```
import numpy as np
import pandas as pd
from sklearn.datasets import load_diabetes
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
# Veri kümesini yükleme
diabetes = load_diabetes()
X_all = diabetes.data # 10 adet özellik
y = diabetes.target # Hedef değişken
feature_names = diabetes.feature_names
# Eğitim ve test verisine ayırma (70% eğitim, 30% test)
X_train_all, X_test_all, y_train, y_test = train_test_split(X_all, y, test_size=0.3,
random_state=42)
### 1. Basit Lineer Regresyon (Sadece BMI)
# 'bmi' sütununun index'ini bulma
bmi_index = feature_names.index('bmi')
# BMI sütununu seçme ve reshape işlemi (tek sütun)
X_train_bmi = X_train_all[:, bmi_index].reshape(-1, 1)
X_test_bmi = X_test_all[:, bmi_index].reshape(-1, 1)
# Basit modelin oluşturulması ve eğitilmesi
simple_model = LinearRegression()
simple_model.fit(X_train_bmi, y_train)
# Test verisi üzerinde tahmin yapma
y_pred_simple = simple_model.predict(X_test_bmi)
# Metriklerin hesaplanması
r2_simple = r2_score(y_test, y_pred_simple)
mae_simple = mean_absolute_error(y_test, y_pred_simple)
mse_simple = mean_squared_error(y_test, y_pred_simple)
print("Basit Lineer Regresyon (BMI) Sonuçları:")
print("R² Skoru: {:.4f}".format(r2_simple))
```

```
print("MAE: {:.4f}".format(mae_simple))
print("MSE: {:.4f}".format(mse_simple))
print("\n---------------------------------------\n")
### 2. Çoklu Lineer Regresyon (Tüm Özellikler)
# Çoklu modelin oluşturulması ve eğitilmesi
multi_model = LinearRegression()
multi_model.fit(X_train_all, y_train)
# Test verisi üzerinde tahmin yapma
y_pred_multi = multi_model.predict(X_test_all)
# Metriklerin hesaplanması
r2_multi = r2_score(y_test, y_pred_multi)
mae_multi = mean_absolute_error(y_test, y_pred_multi)
mse_multi = mean_squared_error(y_test, y_pred_multi)
print("Çoklu Lineer Regresyon Sonuçları:")
print("R² Skoru: {:.4f}".format(r2_multi))
print("MAE: {:.4f}".format(mae_multi))
print("MSE: {:.4f}".format(mse_multi))

# 3. KNN Sınıflandırma — Iris Veri Kümesi
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Veri kümesini yükleme
iris = load_iris()
X_cls, y_cls = iris.data, iris.target

# Train-test split
X_tr_cls, X_te_cls, y_tr_cls, y_te_cls = train_test_split(
    X_cls, y_cls, test_size=0.3, random_state=42, stratify=y_cls
```

```
)

# Özellik ölçeklendirme (KNN mesafeye duyarlı olduğu için)
scaler = StandardScaler().fit(X_tr_cls)
X_tr_s = scaler.transform(X_tr_cls)
X_te_s = scaler.transform(X_te_cls)

# KNN modeli (k=5)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_tr_s, y_tr_cls)
y_pred_cls = knn.predict(X_te_s)

# Performans metrikleri
acc   = accuracy_score(y_te_cls, y_pred_cls)
prec  = precision_score(y_te_cls, y_pred_cls, average='macro')
rec   = recall_score(y_te_cls, y_pred_cls, average='macro')
f1    = f1_score(y_te_cls, y_pred_cls, average='macro')

print("\nKNN Sınıflandırma Sonuçları (Iris):")
print(f"Accuracy       : {acc:.3f}")
print(f"Precision (macro): {prec:.3f}")
print(f"Recall    (macro): {rec:.3f}")
print(f"F1 Score (macro): {f1:.3f}")
```