

Makine Öğrenimi Giriş

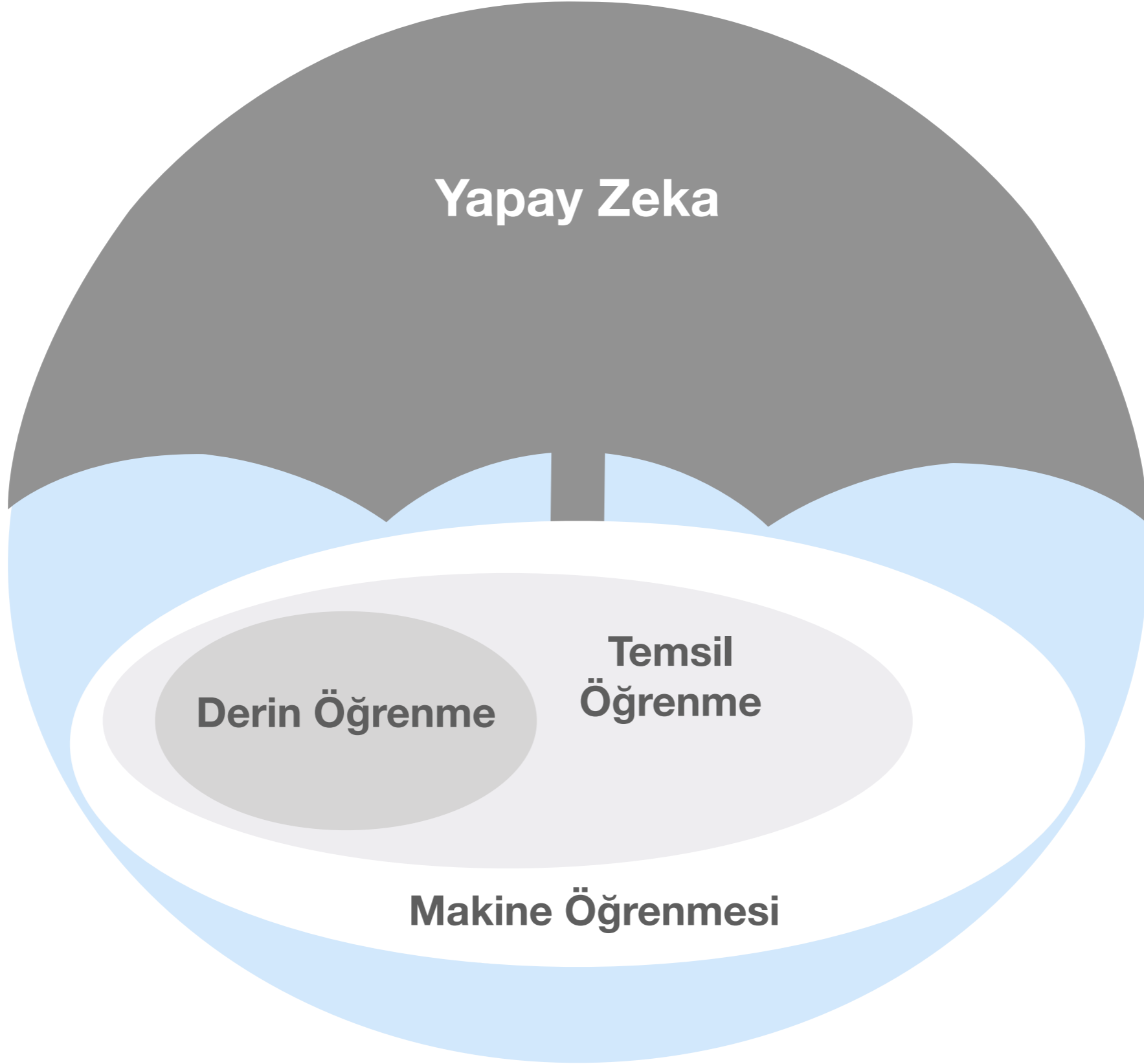
Ece Asilar

Hanyang University

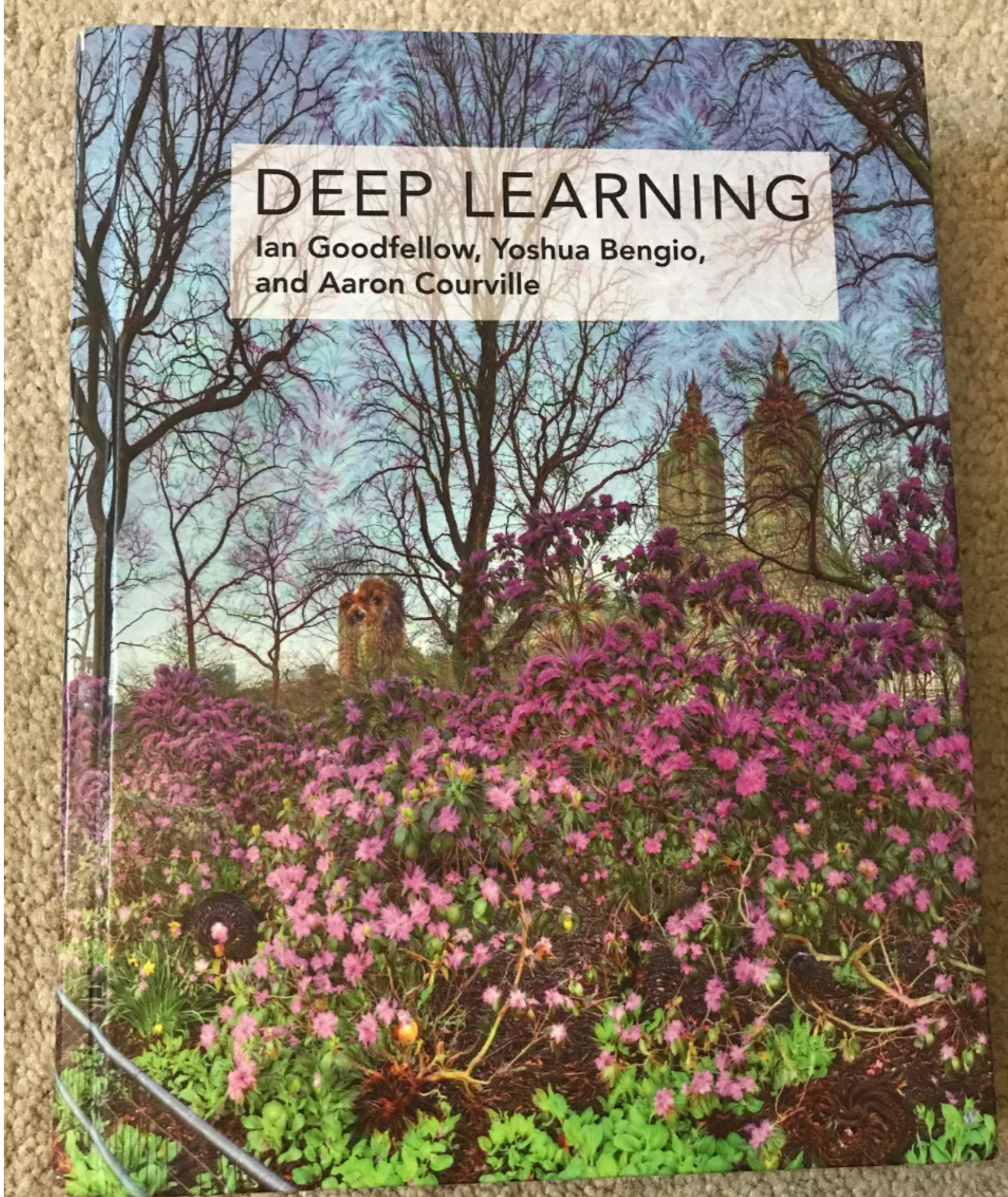
CERN

ece.asilar@cern.ch



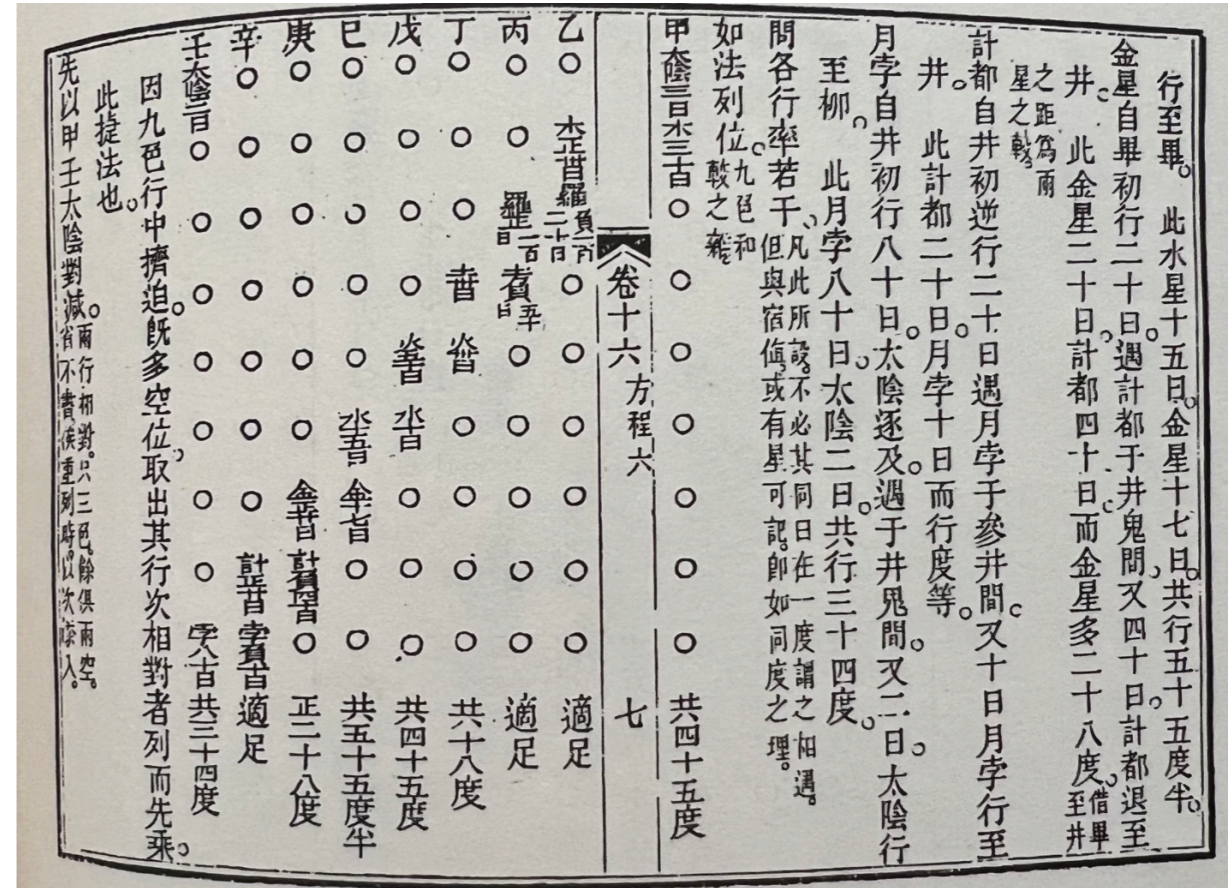


- Tüm seviyeler için uygun
- Matematiğini öğrenmek için şahane



birazcık tarih...

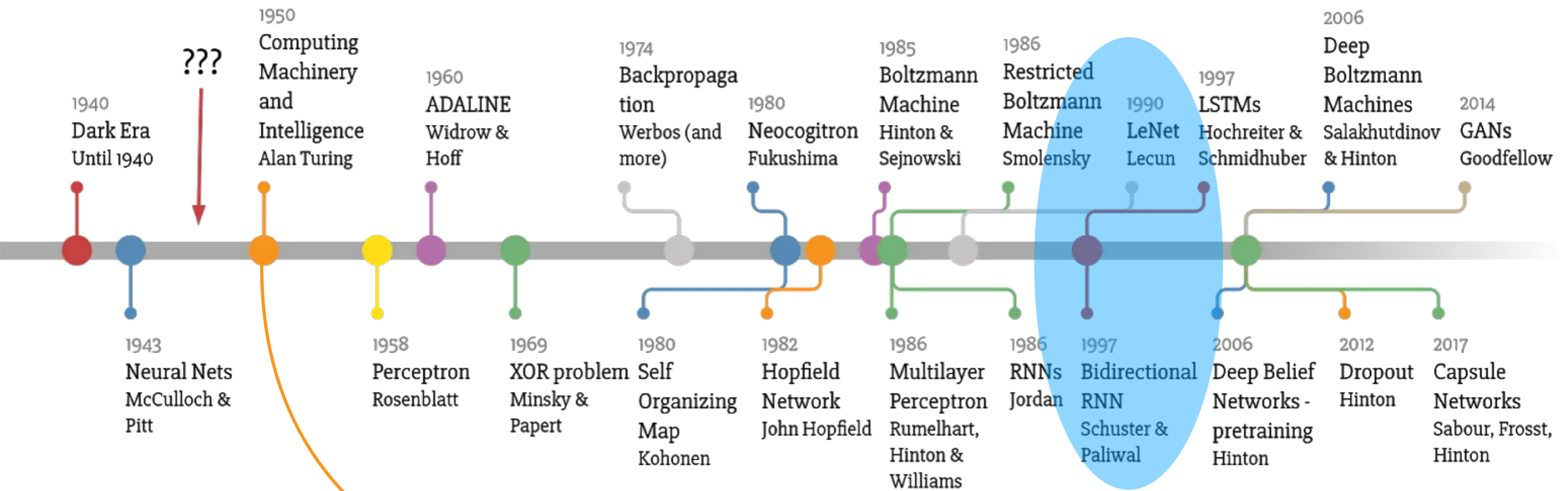
- René Descartes (1596–1650) koordinat düzlemi, cebirsel denklemlerin geometrik temsili.
- Carl Friedrich Gauss (1777–1855) özellikle doğrusal denklemler sistemlerinin çözümünde ve Gauss eliminasyon yönteminde katkıda bulunmuştur.
- Augustin-Louis Cauchy (1789–1857) matris teorisi ve determinantlar.
- Arthur Cayley (1821–1895) ve James Joseph Sylvester (1814–1897) matris teorisini tanıtmış ve modern lineer cebirin temel unsurları olan doğrusal dönüşümleri incelemiştir.



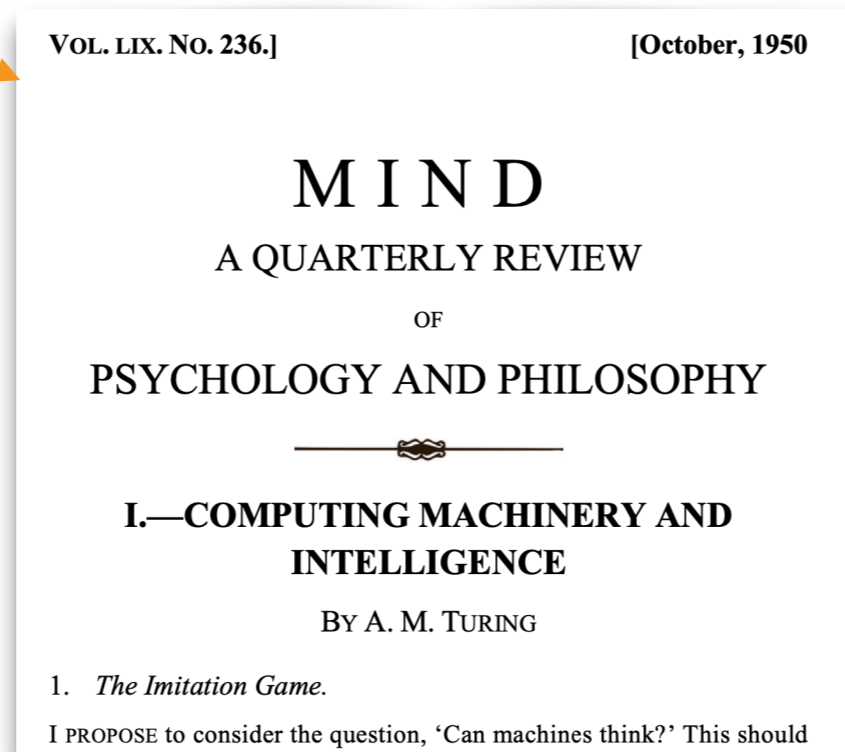
Nine Chapters on the Mathematical Arts (Jiuzhang suanshu 九章算術, c. 1st century CE

Bulmaca olsun:
Önce kim ?

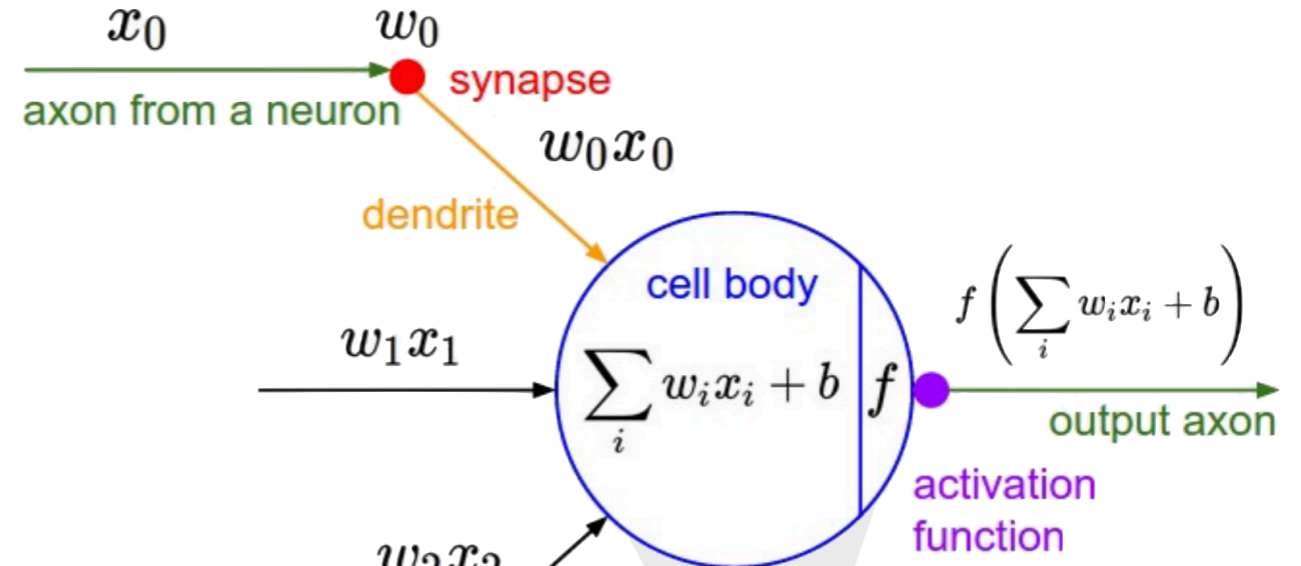
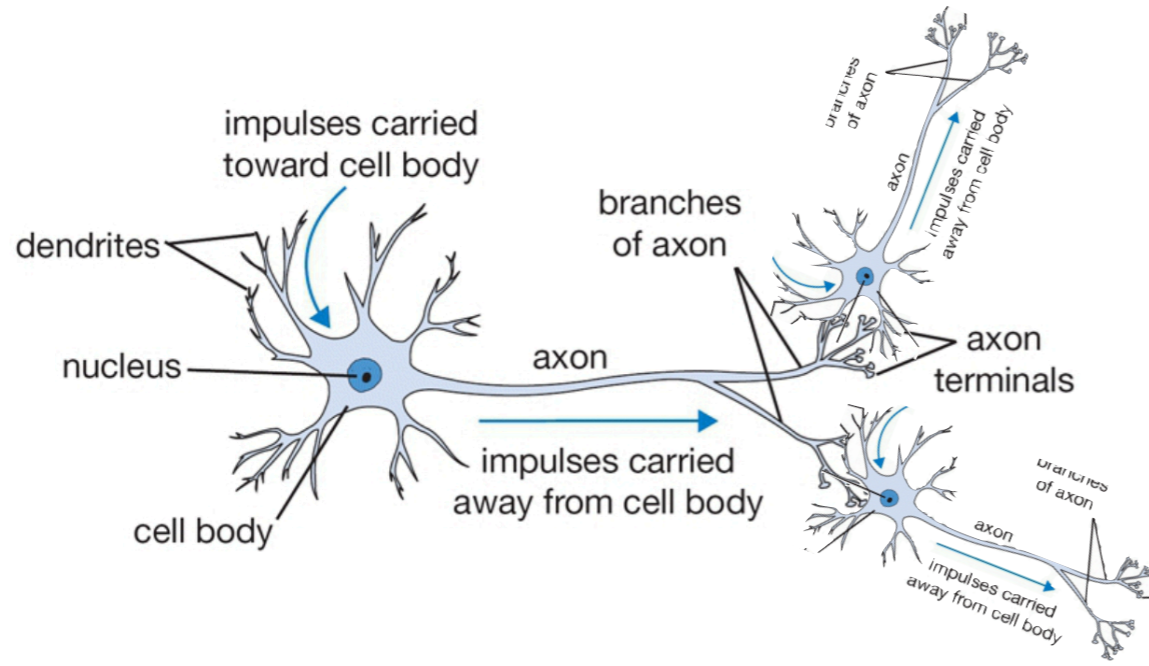
birazcık tarih...



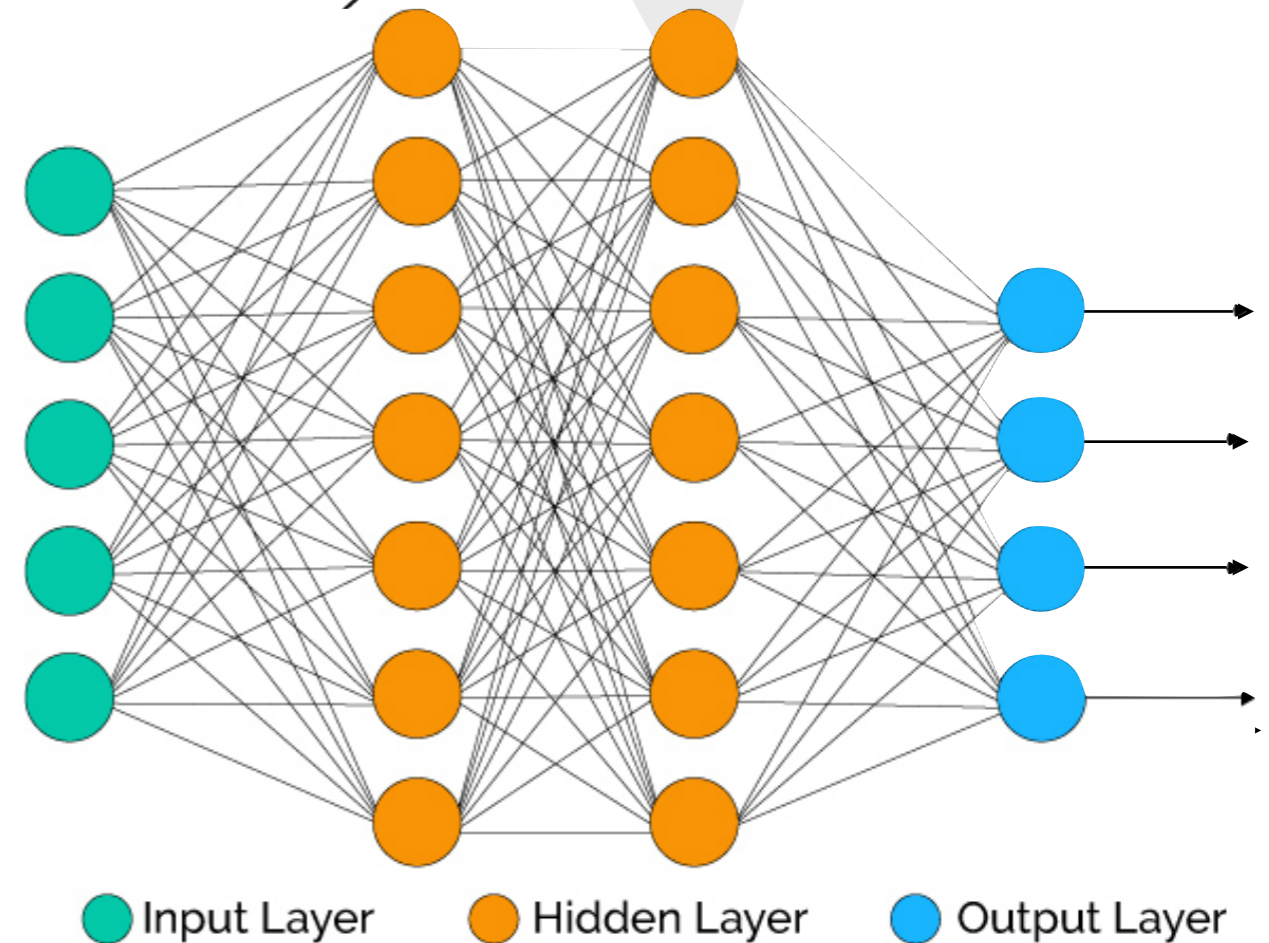
Made by Favio Vázquez



Derin Öğrenme Nedir ?



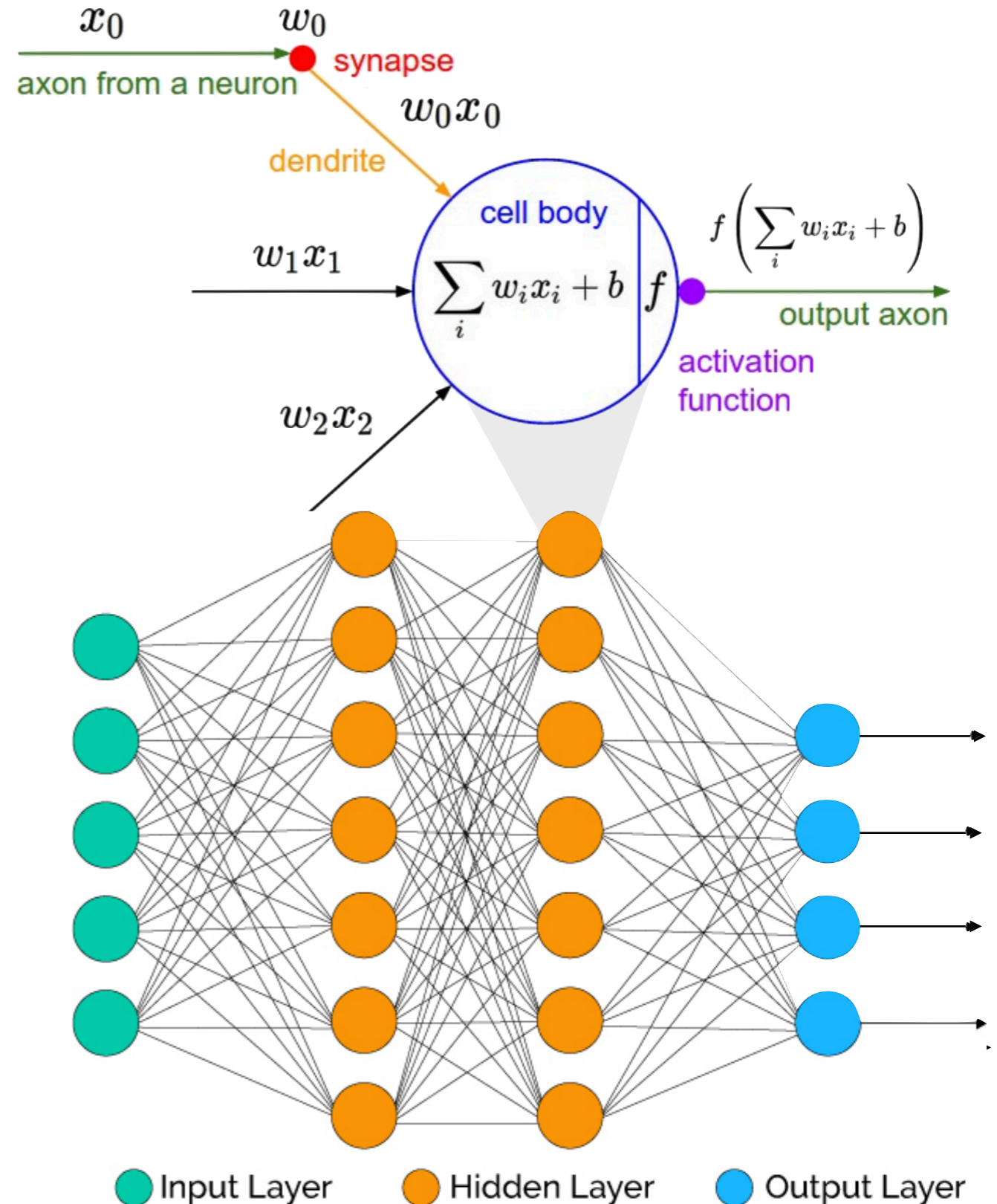
- Verileriniz için iyi temsiller (özellikler) bulma
- Birden fazla katmanlı özellik öğrenme
- Model, yalnızca özelliklerin izin verdiği kadar iyi olabilir



Derin Öğrenme Nedir ?

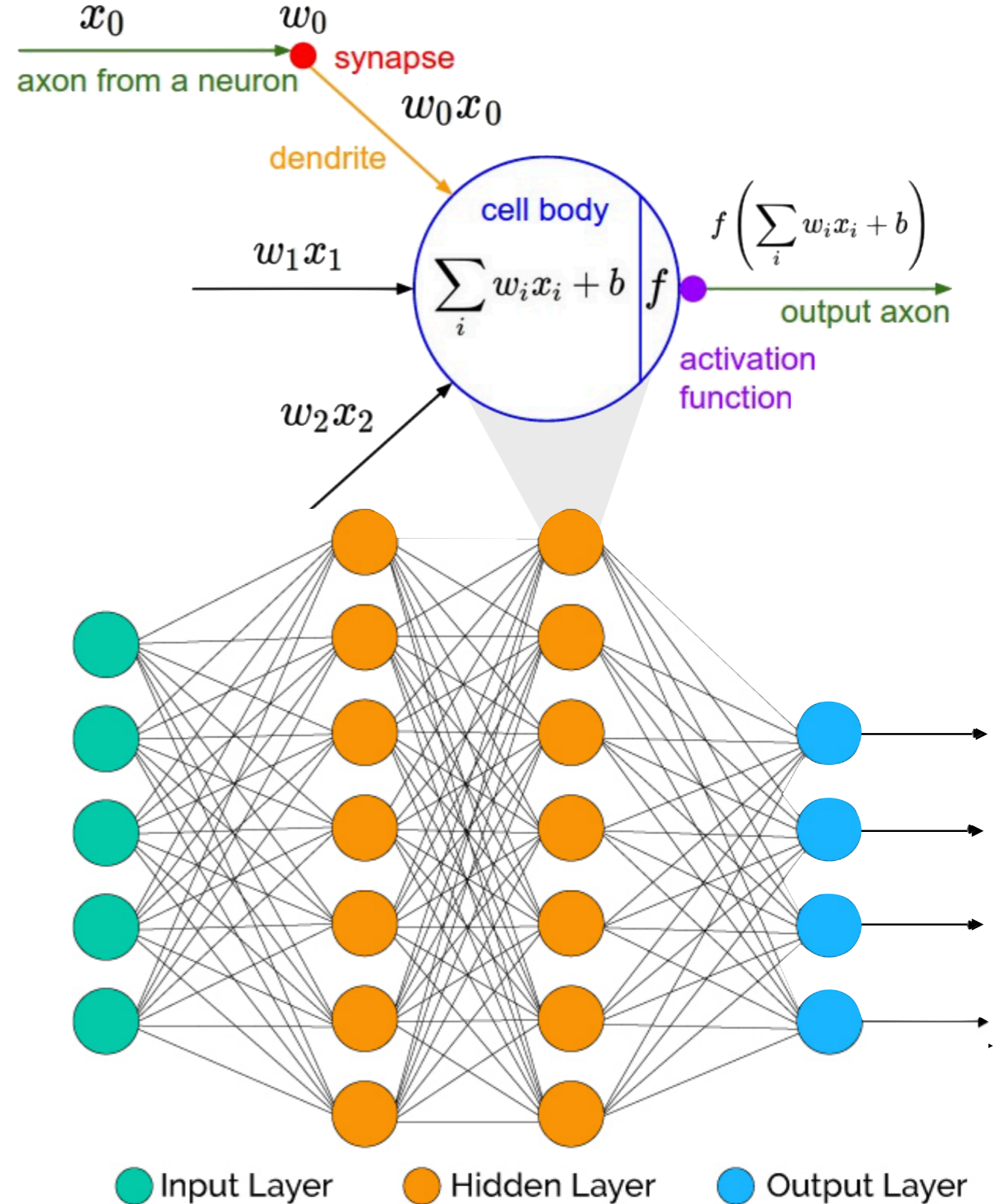
Burada:

- x : Girdi vektörü
- W : Ağırlık matrisi
- b : Bias terimi
- $f(\cdot)$: Aktivasyon fonksiyonu



Model inşa etmek için ne gerekli ?

- Etiketlenmiş veri (zorunlu değil)
- Girdileri başlatan ağırlıklara ihtiyacım var.
- Katmanlar arasındaki bağların dağılımını tanımlamam gerekiyor
- Her bir hücrenin içindeki fonksiyonu seçmem gerekiyor.
- Son katmandaki fonksiyonları ayrıca tanımlamalıyım
- Kayıp fonksiyonu belirlemeliyim
- Optimizasyon algoritması seçmem gerekiyor



Derin öğrenmede temel görevler

- Sınıflandırma (Classification)
- Regresyon (Regression)
- Kümeleme (Clustering)
- Anomali Tespiti (Anomaly Detection)
- Görüntü ve Ses İşleme (Image and Speech Processing)
- Doğal Dil İşleme (Natural Language Processing - NLP)
- Zaman Serisi Tahmini (Time Series Forecasting)

Temel bilgiler

●Aktivasyon fonksiyonları

Neden Aktivasyon Fonksiyonları?

- Doğrusal olmayan dönüşümler sağlar.
- Katmanların daha karmaşık özellikleri öğrenmesine olanak tanır.

Önemli Aktivasyon Fonksiyonları:

- Sigmoid

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- 0 ile 1 arasında değer alır.
- Küçük değişimlere duyarlı ancak **gradyan kaybolma** problemi yaşanabilir.
- **ReLU (Rectified Linear Unit)**

$$f(x) = \max(0, x)$$

- Negatif değerleri sıfırlar, pozitifleri olduğu gibi bırakır.
- Daha hızlı öğrenme sağlar ve gradyan kaybolma problemini azaltır.
- **Softmax** (Çok sınıflı sınıflandırmalarda kullanılır)

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

- Çıktıları olasılıklara dönüştürür (toplamı 1 olacak şekilde normalizasyon yapar).

● Aktivasyon fonksiyonları - python

```
import numpy as np

# Sigmoid Function
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# ReLU Function
def relu(x):
    return np.maximum(0, x)

# Softmax Function
def softmax(x):
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum()

# Generate input range
x = np.linspace(-10, 10, 100)

# Compute function values
y_sigmoid = sigmoid(2)
y_relu = relu(2)
y_softmax = softmax(2)

print(y_sigmoid , y_relu, y_softmax)
```

●Aktivasyon fonksiyonları - python

```
import numpy as np

# Sigmoid Function
def sigmoid(x):
    return 1 / (1 + np.exp(-x))

# ReLU Function
def relu(x):
    return np.maximum(0, x)

# Softmax Function
def softmax(x):
    e_x = np.exp(x - np.max(x))
    return e_x / e_x.sum()

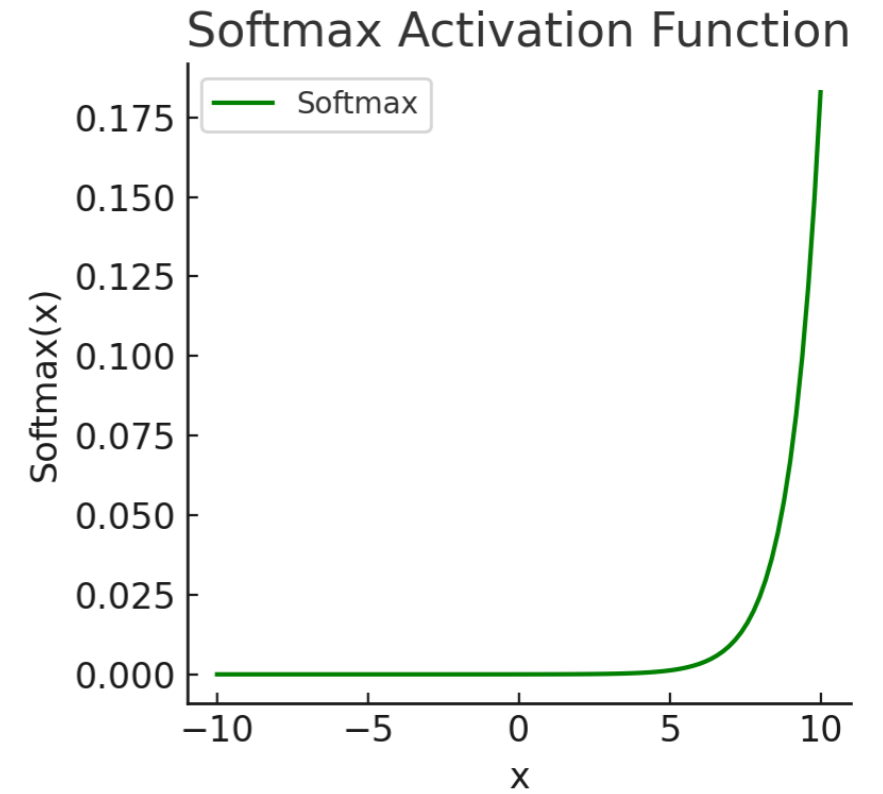
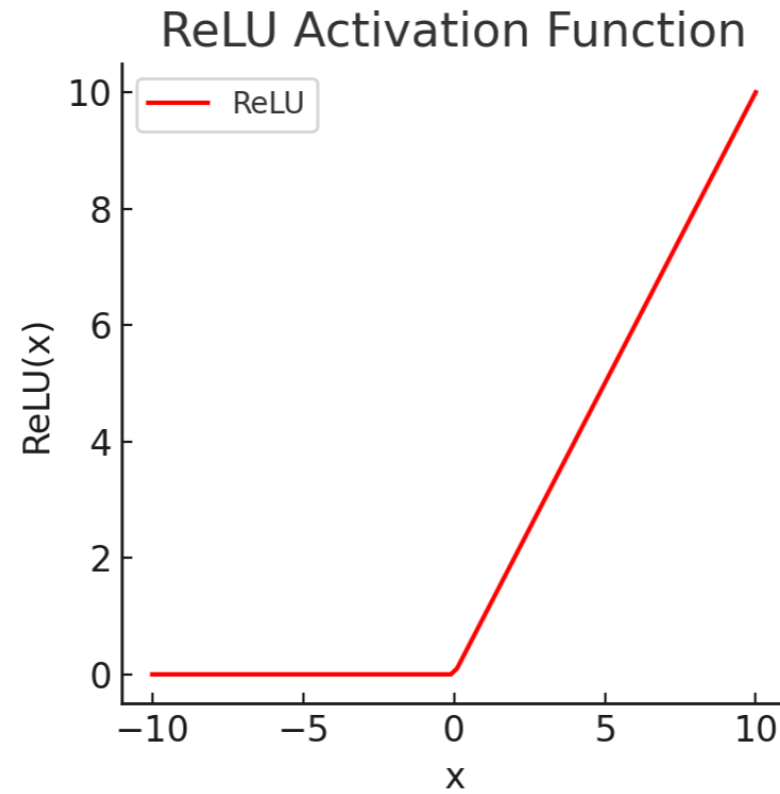
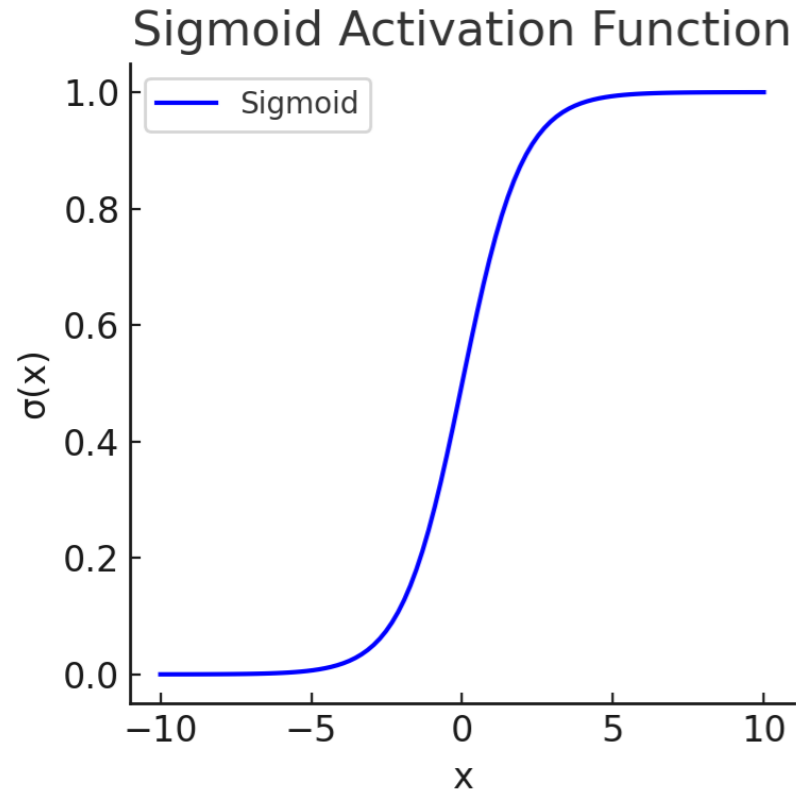
# Generate input range
x = np.linspace(-10, 10, 100)

# Compute function values
y_sigmoid = sigmoid(2)
y_relu = relu(2)
y_softmax = softmax(2)

print(y_sigmoid , y_relu, y_softmax)
```



● Aktivasyon fonksiyonları - python



Burada, **Sigmoid**, **ReLU** ve **Softmax** aktivasyon fonksiyonlarının grafiklerini oluşturdum:

- **Sigmoid**: Küçük girişlerde neredeyse sıfıra, büyük girişlerde neredeyse bire yakınsar.
- **ReLU**: Negatif değerleri sıfıra kırpar, pozitif değerleri değiştirmez.
- **Softmax**: Çıktıları olasılıklara dönüştürerek toplamı 1 yapar.

● Geri yayılım Algoritması ve Optimizasyon (Back propagation)

Nasıl Çalışır?

- İleri yayılım: Girdi, ağdan geçirilerek çıktı hesaplanır.
- Hata hesaplama: Tahmin edilen ve gerçek değer arasındaki fark hesaplanır.
- Geri yayılım: Zincir kuralı kullanılarak hatanın her ağırlık üzerindeki türevleri hesaplanır.
- Ağırlık güncelleme: Gradyan iniş algoritması (Gradient Descent) kullanılarak ağırlıklar güncellenir.



**Minimizasyon (en küçükleme)
deyince aklınıza ne geliyor ?**

● Geri yayılım Algoritması ve Optimizasyon (Back propagation)

◆ Gradyan İniş Güncelleme Kuralı:

$$W = W - \eta \frac{\partial L}{\partial W}$$

Burada:

- W : Güncellenen ağırlıklar
- η (eta) : Öğrenme oranı
- L : Kayıp fonksiyonu

Optimizasyon Algoritmaları

- **SGD (Stochastic Gradient Descent)**
- **Adam (Adaptive Moment Estimation)** (En çok kullanılan, hızlı yakınsama sağlar)
- **RMSProp (Root Mean Square Propagation)**

● Geri yayılım Algoritması ve Optimizasyon

$$y = \sigma(wx + b)$$

Eğer kayıp fonksiyonu $L = (y - y_{true})^2$ yani ortalama kare hata (MSE) olsun.

Ağırlık w için türev almak istediğimizde:

1 Kayıp fonksiyonunun çıkışa göre türevi:

$$\frac{\partial L}{\partial y} = 2(y - y_{true})$$

2 Çıkışın gizli katmana göre türevi (aktivasyon fonksiyonu türevi):

$$\frac{\partial y}{\partial h} = \sigma'(h)$$

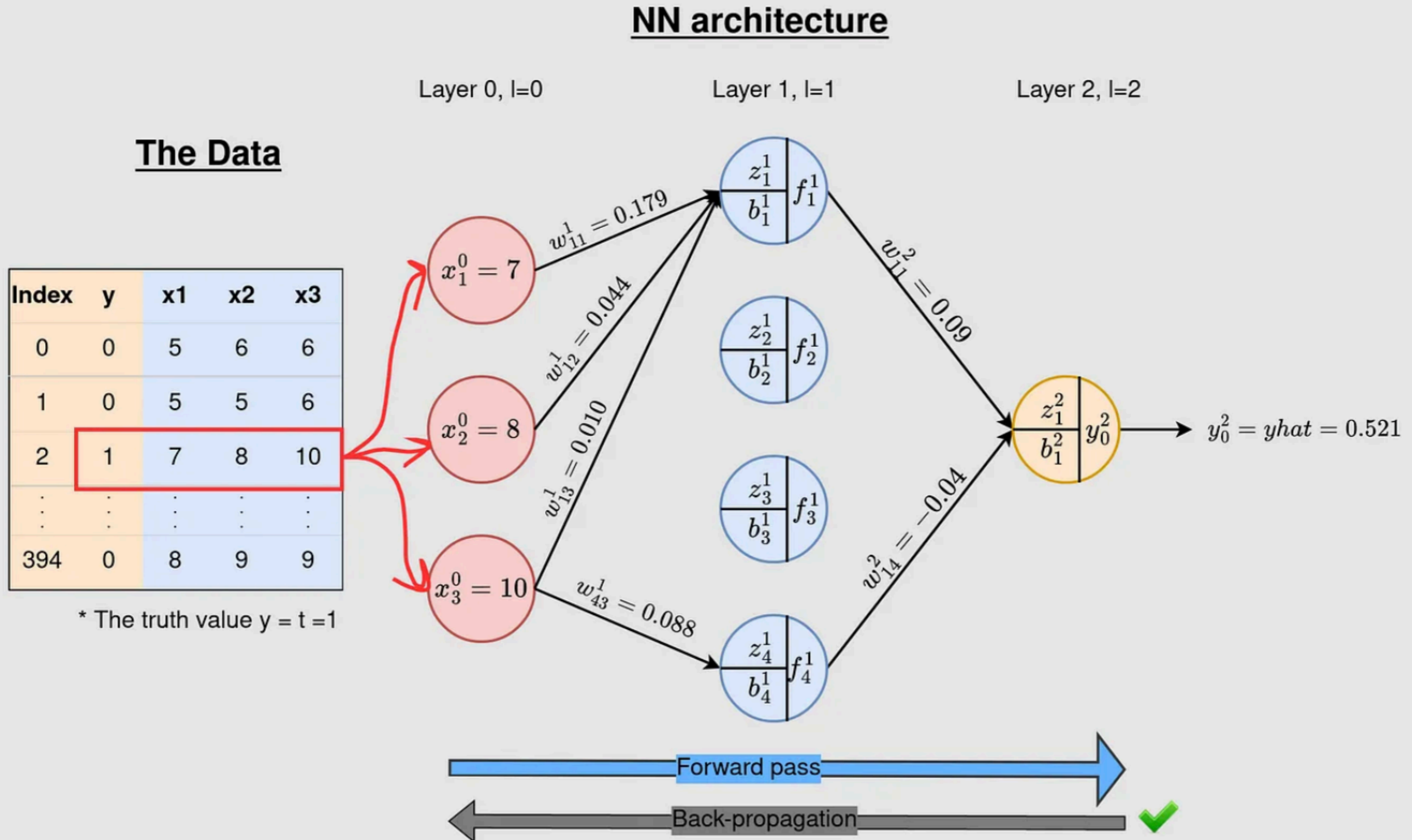
3 Gizli katmanın ağırlığa göre türevi:

$$\frac{\partial h}{\partial w} = x$$

Sonuç olarak:

$$\frac{\partial L}{\partial w} = 2(y - y_{true}) \cdot \sigma'(h) \cdot x$$

● Geri yayılım Algoritması ve Optimizasyon



- ◆ Sol tarafta **veri seti (The Data)** gösteriliyor.
- ◆ Her satırda:
 - **y**: Gerçek çıktı (etiket)
 - **x1, x2, x3**: Modelin giriş özellikleri (features)

İşlenen veri noktası:

- Seçilen örnek (Index 2) $\rightarrow y = 1, x_1 = 7, x_2 = 8, x_3 = 10$
- Bu örnek sinir ağına giriş olarak veriliyor.

◆ Her giriş nöronu, ilk gizli katmandaki (hidden layer) nöronlara bağlanıyor.

◆ Her bağlantının ağırlığı var:

- $w_{11}^1 = 0.179$
- $w_{12}^1 = 0.044$
- $w_{13}^1 = 0.010$
- $w_{14}^1 = 0.088$

📌 **Nöronların hesaplama adımı:**

Her nöron için şu formül uygulanıyor:

$$z = \sum (w_i \cdot x_i) + b$$

Burada:

- $w_i \rightarrow$ Ağırlıklar
- $x_i \rightarrow$ Giriş özellikleri
- $b \rightarrow$ Bias terimi

◆ Gizli katmandaki **çıktılar ikinci katmana iletiliyor.**

◆ **Bağlantı ağırlıkları (w)** ile tekrar çarpılıyor:

- $w_{11}^2 = 0.09$
- $w_{14}^2 = -0.04$

◆ **Çıktı nöronu (y hat = 0.521)** hesaplanıyor.

- **y hat (\hat{y})** \rightarrow Modelin tahmini sonucu.
- Burada $\hat{y} = 0.521$ olarak hesaplanmış.
- Gerçek değer $y = 1$ idi.

◆ Hata hesaplanıyor:

$$L = (y - \hat{y})^2$$

- Gerçek değer $y = 1$ ve model tahmini $\hat{y} = 0.521$
- Hata (L) büyükse, model öğrenmesi için geri yayılım başlar.

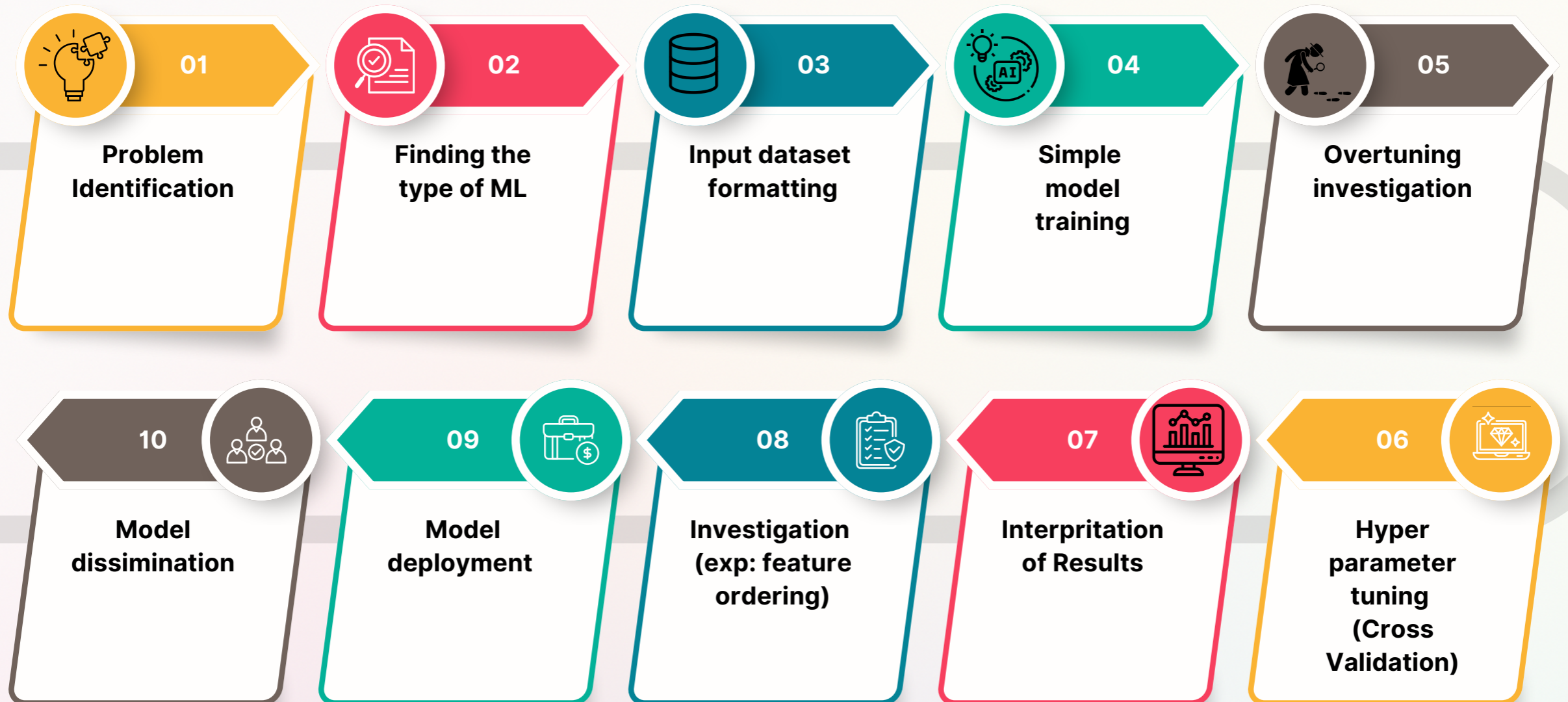
◆ Geri yayılım başlar:

- Hatanın türevi hesaplanır.
- Hata, zincir kuralı ile geriye doğru yayılır.
- **Ağırlıklar güncellenir:**

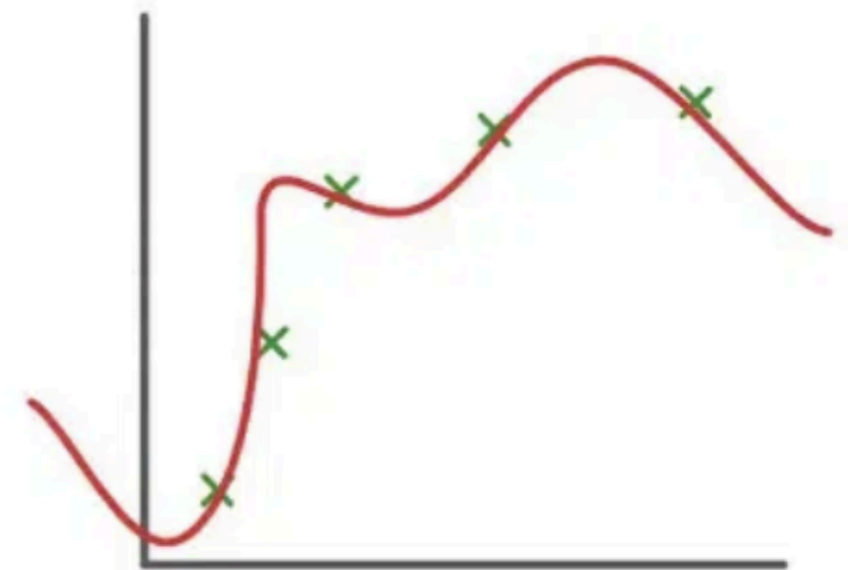
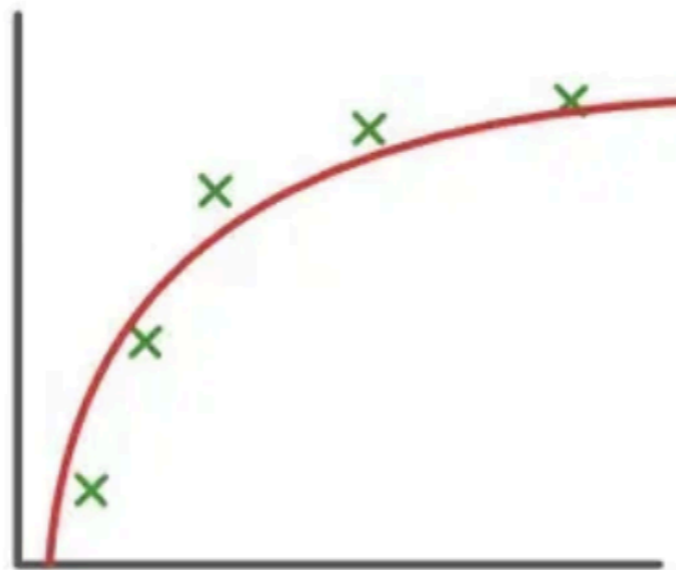
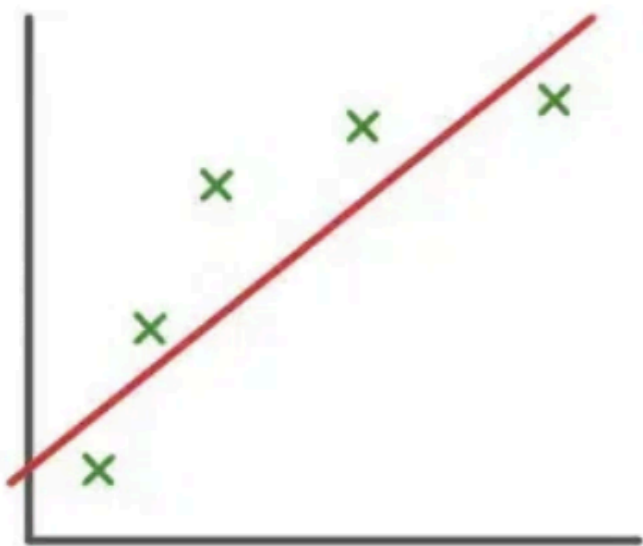
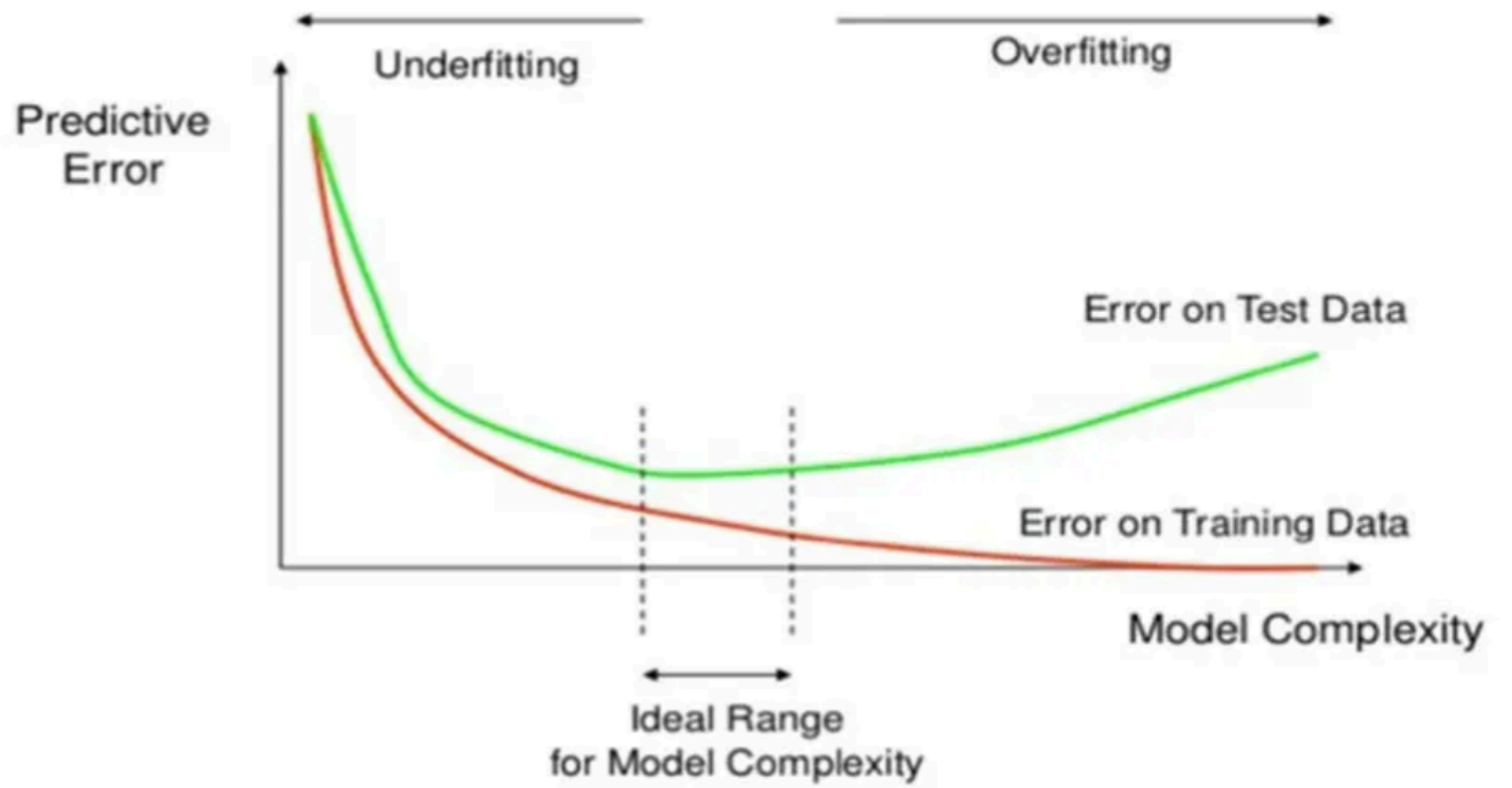
$$w = w - \eta \frac{\partial L}{\partial w}$$

- $\eta \rightarrow$ Öğrenme oranı
- $\frac{\partial L}{\partial w} \rightarrow$ Hatanın ağırlıklara etkisi

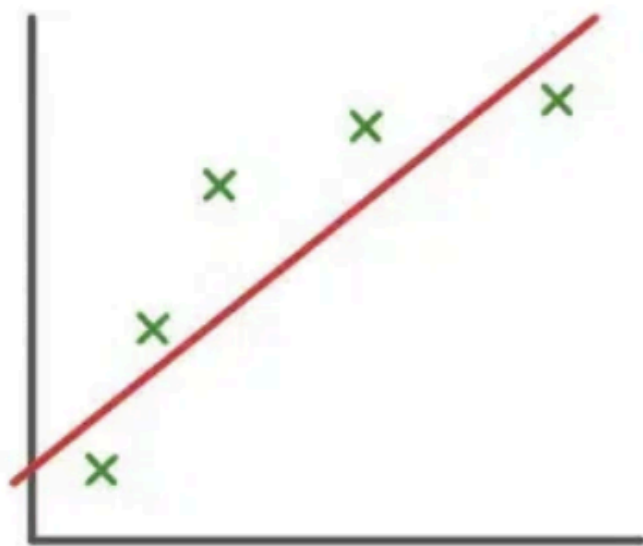
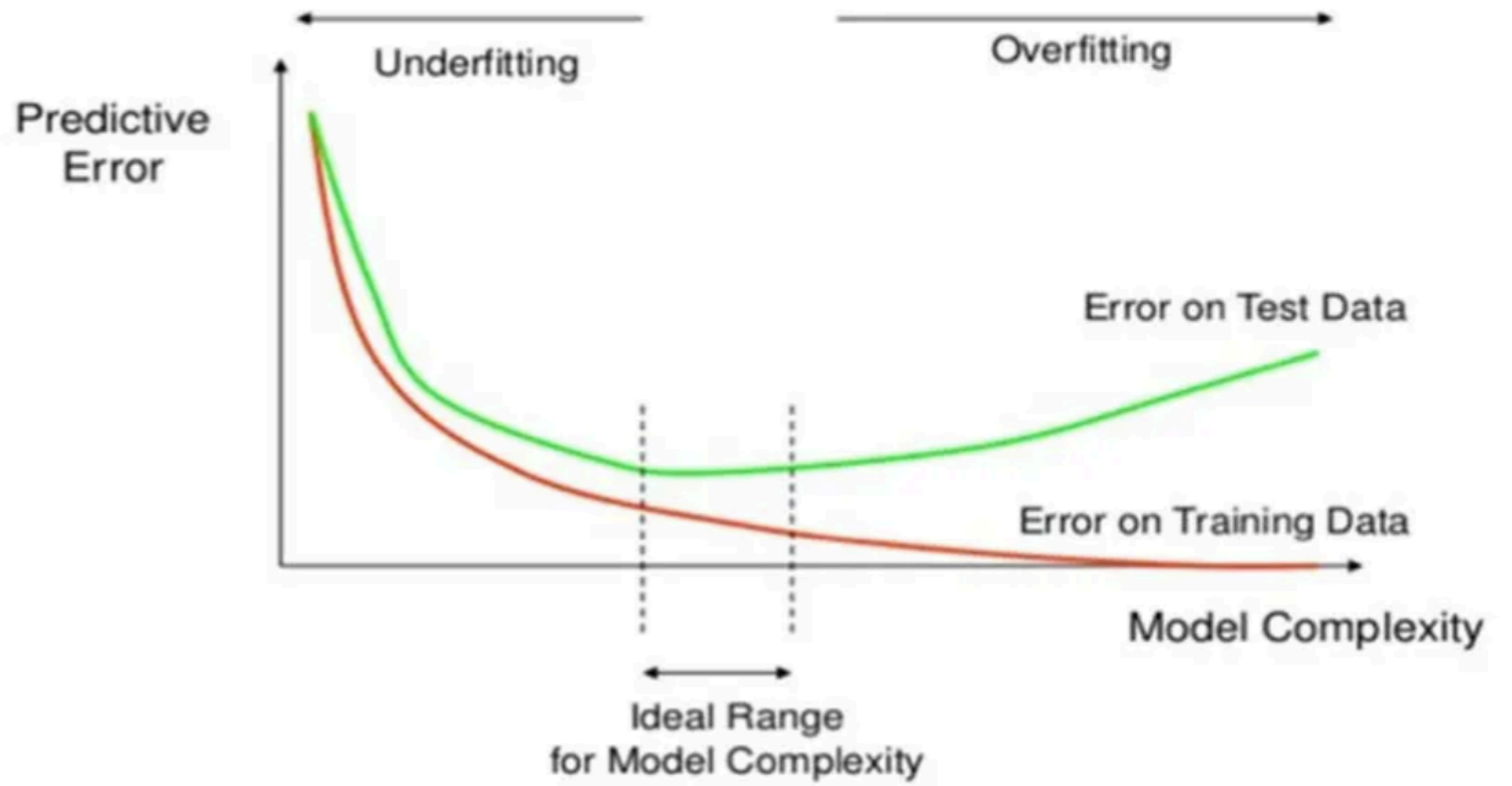
10 Key Steps For Successful Machine Learning Application



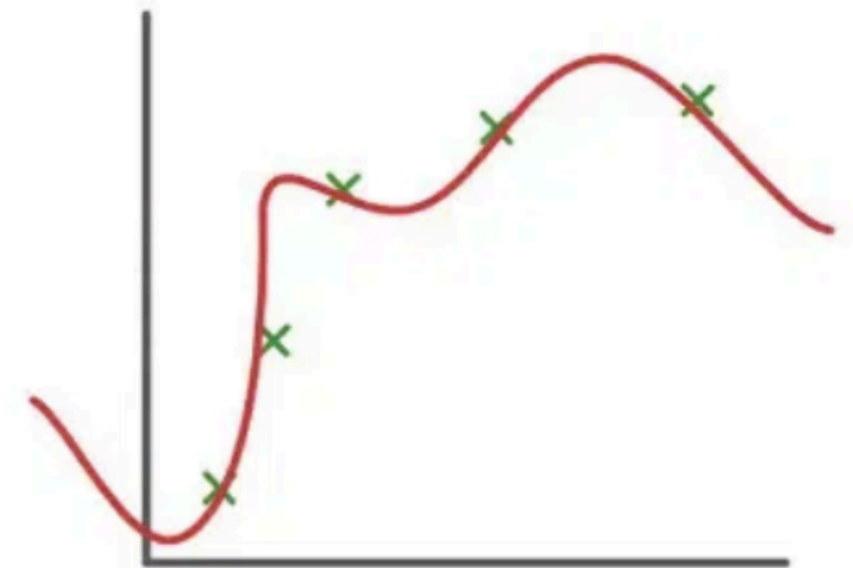
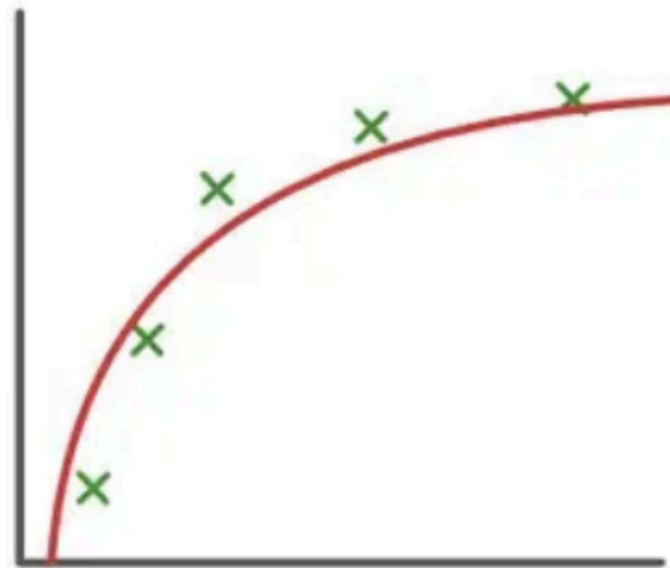
Over fitting



Over fitting

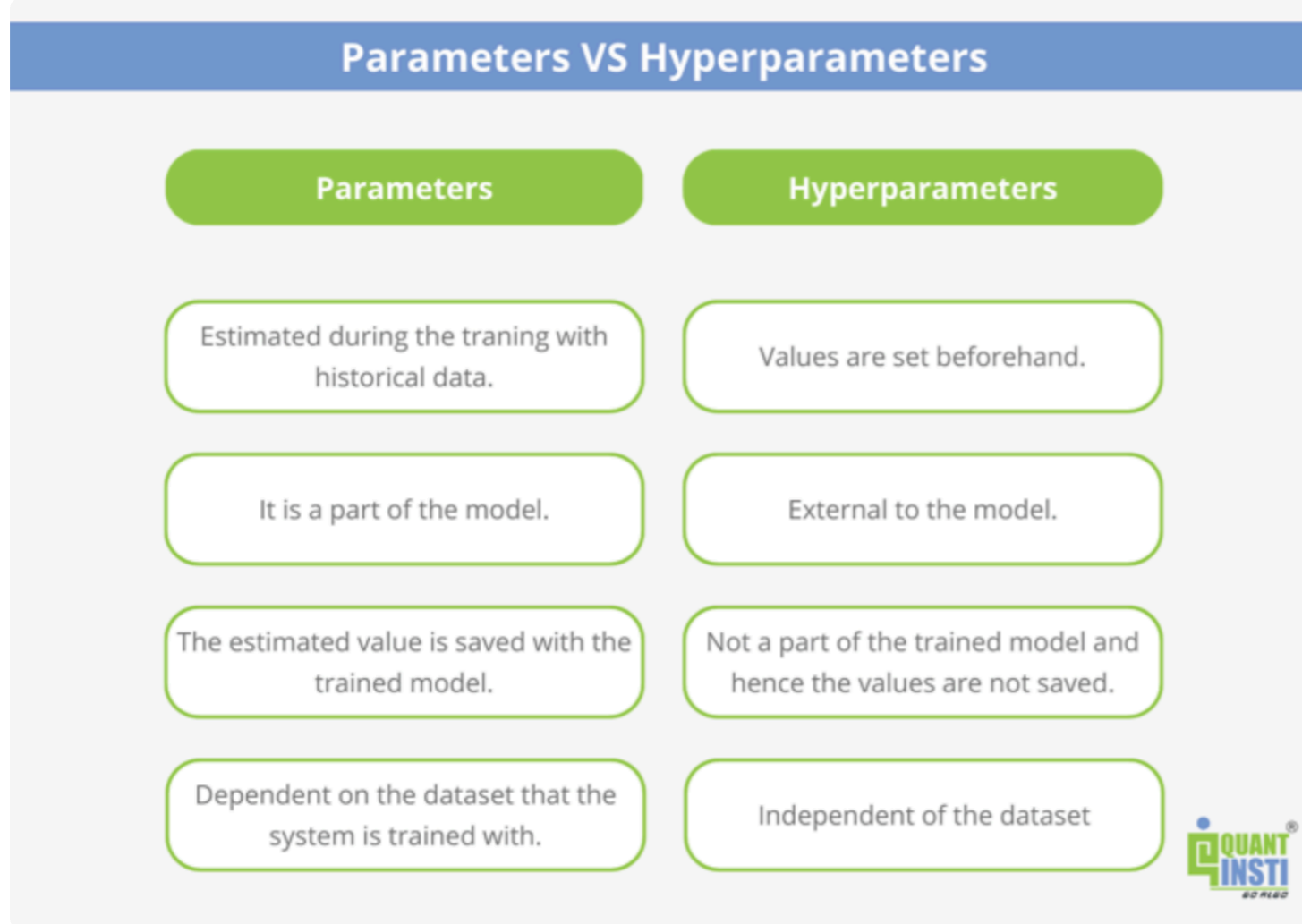


Under fitted



Over fitted

Hyper parameters



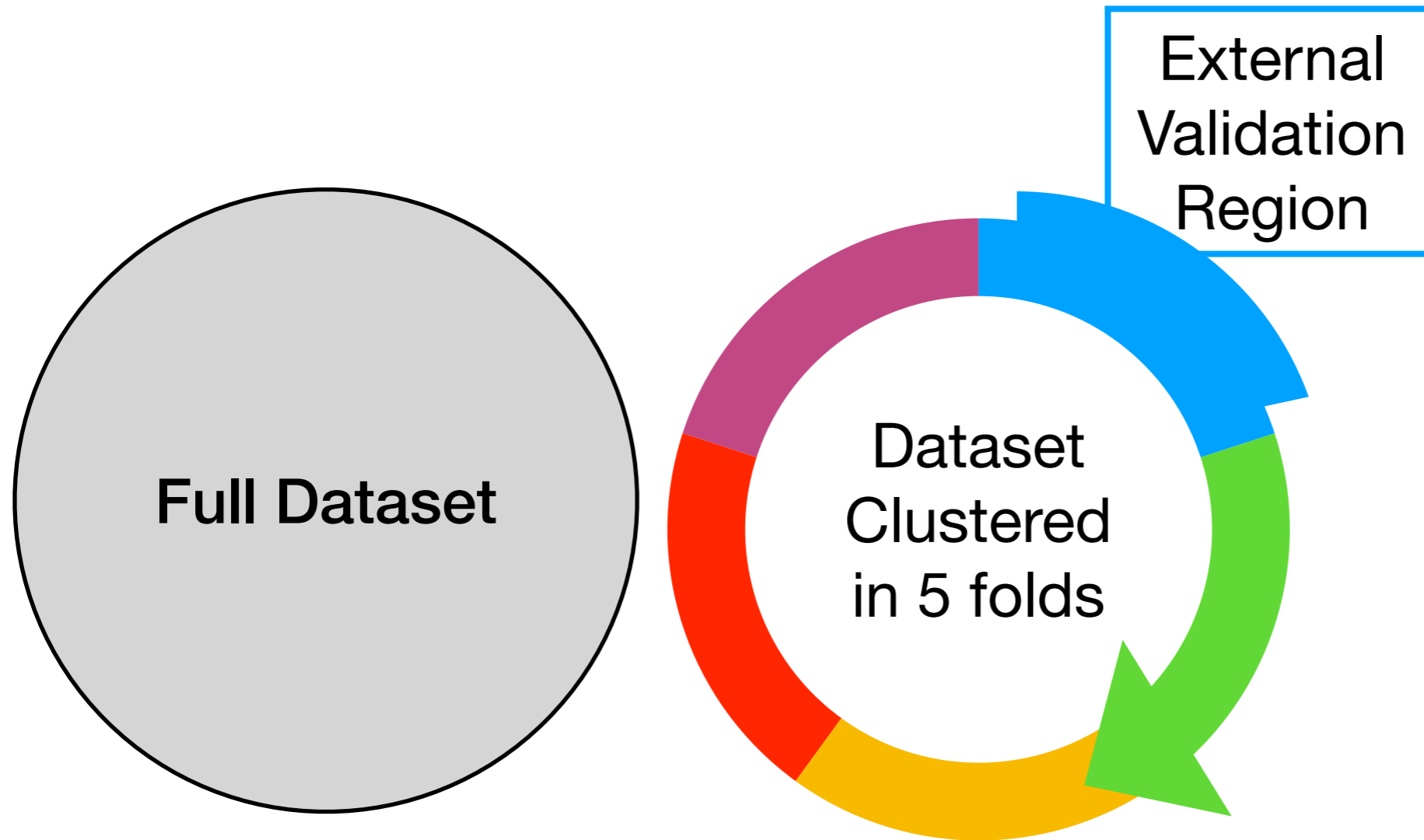
Hiperparametre ayarının avantajları:

- Geliştirilmiş model performansı
- Azaltılmış aşırı uyum ve yetersiz uyum
- Gelişmiş model genelleştirilebilirliği
- Optimize edilmiş kaynak kullanımı
- Geliştirilmiş model yorumlanabilirliği

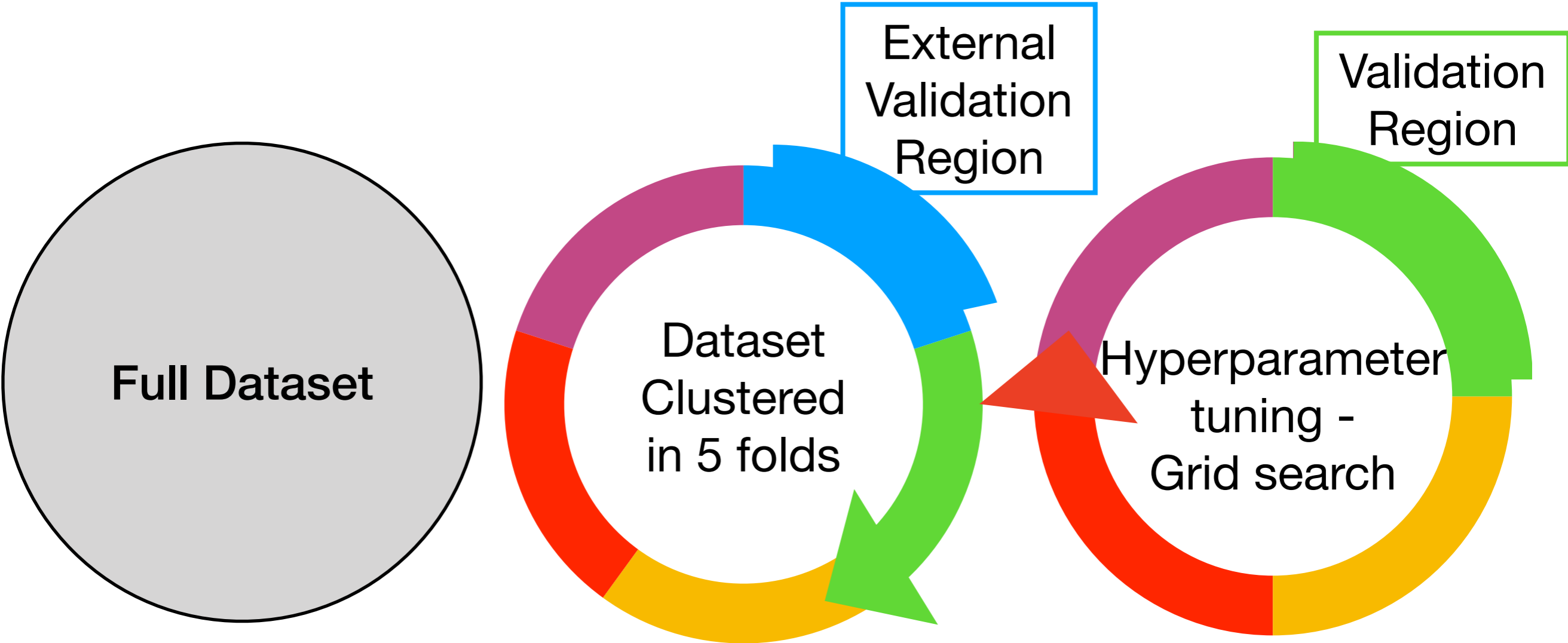
Hiperparametre ayarının dezavantajları:

- Hesaplama maliyeti
- Zaman alıcı süreç
- Aşırı uyum riski
- En iyi performansın garantisi yok
- Uzmanlık gerektirir

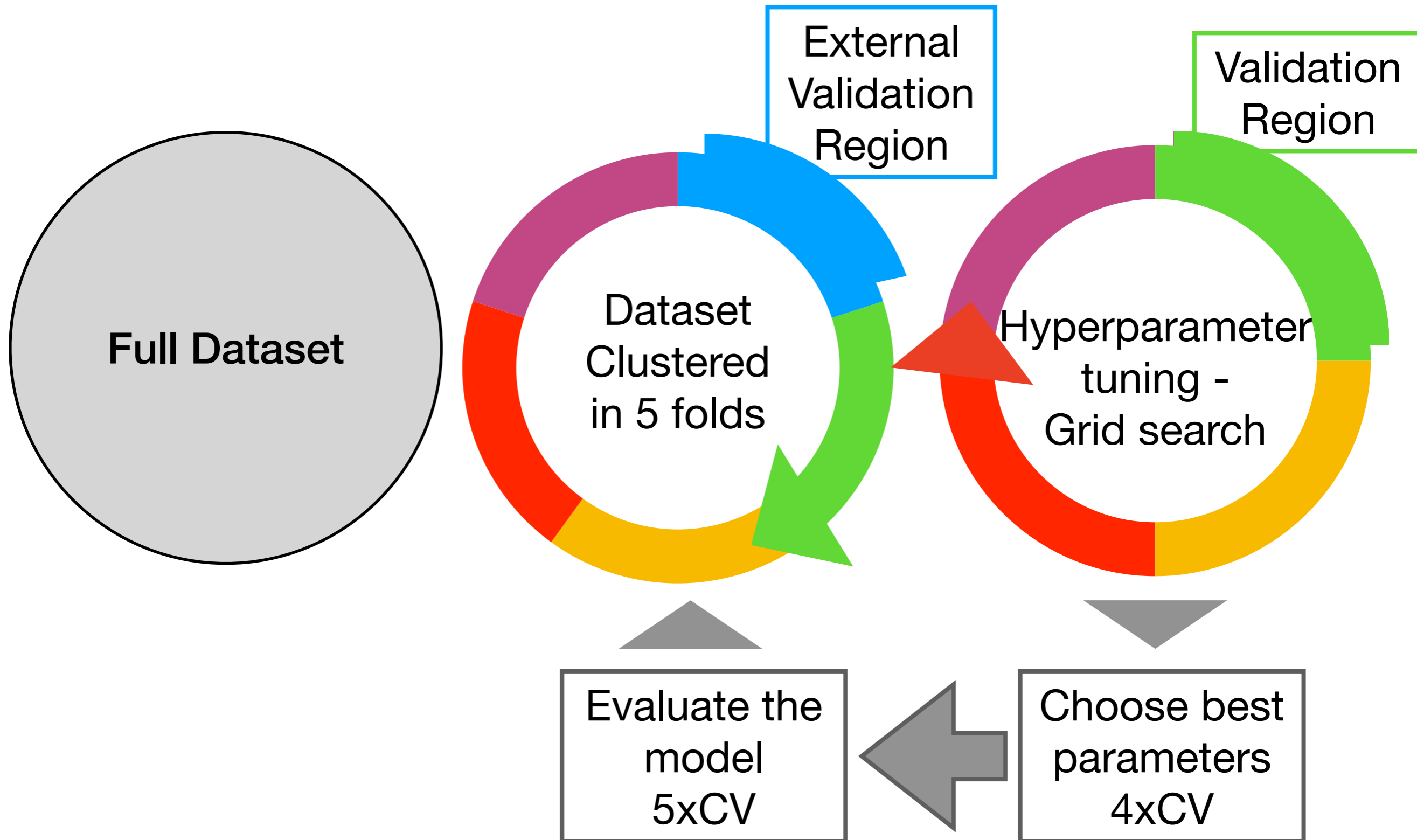
Cross Validation



Advanced Cross Validation

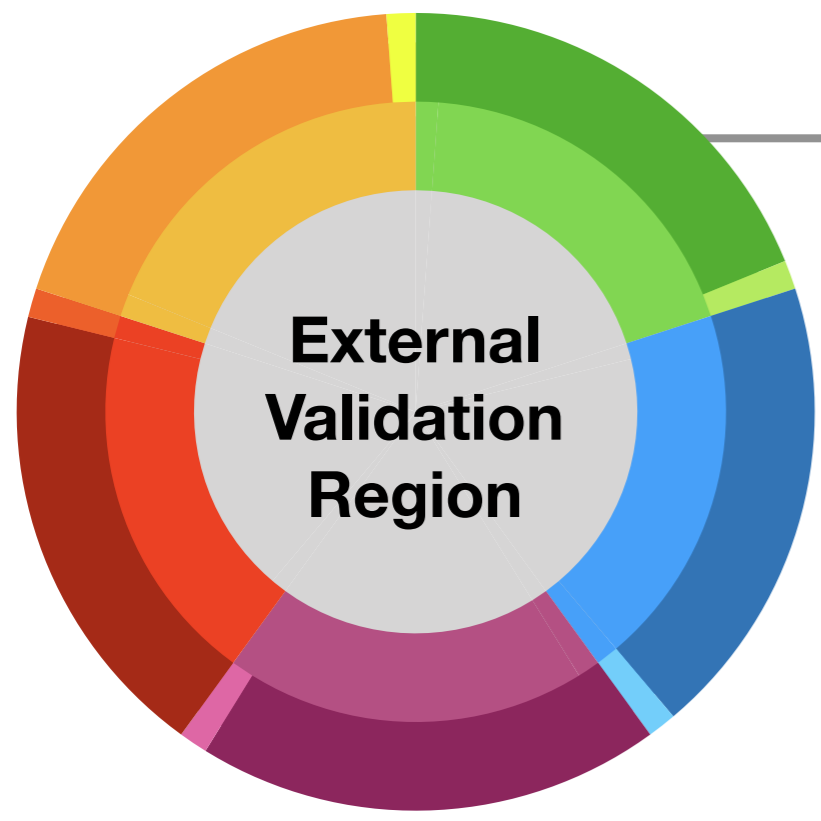


Advanced Cross Validation

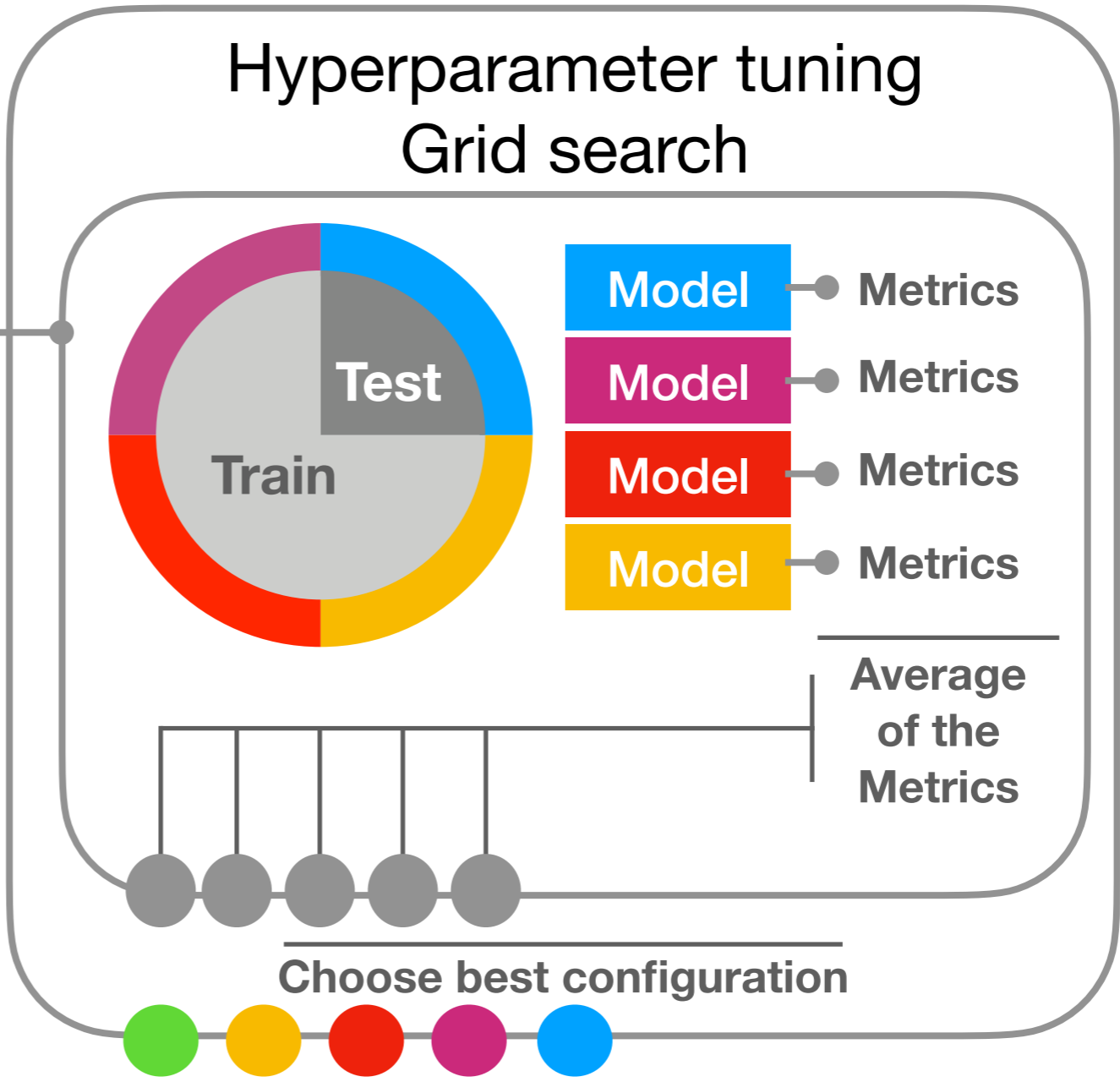


Cross Validation

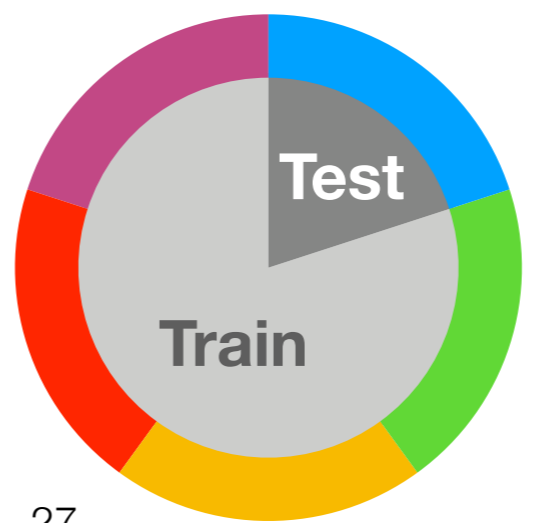
Dataset
Clustered in 5 folds



Hyperparameter tuning
Grid search



Final external validation



Teşekkürler

