Makine Öğrenmesi

1. Temel Kavramlar

1.1 Makine Öğrenmesi Nedir?

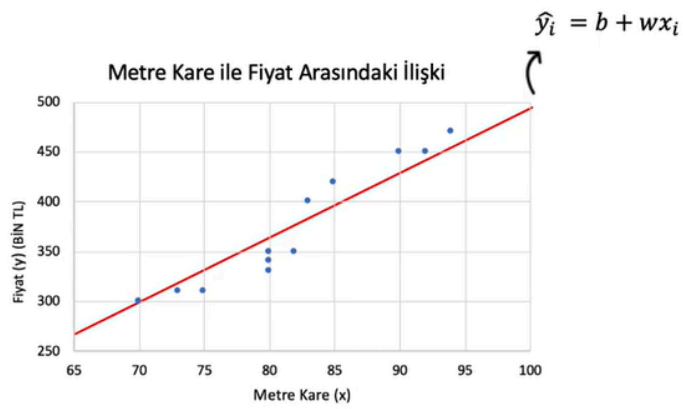
Bilgisayarların insanlara benzer şekilde öğrenmesini sağlamak maksadıyla çeşitli algoritma ve tekniklerin geliştirilmesi için çalışılan bilimsel çalışma alanıdır.

Peki bu ihtiyaç nereden doğdu? (12 Gözlem birimi vs 100.000 Gözlem birimi)

metre_kare (x_i)	fiyat (y_i)
70	300
73	310
75	310
80	330
80	340
80	350
82	350
83	400
85	420
90	450
92	450
94	470

- 80 metrekare bir evin fiyatı ne olabilir?

metre_kare (x_i)	fiyat (y_i)
70	300
73	310
75	310
80	330
80	340
80	350
82	350
83	400
85	420
90	450
92	450
94	470



1.2 Soru: Değişken Türleri nelerdir? (Variable Types)

Soru: Sayısal Değişkenler nelerdir?

Yaş, Gelir, Boy gibi

Soru: Kategorik Değişken Türleri nelerdir?

Soru: Nominal Değişkenler nelerdir?

Nominal değişkenin sınıfları arasında sıralama yapılamaz. Örneğin; Cinsiyet, Futbol Takımları, Medeni durum vb.

Soru: Ordinal Değişkenler nelerdir?

Ordinal değişkenin sınıfları arasında sıralama yapılabilir. Örneğin; Eğitim durumu

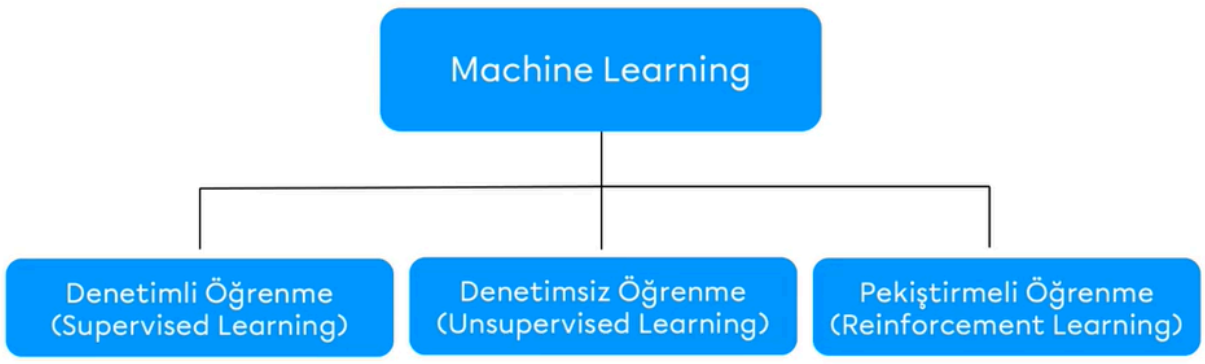
Soru: Bağımlı Değişken nedir? (Target, Dependent, Output, Response)

İlgilenen problemdeki tahminlenmek istenen değişkendir.

Soru: Bağımsız Değişken nedir? (Feature, Independent, Input)

İlgilenen problemdeki tahminlenmek istenen değişkeni etkileyen diğer değişkenlerdir.

▼ 1.3 Soru: Öğrenme Türleri nelerdir? (Learning Types)



1.3.1 Denetimli Öğrenme

Veri setinde label var ise yani tahminlenmek istenen hedef değişken var ise denetimli öğrenmedir.

1.3.2 Denetimsiz Öğrenme

Veri setinde hedef değişken yok ise yani labellar yok ise denetimsiz öğrenmedir. Segmentasyon problemleri gibi.

1.3.3 Pekiştirmeli Öğrenme

Deneme yanılma yoluyla yapılan hatalardan öğrene öğrene eğitilen öğrenme çeşididir. Örneğin bir çocuğun sobayı elleyip sıcak olduğunu anlaması ve bir daha dokunmaması, araba simülasyonları gibi.

▼ 1.4 Soru: Denetimli Öğrenme Problem Türleri nelerdir? (Problem Types)

▼ 1.4.1 Regresyon Problemi

Bağımlı değişken sayısal ise regresyon problemidir. Gelir tahmini, Fiyat tahmini gibi...

▼ 1.4.2 Sınıflandırma Problemi

Bağımlı değişken kategoriktir. Churn modeli gibi

▼ 1.5 Model Başarı Değerlendirme Yöntemleri

▼ Soru: Regresyon problemleri için temel başarı metrikleri nelerdir?

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

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

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

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

▼ **Soru:** Sınıflandırma problemleri için temel başarı metrikleri nelerdir?

Accucary, Recall, Precision, F1 score (Lojistik Regresyon Kısmında Detaylıca anlatılacak)

churn

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

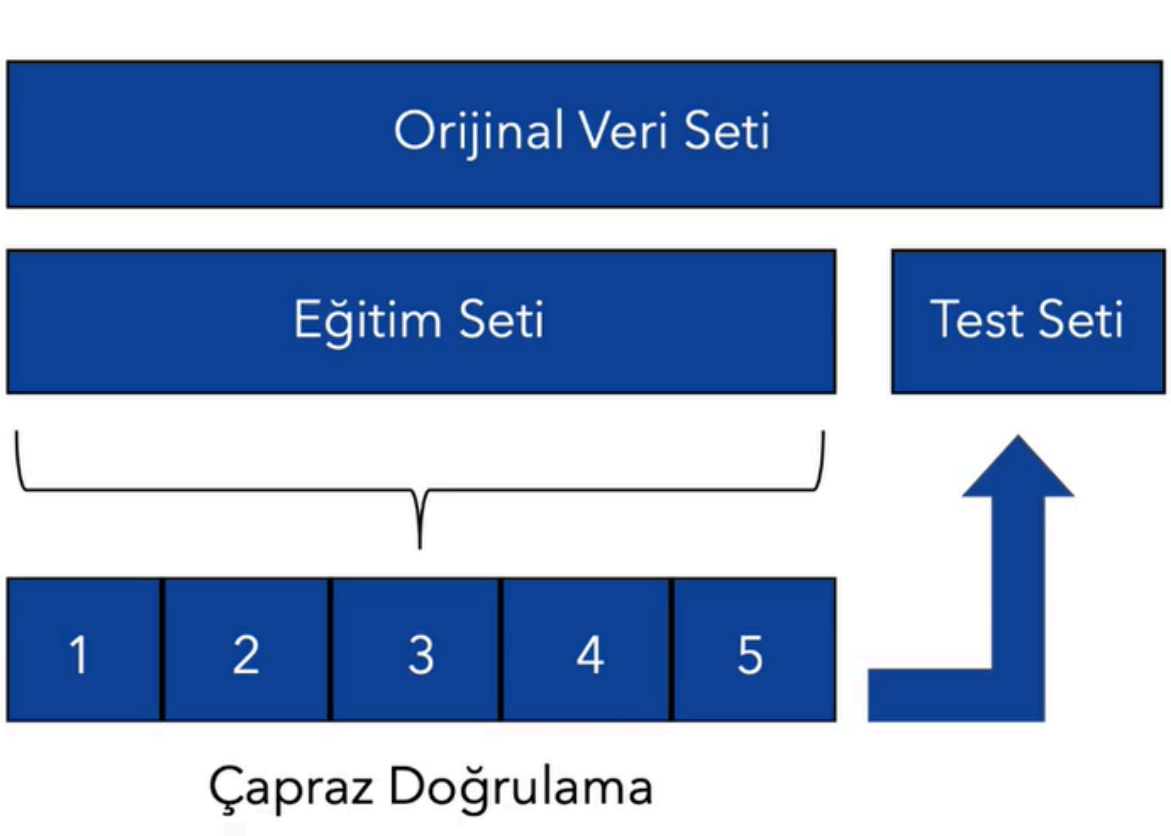
$$\text{Accuracy} = \frac{Doğru\ Sınıflandırma\ Sayısı}{Toplam\ Sınıflandırılan\ Gözlem\ Sayısı}$$

▼ 1.6 Model Doğrulama Yöntemleri

▼ 1.6.1 Holdout (Sınama Seti) Yöntemi

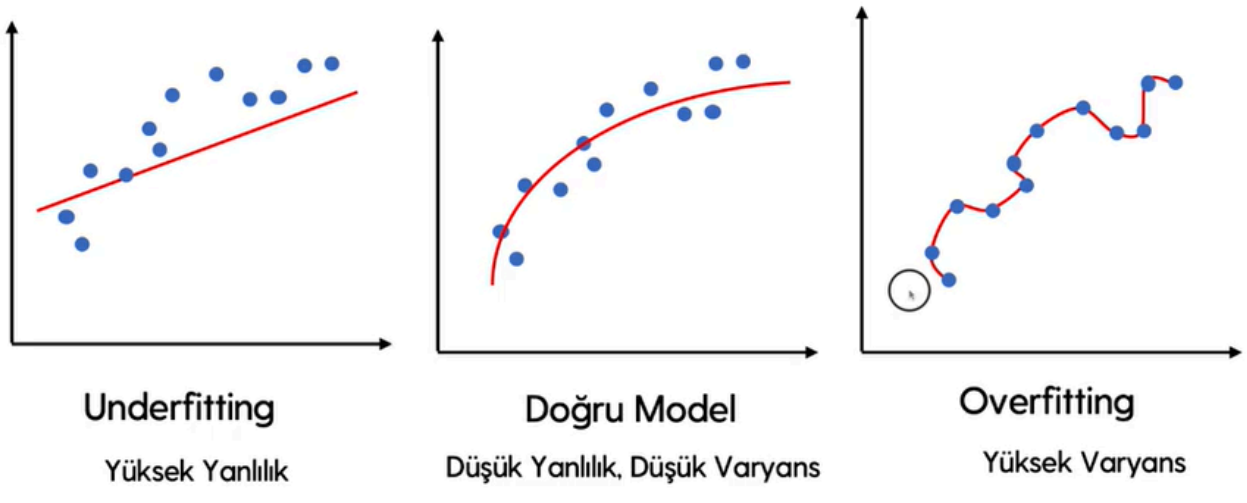
Veri seti eğitim ve test olmak üzere 2 e bölünür. Eğitim seti üzerinde model eğitilerek o modele test verisi üzerinden sorular sorulur.

▼ 1.6.2 K Katlı Çapraz Doğrulama Yöntemi

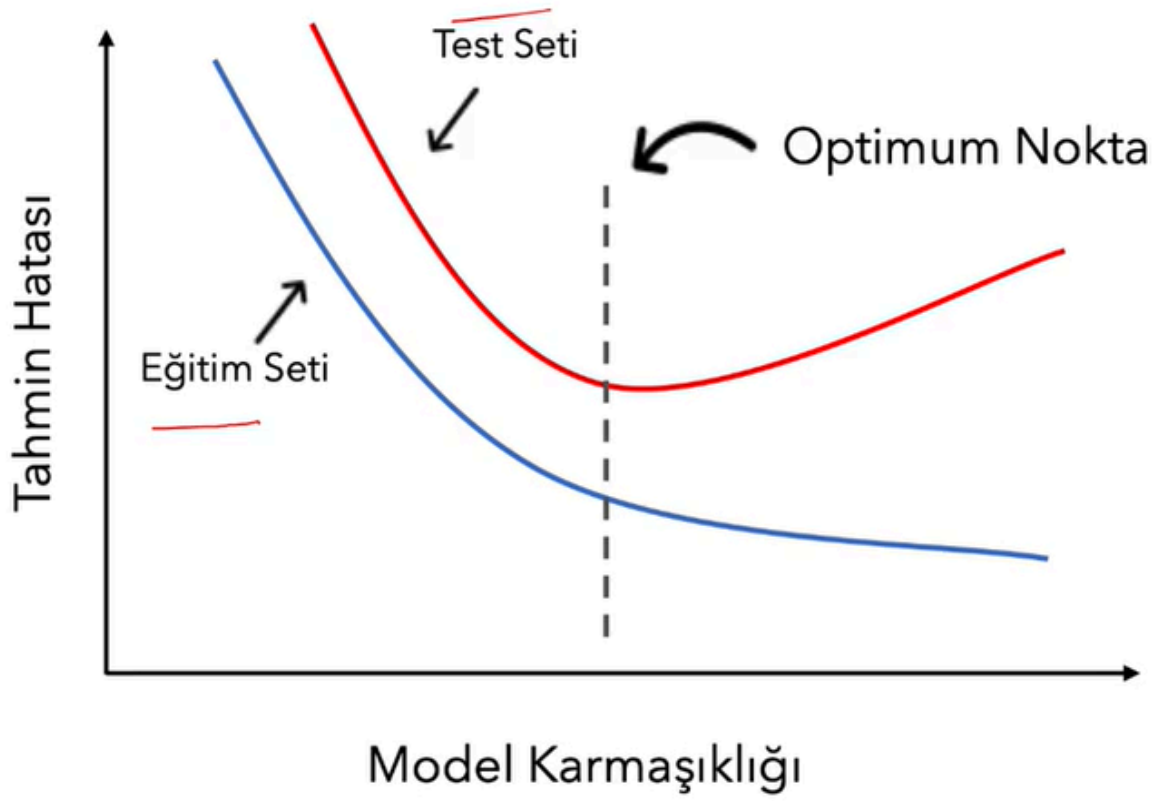


Çapraz Doğrulama

▼ 1.7 Yanlılık Varyans Değiş Tokuşu (Bias Variance Tradeoff)



Aşırı öğrenme (overfitting) sorunu nasıl tespit edilir?



▼ 2. Doğrusal Regresyon (Linear Regression)

▼ Doğrusal Regresyon modellerinde amaç bağımlı ve bağımsız değişken/değişkenler arasındaki ilişkiyi doğrusal olarak modellemektir.

$$\hat{y}_i = b + wx_i$$

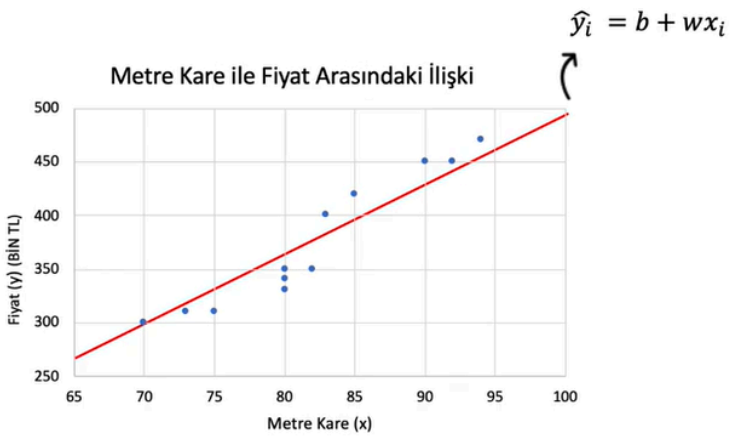
$$\hat{y}_i = b + w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_px_p$$

▼ 2.1 Ağırlıkların Bulunması

Gerçek değerler ile tahmin edilen değerler arasındaki farkların karelerinin toplamını/ortalamasını minimum yapabilecek b ve w değerleri bulunur.

$$\hat{y}_i = b + wx_i$$

metre_kare (x_i)	fiyat (y_i)
70	300
73	310
75	310
80	330
80	340
80	350
82	350
83	400
85	420
90	450
92	450
94	470

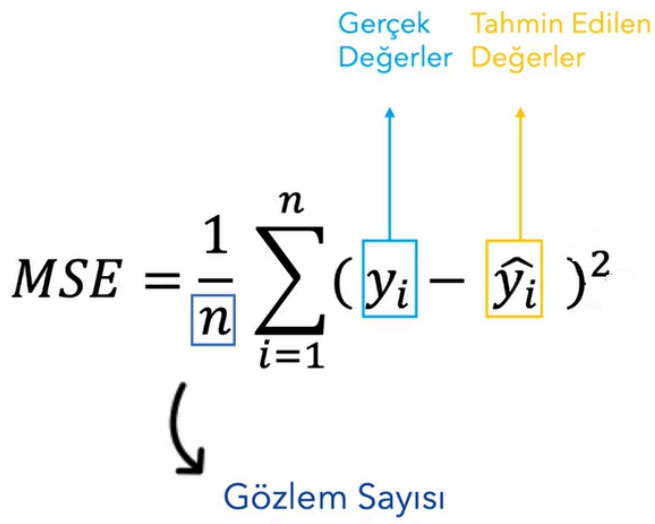


$$Cost(b,w) = \frac{1}{2m}\sum_{i=1}^{m}((b + wx_i) - y_i)^2$$

Sabit değer (b) ve ağırlıklar(w) iteratif bir şekilde değiştirilir. Her değişim sonrasında hata gözlemlenir ve en düşük hatayı veren optimum ağırlıklar belirlenmeye çalışılır.

▼ 2.2 Regresyon Yöntemlerinde Başarı Değerlendirme (MSE, RMSE, MAE)

MSE Nedir?



> Hata metriği hesaplanırken gerçek değerler ile gözlemlenen değerler arasındaki farklar sonucunda oluşan pozitif ve negatif değerler birbirini nötrleyip ölçüm problemine sebep olabilir. Bu nedenle hata kareler ortalaması olarak isimlendirilen MSE hesaplanır.

ÖRNEK;

fiyat (y_i)	fiyat tahmin ($\hat{y}_i$)	hata ($e_i = y_i - \hat{y}_i$)	hata_kare ($e_i = y_i - \hat{y}_i$)^2
300	320	-20	400
310	280	30	900
310	290	20	400
330	329	1	1
340	330	10	100
350	352	-2	4
350	400	-50	2500
400	410	-10	100
420	460	-40	1600
450	420	30	900
450	410	40	1600
470	480	-10	100

RMSE Nedir?

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

MAE Nedir?

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

▼ 2.3 Lineer Regresyon Uygulama

```
##########################################################
# Diamonds Price Prediction with Linear Regression
##########################################################

############################################################
# 1. Kütüphanelerin İçe Aktarılması
############################################################
# Veri işleme, görselleştirme ve modelleme için gerekli kütüphaneleri yüklüyoruz.

import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split

sns.set_theme()

############################################################
# 2. Veri Setinin Yüklenmesi
############################################################
# Seaborn'dan gelen diamonds veri seti, elmasların özelliklerini ve fiyatlarını içerir.

df = sns.load_dataset("diamonds")
print(df.head())
print(df.describe())

############################################################
# 3. Keşifsel Veri Analizi (EDA)
############################################################
# Elmas fiyatlarıyla karat ağırlığı arasındaki ilişkiyi inceliyoruz.

sns.scatterplot(x="carat", y="price", data=df, alpha=0.5)
plt.title("Elmas Karat Ağırlığı ve Fiyat İlişkisi")
plt.xlabel("Karat (carat)")
plt.ylabel("Fiyat (price)")
plt.show()

############################################################
# 4. Basit Lineer Regresyon Modeli (carat → price)
############################################################
# Sadece "carat" değişkeni ile fiyat tahmini yapacağız.

X = df[["carat"]]  # bağımsız değişken (2D)
y = df["price"]    # bağımlı değişken (1D)

model = LinearRegression()
model.fit(X, y)

print(f"Model sabiti (b): {model.intercept_:.2f}")
print(f"Katsayı (w): {model.coef_[0]:.2f}")
```

```
############################################################
# 5. Regresyon Doğrusunun Görselleştirilmesi
############################################################
# Scatter plot üzerine modelin tahmin doğrusunu ekliyoruz.

sns.regplot(x="carat", y="price", data=df, ci=None, line_kws={"color": "red"})
plt.title(f"price = {model.intercept_:.2f} + {model.coef_[0]:.2f} * carat")
plt.xlabel("Karat")
plt.ylabel("Fiyat")
plt.show()

############################################################
# 6. Model Performans Değerlendirmesi
############################################################
# Eğitim verisi üzerinden tahminler yaparak hata metriklerini hesaplıyoruz.

y_pred = model.predict(X)

mse = mean_squared_error(y, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y, y_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")
print(f"R² Score: {r2:.2f}")

############################################################
# 7. Yeni Bir Gözlem Üzerinden Tahmin
############################################################
# 1.5 karatlık bir elmasın tahmini fiyatını hesaplıyoruz.

yeni_elmas = [[1.5]]  # 1.5 karatlık elmas
tahmini_fiyat = model.predict(yeni_elmas)[0]
print(f"1.5 karat elmas için tahmini fiyat: {tahmini_fiyat:.2f} $")
```

▼ 3. Lojistik Regresyon (Sınıflandırma Problemi)

▼ Amaç sınıflandırma problemi için bağımlı ve bağımsız değişkenler arasındaki ilişkiyi doğrusal olarak modellemektir.

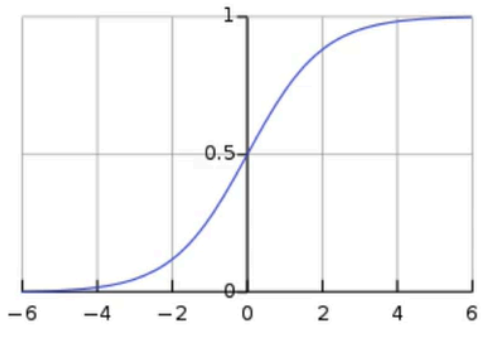
$$\hat{y}_i = \frac{1}{1 + e^{-(z)}}$$

$$z = b + w_1 x_1 + w_2 x_2 + w_3 x_3 + \ldots + w_p x_p$$

Gerçek değerler ile tahmin edilen değerler arasındaki farklara ilişkin logloss değerini minimum yapabilecek ağırlıklar elde edilir.

$$Log\ Loss = \frac{1}{m}\left(\sum_{i=1}^{m} -y_i * \log(p(\hat{y}_i)) - (1-y_i)\log(1-p(\hat{y}_i))\right)$$

$\hat{y}_i = P(Y = 1 \mid X = x)$



$$\frac{1}{1 + e^{-(z)}}$$

$$z = b + w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_px_p$$

▼ 3.1 Lojistik Regresyonda Başarı Değerlendirme Kriterleri

		Tahmin Edilen Sınıf	
		Sınıf = 1	Sınıf = 0
Gerçek Sınıf	Sınıf = 1	True Pozitif (TP)	False Negatif (FN)
	Sınıf = 0	False Pozitif (FP)	True Negatif (TN)

- **Accuracy**: Doğru sınıflandırma oranıdır: **(TP+TN) / (TP+TN+FP+FN)**
- **Precision**: Pozitif sınıf (1) tahminlerinin başarı oranıdır: **TP / (TP+FP)**
- **Recall**: Pozitif sınıfın (1) doğru tahmin edilme oranıdır: **TP / (TP + FN)**
- **F1 SCORE:** 2 * (Precision * Recall) / (Precision + Recall)

Sınıf dağılımlarının dengesiz olduğu durumlarda *Accuracy* kullanılmaz. Bu durumlarda ise ***Precision*** ve ***Recall*** değerleri incelenir:

- ***Precision:***

  Pozitif sınıf (1) tahminlerinin başarı oranıdır: ***TP/(TP+FP)***

  Tahmin edilen değerlerde ne kadar başarılı olunduğunun ölçüsüdür. **Tahmin edilen pozitif sınıfların (1 olarak tahmin edilen sınıfların) gerçekte ne kadarının pozitif olduğunu gösterir.** 1. tip hata ile ilgilenilir. Tahminlerin başarısına odaklanır.

- ***Recall:*** Pozitif sınıfın (1) doğru tahmin edilme oranıdır. ***TP/(TP+FN)*** **Tahmin edilen pozitif sınıfların ne kadarının doğru tahmin edildiğini betimler**. 2. tip hata ile ilgilenir. Önemli bir ölçüdür. Gözden kaçırmaların maliyeti hakkında bilgi verir. Gerçekleri yakalama başarısına odaklanır.

▼ 3.2 Karmaşıklık Matrisi (Confusion Matrix)

Soru: 1000 kredi kartı işlemi var. 990 normal işlem. 10 sahtekar işlem.

Buna göre Confusion matrisi doldurunuz ve başarı metriklerini hesaplayınız.

		Tahmin Edilen Sınıf Değerleri		
		Fraud İşlem (1)	Normal İşlem (0)	
Gerçek Sınıf Değerleri	Fraud İşlem (1)	5	5	**10**
	Normal İşlem (0)	90	900	**990**
		95	**905**	

- Accuracy = (5 + 900) / 1000 = 0.905
- Precision = 5/(5+90) = 0.05
- Recall = 5 / (5+5) = 0.50
- F1 Score = 2*0.025/ 0.55 = 0.09

▼ 3.3 Lojistik Regresyon Uygulama

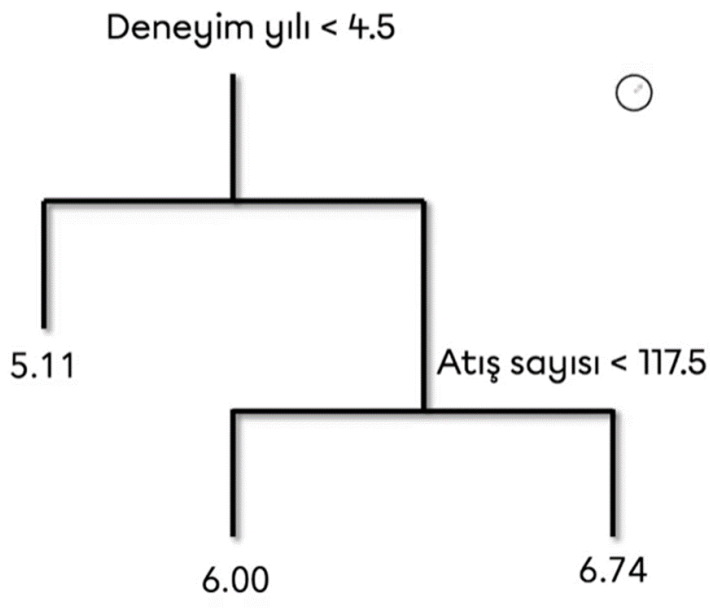
```
#############################################################
# Diabetes Prediction with Logistic Regression
#############################################################

# İş Problemi:

# Özellikleri belirtildiğinde kişilerin diyabet hastası olup
# olmadıklarını tahmin edebilecek bir makine öğrenmesi
# modeli geliştirebilir misiniz?

# Veri seti ABD'deki Ulusal Diyabet-Sindirim-Böbrek Hastalıkları Enstitüleri'nde tutulan büyük veri setinin
# parçasıdır. ABD'deki Arizona Eyaleti'nin en büyük 5. şehri olan Phoenix şehrinde yaşayan 21 yaş ve üzerinde olan
# Pima Indian kadınları üzerinde yapılan diyabet araştırması için kullanılan verilerdir. 768 gözlem ve 8 sayısal
# bağımsız değişkenden oluşmaktadır. Hedef değişken "outcome" olarak belirtilmiş olup; 1 diyabet test sonucunun
# pozitif oluşunu, 0 ise negatif oluşunu belirtmektedir.

# Değişkenler
# Pregnancies: Hamilelik sayısı
# Glucose: Glikoz.
# BloodPressure: Kan basıncı.
# SkinThickness: Cilt Kalınlığı
# Insulin: İnsülin.
# BMI: Beden kitle indeksi.
# DiabetesPedigreeFunction: Soyumuzdaki kişilere göre diyabet olma ihtimalimizi hesaplayan bir fonksiyon.
# Age: Yaş (yıl)
# Outcome: Kişinin diyabet olup olmadığı bilgisi. Hastalığa sahip (1) ya da değil (0)


# 1. Exploratory Data Analysis
# 2. Data Preprocessing
# 3. Model & Prediction
# 4. Model Evaluation
# 5. Model Validation: Holdout
# 6. Model Validation: 10-Fold Cross Validation
# 7. Prediction for A New Observation
```

```
# 1. Gerekli Kütüphaneler
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split, cross_validate
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, roc_auc_score, confusion_matrix, classification_report, RocCurveDisp


# 2. Yardımcı Fonksiyonlar
def outlier_thresholds(df, col, q1=0.05, q3=0.95):
    q1_val = df[col].quantile(q1)
    q3_val = df[col].quantile(q3)
    iqr = q3_val - q1_val
    low_limit = q1_val - 1.5 * iqr
    up_limit = q3_val + 1.5 * iqr
    return low_limit, up_limit


def replace_with_thresholds(df, col):
    low, up = outlier_thresholds(df, col)
    df[col] = np.where(df[col] < low, low, np.where(df[col] > up, up, df[col]))


def plot_confusion_matrix(y_true, y_pred):
    acc = round(accuracy_score(y_true, y_pred), 2)
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(cm, annot=True, fmt="d")
    plt.xlabel("Predicted")
    plt.ylabel("Actual")
    plt.title(f"Accuracy: {acc}")
    plt.show()


# 3. Veri Yükleme ve İlk İnceleme
df = pd.read_csv(r"\Miuul Yaz Kampı\diabetes.csv")

target = "Outcome"
features = [col for col in df.columns if col != target]

# 4. Eksik ve Aykırı Değer İşleme
for col in features:
    replace_with_thresholds(df, col)

# 5. Ölçekleme
scaler = RobustScaler()
df[features] = scaler.fit_transform(df[features])

# 6. Modelleme
X = df[features]
y = df[target]

log_model = LogisticRegression().fit(X, y)
y_pred = log_model.predict(X)
```

```
y_prob = log_model.predict_proba(X)[:, 1]

# 7. Başarı Değerlendirmesi (Tüm veri seti ile)
print("Training Classification Report:\n", classification_report(y, y_pred))
plot_confusion_matrix(y, y_pred)
print("ROC AUC Score (Train):", roc_auc_score(y, y_prob))

# 8. Holdout Doğrulama
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=17)

log_model = LogisticRegression().fit(X_train, y_train)
y_pred = log_model.predict(X_test)
y_prob = log_model.predict_proba(X_test)[:, 1]

print("Holdout Classification Report:\n", classification_report(y_test, y_pred))
RocCurveDisplay.from_estimator(log_model, X_test, y_test)
plt.title("ROC Curve (Test Set)")
plt.plot([0, 1], [0, 1], "r--")
plt.show()
print("ROC AUC Score (Test):", roc_auc_score(y_test, y_prob))

# 9. 10-Fold Cross Validation
cv_results = cross_validate(log_model, X, y, cv=10,
                            scoring=["accuracy", "precision", "recall", "f1", "roc_auc"])

print("CV Accuracy:", cv_results["test_accuracy"].mean())
print("CV Precision:", cv_results["test_precision"].mean())
print("CV Recall:", cv_results["test_recall"].mean())
print("CV F1 Score:", cv_results["test_f1"].mean())
print("CV ROC AUC:", cv_results["test_roc_auc"].mean())

# 10. Yeni Gözlem Üzerinden Tahmin
new_sample = X.sample(1, random_state=45)
prediction = log_model.predict(new_sample)
print("New Sample Prediction:", prediction)
```

▼ 4. CART (Classification and Regression Tree)

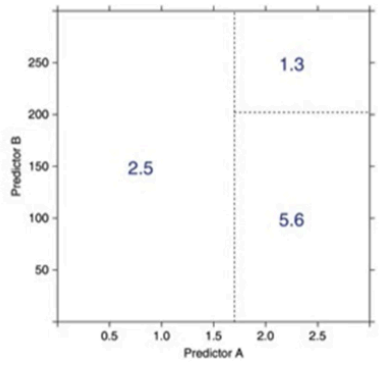
▼ Bir karar ağacı yöntemidir. Random Forest'ın temelini oluşturur. Amacı, veri seti içerisindeki karmaşık yapıları basit karar yapılarına dönüştürmektir. Heterojen veri setleri belirlenmiş bir hedef değişkene göre homojen alt gruplara ayrılır.



Hitters (Baseball) veri setini ağaç yapısında gösterecek olursak, deneyim yılı ve atış sayısı bağımsız değişkenleri için, deneyim yılı 4.5 yıldan büyükse ve atış sayısı 117.5'den fazlaysa maaş 6.74'tür. Deneyim yılı 4.5'ten küçükse atış sayısına direk bakılmadan maaş 5.11 denir.
Burada 2 tane internal nod yani 2 iç düğümü vardır. 3 tane de terminal nod (yani kural seti-nihai sonuç) vardır.
Burada deneyim yılı değişkeni, ilk kırılımı sağlayan değişken olduğundan maaşı etkileyen en önemli faktördür. En tepede olmasının sebebi de hatada en çok harekete sebep olmasıdır.

Bu yapıyı 2 eksen üzerinde gözlemlemek istersek de aşağıdaki örneği inceleyebiliriz.

```
if Predictor A >= 1.7 then
|   if Predictor B >= 202.1 then Outcome = 1.3
|   else Outcome = 5.6
else Outcome = 2.5
```



Karar ağaçlarındaki ayrımın neye göre olacağını çeşitli homojenlik ölçütleriyle belirleyebiliriz. (***ki-kare***, ***gini*** veya ***entropi*** gibi.)
Regresyon modelleri için karar ağaçları kullanıldığında tahmin edilen değerler, bölme işlemi neticesinde ortaya çıkan kutularda kalan gözlem birimlerinin ortalamasıdır. Bu bölme işlemleri ile (Sum of Squared Errors) SSE'nin minimum olması istenir. Yani gerçek değerlerden kutucuğun değerinin farkının (hatanın) kareler toplamını minimum yapmaya çalışıyoruz. Algoritma tek tek bütün kombinasyonlardan kutular oluşturur ve daha sonrasında hata kareler toplamı en az olan kombinasyonu seçer ve orası kırılım noktası olur.

Regresyon Problemleri için Cost/Loss/Objective Fonksiyonu

RSS (SSE): $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$

R_j: bölge/yaprak/kutu

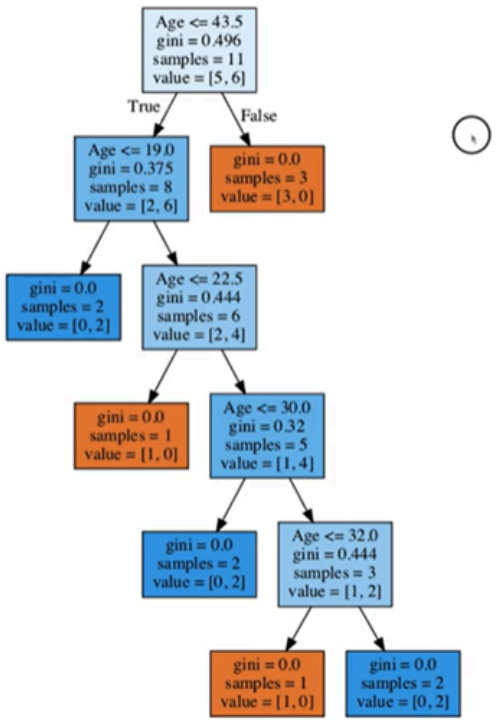
Sınıflandırma Problemleri için Cost/Loss/Objective fonksiyonu

$$\mathcal{L}_G(\mathcal{N}_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

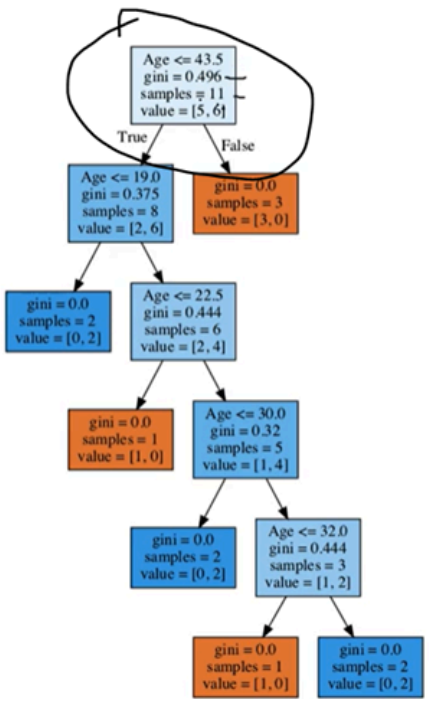
$$\mathcal{L}_E(\mathcal{N}_m) = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

The weighted loss: $\mathcal{L}(S_m) = f_L \cdot \mathcal{L}(C_m^L) + f_R \cdot \mathcal{L}(C_m^R).$

Sınıflandırma için karar ağaçlarında gini katsayısı, entropiye göre kırılımlar gerçekleşir. LG (gini) olarak gösterilen ifadede olayın olma olasılığı ve olmama olasılığı çarpılır. LE (entropi) ise olma olasılığı ve bu olasılığın logaritma karşılığına denk gelir. Sonrasında 1 sınıfının olma olasılığı ve 1 sınıfının olmama olasılığını çarpıp, 0 sınıfının olma olasılığı ve 0 sınıfının olmama olasılığını çarparız. Gelen sonuç gini bilgisini döndürür.

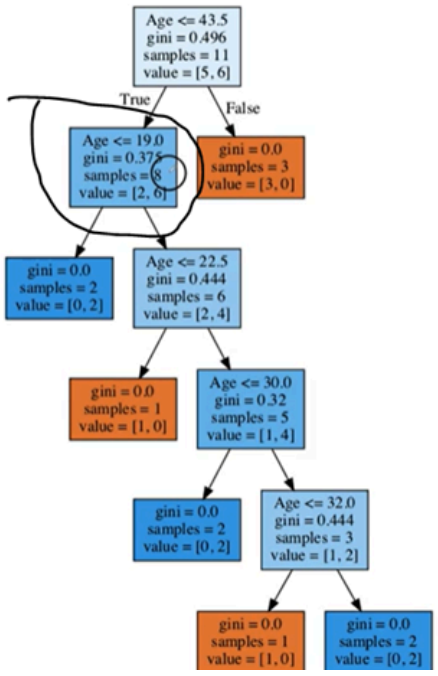


$$\mathcal{L}_G(\mathcal{N}_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$



$$\mathcal{L}_G(\mathcal{N}_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

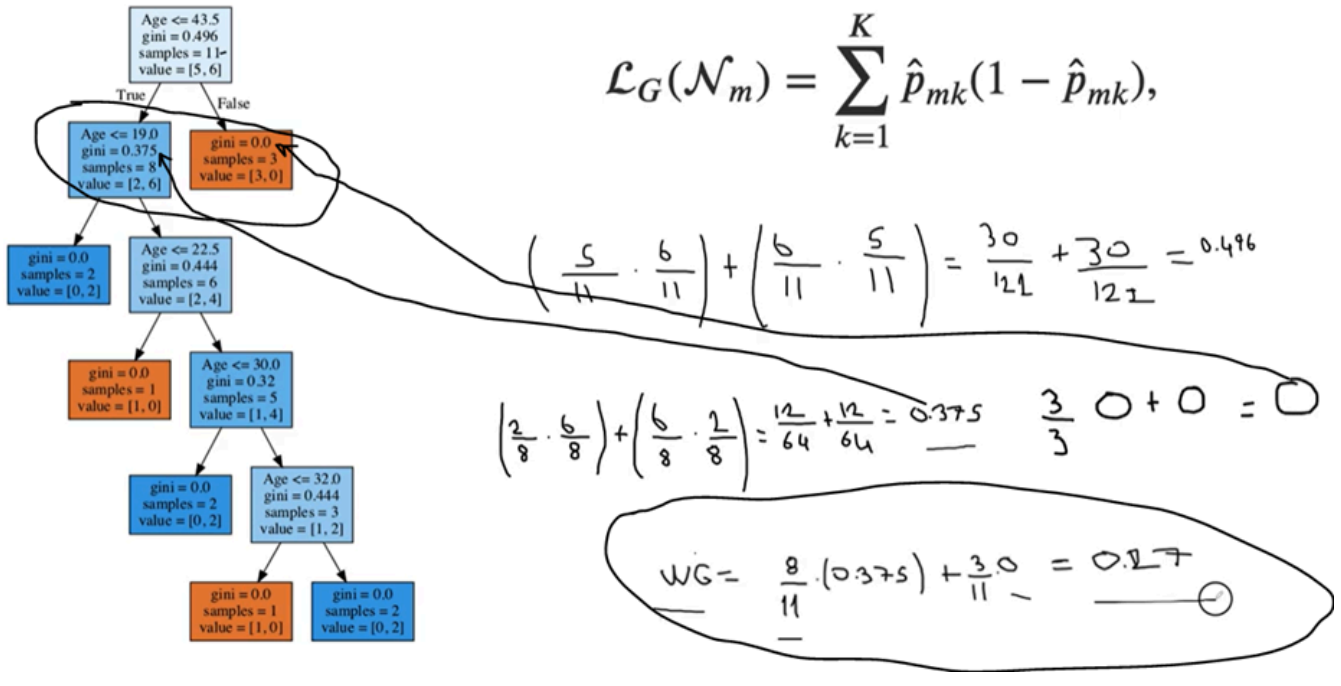
$$\left(\frac{5}{11} \cdot \frac{6}{11}\right) + \left(\frac{6}{11} \cdot \frac{5}{11}\right) = \frac{30}{121} + \frac{30}{121} = 0.496$$



$$\mathcal{L}_G(\mathcal{N}_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

$$\left(\frac{5}{11} \cdot \frac{6}{11}\right) + \left(\frac{6}{11} \cdot \frac{5}{11}\right) = \frac{30}{121} + \frac{30}{121} = 0.496$$

$$\left(\frac{2}{8} \cdot \frac{6}{8}\right) + \left(\frac{6}{8} \cdot \frac{2}{8}\right) = \frac{12}{64} + \frac{12}{64} = 0.375$$



Weighted Loss fonksiyonu, ilgili kutudaki gözlem sayısının tüm data setine oranıyla gini katsayılar çarpılıp, diğer kutular içinde yapılan bu işlemin toplamından oluşur.

▼ CART Uygulama

```
##################################################
# CART (Classification and Regression Trees)
##################################################

# Gerekli kütüphaneler
import warnings
import joblib
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import pydotplus
from sklearn.tree import DecisionTreeClassifier, export_graphviz, export_text, plot_tree
from sklearn.metrics import classification_report, roc_auc_score
from sklearn.model_selection import train_test_split, GridSearchCV, cross_validate, validation_curve
from skompiler import skompile
from sklearn.tree import export_graphviz
from graphviz import Source

warnings.simplefilter(action='ignore', category=Warning)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', 500)

##################################################
# 1. Veri Yükleme ve Ön İşleme
##################################################

df = pd.read_csv(r"\Miuul Yaz Kampı\diabetes.csv")
df.head()
y = df["Outcome"]
X = df.drop("Outcome", axis=1)

##################################################
# 2. Modelleme: Temel CART
##################################################
```

```
model = DecisionTreeClassifier(random_state=1).fit(X, y)
y_pred = model.predict(X)
y_prob = model.predict_proba(X)[:, 1]

print("Confusion Matrix:\n", classification_report(y, y_pred))
print("ROC AUC:", roc_auc_score(y, y_prob))

##################################################
# 3. Holdout Yöntemiyle Değerlendirme
##################################################

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=45)
model = DecisionTreeClassifier(random_state=17).fit(X_train, y_train)

# Eğitim verisi değerlendirme
print("Train Score:\n", classification_report(y_train, model.predict(X_train)))
print("Train ROC AUC:", roc_auc_score(y_train, model.predict_proba(X_train)[:, 1]))

# Test verisi değerlendirme
print("Test Score:\n", classification_report(y_test, model.predict(X_test)))
print("Test ROC AUC:", roc_auc_score(y_test, model.predict_proba(X_test)[:, 1]))

##################################################
# 4. Cross Validation (CV)
##################################################

cv_model = DecisionTreeClassifier(random_state=17)
cv_results = cross_validate(cv_model, X, y, cv=5, scoring=["accuracy", "f1", "roc_auc"])
print("CV Accuracy:", cv_results['test_accuracy'].mean())
print("CV F1:", cv_results['test_f1'].mean())
print("CV ROC AUC:", cv_results['test_roc_auc'].mean())

##################################################
# 5. Hiperparametre Optimizasyonu
##################################################

params = {"max_depth": range(1, 11), "min_samples_split": range(2, 20)}
grid = GridSearchCV(cv_model, params, cv=5, n_jobs=-1, verbose=1).fit(X, y)
print("Best Params:", grid.best_params_)
print("Best CV Score:", grid.best_score_)

##################################################
# 6. Final Model
##################################################

final_model = DecisionTreeClassifier(**grid.best_params_, random_state=17).fit(X, y)
final_cv = cross_validate(final_model, X, y, cv=5, scoring=["accuracy", "f1", "roc_auc"])
print("Final CV Accuracy:", final_cv['test_accuracy'].mean())
print("Final CV F1:", final_cv['test_f1'].mean())
print("Final CV ROC AUC:", final_cv['test_roc_auc'].mean())

##################################################
# 7. Özellik Önem Grafiği
##################################################

def plot_importance(model, features, top_n=10):
    importance_df = pd.DataFrame({'Feature': features.columns, 'Importance': model.feature_importances_})
    importance_df = importance_df.sort_values("Importance", ascending=False).head(top_n)
```

```
    plt.figure(figsize=(8, 6))
    sns.barplot(x="Importance", y="Feature", data=importance_df)
    plt.title("Feature Importances")
    plt.tight_layout()
    plt.show()

plot_importance(final_model, X)

##################################################
# 8. Validation Curve (BONUS)
##################################################

def val_curve(model, X, y, param_name, param_range, scoring="roc_auc", cv=10):
    train_scores, test_scores = validation_curve(
        estimator=model,
        X=X,
        y=y,
        param_name=param_name,
        param_range=param_range,
        scoring=scoring,
        cv=cv)

    plt.plot(param_range, train_scores.mean(axis=1), label="Train")
    plt.plot(param_range, test_scores.mean(axis=1), label="Validation")
    plt.title(f"Validation Curve: {param_name}")
    plt.xlabel(param_name)
    plt.ylabel(scoring.upper())
    plt.legend()
    plt.tight_layout()
    plt.show()

val_curve(final_model, X, y, param_name="max_depth", param_range=range(1, 11))
val_curve(final_model, X, y, param_name="max_depth", param_range=range(2, 20))

##################################################
# 9. Karar Ağacını Görselleştirme
##################################################

def plot_tree_inline(model, feature_names):
    plt.figure(figsize=(40, 20))
    plot_tree(model, feature_names=feature_names, filled=True, class_names=True, rounded=True)
    plt.title("Decision Tree")
    plt.show()

plot_tree_inline(final_model, X.columns)

##################################################
# 10. Karar Kurallarını ve Kodları Çıkarma
##################################################

print(export_text(final_model, feature_names=list(X.columns)))
print(skompile(final_model.predict).to("python/code"))

# Örnek tahmin
sample = pd.DataFrame([X.iloc[1].values], columns=X.columns)
print("Prediction:", final_model.predict(sample))
```

▼ Makine Öğrenmesi Proje Örnekleri

Beyzbol Maaş Tahmini - https://www.kaggle.com/datasets/creepycrap/baseball-player-salary-prediction

Telco Müşteri Terk Modeli (Churn) - https://www.kaggle.com/datasets/blastchar/telco-customer-churn

Titanic Hayatta Kalma Tahmin Modeli - https://www.kaggle.com/competitions/titanic/data

Ev Fiyat Tahmin Modeli - https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques

Diyabet Tahmin Modeli - https://www.kaggle.com/datasets/akshaydattatraykhare/diabetes-dataset/data