Machine Learning - Fundamentals

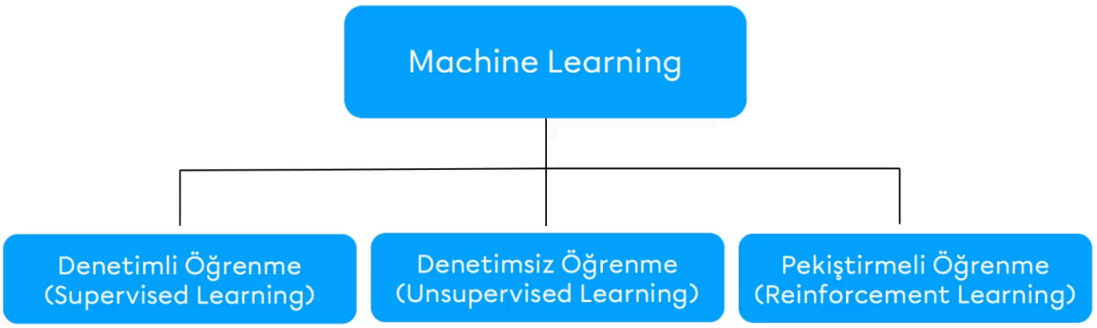
- Temel Kavramlar
- Doğrusal Regresyon
- Lojistik Regresyon
- KNN
- CART
- Gelişmiş Ağaç Yöntemleri
- Denetimsiz Öğrenme
- Yapay Öğrenme Pipeline

Variable Types:

- Sayısal Degişkenler
- Kategorik Degişkenler (Nominal, Ordinal)
- Bağimli Degişken (target, dependent, output, response
- Bağımsız Degişken (feature, independent input, column, predictor, explanatory)

TARGET	NAME_CONTRACT_TYPE	FLAG_OWN_REALTY	AMT_INCOME_TOTAL
1	Cash loans	Y	202500.0
0	Cash loans	N	270000.0
0	Revolving loans	Y	67500.0
1	Cash loans	Y	135000.0
0	Cash loans	Y	121500.0
1	Cash loans	Y	99000.0
1	Cash loans	Y	171000.0
0	Cash loans	Y	360000.0
1	Cash loans	Y	112500.0
0	Revolving loans	Y	135000.0

Öğrenme Türleri (Learning Types)



Özellik / Öğrenme Tipi	**Denetimli Öğrenme (Supervised Learning)**	**Denetimsiz Öğrenme (Unsupervised Learning)**	**Yarı Denetimli Öğrenme (Semi-Supervised Learning)**	**Pekiştirmeli Öğrenme (Reinforcement Learning)**
Giriş Verisi	Etiketli veriler (giriş ve karşılık gelen çıkış)	Etiketsiz veriler (sadece giriş)	Hem etiketli hem de etiketsiz veriler (genellikle küçük etiketli, büyük etiketsiz set)	Çevre, eylemler ve ödül/ceza sinyalleri
Amaç	Giriş verisinden çıkış etiketini tahmin etme/sınıflandırma	Veri içindeki gizli yapıları/desenleri keşfetme	Kısıtlı etiketli veriyle bile daha iyi tahminler yapma	Maksimum kümülatif ödülü elde etmek için en uygun eylem dizisini öğrenme
Temel Yaklaşım	Giriş-çıkış eşleşmeleri arasında bir fonksiyon öğrenme	Veri noktaları arasındaki benzerlikleri/farklılıkları bulma	Etiketli veriyi kullanarak etiketsiz veriyi tahmin etme ve bu tahminleri öğrenmede kullanma	Deneme yanılma yoluyla öğrenme; eylem-ödül döngüsü
Algoritma Örnekleri	Lineer/Lojistik Regresyon, SVM, Karar Ağaçları, Rastgele Orman, Sinir Ağları (Classification/Regression)	K-Means, Hiyerarşik Kümeleme, PCA, Anomali Tespiti (Isolation Forest, LOF)	Genellikle Denetimli/Denetimsiz algoritmaların birleşimi (Co-training, Self-training)	Q-Learning, SARSA, Derin Q Ağları (DQN), Politik Gradyan Metotları
Uygulama Alanları	Spam tespiti, görüntü sınıflandırma, fiyat tahmini, hastalık teşhisi, müşteri churn tahmini	Müşteri segmentasyonu, anomali tespiti, boyut azaltma, tavsiye sistemleri (ön-işleme)	Web sayfaları sınıflandırma (az etiketli), tıbbi görüntü analizi, konuşma tanıma	Otonom sürüş, robotik, oyun oynama (AlphaGo, Atari), kaynak yönetimi

Avantajları	Yüksek doğruluk potansiyeli, iyi tanımlanmış görevler için uygun	Etiketleme maliyetinden kaçınma, yeni desenleri keşfetme	Etiketli veri kıtlığına rağmen iyi performans, öğrenme sürecini hızlandırma	Karmaşık ortamlarda en uygun davranışları öğrenebilme, insan müdahalesi olmadan öğrenme
Dezavantajları	Etiketli veri toplama maliyetli ve zaman alıcı, etiketleme hatalarına duyarlı	Keşfedilen yapıların yorumlanması zor olabilir, kesin bir hedefi yoktur, sonuçlar her zaman "doğru" değildir	Etiketli ve etiketsiz veri arasındaki dengesizlik sorun yaratabilir, yanlış etiketler modelin performansını düşürebilir	Çok fazla deneme yanılma gerektirebilir, ödül fonksiyonu tasarımı zor olabilir, "kara kutu" doğası
Veri Ölçeği	Küçükten büyüğe değişebilir, ancak genellikle yeterli etiketli veriye ihtiyaç duyar	Genellikle büyük veri setleri ile kullanılır, yapıları ortaya çıkarmak için daha fazla veriye ihtiyaç duyar	Çok büyük etiketsiz veri setleri ve küçük etiketli veri setleri ile idealdir	Genellikle simülasyon ortamlarında başlar, sonra gerçek dünyaya uygulanır

Problem Türleri (Problem Types)

Sınıflandırma problemlerinde bağımlı değişken kategorik

churn

Age	Total_Purchase	Years	Churn
42.0	11066.8	7.22	1
41.0	11916.22	6.5	1
38.0	12884.75	6.67	0
42.0	8010.76	6.71	1
37.0	9191.58	5.56	0
48.0	10356.02	5.12	1
44.0	11331.58	5.23	0
32.0	9885.12	6.92	0
43.0	14062.6	5.46	1
40.0	8066.94	7.11	1

Regresyon problemlerinde bağımlı değişken sayısal

advertising

TV	radio	newspaper	sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	9.3
151.5	41.3	58.5	18.5
180.8	10.8	58.4	12.9
8.7	48.9	75.0	7.2
57.5	32.8	23.5	11.8
120.2	19.6	11.6	13.2
8.6	2.1	1.0	4.8
199.8	2.6	21.2	10.6

Model Başarı Değerlendirme Yöntemleri

advertising

TV	radio	newspaper	sales	predicted_sales
230.1	37.8	69.2	22.1	20
44.5	39.3	45.1	10.4	11
17.2	45.9	69.3	9.3	9
151.5	41.3	58.5	18.5	17
180.8	10.8	58.4	12.9	12
8.7	48.9	75.0	7.2	7
57.5	32.8	23.5	11.8	11
120.2	19.6	11.6	13.2	14
8.6	2.1	1.0	4.8	4.5

advertising

TV	radio	newspaper	sales	predicted_sales
230.1	37.8	69.2	22.1	20
44.5	39.3	45.1	10.4	11
17.2	45.9	69.3	9.3	9
151.5	41.3	58.5	18.5	17
180.8	10.8	58.4	12.9	12
8.7	48.9	75.0	7.2	7
57.5	32.8	23.5	11.8	11
120.2	19.6	11.6	13.2	14
8.6	2.1	1.0	4.8	4.5

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Model Doğrulama Yöntemleri (Model Validation)

- Holdout Yöntemi

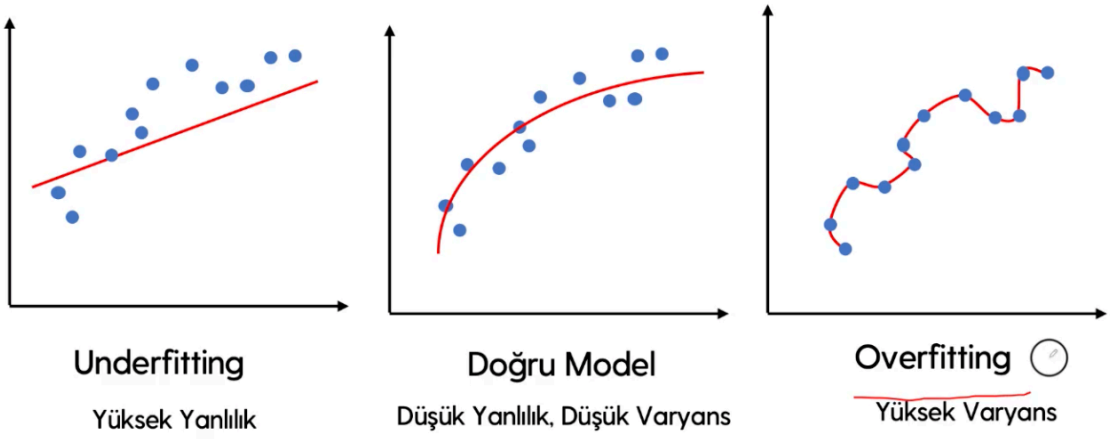
- K fold Cross Validation

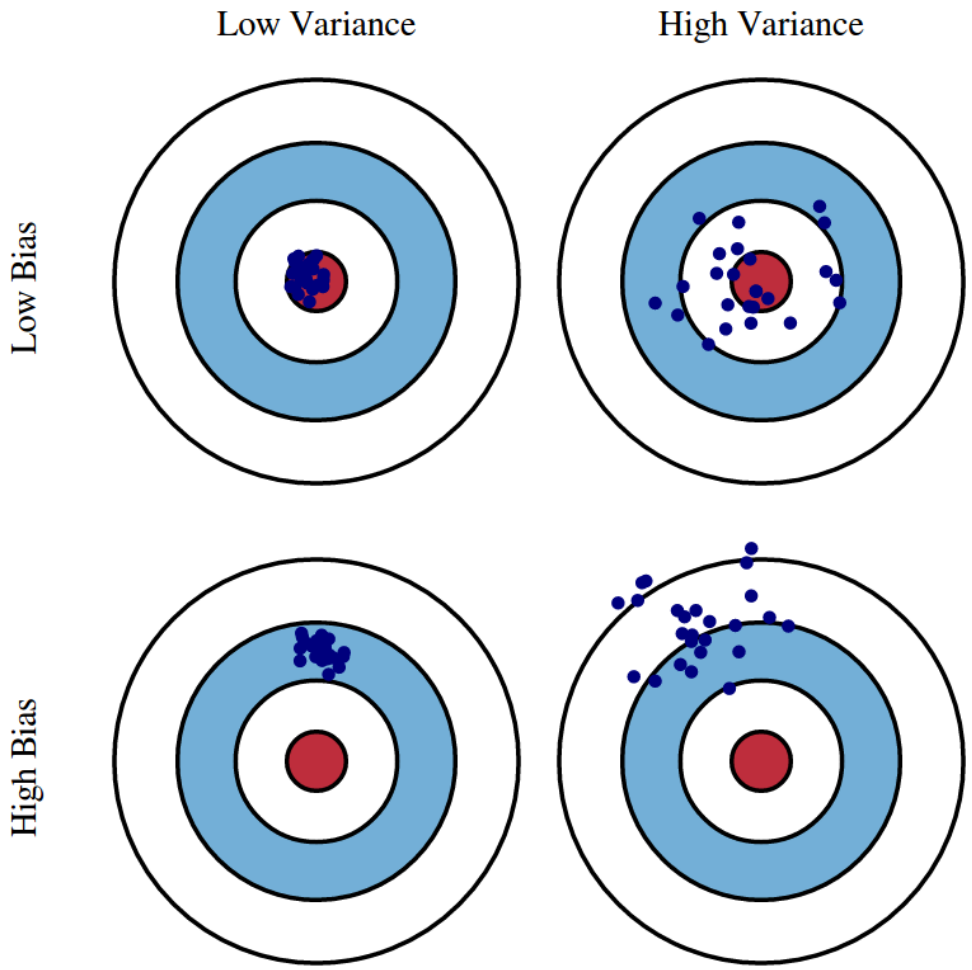
  Nasıl Çalışır?

  1. Veri setiniz **K eşit (veya yaklaşık eşit) parçaya** (katmanlara/folds) bölünür.
  2. Bu K parçadan **biri test seti** olarak ayrılır.
  3. Kalan **K-1 parça, eğitim seti** olarak kullanılır ve model bu veriyle eğitilir.
  4. Eğitilen model, ayrılan test seti üzerinde **değerlendirilir** ve bir performans metrik değeri (örn. doğruluk, hata oranı, F1-skoru) kaydedilir.
  5. Bu süreç, **K kez tekrarlanır**, her seferinde farklı bir parça test seti olarak kullanılır.
  6. Son olarak, K adet performans metrik değeri toplanır ve bu değerlerin **ortalaması alınarak** modelin genel performansı hakkında daha güvenilir bir tahmin elde edilir. Standart sapma da, modelin performansının katmanlar arasındaki tutarlılığını gösterir.

Özellik	**Holdout Yöntemi (Eğitim-Test Bölmesi)**	**K-Fold Cross-Validation**
Temel Yaklaşım	Veri seti bir kez eğitim ve test setine bölünür.	Veri seti K parçaya bölünür, K kez eğitim-test döngüsü yapılır.
Performans Tahmini	Tek bir tahmin; veri bölmesine duyarlı olabilir.	K adet tahminin ortalaması; daha istikrarlı ve güvenilir.
Veri Kullanımı	Test seti hiç eğitimde kullanılmaz; eğitim seti hiç testte kullanılmaz. Veri israfı olabilir.	Verinin her parçası hem eğitim hem de test için kullanılır; veri kullanım verimliliği yüksektir.
Aşırı Uyum Tespiti	Yetersiz; model eğitim setine uyum sağladığında test setinde kötü performans gösterir, ancak bu sadece bir tek testle sınırlıdır.	Daha iyi; her fold'da test seti değiştiği için aşırı uyumun belirtileri (yüksek varyanslı performans) daha net gözlemlenebilir.
Hesaplama Maliyeti	Düşük; model sadece bir kez eğitilir ve test edilir.	Yüksek; model K kez eğitilir ve test edilir.
Kullanım Senaryosu	Çok büyük veri setleri, hızlı model değerlendirmesi gerektiğinde veya kaynak kısıtlı olduğunda başlangıç aşaması için.	Küçük/orta boyutlu veri setleri, güvenilir performans tahmini ve model seçimi/hiperparametre ayarı için.
Varyans (Model Performansı)	Yüksek varyansa sahip olabilir; farklı rastgele bölmeler farklı sonuçlar verebilir.	Daha düşük varyans; K farklı test setinin ortalaması alındığı için daha stabil sonuçlar verir.

Yanlılık-Varyans Değiş Tokuşu (Bias-Variance Tradeoff)

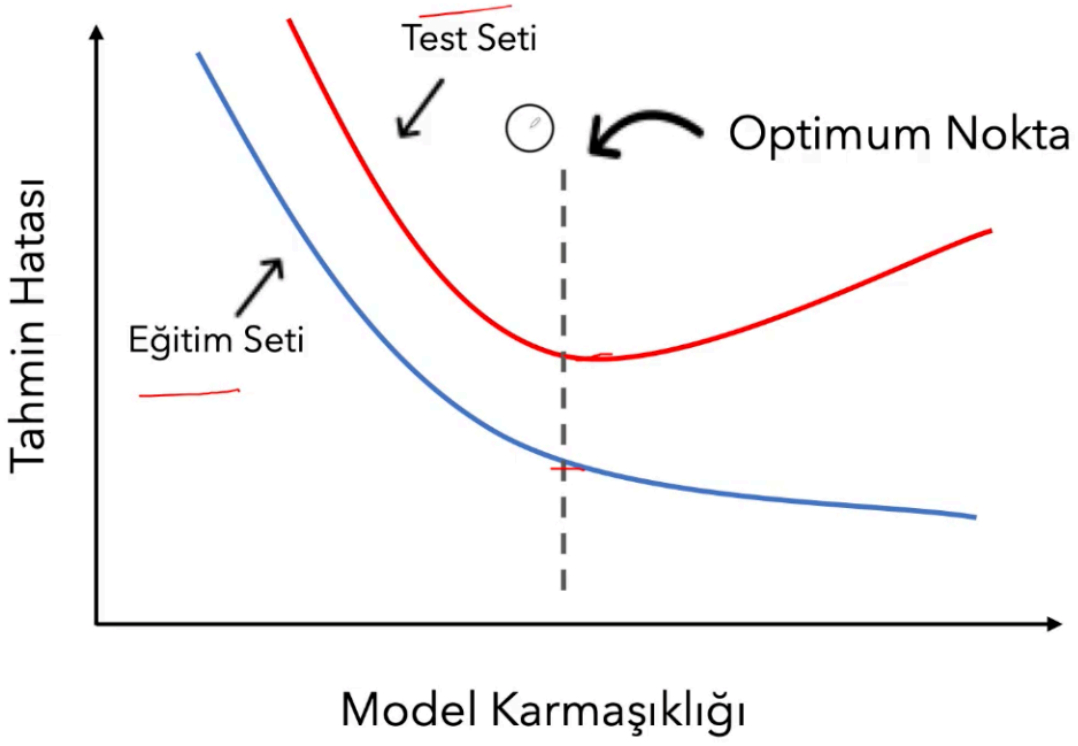


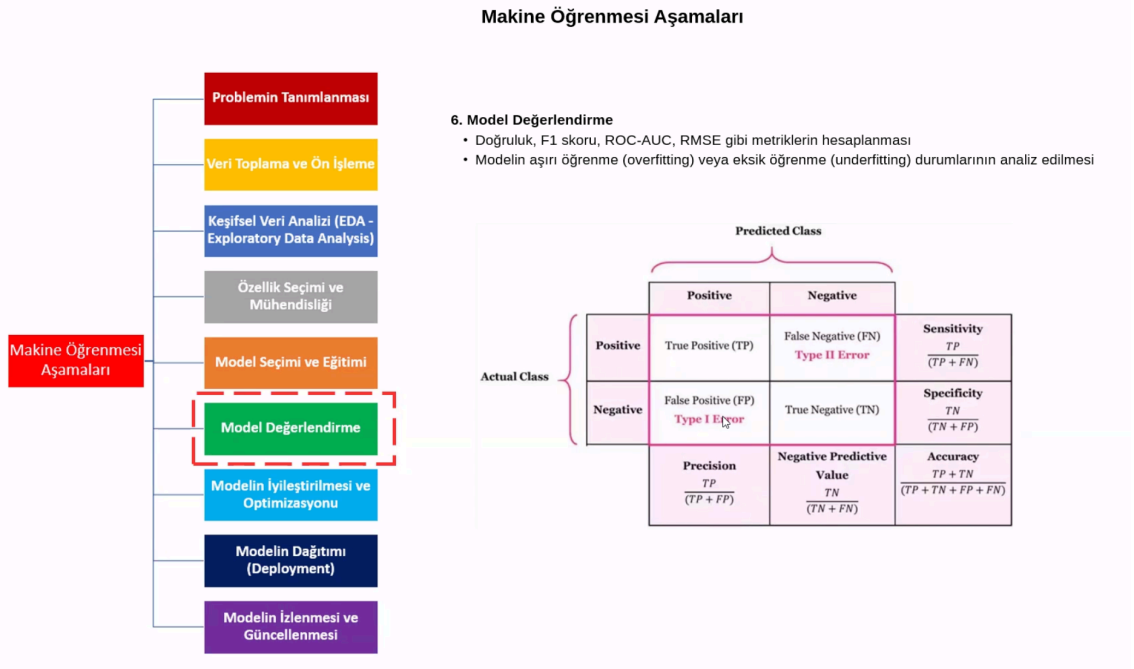


Overfitting durumunu nasıl anlarız?:

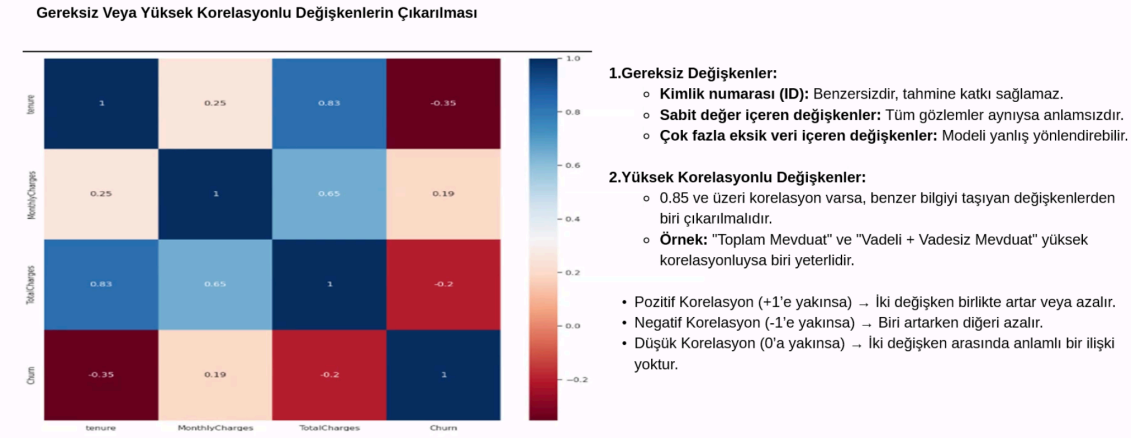
Belirti: Bir modelin eğitim veri seti üzerindeki performansı (doğruluk oranı yüksek, hata oranı düşük) **çok iyi veya mükemmel** iken, test (veya doğrulama) veri seti üzerindeki performansı **önemli ölçüde düşüktür**.

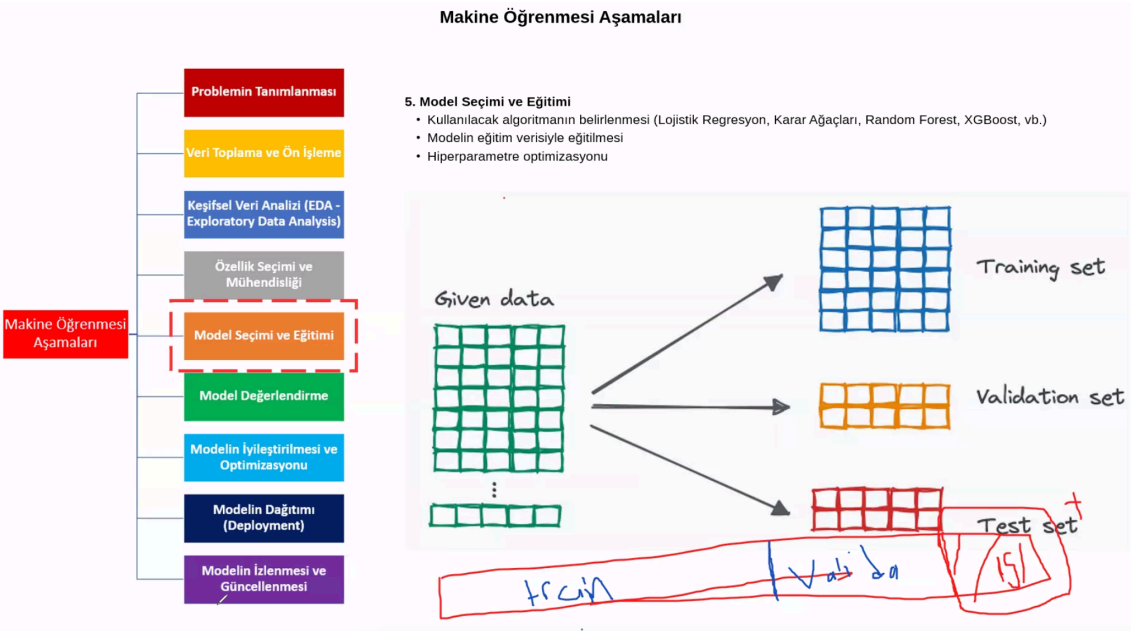
1. **Verinizi Eğitim ve Test Setlerine Ayırın:** Bu, overfitting'i tespit etmenin ilk ve en temel adımıdır. Genellikle %70-%30 veya %80-%20 gibi oranlar kullanılır.
2. **Modelinizi Eğitin:** Sadece eğitim seti üzerinde modelinizi eğitin.
3. **Performansı Değerlendirin:**
   - Hem eğitim seti üzerinde modelin performansını (doğruluk, hata oranı, F1-skoru vb.) ölçün.
   - Hem de **test seti** üzerinde modelin performansını ölçün.
4. **Sonuçları Karşılaştırın:** Eğer eğitim performansı test performansından belirgin şekilde daha yüksekse, overfitting mevcuttur.
5. **Öğrenme Eğrilerini Çizin:** Özellikle derin öğrenme modellerinde, eğitim ve doğrulama kayıp/doğruluk eğrilerini görselleştirmek overfitting'i net bir şekilde gösterir.
6. **K-Fold Cross-Validation Kullanın:** Daha sağlam bir değerlendirme için bu yöntemi kullanarak modelin farklı veri parçaları üzerindeki performans tutarlılığını kontrol edin.





Metrik	Açıklama	Yüksek Değer Ne Anlama Gelir?
MAE	Ortalama mutlak hata	Modelin tahmini daha doğru
MSE	Kareli hata ortalaması	Büyük hataların sayısı daha az
RMSE	Kök ortalama kare hata	Modelin tahmini gerçek değere daha yakın
R^2	Modelin açıklama gücü	Model değişkenleri daha iyi açıklıyor
Accuracy	Doğru tahmin oranı	Model genel olarak daha doğru
Precision	Doğru pozitif oranı	Yanlış pozitifler az
Recall	Gerçek pozitifleri bulma oranı	Yanlış negatifler az
F1 Score	Precision & Recall dengesi	Model dengeli tahmin yapıyor





Mülakat Soruları

Aşırı Öğrenme (Overfitting)

- Aşırı öğrenme, bir makine öğrenimi modelinin **eğitim verisindeki gürültüyü veya tesadüfi desenleri ezberleyerek**, yeni ve daha önce görmediği veriler üzerinde **genelleme yapma yeteneğini kaybetmesi** durumudur. Model, eğitim veri setine o kadar iyi uyum sağlar ki, bu uyum veri setindeki gerçek ilişkilerin ötesine geçer ve modelin **performansı eğitim setinde yüksek görünse de, test setinde önemli ölçüde düşer**. Bu genellikle modelin çok karmaşık olması veya eğitim verisinin yetersiz olmasından kaynaklanır.

Model Karmaşıklığı:

- Model karmaşıklığı, bir makine öğrenimi modelinin **öğrenebileceği desenlerin ve ilişkilerin çeşitliliğini ve derinliğini** ifade eder. Daha karmaşık bir model, veri içindeki daha ince ve detaylı ilişkileri yakalama potansiyeline sahiptir; örneğin, daha fazla katmana veya nörona sahip bir sinir ağı ya da daha fazla karar düğümüne sahip bir karar ağacı. Ancak, **aşırı karmaşık modeller aşırı öğrenmeye (overfitting)** eğilimli olabilirken, **yetersiz karmaşık modeller yetersiz öğrenmeye (underfitting)** yol açarak verideki önemli desenleri kaçırabilir. İdeal model karmaşıklığı, hem eğitim hem de test verilerinde iyi genelleme yapabilen dengeyi bulmaktır.

Analitik çözüm ile optimizasyon çözümü arasındaki fark nedir?

- Analitik çözüm ile optimizasyon çözümü arasındaki temel fark, **bir problemin en iyi çözümüne ulaşma yöntemleridir**. **Analitik çözüm**, bir fonksiyonun türevini sıfıra eşitlemek gibi matematiksel formüller veya cebirsel manipülasyonlar kullanarak **kapalı formda, kesin bir çözüm bulmayı** hedefler. Bu yöntem, çözümün doğrudan ve tam olarak hesaplanabildiği durumlarda, örneğin basit regresyon problemlerinde en uygun katsayıları bulmak için kullanılır.

  Buna karşılık, **optimizasyon çözümü** (Gradient Descent gibi), kapalı form bir çözümün bulunamadığı veya hesaplama açısından çok pahalı olduğu **karmaşık problemlerde, iteratif bir yaklaşımla en iyi çözüme adım adım yaklaşmayı** amaçlar. Gradient Descent özelinde, modelin kayıp fonksiyonunun (loss function) gradyanını (eğimi) hesaplayarak, bu eğimin tersi yönde küçük adımlarla ilerleyerek kayıp fonksiyonunu minimize etmeye çalışır. Bu yöntem, derin öğrenme modellerinin eğitimi gibi yüksek boyutlu ve doğrusal olmayan problemlerde yaygın olarak kullanılır, ancak **yaklaşık bir çözüm sunar ve global optimuma ulaşma garantisi yoktur**.

Çapraz doğrulama nedir açıklayınız. Model performansını artırır mı?

- Çapraz doğrulama (cross-validation), bir makine öğrenimi modelinin **performansını daha güvenilir bir şekilde değerlendirmek ve modelin yeni, görünmeyen verilere ne kadar iyi genellenebileceğini tahmin etmek için kullanılan bir tekniktir**. Temel olarak, mevcut veri setini eğitim ve test setleri olarak tek bir kez bölmek yerine, veri setini birden çok alt kümeye (katmanlara veya "fold"lara) ayırır. Her iterasyonda, bu alt kümelerden biri test seti olarak kullanılırken, kalanlar eğitim seti olarak kullanılır. Bu süreç, her alt küme en az bir kez test seti olarak kullanılana kadar tekrarlanır ve sonunda her iterasyonda elde edilen performans metriklerinin ortalaması alınır.

  Çapraz doğrulama, **model performansını doğrudan artırmaz**, yani modelin öğrenme yeteneğini veya içsel parametrelerini iyileştirmez. Ancak, **modelin gerçek genelleme yeteneğine dair daha sağlam ve daha az yanlı bir tahmin sağlayarak dolaylı olarak modelin iyileştirilmesine yardımcı olur**. Bu sayede, aşırı öğrenme (overfitting) veya yetersiz öğrenme (underfitting) gibi sorunları daha doğru bir şekilde tespit etmeye ve model seçimi (örneğin, farklı algoritmalar veya hiperparametreler arasında seçim yapma) ve model ayarlama (tuning) süreçlerinde daha bilinçli kararlar vermeye olanak tanır. Bu da, uzun vadede daha iyi performans gösteren bir modelin geliştirilmesine katkıda bulunur.

Makine öğrenmesi tekniklerinden regresyon ve sınıflandırmanın ortak özellikleri ve farkları nelerdir?

Regresyon ve sınıflandırma, makine öğreniminin en temel ve yaygın olarak kullanılan iki denetimli öğrenme tekniğidir. Her ikisi de **giriş verileri (özellikler) ile çıkış verileri (hedefler) arasındaki ilişkileri öğrenmeyi** ve bu ilişkileri kullanarak **yeni, görünmeyen veriler için tahminler yapmayı** amaçlar.

Ortak Özellikleri:

- **Denetimli Öğrenme:** Her iki teknik de, doğru çıktıların (etiketlerin) bilindiği etiketli veri setleri üzerinde eğitilir.
- **Tahmin Amacı:** Her ikisi de, bir model oluşturarak gelecekteki veya bilinmeyen veri noktaları için tahminler yapmayı hedefler.
- **Model Değerlendirme:** Performanslarını değerlendirmek için çeşitli metrikler (örneğin, doğruluk, hata kareler ortalaması) kullanılır.
- **Veri Ön İşleme:** Her iki teknik de, modelin daha iyi performans göstermesi için veri temizleme, özellik mühendisliği ve ölçeklendirme gibi ön işleme adımlarına ihtiyaç duyar.

Farkları:

- **Çıktı Türü:**
  - **Regresyon:** Sürekli, sayısal bir çıktı (hedef değişken) tahmin eder. Örneğin, bir evin fiyatını, bir kişinin yaşını veya bir şirketin satışlarını tahmin etmek. Çıktı sonsuz sayıda değer alabilir.
  - **Sınıflandırma:** Ayrık, kategorik bir çıktı (hedef değişken) tahmin eder. Yani, bir veri noktasını belirli bir sınıfa veya kategoriye atar. Örneğin, bir e-postanın spam olup olmadığını (evet/hayır), bir resimdeki nesnenin ne olduğunu (kedi/köpek/kuş) veya bir müşterinin ürünü satın alıp almayacağını (satın alır/satın almaz) tahmin etmek. Çıktı sınırlı sayıda önceden tanımlanmış kategoriye aittir.
- **Problem Tanımı:**
  - **Regresyon:** "Ne kadar?" veya "Kaç tane?" gibi sorulara cevap arar.
  - **Sınıflandırma:** "Hangi kategoriye ait?" veya "Evet mi, Hayır mı?" gibi sorulara cevap arar.
- **Kullanılan Algoritmalar ve Metrikler:** Her ne kadar bazı algoritmalar (örneğin, Karar Ağaçları, Destek Vektör Makineleri) hem regresyon hem de sınıflandırma için uyarlanabilse de, genellikle farklı varyasyonları kullanılır. Ayrıca, performans değerlendirme metrikleri de farklıdır:
  - **Regresyon:** Ortalama Mutlak Hata (MAE), Ortalama Kare Hata (MSE), Kök Ortalama Kare Hata (RMSE), R-kare (R-squared) gibi metrikler kullanılır.
  - **Sınıflandırma:** Doğruluk (Accuracy), Hassasiyet (Precision), Duyarlılık (Recall), F1 Skoru, ROC Eğrisi, Karmaşıklık Matrisi (Confusion Matrix) gibi metrikler kullanılır.

Random Forest'ı teknik olmayan birine nasıl anlatırsınız?

Random Forest, karmaşık kararlar alması gereken bir grup uzmandan oluşan bir ekibe benzer. Tek bir uzmana güvenmek yerine, bu model birçok farklı "karar ağacı" oluşturur. Her bir ağaç, verinin farklı yönlerine odaklanan ve kendi kararlarını veren küçük bir uzmandır.

Model, bir tahmin yapması gerektiğinde, tüm bu bireysel ağaçlara sorar ve onların verdiği cevapların **çoğunluğunu veya ortalamasını alır.** Böylece, tek bir uzmanın hata yapma olasılığını azaltırken, çok daha **güçlü ve güvenilir bir tahmin** elde ederiz. Tıpkı önemli bir kararı tek bir kişiye bırakmak yerine, farklı perspektiflerden bakıp oy birliğiyle karar veren bir kurul gibi düşünebilirsiniz. Bu sayede model, verideki ufak değişikliklerden daha az etkilenir ve daha genel geçer sonuçlar verir.

Gradient Boosting Machine(GBM) nedir? Nasıl çalışır?

Gradient Boosting Machine (GBM) Nedir?

Gradient Boosting Machine (GBM), tahmin yapmak için birçok basit "öğrenici" modeli (genellikle **karar ağaçları**) **ardışık** olarak bir araya getiren güçlü bir makine öğrenimi algoritmasıdır. Tek bir büyük ve karmaşık model oluşturmak yerine, GBM küçük, zayıf modellerden oluşan bir ekip kurar ve her yeni model, bir önceki modelin yaptığı hataları düzeltmeye odaklanır. Bu "takım çalışması" sayesinde, nihai tahminler çok daha doğru ve sağlam hale gelir.

GBM Nasıl Çalışır?

GBM'nin çalışma mantığı, bir problemi adım adım ve art arda düzelten bir ekip yaklaşımına benzer. İşte basitleştirilmiş adımları:

1. **İlk Tahmin (Başlangıç Noktası):** İlk olarak, model basit bir başlangıç tahmini yapar. Bu genellikle verilerdeki hedef değerin ortalaması gibi çok temel bir tahmindir. Tahmin, büyük olasılıkla doğru değildir ve birçok hata içerir.
2. **Hataları Hesaplama:** İlk tahmin yapıldıktan sonra, gerçek değerler ile bu ilk tahmin arasındaki **hatalar (artıklar)** hesaplanır. Bu hatalar, modelin nerede yanlış yaptığını gösterir.
3. **Hataları Düzeltmek İçin Yeni Bir Model Oluşturma: İşte burada "boosting" ve "gradient" kısmı devreye girer.** GBM, önceki modelin yaptığı hataları öğrenmek ve düzeltmek için **yeni, basit bir model (genellikle küçük bir karar ağacı)** eğitir. Bu yeni ağaç, doğrudan hedef değeri tahmin etmek yerine, **önceki modelin yaptığı hataları tahmin etmeye çalışır.** Yani, "Ne kadar yanıldık?" sorusuna cevap bulmaya odaklanır.
4. **Tahmini Güncelleme:** Yeni eğitilen bu "hata düzeltici" modelin tahminleri, bir **öğrenme oranı (learning rate)** ile çarpılarak önceki toplu tahmine eklenir. Öğrenme oranı, her yeni modelin önceki hataları ne kadar güçlü bir şekilde düzeltmesi gerektiğini kontrol eden küçük bir sayıdır. Bu, modelin çok hızlı öğrenip "aşırı öğrenmesini" (ezberlemesini) engeller.
5. **Tekrarlama:** Bu adımlar (hata hesaplama, yeni model eğitme ve tahmini güncelleme) belirli bir sayı kadar veya hata belirli bir seviyeye düşene kadar tekrarlanır. Her yeni ağaç, önceki ağaçların bıraktığı "kalan" hatalara odaklanır ve bu hataları adım adım azaltır. Bu süreç, bir yokuş aşağı inen (gradient descent) bir süreç gibidir; model her adımda hatayı azaltmak için en dik yokuş aşağı yönde ilerler.

Sonuç olarak, GBM, her biri bir öncekinin eksiklerini kapatan birçok basit modelin bir araya gelmesiyle, tek başına çok daha güçlü ve doğru tahminler yapabilen bir süper model oluşturur. Bu "ekip çalışması", GBM'i veri bilimindeki en etkili algoritmalardan biri yapar.

Hiperparametre nedir? Ağaç yöntemleri için en sık kullanılan hiperparametrelerden iki tanesini açıklayınız.

Basitçe ifade etmek gerekirse, **hiperparametreler, bir makine öğrenimi modelinin "öğrenme süreci" başlamadan önce, bizim tarafımızdan elle ayarlanan konfigürasyonlardır.** Modelin kendisinin veriden öğrendiği parametrelerden (örneğin, bir regresyon denklemindeki katsayılar veya bir sinir ağındaki ağırlıklar) farklıdırlar. Hiperparametreler, modelin ne kadar karmaşık olacağını, ne kadar derinlemesine öğrenme yapacağını veya öğrenme sürecinin nasıl ilerleyeceğini belirler. Doğru hiperparametre ayarları, bir modelin performansını ve genelleme yeteneğini önemli ölçüde artırabilirken, yanlış ayarlar modelin aşırı öğrenmesine (ezberlemesine) veya yeterince öğrenememesine neden olabilir.

Ağaç tabanlı algoritmalar (Karar Ağaçları, Random Forest, Gradient Boosting gibi) için en sık kullanılan ve en etkili hiperparametrelerden ikisi şunlardır:

1. `max_depth` (Maksimum Derinlik): Bu hiperparametre, bir karar ağacının **ne kadar derin olabileceğini**, yani bir düğümden kök düğüme olan en uzun yolun uzunluğunu belirler. Başka bir deyişle, bir ağacın bir karar vermek için **kaç tane ardışık soru sorabileceğini** sınırlar.

- **Etkisi:**
  - **Küçük `max_depth` değeri:** Daha sığ ağaçlar anlamına gelir. Bu ağaçlar daha basittir ve veriyi "ezberleme" (aşırı öğrenme) olasılığı daha düşüktür, ancak karmaşık ilişkileri yakalamakta zorlanabilirler (eksik öğrenme riski).
  - **Büyük `max_depth` değeri:** Daha derin ağaçlar anlamına gelir. Bu ağaçlar daha karmaşıktır ve eğitim verisindeki ince detayları bile yakalayabilirler. Ancak, çok derin ağaçlar eğitim verisini ezberleyerek yeni, görünmeyen verilerde kötü performans gösterme (aşırı öğrenme) riskini taşır.
- **Benzetme:** Bir karar ağacını bir "Evet/Hayır" sorulardan oluşan bir akış şeması olarak düşünün. `max_depth`, şemadaki en uzun soru zincirinin kaç adımdan oluşabileceğini belirler.

2. `n_estimators` (Tahminleyici Sayısı - Özellikle Ensemble Modeller İçin): Bu hiperparametre, Random Forest veya Gradient Boosting gibi **topluluk (ensemble) öğrenme algoritmalarında** kullanılan **bağımsız tahminleyici sayısını** (yani modelin içinde oluşturulan karar ağacı sayısını) belirler.

- **Etkisi:**
  - **Küçük `n_estimators` değeri:** Daha az sayıda ağaç anlamına gelir. Model daha az karmaşık olur ve eğitim süresi kısalır, ancak topluluğun gücünden tam olarak yararlanamadığı için performansı daha düşük olabilir.
  - **Büyük `n_estimators` değeri:** Daha fazla sayıda ağaç anlamına gelir. Bu genellikle modelin performansını artırır ve daha sağlam, genellenebilir tahminler yapmasını sağlar. **Ancak, çok fazla ağaç, eğitim süresini ve bellek kullanımını artırabilir ve belirli bir noktadan sonra performans artışı durur veya çok yavaşlar.** Aşırı öğrenme riskini tek başına artırmaz, aksine bazen azaltabilir çünkü farklı ağaçlar hataları dengeleyebilir.
- **Benzetme:** Random Forest'ı bir uzmandan oluşan komite olarak düşündüğümüzde, `n_estimators` komitedeki uzman sayısıdır. Ne kadar çok uzman fikirlerini birleştirirse, nihai kararın o kadar sağlam ve doğru olması beklenir.

PCA nedir? Hangi amaçlar için kullanılır?

Temel Bileşen Analizi (PCA), yüksek boyutlu veri setlerini daha düşük boyutlu bir temsiline indirgemek için kullanılan istatistiksel bir tekniktir. Bunu yaparken, orijinal verideki en önemli bilgiyi (yani en fazla varyansı) korumayı hedefler.

Basitçe söylemek gerekirse, PCA bir veri setindeki **ilişkili değişkenleri, birbirleriyle ilişkisi olmayan (ortogonal) yeni değişkenlere** dönüştürür. Bu yeni değişkenlere **"Temel Bileşenler"** denir. İlk temel bileşen, verideki en fazla varyansı açıklarken, ikinci bileşen kalan varyansı açıklar ve bu şekilde devam eder.

Hangi Amaçlar İçin Kullanılır?

PCA'nın veri biliminde birçok önemli kullanım alanı vardır:

- **Boyut Azaltma (Dimensionality Reduction):** En yaygın kullanım amacıdır. Çok fazla özelliğe (değişkene) sahip veri setleri, makine öğrenimi modellerinin eğitilmesi için hem yavaş hem de karmaşık olabilir (boyutluluk laneti). PCA, bu özellikleri daha az sayıda temel bileşene dönüştürerek bu karmaşıklığı azaltır ve modelin daha hızlı çalışmasını ve genelleme yeteneğini artırmasını sağlar.
- **Veri Görselleştirme:** Yüksek boyutlu verileri doğrudan görselleştirmek zordur. PCA, verileri iki veya üç ana bileşene indirgeyerek, veri setindeki kalıpları, kümeleri veya aykırı değerleri görsel olarak keşfetmeyi kolaylaştırır.
- **Gürültü Azaltma (Noise Reduction):** Düşük varyansa sahip temel bileşenler genellikle verideki gürültüyü temsil eder. Bu bileşenleri atarak, veri setindeki sinyal-gürültü oranını iyileştirebilir ve model performansını artırabiliriz.
- **Öznitelik Mühendisliği (Feature Engineering):** PCA, mevcut özelliklerin doğrusal kombinasyonlarından yeni, bağımsız özellikler oluşturarak model performansını artırmaya yardımcı olabilir.
- **Çoklu Doğrusallıkla Başa Çıkma:** Bağımsız değişkenler arasında yüksek korelasyon olduğunda (çoklu doğrusallık), bu durum regresyon modellerinde sorunlara yol açabilir. PCA, korelasyonlu değişkenleri bağımsız temel bileşenlere dönüştürerek bu sorunu çözmeye yardımcı olur.

Özetle, PCA, karmaşık veri setlerini daha yönetilebilir ve anlaşılır hale getirerek veri analizi ve makine öğrenimi süreçlerini optimize eden güçlü bir araçtır.

Unsupervised learning sürecinden supervised learning'e nasıl geçeriz?

Unsupervised öğrenmeden supervised öğrenmeye geçiş, esasen etiketsiz veriye anlamlı etiketler atama sürecidir. Bu geçiş, öncelikle kümeleme (veride doğal grupları bulma) veya boyut azaltma gibi denetimsiz tekniklerle verideki gizli yapıları ve kalıpları keşfederek başlar. Bu keşifler, daha sonra ya insan uzmanların manuel olarak etiketlemesi için bir temel oluşturur ya da yarı denetimli öğrenme (az etiketli veriyle modeli eğitip etiketsiz verileri tahmin etme) ve aktif öğrenme (modelin etiketlemesi gereken en faydalı noktaları seçmesi) gibi otomatik veya yarı otomatik yöntemlerle veriye sentetik etiketler atamak için kullanılır. Etiketleme süreci tamamlandığında, elde edilen bu etiketli veri seti artık sınıflandırma veya regresyon gibi denetimli öğrenme modellerini eğitmek ve belirli tahmin görevlerini gerçekleştirmek için hazırdır. Bu sayede, başlangıçta sadece kalıpları keşfetmekle yetinen sistem, belirli bir çıktıyı tahmin edebilen güçlü bir tahminsel bir araca dönüşür.

İşte bu geçişi sağlamanın temel adımları ve yöntemleri:

Unsupervised'dan Supervised Learning'e Geçiş Adımları

1. **Unsupervised Öğrenmeyle Veri Keşfi ve Yapılandırma:**
    - **Kümeleme (Clustering):** Veri setindeki doğal grupları veya kümeleri belirlemek için K-Means, DBSCAN, Hierarchical Clustering gibi algoritmalar kullanılır. Örneğin, müşteri segmentasyonu yaparak benzer davranışa sahip müşterileri gruplayabilirsiniz. Bu kümeler, daha sonra supervised öğrenme için **"sentetik etiketler"** olarak kullanılabilir.
    - **Boyut Azaltma (Dimensionality Reduction):** PCA, t-SNE, UMAP gibi tekniklerle veriyi daha düşük boyutlu bir uzaya indirgeyerek görselleştirebilir ve önemli özelliklerini belirleyebilirsiniz. Bu, etiketleme sürecini kolaylaştırabilir veya yeni, daha anlamlı özellikler (feature engineering) oluşturmanıza yardımcı olabilir.
    - **Anomali Tespiti (Anomaly Detection):** Aykırı değerleri belirlemek, supervised öğrenme modelinde "anormal" veya "normal" etiketleri oluşturmak için bir temel sağlayabilir.
2. **Etiketleme (Labeling):**
    - **Manuel Etiketleme:** Unsupervised öğrenmeyle keşfedilen kalıpları veya kümeleri kullanarak, insan uzmanlar tarafından verilere manuel olarak etiket atanmasıdır. Bu, en doğru yöntemdir ancak zaman alıcı ve maliyetli olabilir, özellikle büyük veri setleri için. Örneğin, kümeleme sonucu oluşan müşteri segmentlerine "yüksek değerli", "riskli" gibi iş odaklı etiketler atayabilirsiniz.
    - **Yarı Denetimli Öğrenme (Semi-Supervised Learning):** Elinizde az miktarda etiketli veri ve çok miktarda etiketsiz veri olduğunda bu yaklaşım kullanılır. **Öncelikle az etiketli veriyle bir başlangıç modeli eğitilir. Bu model daha sonra etiketsiz verileri tahmin etmek (pseudo-labeling) için kullanılır.** Yüksek güvenilirlikteki tahminler, etiketli veri setine eklenerek model yeniden eğitilir. **Bu süreç iteratif olarak devam edebilir.**
    - **Aktif Öğrenme (Active Learning):** Modelin, etiketlenmesi en çok fayda sağlayacak veri noktalarını insan uzmanlardan talep etmesi prensibine dayanır. Bu, manuel etiketleme çabasını optimize ederken model performansını artırmaya yardımcı olur.
3. **Supervised Model Eğitimi:**
    - Etiketleme süreci tamamlandıktan sonra, artık etiketli bir veri setiniz olur. Bu veri setiyle sınıflandırma (classification) veya regresyon (regression) gibi **supervised öğrenme algoritmaları** (örneğin, Karar Ağaçları, Destek Vektör Makineleri, Sinir Ağları, Lojistik Regresyon) kullanarak tahmin modellerinizi eğitebilirsiniz.
4. **Değerlendirme ve İyileştirme:**
    - Eğittiğiniz supervised modelin performansını (doğruluk, kesinlik, duyarlılık, F1 skoru, RMSE vb.) değerlendirin. Gerekirse etiketleme sürecini veya modelin hiperparametrelerini iyileştirerek performansı artırabilirsiniz.

Neden böyle bir geçiş yaparız?

- **Etiketli Veri Kıtlığı:** Gerçek dünyada etiketli veri elde etmek çoğu zaman çok pahalı veya zordur. Unsupervised öğrenme, bol miktarda bulunan etiketsiz veriden değerli bilgiler çıkararak etiketleme yükünü azaltır.

- **Keşif ve Anlama:** Unsupervised öğrenme, veri setindeki gizli yapıları ve ilişkileri anlamak için güçlü bir başlangıç noktası sağlar. Bu bilgiler, daha sonra supervised bir problem için nasıl etiketleme yapılacağına dair değerli ipuçları verebilir.
- **Öznitelik Mühendisliği:** Unsupervised teknikler (özellikle boyut azaltma), supervised modeller için daha anlamlı ve kullanışlı yeni özellikler oluşturmaya yardımcı olabilir.

Özetle, unsupervised öğrenme genellikle veriyi hazırlar, anlamlandırır ve potansiyel etiketleri ortaya çıkarır; ardından bu etiketler supervised öğrenme modellerinin eğitilmesi için kullanılır.

K-Means kümeleme algoritması nasıl çalışır?

K-Means, makine öğrenmesinde sıkça kullanılan, denetimsiz (unsupervised) bir kümeleme algoritmasıdır. Amacı, etiketlenmemiş veri noktalarını, benzer özelliklere sahip olanları bir araya getirerek önceden belirlenmiş "K" sayıda kümeye ayırmaktır. Algoritma, her bir kümenin merkezini (centroid) bulmaya ve veri noktalarını bu merkezlere olan uzaklıklarına göre atayarak çalışır.

İşte K-Means kümeleme algoritmasının adım adım çalışma prensibi:

K-Means Kümeleme Algoritması Nasıl Çalışır?

1. **K Değerini Belirleme:**
   - Algoritmaya başlamadan önce, veri setini kaç kümeye ayırmak istediğimizi (yani **K değerini**) belirlememiz gerekir. Bu, genellikle alan bilgisine veya dirsek metodu (Elbow Method) gibi tekniklere dayanarak yapılır.
2. **Başlangıç Centroidlerini Seçme:**
   - Veri setinden **rastgele K adet veri noktası seçilir** ve bunlar ilk küme merkezleri (centroidler) olarak atanır. Bazen daha akıllıca seçimler için K-Means++ gibi yöntemler de kullanılır.
3. **Veri Noktalarını Kümelere Atama (Atama Adımı - Expectation Step):**
   - Her bir veri noktası için, belirlenen **K adet centroid'e olan uzaklıklar hesaplanır**. Genellikle Öklid mesafesi kullanılır, ancak başka mesafe ölçümleri de tercih edilebilir.
   - Her veri noktası, kendisine **en yakın centroid'in ait olduğu kümeye atanır.**
4. **Centroidleri Güncelleme (Güncelleme Adımı - Maximization Step):**
   - Her kümedeki tüm veri noktaları atandıktan sonra, her bir kümenin **yeni centroid'i hesaplanır. Yeni centroid, o kümeye atanmış tüm veri noktalarının ortalama pozisyonudur. Yani kümenin merkezi güncellenir.**
5. **Tekrarlama ve Yakınsama:**
   - Algoritma, 3. ve 4. adımları **centroidlerin pozisyonu artık önemli ölçüde değişmeyene kadar** veya **maksimum iterasyon sayısına ulaşana kadar** tekrarlar. Bu duruma **yakınsama (convergence)** denir. Centroidler sabitlendiğinde veya çok az hareket ettiğinde, kümelerin stabilize olduğu kabul edilir ve algoritma durur.

Bu iteratif süreç sayesinde K-Means algoritması, veri noktaları ile kendi kümelerinin centroidleri arasındaki toplam mesafeyi (yani küme içi varyansı) minimuma indirmeyi hedefler. Sonuç olarak,

her küme içindeki noktalar birbirine benzerken, farklı kümelerdeki noktalar birbirinden daha farklı özelliklere sahip olur.