NAME : PAREENITA A.SHIRSATH PRN : 221101062 ROLL.NO : 57 T.E.A.I.&.D.S.

**DAV EXPERIMENT : 3**

**MULTIPLE LINEAR REGRESSION**

import numpy as np

import matplotlib as mpl

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

def generate\_dataset(n):

x = []

y = []

random\_x1 = np.random.rand()

random\_x2 = np.random.rand()

for i in range(n):

x1 = i

x2 = i/2 + np.random.rand()\*n

x.append([1, x1, x2])

y.append(random\_x1 \* x1 + random\_x2 \* x2 + 1)

return np.array(x), np.array(y)

x, y = generate\_dataset(200)

mpl.rcParams['legend.fontsize'] = 12

fig = plt.figure()

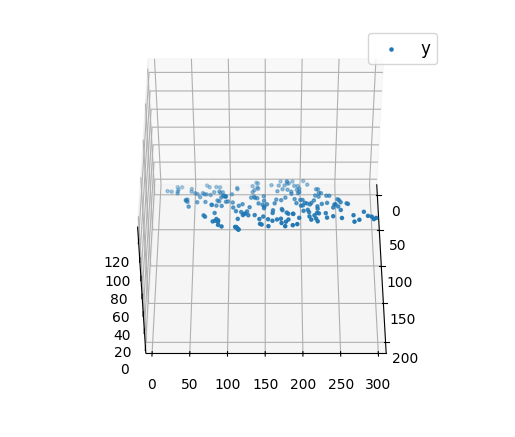
ax = fig.add\_subplot(projection='3d')

ax.scatter(x[:, 1], x[:, 2], y, label='y', s=5)

ax.legend()

ax.view\_init(45, 0)

plt.show()



# Sample data

Y = [-3.7, 3.5, 2.5, 11.5, 5.7]

X1 = [3, 4, 5, 6, 2]

X2 = [8, 5, 7, 3, 1]

# Calculate the coefficients (beta values) - same as before

mean\_X1 = sum(X1) / len(X1)

mean\_X2 = sum(X2) / len(X2)

mean\_Y = sum(Y) / len(Y)

beta1 = sum((X1[i] - mean\_X1) \* (Y[i] - mean\_Y) for i in range(len(X1))) / sum((X1[i] - mean\_X1) \*\* 2 for i in range(len(X1)))

beta2 = sum((X2[i] - mean\_X2) \* (Y[i] - mean\_Y) for i in range(len(X2))) / sum((X2[i] - mean\_X2) \*\* 2 for i in range(len(X2)))

beta0 = mean\_Y - beta1 \* mean\_X1 - beta2 \* mean\_X2

# New values for prediction

new\_X1 = 3

new\_X2 = 2

# Predict Y for new values of X1 and X2

predicted\_Y = beta0 + beta1 \* new\_X1 + beta2 \* new\_X2

# Print the predicted Y

print("Y = a + β1X1 + β2X2")

print(f"Y = {beta0} + {beta1} \* X1 + {beta2} \* X2")

print("Predicted Y for new values of X1 and X2:", predicted\_Y)



# Sample data

Y = [-3.7, 3.5, 2.5, 11.5, 5.7]

X1 = [3, 4, 5, 6, 2]

X2 = [8, 5, 7, 3, 1]

# New data for prediction

new\_x1 = 3

new\_x2 = 2

# Number of data points

n = len(X1)

# Calculate the means

mean\_x1 = sum(X1) / n

mean\_x2 = sum(X2) / n

mean\_y = sum(Y) / n

numerator1 = sum((X1[i] - mean\_x1) \* (Y[i] - mean\_y) for i in range(n))

denominator1 = sum((X1[i] - mean\_x1) \*\* 2 for i in range(n))

beta1 = numerator1 / denominator1

numerator2 = sum((X2[i] - mean\_x2) \* (Y[i] - mean\_y) for i in range(n))

denominator2 = sum((X2[i] - mean\_x2) \*\* 2 for i in range(n))

beta2 = numerator2 / denominator2

a = mean\_y - beta1 \* mean\_x1 - beta2 \* mean\_x2

print("Intercept a =", a)

print("β1 =", beta1)

print("β2 =", beta2)

print("Estimate X1 =", new\_x1)

print("Estimate X2 =", new\_x2)

print("\n")

# Make predictions for new values

new\_y = a + beta1 \* new\_x1 + beta2 \* new\_x2

print("Y = a + β1X1 + β2X2")

print(f"Y = {a} + {beta1} \* X1 + {beta2} \* X2")

# Make predictions for new values

new\_y = a + beta1 \* new\_x1 + beta2 \* new\_x2

print("\n")

print("Predicted y for new X1 and X2:", new\_y)



**RMSE:**

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)

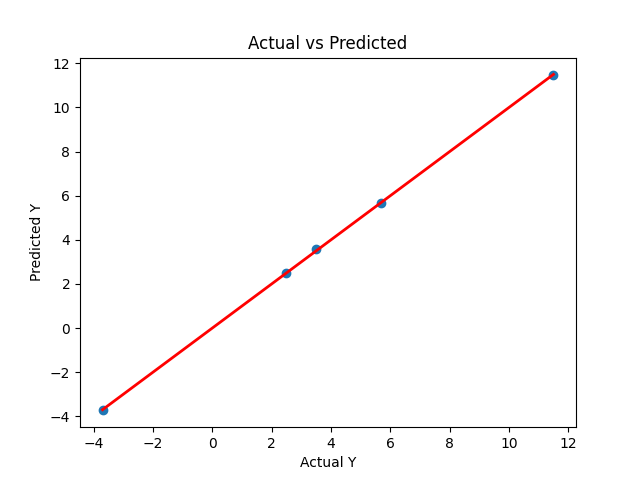
plt.plot([min(Y), max(Y)], [min(Y), max(Y)], color='red', linewidth=2) # Ideal line

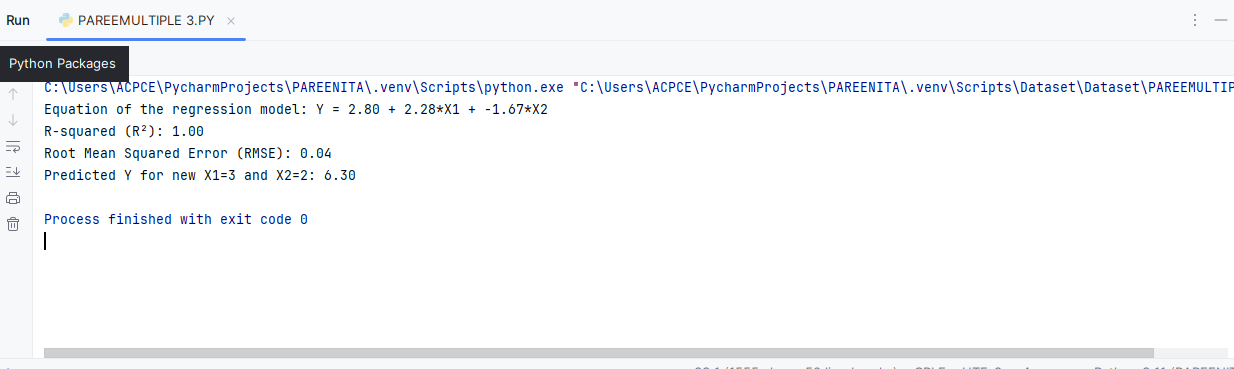
plt.xlabel("Actual Y")

plt.ylabel("Predicted Y")

plt.title("Actual vs Predicted")

plt.show()





**MSE:**

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]).reshape(-1, 1) # Feature 1

X2 = np.array([8, 5, 7, 3, 1]).reshape(-1, 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.hstack((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")

# Calculate the predicted values

Y\_pred = model.predict(X)

# Calculate Mean Squared Error (MSE)

mse = mean\_squared\_error(Y, Y\_pred)

print(f"Mean Squared Error (MSE): {mse:.2f}")

# Calculate Root Mean Squared Error (RMSE)

rmse = np.sqrt(mse)

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 (Y = Y\_pred)

plt.xlabel("Actual Y")

plt.ylabel("Predicted Y")

plt.title("Actual vs Predicted")

plt.show()

