGİ

Deney, konsol uygulaması yerine Jupyter Notebook ortamında ve Python dilinde, bilimsel hesaplama için geliştirilmiş bir dizi kütüphane kullanılarak yürütülür. Aşağıda deney boyunca kullanılacak temel kütüphaneler ve önemli yapılar tanıtılmaktadır.

2.1. NumPy

NumPy (Numerical Python), Python'da sayısal hesaplamanın temel kütüphanesidir. C/C++ dizilerine benzer şekilde sabit bellek alanı tutan, hızlı ve verimli n-boyutlu dizi yapısı (ndarray) sunar. Makine öğrenmesi uygulamalarında veri seti, özellik vektörleri ve etiketler hemen her zaman NumPy dizileri olarak temsil edilir.

Klasik Python listeleri yerine NumPy dizilerinin tercih edilmesinin iki önemli sebebi vardır: matris işlemleri (toplama, çarpma, dilimleme) çok daha hızlı yürütülür ve scikit-learn gibi kütüphaneler girdi olarak doğrudan ndarray bekler. ndarray üzerinde sıkça kullanılan işlemler arasında dizinin boyutunu döndüren

shape niteliği, diziyi farklı şekle dönüştüren reshape ve bir dizideki benzersiz değerleri çıkaran unique fonksiyonu sayılabilir.

2.2. Matplotlib

Matplotlib, Python'un en yaygın görselleştirme kütüphanesidir; bu deneyde matplotlib.pyplot modülü kullanılır. Veri setindeki örnek görüntüleri ekrana çizmek, eğitim sonuçlarını grafikle göstermek ve confusion matrix çıktısını görselleştirmek için tercih edilir. subplots fonksiyonu bir veya birden çok grafik için figür ve eksen oluşturur, imshow iki boyutlu bir matrisi (örneğin bir piksel görüntüsünü) renk haritasıyla çizer, show ise hazırlanan grafiği ekrana getirir.

2.3. scikit-learn

scikit-learn (kısaca sklearn), Python ekosisteminin en yaygın makine öğrenmesi kütüphanesidir. Veri seti yükleme, modeli eğitme, tahmin yapma ve sonuçları değerlendirme adımlarının tamamı bu kütüphane üzerinden yürütülür. Kütüphanenin en güçlü yanı, tüm sınıflandırıcı modellerin tutarlı bir arayüze sahip olmasıdır: model nesnesi oluşturulur, fit(X, y) ile eğitilir, predict(X) ile tahmin yaptırılır. Bu tutarlılık, farklı modeller arasında geçişi yalnızca model nesnesinin değiştirilmesi kadar basit hale getirir.

Bu deneyde scikit-learn'in birkaç farklı modülü birlikte kullanılacaktır. sklearn.datasets modülü hazır veri setlerini sağlar; sklearn.model_selection eğitim/test ayrımı için train_test_split fonksiyonunu içerir; sklearn.metrics performans değerlendirmesi için accuracy_score, confusion_matrix ve ConfusionMatrixDisplay yardımcı yapılarını sunar. Kullanılacak somut sınıflandırma modeli ise scikit-learn'in lineer modeller, ağaç tabanlı modeller, komşuluk tabanlı modeller veya destek vektör makineleri gibi farklı modüllerinden biri olabilir; hangisi olduğu derste duyurulacaktır.

2.4. Jupyter Notebook

Jupyter Notebook, Python kodunun "hücre" denilen bağımsız bloklar halinde çalıştırıldığı etkileşimli bir geliştirme ortamıdır. Her hücre tek başına çalıştırılabilir; yazdırılan değerler veya çizilen grafikler hücrenin hemen altında görünür. Bu yapı, makine öğrenmesi gibi adım adım deneyleme yapılan alanlarda kullanışlıdır çünkü veri yükleme, model eğitimi ve sonuç analizi gibi aşamalar ayrı ayrı çalıştırılarak sonuçlar anında gözlemlenebilir.

İki temel hücre tipi vardır: kod hücreleri Python kodu içerir ve Shift+Enter ile çalıştırılır, markdown hücreleri ise biçimlendirilmiş yazı (başlık, liste, açıklama) yazmak için kullanılır. Lab ortamında Jupyter kurulu değilse pip install jupyter komutu ile kurulabilir; alternatif olarak Google Colab, tarayıcı üzerinden hiçbir kurulum gerektirmeden kullanım imkânı sunar.

3. DERSİN ÖĞRENME ÇIKTILARI

Bu deney çalışmasının sonunda aşağıdaki kazanımların elde edilmesi beklenmektedir:

- Yapay zeka ve makine öğrenmesi kavramlarını birbiriyle ilişkilendirebilmek,
- Klasik programlama ile veri-temelli öğrenmeli yaklaşım arasındaki farkı açıklayabilmek,
- Denetimli öğrenmenin mantığını ve sınıflandırma görevini tanıyabilmek,
- Bir veri setini öznitelik (X) ve etiket (y) bileşenleriyle temsil edebilmek,

- Eğitim ve test ayrımının neden gerekli olduğunu ve random_state ile nasıl tekrarlanabilir hâle getirildiğini bilmek,
- scikit-learn'in fit/predict arayüzünü kullanabilmek,
- Jupyter Notebook ortamında hücre tabanlı bir deneyi adım adım yürütebilmek,
- Accuracy ve confusion matrix kullanarak sınıflandırma modelinin performansını değerlendirebilmek,
- Confusion matrix üzerinden en çok karıştırılan sınıfları tespit edebilmek ve sonuçları yorumlayabilmek.