import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Sample data (X1, X2, and Y)

X1 = np.array([3, 4, 5, 6, 2]) # Feature 1

X2 = np.array([8, 5, 7, 3, 1]) # Feature 2

Y = np.array([-3.7, 3.5, 2.5, 11.5, 5.7]) # Target variable

# Combine X1 and X2 into a single 2D array for fitting the model

X = np.column\_stack((X1, X2))

# Create a Linear Regression model

model = LinearRegression()

# Fit the model to the data

model.fit(X, Y)

# Get the coefficients and intercept

a = model.intercept\_ # Intercept (a)

beta1, beta2 = model.coef\_ # Coefficients (β1, β2)

# Print the equation of the model

print(f"Equation of the regression model: Y = {a:.2f} + {beta1:.2f}\*X1 + {beta2:.2f}\*X2")

# Make predictions on the training data

Y\_pred = model.predict(X)

# Calculate R-squared (R²)

r\_squared = model.score(X, Y)

print(f"R-squared (R²): {r\_squared:.2f}")

# Calculate the Root Mean Squared Error (RMSE)

rmse = np.sqrt(mean\_squared\_error(Y, Y\_pred))

print(f"Root Mean Squared Error (RMSE): {rmse:.2f}")

# New data for prediction

new\_X = np.array([[3, 2]]) # New X1 and X2 values

new\_Y\_pred = model.predict(new\_X)

print(f"Predicted Y for new X1=3 and X2=2: {new\_Y\_pred[0]:.2f}")

# Visualization (Optional)

# Scatter plot for the actual vs predicted values

plt.scatter(Y, Y\_pred)

plt.plot([min(Y), max(Y)], [min(Y), max(Y)], color='red', linewidth=2) # Ideal line

plt.xlabel("Actual Y")

plt.ylabel("Predicted Y")

plt.title("Actual vs Predicted")

plt.show()



