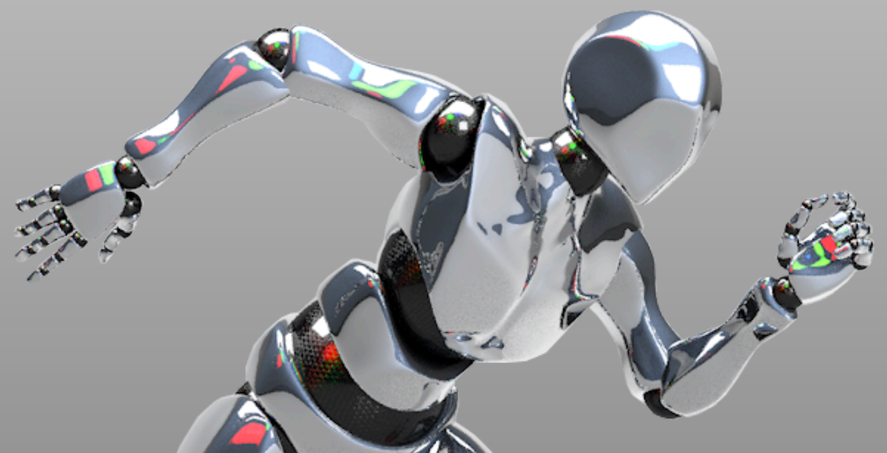
7.HAFTA

# YAPAY SİNİR AĞLARI



SAMSUN ÜNİVERSİTESİ

DR. ÖĞR. ÜYESİ ALPER TALHA KARADENİZ

# MODEL PERFORMANSININ DEĞERLENDIRILMESI :

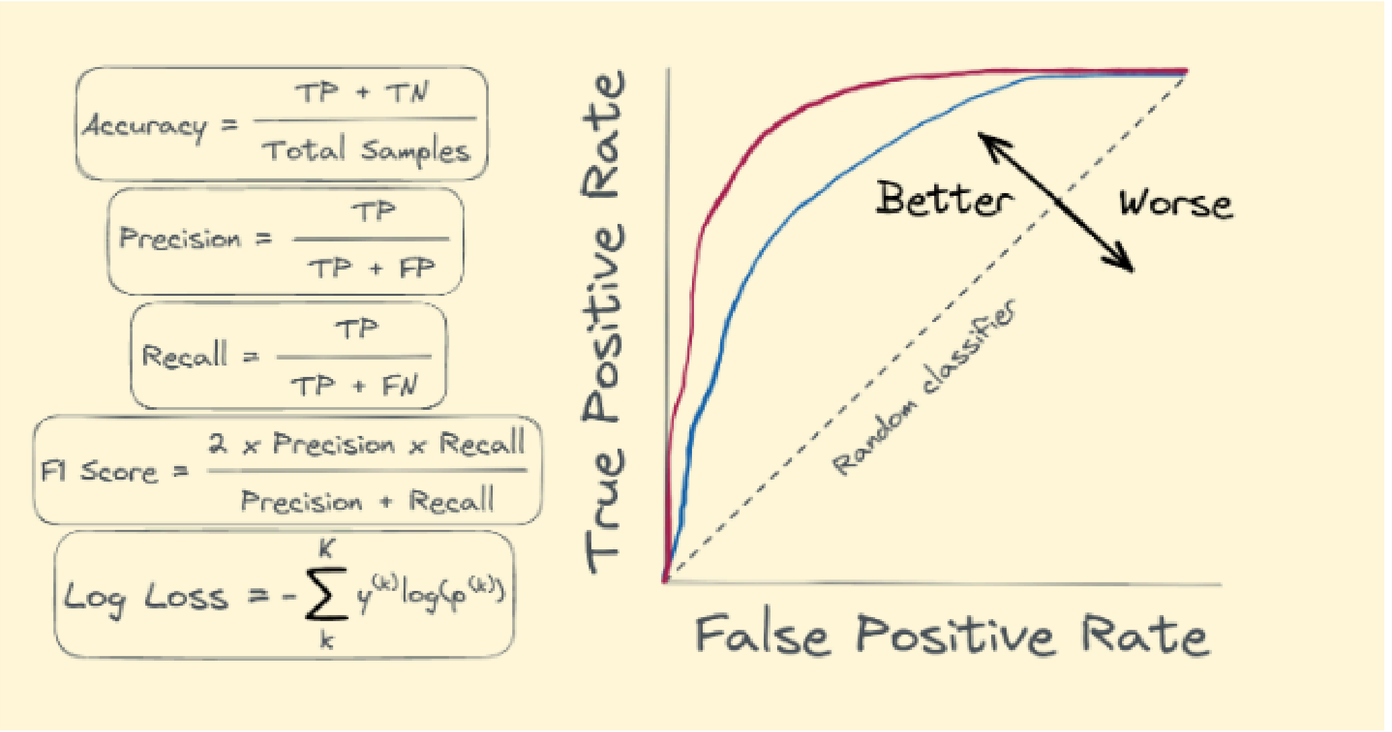
Makine öğrenmesi veya derin öğrenme ile kurduğumuz bir modelin, gerçek hayatta ne kadar doğru tahmin yapabildiğini ölçme sürecidir.

Bir modeli sadece **eğitim verisi** üzerinde iyi çalıştırmak yetmez, çünkü model ezberlemiş olabilir. Bu yüzden performans değerlendirmesi, modelin **genelleme yeteneğini** (yani yeni, daha önce görmediği verilerde ne kadar başarılı olduğunu) ölçmek için yapılır.

Genellikle şu adımlarla değerlendirme yapılır:

- **Eğitim, Doğrulama ve Test Verileri** kullanılır.
  - Eğitim seti: Modelin öğrenmesi için.
  - Doğrulama seti: Hiperparametre ayarları için.
  - Test seti: Gerçek performansı ölçmek için.
- **Performans Ölçütleri (Metrikler)** kullanılır:
  - Sınıflandırma problemlerinde: Doğruluk (Accuracy), Kesinlik (Precision), Duyarlılık (Recall), F1 Skoru, ROC-AUC vb.
  - Regresyon problemlerinde: Ortalama Kare Hatası (MSE), Ortalama Mutlak Hata (MAE), $R^2$ skoru vb.

- **Aşırı öğrenme (overfitting) / yetersiz öğrenme (underfitting)** kontrol edilir.
  Eğer model eğitimde yüksek başarı, testte düşük başarı gösteriyorsa overfitting vardır.
- **Çapraz Doğrulama (Cross Validation)** gibi yöntemler de, modelin daha güvenilir şekilde değerlendirilmesini sağlar.



# Confusion Matrix (Karmaşıklık Matrisi)

Confusion Matrix, bir sınıflandırma modelinin tahminleri ile gerçek değerleri karşılaştıran, hataların ve doğruların detaylı olarak görülebildiği bir tablodur. Bu tablo sayesinde modelin sadece genel doğruluğu değil, hangi tür hataları yaptığı da anlaşılır.

---

## İkili Sınıflandırmada Yapısı

Varsayalım ki iki sınıfımız var: Pozitif (1) ve Negatif (0). Karmaşıklık matrisi şöyle düzenlenir:

| Tahmin \ Gerçek | Gerçek Pozitif (1) | Gerçek Negatif (0) |
|---|---|---|
| Tahmin Pozitif (1) | True Positive (TP) | False Positive (FP) |
| Tahmin Negatif (0) | False Negative (FN) | True Negative (TN) |

## Kavramlar

- **True Positive (TP):**

  Modelin pozitif dediği ve gerçekten pozitif olan örnekler.

  Örnek: Hasta olan bir kişinin "hasta" olarak tahmin edilmesi.
- **True Negative (TN):**

  Modelin negatif dediği ve gerçekten negatif olan örnekler.

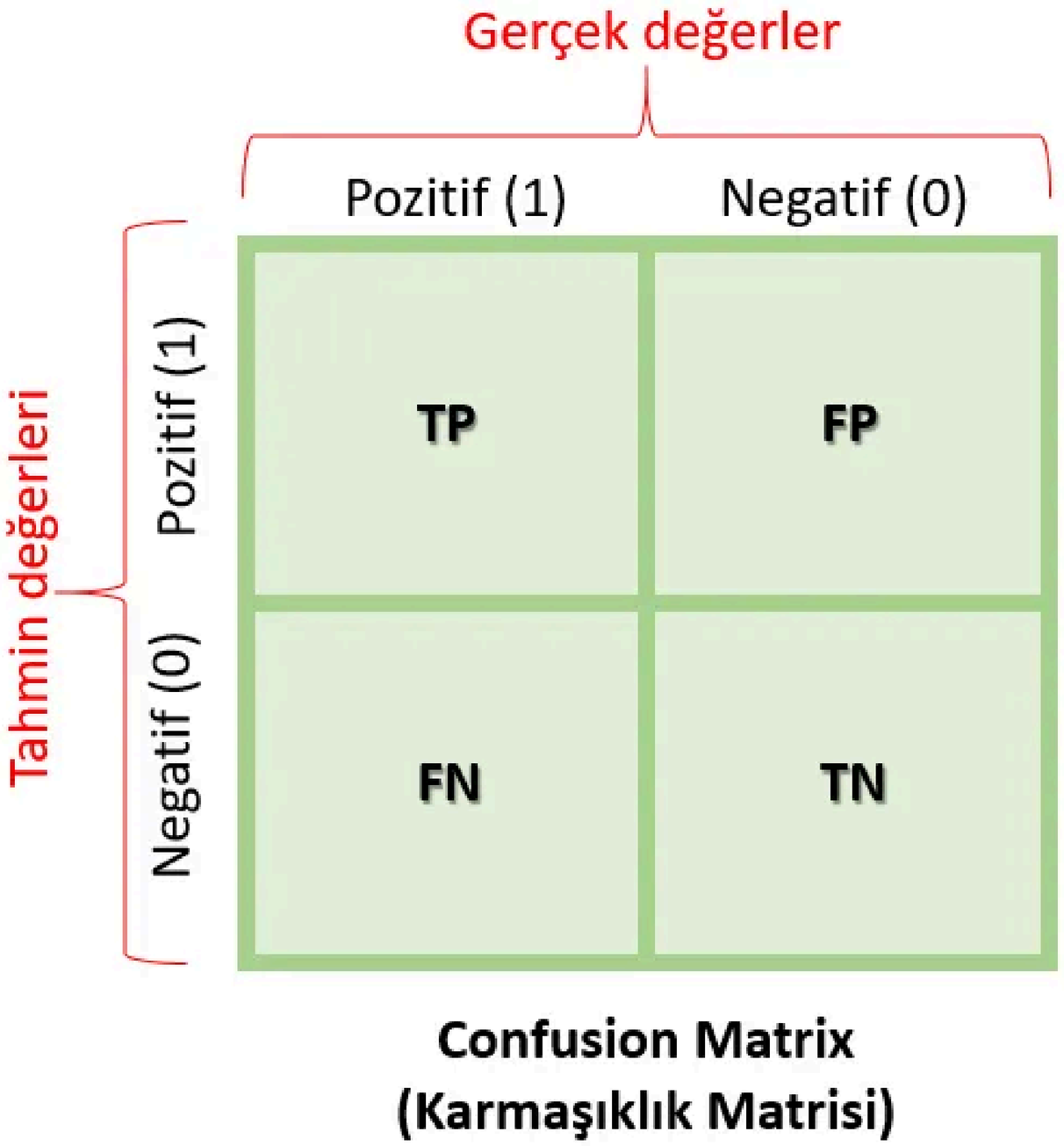
  Örnek: Sağlıklı bir kişinin "sağlıklı" olarak tahmin edilmesi.
- **False Positive (FP):**

  Modelin pozitif dediği ama aslında negatif olan örnekler.

  Örnek: Sağlıklı bir kişiyi yanlışlıkla "hasta" olarak tahmin etmek.
- **False Negative (FN):**

  Modelin negatif dediği ama aslında pozitif olan örnekler.

  Örnek: Hasta bir kişiyi yanlışlıkla "sağlıklı" olarak tahmin etmek.




**Confusion Matrix**
**(Karmaşıklık Matrisi)**

- Hedef değişkenin iki değeri vardır: **Pozitif** veya **Negatif**
- Sütunlar hedef değişkenin **gerçek değerlerini** temsil **eder**
- Satırlar hedef değişkenin **tahmin edilen değerlerini temsil** eder

## Doğruluk (Accuracy)

- Bir sınıflandırma modelinin **doğru tahmin ettiği örneklerin tüm örneklere oranıdır.**
- Formül:

$$Accuracy = \frac{\text{Doğru Tahmin Sayısı}}{\text{Toplam Tahmin Sayısı}}$$

- Örnek: 100 test verisinden 90'ını doğru tahmin ettiyse, doğruluk = %90'dır.
- Avantajı: Basit ve anlaşılır bir ölçüttür.
- Dezavantajı: Dengeli olmayan (örneğin %95'i "0", %5'i "1" olan) veri setlerinde yanıltıcı olabilir.

**Soru:**

Bir sınıflandırma modeli 200 adet test verisi üzerinde denenmiştir.

- Model, 160 örneği doğru tahmin etmiştir.
- 40 örneği ise yanlış tahmin etmiştir.

Bu modelin **doğruluk (accuracy)** değeri nedir?

---

**Çözüm:**

Doğruluk formülü:

$$Accuracy = \frac{\text{Doğru Tahmin Sayısı}}{\text{Toplam Tahmin Sayısı}}$$

$$Accuracy = \frac{160}{200} = 0.80$$

Yani modelin doğruluğu **%80**'dir.

## Kayıp (Loss)

- Modelin tahminleri ile gerçek değerler arasındaki farkı ölçen **hata fonksiyonudur.**
- Eğitim sırasında **optimizasyon algoritmaları (ör. Gradient Descent)** bu kaybı azaltmaya çalışır.
- Kullanılan probleme göre farklı kayıp fonksiyonları vardır:
  - Sınıflandırmada: **Cross-Entropy Loss**
  - Regresyonda: **Mean Squared Error (MSE)** veya **Mean Absolute Error (MAE)**
- Kayıp değeri ne kadar küçükse, modelin tahminleri gerçek değerlere o kadar yakındır.

## Soru (Mean Squared Error - MSE Loss):

Bir regresyon modeli, aşağıdaki 4 veri noktası için tahminler yapmıştır:

- Gerçek değerler ($y$): [3, 5, 2, 8]
- Model tahminleri ($\hat{y}$): [2.5, 5.5, 2, 7]

Bu modele göre **Mean Squared Error (MSE) Loss** değerini hesaplayınız.

**Çözüm:**

MSE formülü:

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

Adım adım:

1. Hatalar:
   - $(3 - 2.5)^2 = 0.25$
   - $(5 - 5.5)^2 = 0.25$
   - $(2 - 2)^2 = 0$
   - $(8 - 7)^2 = 1$
2. Toplam hata:

   $0.25 + 0.25 + 0 + 1 = 1.5$
3. Ortalama hata (n = 4):

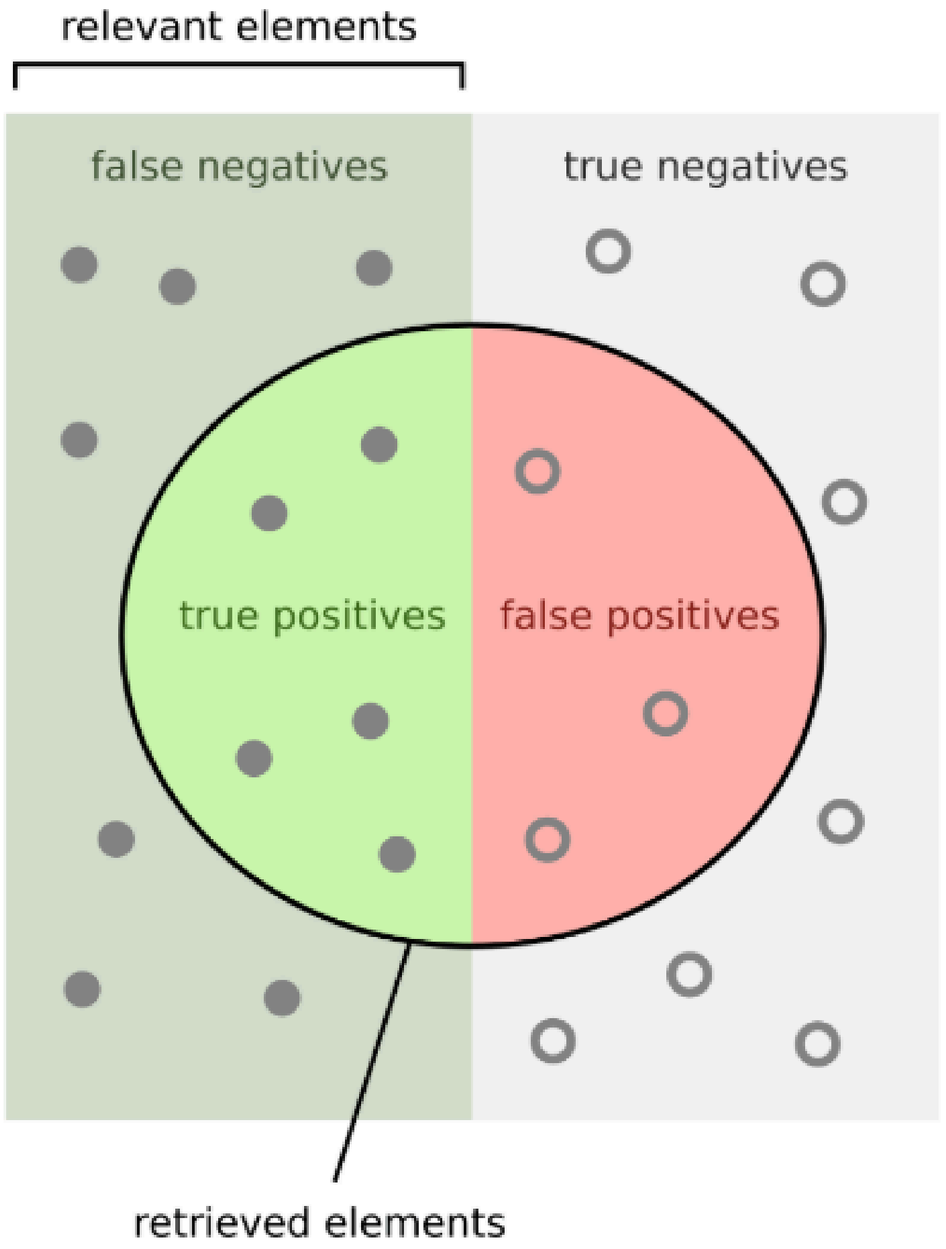
$$MSE = \frac{1.5}{4} = 0.375$$

## Precision (Kesinlik)

- Modelin "**pozitif" tahmin ettiği örneklerin gerçekten ne kadarının doğru olduğunu** gösterir.
- Formül:

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

- TP: True Positive (Gerçek Pozitif), FP: False Positive (Yanlış Pozitif)
- Yani: **Yanlış alarm yapmama becerisidir.**
- Örnek: Bir e-posta filtreleme sisteminde, "spam" dediği e-postaların gerçekten spam olma oranı.




How many retrieved items are relevant?

$$\text{Precision} = \frac{\text{green}}{\text{green} + \text{red}}$$

**Soru :**

Bir e-posta filtreleme sistemi, 200 test e-postası üzerinde denenmiştir. Sonuçlar şu şekildedir:

- **Gerçek Pozitif (TP)**: 70 (Spam olan e-postaları doğru yakaladı)
- **Yanlış Pozitif (FP)**: 30 (Spam olmayanları yanlışlıkla spam dedi)
- **Gerçek Negatif (TN)**: 80
- **Yanlış Negatif (FN)**: 20

Bu sistemin **Precision (Kesinlik)** değerini hesaplayınız.

**Çözüm:**

Precision formülü:

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

Yerine koyalım:

$$Precision = \frac{70}{70 + 30} = \frac{70}{100} = 0.70$$

**Sonuç:** Modelin **Precision değeri %70**'tir.

## Recall (Duyarlılık / Hassasiyet)

- Modelin **gerçek pozitifleri ne kadar yakaladığını** gösterir.
- Formül:

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

- FN: False Negative (Yanlış Negatif)
- Yani: **Gerçek pozitifleri atlamama becerisidir.**
- Örnek: Kanser testinde, hasta olan kişileri gerçekten hasta diye bulma oranı.

**Soru :**

Bir kanser tespit modeli, 150 hastalık testinde şu sonuçları vermiştir:

- **Gerçek Pozitif (TP)**: 45 (Hasta olan kişileri doğru hasta buldu)
- **Yanlış Negatif (FN)**: 15 (Hasta olan kişileri yanlışlıkla sağlıklı dedi)
- **Gerçek Negatif (TN)**: 70
- **Yanlış Pozitif (FP)**: 20

Bu modele göre **Recall (Duyarlılık)** değerini hesaplayınız.

**Çözüm:**

Recall formülü:

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

Yerine koyalım:

$$Recall = \frac{45}{45 + 15} = \frac{45}{60} = 0.75$$

**Sonuç:** Modelin **Recall değeri %75**'tir.

## F1 Score

- **Precision ve Recall'un harmonik ortalamasıdır.**
- İki metriği tek bir sayıda dengeler.
- Formül:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- Eğer Precision yüksek ama Recall düşükse (ya da tersi), F1 bu dengesizliği gösterir.
- **0 ile 1 arasında değer alır.** 1'e yaklaştıkça model daha iyidir.

**Soru :**

Bir hastalık tespit modeli için sonuçlar aşağıdaki gibidir:

- **Gerçek Pozitif (TP)**: 90
- **Yanlış Pozitif (FP)**: 30
- **Yanlış Negatif (FN)**: 30
- **Gerçek Negatif (TN)**: 150

Bu modele göre:

1. **Precision (Kesinlik)** değerini bulunuz.
2. **Recall (Duyarlılık)** değerini bulunuz.
3. Bu değerleri kullanarak **F1 Score**'u hesaplayınız.

Çözüm:

1. Precision:

$$Precision = \frac{TP}{TP + FP} = \frac{90}{90 + 30} = \frac{90}{120} = 0.75$$

2. Recall:

$$Recall = \frac{TP}{TP + FN} = \frac{90}{90 + 30} = \frac{90}{120} = 0.75$$

3. F1 Score:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$F1 = 2 \times \frac{0.75 \times 0.75}{0.75 + 0.75}$$

$$F1 = 2 \times \frac{0.5625}{1.5} = \frac{1.125}{1.5} = 0.75$$

**Sonuç: Modelin F1 Score değeri %75'tir.**

| | |
|---|---|
| **Accuracy** | $= \frac{\text{Doğru tahmin edilen değerler}}{\text{Tüm değerler}}, \left(\frac{TP+TN}{TP+TN+FP+FN}\right)$ |
| **Specificity** | $= \frac{TN}{TN+FP}$ |
| **Recall (Sensitivity)** | $= \frac{TP}{TP+FN}$ |
| **Precision** | $= \frac{TP}{TP+FP}$ |
| **F1 Score** | $= 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$ |
| **Geo. Ortalama** | $= \sqrt{\text{Sensitivity} \times \text{Specificity}}$ |

**Soru:**

Bir e-ticaret sitesi, ürünleri **Elektronik (Pozitif, 1)** ve **Diğer (Negatif, 0)** olarak sınıflandıran bir model geliştirmiştir. 150 ürünlük test setinde modelin tahmin sonuçları aşağıdaki gibidir:

| **Tahmin \ Gerçek** | **Elektronik (1)** | **Diğer (0)** |
|---|---|---|
| Tahmin Elektronik (1) | 40 | 15 |
| Tahmin Diğer (0) | 10 | 85 |

Ayrıca modelin eğitim sırasında **Kayıp (Loss) değeri = 0.05** olarak kaydedilmiştir.

1. Confusion Matrix'teki TP, TN, FP ve FN değerlerini belirtiniz.
2. Accuracy (Doğruluk) değerini hesaplayınız.
3. Precision (Kesinlik) değerini hesaplayınız.
4. Recall (Duyarlılık) değerini hesaplayınız.
5. F1 Score değerini hesaplayınız.
6. Loss değerinin model performansına etkisini açıklayınız.

## Çözüm:

### 1. Confusion Matrix değerleri:

- TP = 40
- TN = 85
- FP = 15
- FN = 10

### 2. Accuracy (Doğruluk):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{40 + 85}{40 + 85 + 15 + 10} = \frac{125}{150} \approx 0.833$$

**Doğruluk:** %83.3

3. Precision (Kesinlik):

$$Precision = \frac{TP}{TP + FP} = \frac{40}{40 + 15} = \frac{40}{55} \approx 0.727$$

Precision: %72.7

4. Recall (Duyarlılık):

$$Recall = \frac{TP}{TP + FN} = \frac{40}{40 + 10} = \frac{40}{50} = 0.8$$

Recall: %80

**5. F1 Score:**

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} = 2 \times \frac{0.727 \times 0.8}{0.727 + 0.8}$$

$$F1 = 2 \times \frac{0.5816}{1.527} \approx 0.761$$

**F1 Score:** %76.1

**6. Kayıp (Loss) açıklaması:**

- Loss değeri 0.05, modelin tahminleri ile gerçek değerler arasındaki ortalama hata seviyesini gösterir.
- Küçük bir loss, modelin tahminlerinin çoğunlukla gerçek sınıflara yakın olduğunu ifade eder ve modelin güvenilir olduğunu gösterir.

# AUC-ROC Eğrileri :

## ROC Eğrisi

ROC (Receiver Operating Characteristic) eğrisi, bir **sınıflandırma modelinin performansını eşik değerlerine göre görselleştiren bir grafiktir.**

- Yatay eksende: **False Positive Rate (FPR / Yanlış Pozitif Oranı)**
- Dikey eksende: **True Positive Rate (TPR / Gerçek Pozitif Oranı / Recall)**

ROC eğrisi, farklı sınıflandırma eşik değerlerinde modelin **pozitif ve negatif sınıfları ayırt etme başarısını** gösterir.

## True Positive Rate (TPR / Recall)

Modelin gerçekten pozitif olan örnekleri doğru tahmin etme oranıdır.

$$TPR = \frac{TP}{TP + FN}$$

- TP (True Positive): Modelin doğru şekilde pozitif tahmin ettiği örnekler
- FN (False Negative): Modelin yanlış şekilde negatif tahmin ettiği pozitif örnekler

**Anlamı:** Modelin pozitif örnekleri yakalama yeteneğini gösterir.

## False Positive Rate (FPR)

Modelin gerçekten negatif olan örnekleri yanlışlıkla pozitif tahmin etme oranıdır.

$$FPR = \frac{FP}{FP + TN}$$

- FP (False Positive): Modelin yanlış şekilde pozitif tahmin ettiği negatif örnekler
- TN (True Negative): Modelin doğru şekilde negatif tahmin ettiği örnekler

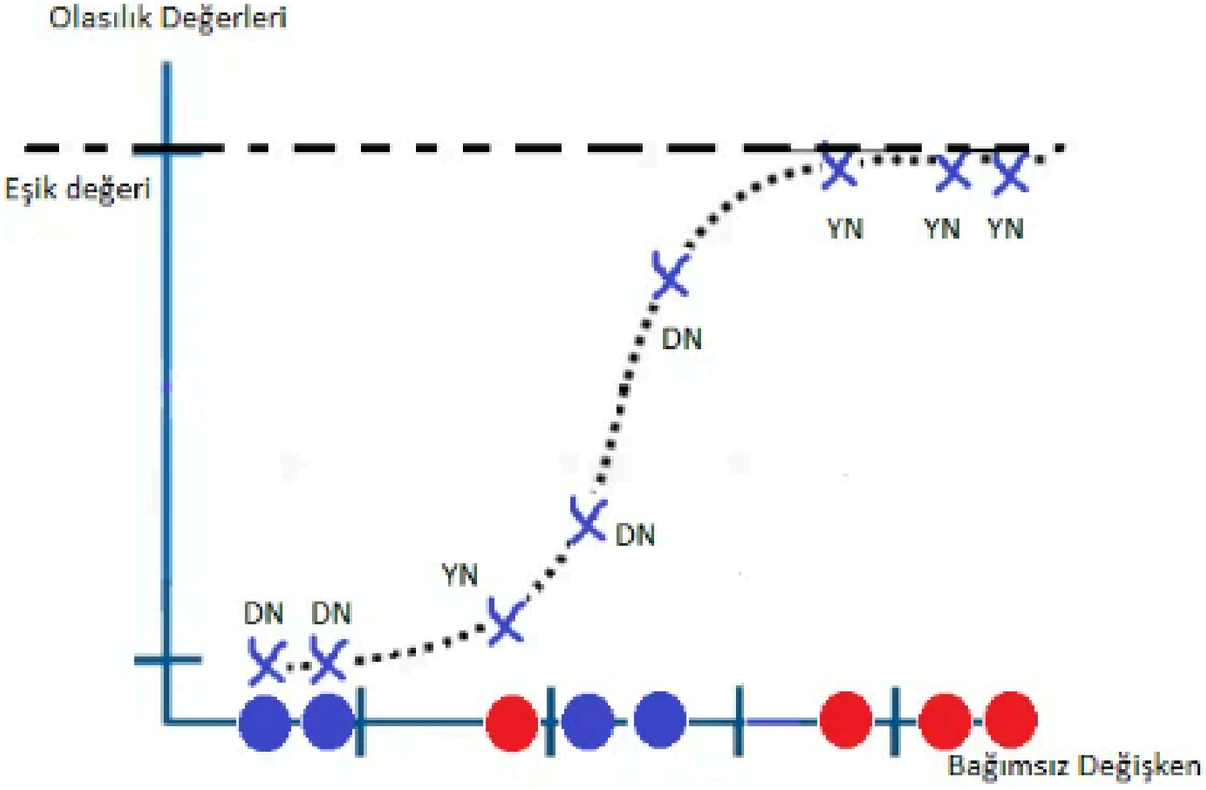
**Anlamı:** Modelin negatifleri yanlış pozitif olarak etiketleme hatasını gösterir.

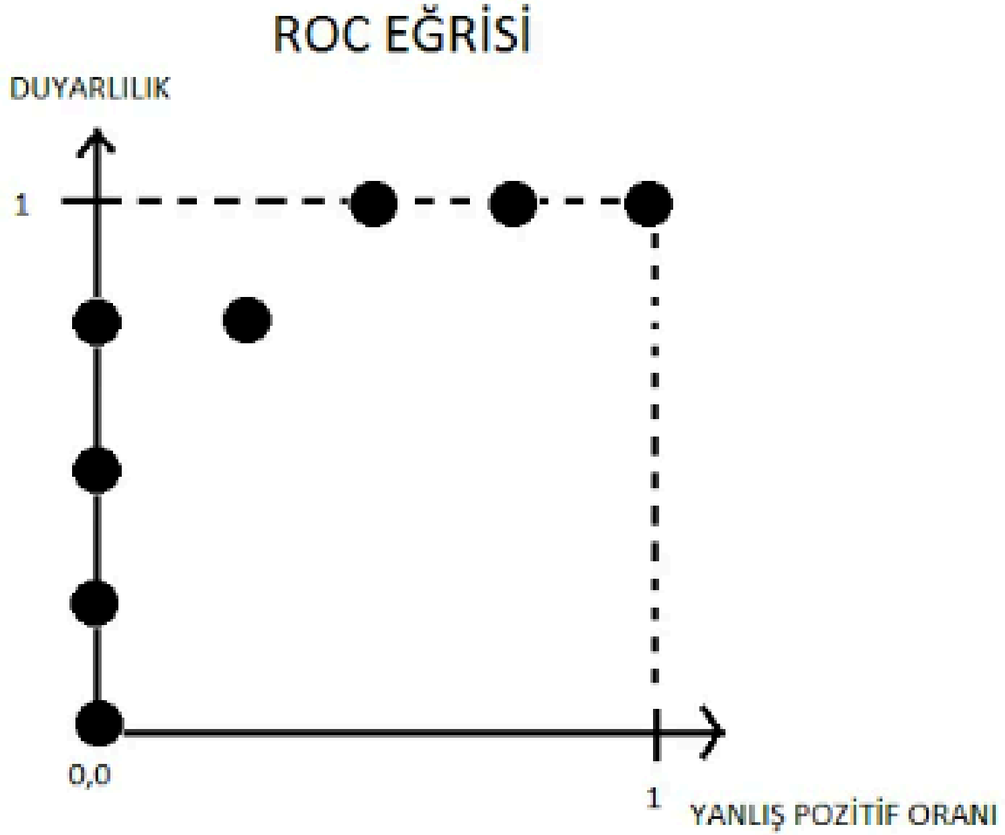
## Eşik (Threshold)

- Modelin bir örneği pozitif veya negatif olarak sınıflandırmak için kullandığı sınır değer.
- Örnek: Bir olasılık tahmin modeli, olasılık ≥ 0.5 ise pozitif, < 0.5 ise negatif sınıf olarak kabul edebilir.
- Eşik değiştirildiğinde TPR ve FPR değerleri değişir ve ROC eğrisi ortaya çıkar.

= gerçekte hasta

= gerçekte sağlıklı




Duyarlılık = 0 / 4 = 0

YPO = 0 / 4 = 0

# AUC (Area Under the Curve)

**AUC**, "Area Under the Curve" yani "Eğrinin Altındaki Alan" demektir.

- Genellikle **ROC eğrisi (Receiver Operating Characteristic Curve)** ile birlikte kullanılır.
- Bir sınıflandırma modelinin **pozitif ve negatif sınıfları ayırt etme başarısını sayısal olarak ölçer.**

## Hesaplanan Eğri :

- ROC eğrisi çizilir:
  - **Y ekseni:** True Positive Rate (TPR / Recall)
  - **X ekseni:** False Positive Rate (FPR)
- AUC, bu eğrinin altında kalan alanı ifade eder.

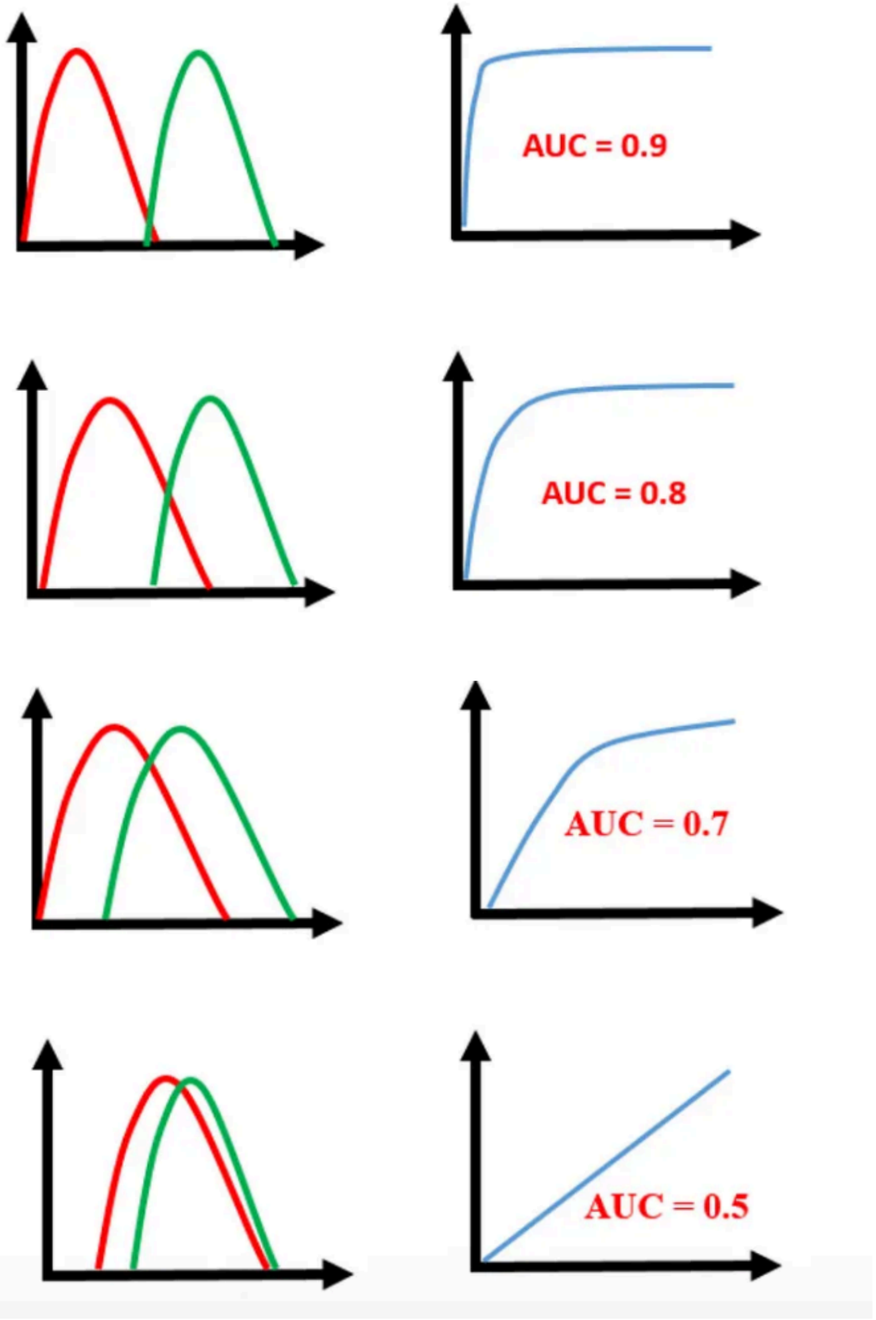
- **AUC = 1** → Mükemmel model: Tüm pozitif ve negatif örnekleri doğru sınıflandırır.
- **AUC = 0.5** → Rastgele tahmin yapan model: Modelin sınıflandırma gücü yoktur.
- **0.7 ≤ AUC < 0.8** → İyi model
- **0.8 ≤ AUC < 0.9** → Çok iyi model
- **AUC ≥ 0.9** → Mükemmel model

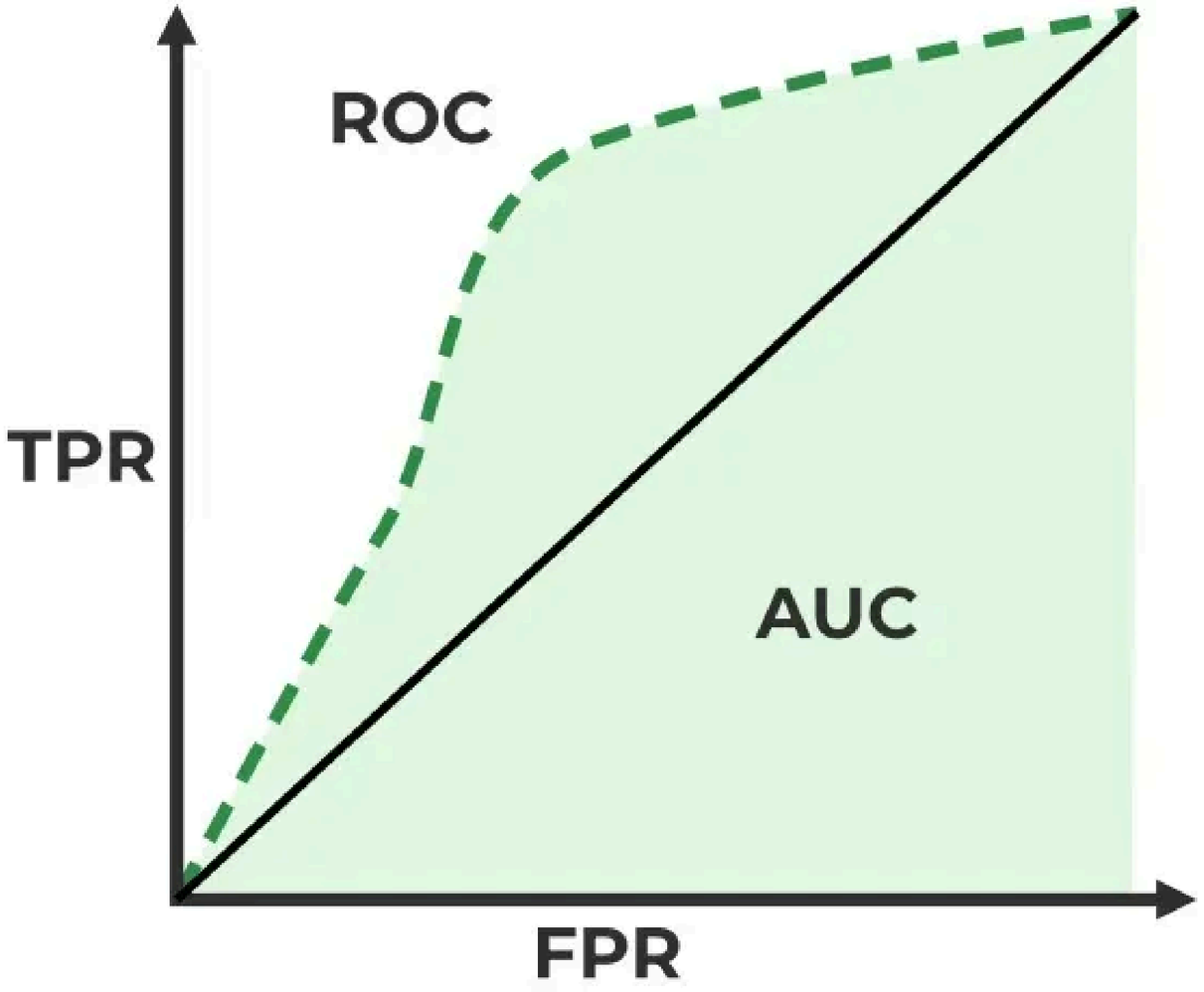
AUC, özellikle sınıflar arasında dengesizlik olduğunda **accuracy'den daha güvenilir bir performans ölçütüdür.**

# Özellikler :

1. **Sınıflar arası ayırt edebilirlik:** AUC, modelin rastgele seçilmiş bir pozitif örneği negatif bir örnekten ayırma olasılığını ifade eder.
2. **Eşik bağımsızdır:** Modelin tahmin eşik değerinden bağımsız olarak genel performansı gösterir.
3. **0.5–1 arası değer alır:** 0.5, şans seviyesini, 1 ise mükemmelliği gösterir.

- AUC, modelin **pozitif ve negatif sınıfları ayırt etme yeteneğini** özetler.
- Yüksek AUC → Model sınıflandırmada başarılıdır.
- ROC eğrisi ile birlikte kullanılır; eğrinin altında kalan alan, AUC değerini verir.


AUC = 0.9
AUC = 0.8
AUC = 0.7
AUC = 0.5


ROC
TPR
AUC
FPR

# Scikit-Learn (sklearn) :

**Scikit-Learn**, Python ile makine öğrenmesi uygulamaları geliştirmek için kullanılan **açık kaynaklı bir kütüphanedir.**

- Hem **denetimli (supervised)** hem de **denetimsiz (unsupervised)** öğrenme algoritmalarını içerir.
- Veri ön işleme, model eğitimi, model değerlendirme ve tahmin süreçlerini kolaylaştırır.

## Özellikleri

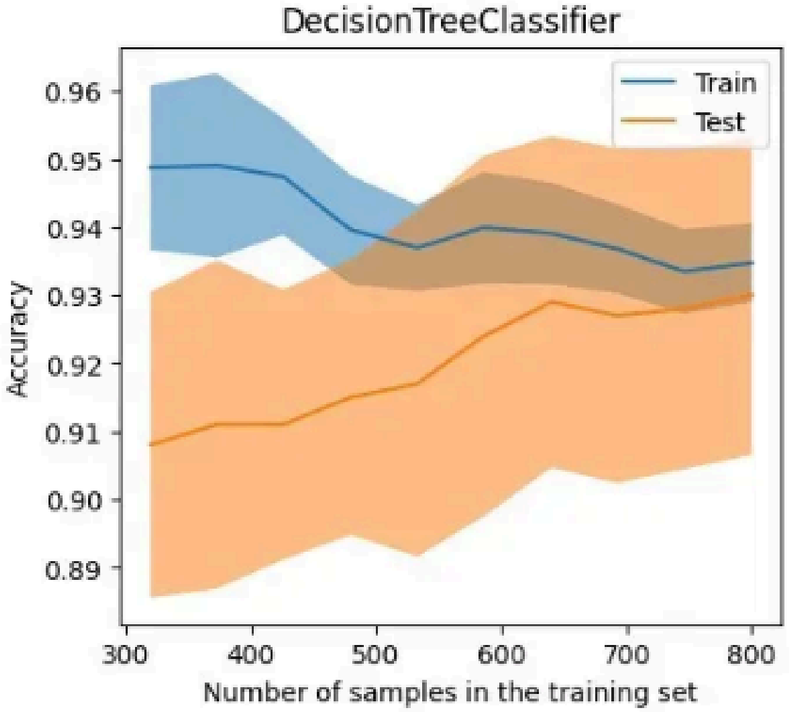
1. **Geniş Algoritma Desteği:**
    - Sınıflandırma: Logistic Regression, Random Forest, SVM, KNN vb.
    - Regresyon: Linear Regression, Ridge, Lasso vb.
    - Kümeleme: K-Means, DBSCAN, Hierarchical Clustering vb.
2. **Veri Ön İşleme:**
    - Ölçekleme: StandardScaler, MinMaxScaler
    - Kodlama: OneHotEncoder, LabelEncoder
    - Eksik değer doldurma: SimpleImputer

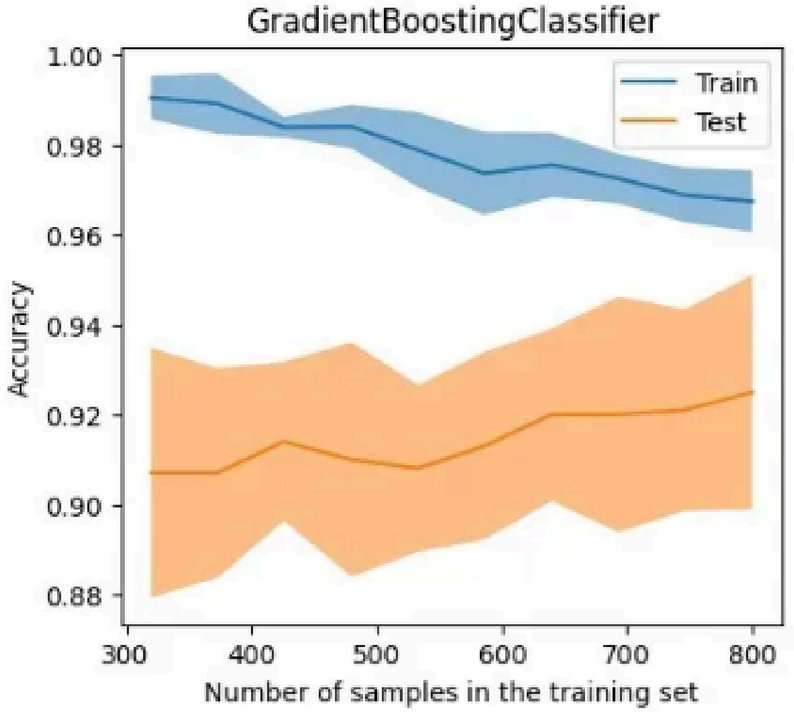
3. **Model Seçimi ve Değerlendirme:**
    - Veri seti bölme: `train_test_split`
    - Çapraz doğrulama: `cross_val_score`
    - Hiperparametre optimizasyonu: `GridSearchCV`, `RandomizedSearchCV`
4. **Model Değerlendirme Metrikleri:**
    - Sınıflandırma: Accuracy, Precision, Recall, F1 Score, ROC-AUC
    - Regresyon: MSE, RMSE, $R^2$

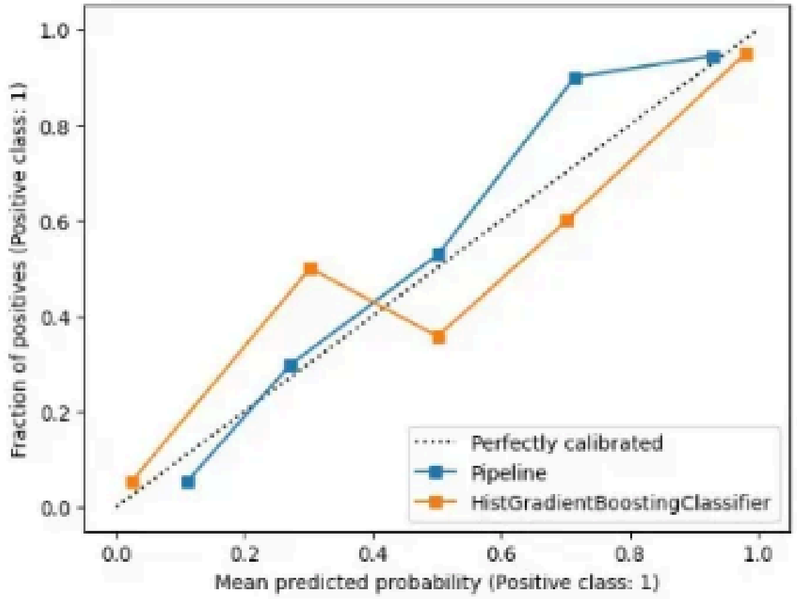
## Kurulum

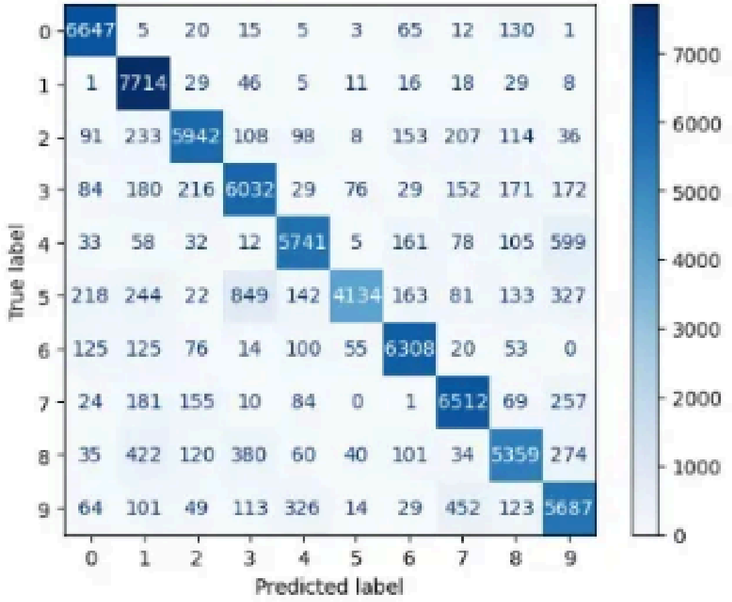
bash

```
pip install scikit-learn
```


DecisionTreeClassifier
Train
Test
Accuracy
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
300
400
500
600
700
800
Number of samples in the training set


GradientBoostingClassifier
Train
Test
Accuracy
1.00
0.98
0.96
0.94
0.92
0.90
0.88
300
400
500
600
700
800
Number of samples in the training set


Fraction of positives (Positive class: 1)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Mean predicted probability (Positive class: 1)
Perfectly calibrated
Pipeline
HistGradientBoostingClassifier


True label
Predicted label
0 6647 5 20 15 5 3 65 12 130 1
1 1 7714 29 46 5 11 16 18 29 8
2 91 233 5942 108 98 8 153 207 114 36
3 84 180 216 6032 29 76 29 152 171 172
4 33 58 32 12 5741 5 161 78 105 599
5 218 244 22 849 142 4134 163 81 133 327
6 125 125 76 14 100 55 6308 20 53 0
7 24 181 155 10 84 0 1 6512 69 257
8 35 422 120 380 60 40 101 34 5359 274
9 64 101 49 113 326 14 29 452 123 5687
0 1 2 3 4 5 6 7 8 9
7000
6000
5000
4000
3000
2000
1000
0

# Uygulama: Scikit-learn & Keras ile model sonuçlarının görselleştirilmesi :

### Adım 1: Gerekli Kütüphaneleri Yükleme

İlk olarak, projenin çalışması için gerekli olan tüm kütüphaneleri içe aktarırız. Bu adım, kullanacağımız tüm araçları çağırır.

- `tensorflow` & `keras` : Yapay sinir ağı modelini oluşturmak ve eğitmek için.
- `sklearn` : Veriyi hazırlamak ve modelin performansını ölçmek için.
- `numpy` : Sayısal verilerle çalışmak için.
- `matplotlib` & `seaborn` : Görselleştirmeleri oluşturmak için.

```
import tensorflow as tf
from tensorflow import keras
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

**Adım 2: Veri Kümesini Hazırlama**

Modelimizi eğitebilmek için öncelikle veriyi düzenlememiz gerekir. Bu aşama, veriyi modelin anlayacağı formata dönüştürür.

- **Veriyi Yükleme:** Scikit-learn kütüphanesinden **Iris veri kümesini** yüklüyoruz. Bu veri kümesi, üç farklı çiçek türünü ayırt etmeye yarar.
- **Veriyi Bölme:** Veriyi, modelin öğrenmesi için **eğitim seti** ve performansını test etmek için **test seti** olmak üzere ikiye ayırırız.
- **Etiketleri Dönüştürme:** Keras modelleri, sınıf etiketlerinin **one-hot encoding** formatında olmasını bekler. Bu, her bir sınıfı ayrı bir binary vektörle temsil etmektir (örneğin, 0 yerine `[1, 0, 0]`).

```
from sklearn.datasets import load_iris
iris = load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

y_train_one_hot = keras.utils.to_categorical(y_train, num_classes=3)
y_test_one_hot = keras.utils.to_categorical(y_test, num_classes=3)

print(f"Eğitim seti boyutu: {X_train.shape}")
print(f"Test seti boyutu: {X_test.shape}")
```

**Adım 3: Keras Modelini Oluşturma ve Eğitme**

Veri hazırlandıktan sonra, onu öğrenmeye hazır bir model oluştururuz.

- **Modelin Yapısı:** Bir **Sequential** model oluştururuz. Bu, katmanları sırayla ekleyebileceğimiz anlamına gelir.
    - İlk katman **(Dense)**, 10 nöron içerir ve modelin veriden özellikler öğrenmesini sağlar.
    - İkinci ve son katman **(Dense)**, 3 nöron içerir (her bir çiçek türü için bir tane). Bu katman, modelin nihai tahminini yapar.
- **Modeli Derleme:** Modelin nasıl öğreneceğini belirtiriz. Bu adımda **optimizer**, **loss** fonksiyonu ve izlenecek **metrics** belirlenir.
- **Modeli Eğitme:** Modeli, eğitim verileriyle besler ve belirli bir döngü sayısı (epoch) boyunca eğitiriz. Bu süreçte modelin performansı kaydedilir.

```
model = keras.Sequential([
    keras.layers.Dense(10, activation='relu', input_shape=(4,)),
    keras.layers.Dense(3, activation='softmax')
])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history = model.fit(X_train, y_train_one_hot,
                    epochs=50,
                    validation_data=(X_test, y_test_one_hot),
                    verbose=0)
```

**Adım 4: Model Sonuçlarını Görselleştirme**

Şimdi, asıl konumuz olan görselleştirme aşamasına geçiyoruz. Bu aşamada, **Matplotlib** ve **Seaborn** kütüphanelerini kullanarak modelin performansını grafiklerle analiz ederiz.

**Görselleştirme 1: Eğitim Geçmişi Grafikleri**

Bu grafikler, modelin eğitim ve doğrulama setleri üzerindeki **kayıp (loss)** ve **doğruluk (accuracy)** değerlerinin eğitim süresince nasıl değiştiğini gösterir. Bu, modelin **aşırı öğrenme (overfitting)** gibi sorunlar yaşayıp yaşamadığını anlamak için çok önemlidir.

```python
plt.figure(figsize=(12, 5))

# Kayıp (Loss) grafiği
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Eğitim Kaybı')
plt.plot(history.history['val_loss'], label='Doğrulama Kaybı')
plt.title('Model Kaybı (Loss)')
plt.ylabel('Kayıp Değeri')
plt.xlabel('Epoch')
plt.legend()

# Doğruluk (Accuracy) grafiği
plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Eğitim Doğruluğu')
plt.plot(history.history['val_accuracy'], label='Doğrulama Doğruluğu')
plt.title('Model Doğruluğu (Accuracy)')
plt.ylabel('Doğruluk Değeri')
plt.xlabel('Epoch')
plt.legend()

plt.tight_layout()
plt.show()
```

**Görselleştirme 2: Karışıklık Matrisi (Confusion Matrix)**

Karışıklık matrisi, modelin her bir sınıfı ne kadar doğru ve yanlış sınıflandırdığını net bir şekilde gösteren bir tablodur. Bu matris, hangi sınıfların birbiriyle karıştırıldığını anlamamıza yardımcı olur.

Python

```
# Test setindeki tahminleri al
y_pred_probs = model.predict(X_test)
y_pred = np.argmax(y_pred_probs, axis=1)

# Karışıklık matrisini oluştur
cm = confusion_matrix(y_test, y_pred)

# Matrisi görselleştir
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=iris.target_names,
            yticklabels=iris.target_names)
plt.title('Karışıklık Matrisi')
plt.ylabel('Gerçek Sınıflar')
plt.xlabel('Tahmin Edilen Sınıflar')
plt.show()
```

**Sonuçların Yorumlanması**

- **Grafikler:** Kayıp değerlerinin (loss) düşmesi ve doğruluk değerlerinin (accuracy) yükselmesi, modelin başarılı bir şekilde öğrendiğini gösterir. Eğer eğitim ve doğrulama grafikleri birbirinden çok uzaklaşırsa, bu durum **aşırı öğrenme** (modelin sadece eğitim verisini ezberlediği) anlamına gelir.
- **Karışıklık Matrisi:** Bu matrisin ana köşegenindeki (sol üstten sağ alta) sayılar, modelin doğru tahminlerini temsil eder. Diğer yerlerdeki sayılar ise yanlış tahminlerdir. Bu matris sayesinde modelin hangi sınıfları birbiriyle karıştırdığını kolayca anlayabiliriz. Bu örnekte, modelin tüm tahminlerinin doğru olduğunu görüyoruz.