1. **Ortalama gelir (MedInc) değişkenine bakarak konut değerini (Target) tahmin etmek.**

# Gerekli kütüphaneler

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import fetch\_california\_housing

from sklearn.linear\_model import LinearRegression

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_error

# California konut veri setini yükle

data = fetch\_california\_housing()

# Veriyi pandas DataFrame'e dönüştür

df = pd.DataFrame(data.data, columns=data.feature\_names)

df['Target'] = data.target # Hedef sütun: konut fiyatları

# Sadece bir bağımsız değişken ile regresyon yapalım (örnek: MedInc - Ortalama Gelir)

X = df[['MedInc']] # Girdi (bağımsız değişken)

y = df['Target'] # Çıktı (bağımlı değişken)

# Veriyi %80 eğitim, %20 test olarak ayırıyoruz

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Modeli oluşturuyoruz ve eğitiyoruz

model = LinearRegression()

model.fit(X\_train, y\_train)

# Test verisi üzerinde tahmin yapıyoruz

y\_pred = model.predict(X\_test)

# Ortalama Kare Hata (MSE) hesaplama

mse = mean\_squared\_error(y\_test, y\_pred)

print(f"Ortalama Kare Hata (MSE): {mse:.2f}")

print(f"Regresyon Katsayısı (Eğim): {model.coef\_[0]:.4f}")

# Test verisindeki gerçek ve tahmin edilen değerleri karşılaştıralım

plt.scatter(X\_test, y\_test, color='gray', label="Gerçek Değerler")

plt.plot(X\_test, y\_pred, color='blue', linewidth=2, label="Model Tahminleri")

plt.xlabel("Ortalama Gelir (MedInc)")

plt.ylabel("Ev Değeri (Target)")

plt.title("Basit Doğrusal Regresyon")

plt.legend()

plt.show()

**2. Çoklu Doğrusal Regresyon (Multiple Linear Regression)**

**📌 Amaç:**

Tüm özellikleri kullanarak daha isabetli tahmin yapmak.

# Giriş değişkenleri: tüm sütunlar (Target hariç)

X\_multi = df.drop(columns='Target')

y\_multi = df['Target']

# Eğitim/test bölme

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_multi, y\_multi, test\_size=0.2, random\_state=0)

# Regresyon modelini tanımlayıp eğitiyoruz

model\_multi = LinearRegression()

model\_multi.fit(X\_train, y\_train)

# Tahmin ve hata hesabı

y\_pred\_multi = model\_multi.predict(X\_test)

mse\_multi = mean\_squared\_error(y\_test, y\_pred\_multi)

print(f"Çoklu Regresyon MSE: {mse\_multi:.2f}")

print("Özelliklere ait regresyon katsayıları:")

for feature, coef in zip(X\_multi.columns, model\_multi.coef\_):

print(f"{feature}: {coef:.4f}")

import seaborn as sns

# Özellik katsayılarını görselleştir

coef\_df = pd.DataFrame({

"Feature": X\_multi.columns,

"Coefficient": model\_multi.coef\_

})

plt.figure(figsize=(10, 5))

sns.barplot(data=coef\_df, x="Coefficient", y="Feature")

plt.title("Özelliklerin Ev Değerine Etkisi (Regresyon Katsayıları)")

plt.tight\_layout()

plt.show()

**3. Polinomsal Regresyon (Polynomial Regression)**

**📌 Amaç:**

Veri doğrusal değilse, doğrusal olmayan (eğrisel) modellerle tahmin yapmak.

from sklearn.preprocessing import PolynomialFeatures

# PolynomialFeatures ile 'MedInc' özelliğini ikinci dereceden polinoma çeviriyoruz

poly = PolynomialFeatures(degree=2)

X\_poly = poly.fit\_transform(df[['MedInc']]) # [x, x^2] şeklinde dönüşür

# Eğitim ve test setlerini oluştur

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_poly, y, test\_size=0.2, random\_state=42)

# Modeli eğit

model\_poly = LinearRegression()

model\_poly.fit(X\_train, y\_train)

# Tahmin ve değerlendirme

y\_pred\_poly = model\_poly.predict(X\_test)

mse\_poly = mean\_squared\_error(y\_test, y\_pred\_poly)

print(f"Polinomsal Regresyon MSE: {mse\_poly:.2f}")

**4. Ridge ve Lasso Regresyon – Regularizasyon Teknikleri**

**📌 Amaç:**

Çoklu doğrusal regresyonda aşırı öğrenmeyi (overfitting) engellemek.

from sklearn.linear\_model import Ridge, Lasso

# Ridge Regresyon - L2 regularizasyonu (ağırlıkları küçültür ama sıfırlamaz)

ridge = Ridge(alpha=1.0)

ridge.fit(X\_train, y\_train)

ridge\_mse = mean\_squared\_error(y\_test, ridge.predict(X\_test))

print(f"Ridge Regresyon MSE: {ridge\_mse:.2f}")

# Lasso Regresyon - L1 regularizasyonu (gereksiz değişkenleri sıfırlar)

lasso = Lasso(alpha=0.1)

lasso.fit(X\_train, y\_train)

lasso\_mse = mean\_squared\_error(y\_test, lasso.predict(X\_test))

print(f"Lasso Regresyon MSE: {lasso\_mse:.2f}")