**Practical: 3**

**To Implement Linear Regression Model**

**Code:**

import numpy as np

import matplotlib.pyplot as plt

# MSE (Mean Squared Error) function

def equation(x, y):

x\_mean, y\_mean = np.mean(x), np.mean(y)

x\_ = [i - x\_mean for i in x]

y\_ = [i - y\_mean for i in y]

# Covariance(x, y) = (np.array(x\_) \* np.array(y\_)).sum()

# Variance = (np.array(x\_)\*\*2).sum()

b1 = (np.array(x\_) \* np.array(y\_)).sum() / (np.array(x\_)\*\*2).sum()

b0 = y\_mean - (b1 \* x\_mean)

# b1, b0 are coefficients of the line

y\_pred = [b1 \* i + b0 for i in x]

# SST is the total sum of squares and

# SSR is the total sum of squares of residuals.

ssr = ((np.array(y\_pred) - y\_mean)\*\*2).sum()

sst = ((np.array(y\_))\*\*2).sum()

# R² Score usually ranges from 0 to 1.

r\_squared = ssr / sst

score = 1 - round(r\_squared, 2)

return (b1, b0, score)

x = np.array([np.random.randint(1, 10) for i in range(20)])

y = np.array([np.random.randint(1, 10) for i in range(20)])

# x = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

# y = np.array([3, 1, 2, 7, 5, 6, 4, 10, 9, 5])

plt.plot(x, y, 'bo', markersize=10)

# Generate a line of best fit

b1, b0, score = equation(x, y)

y\_pred = b1 \* x + b0

plt.plot(x, y\_pred, 'go-', markersize=10)

# Print the accuracy of the model

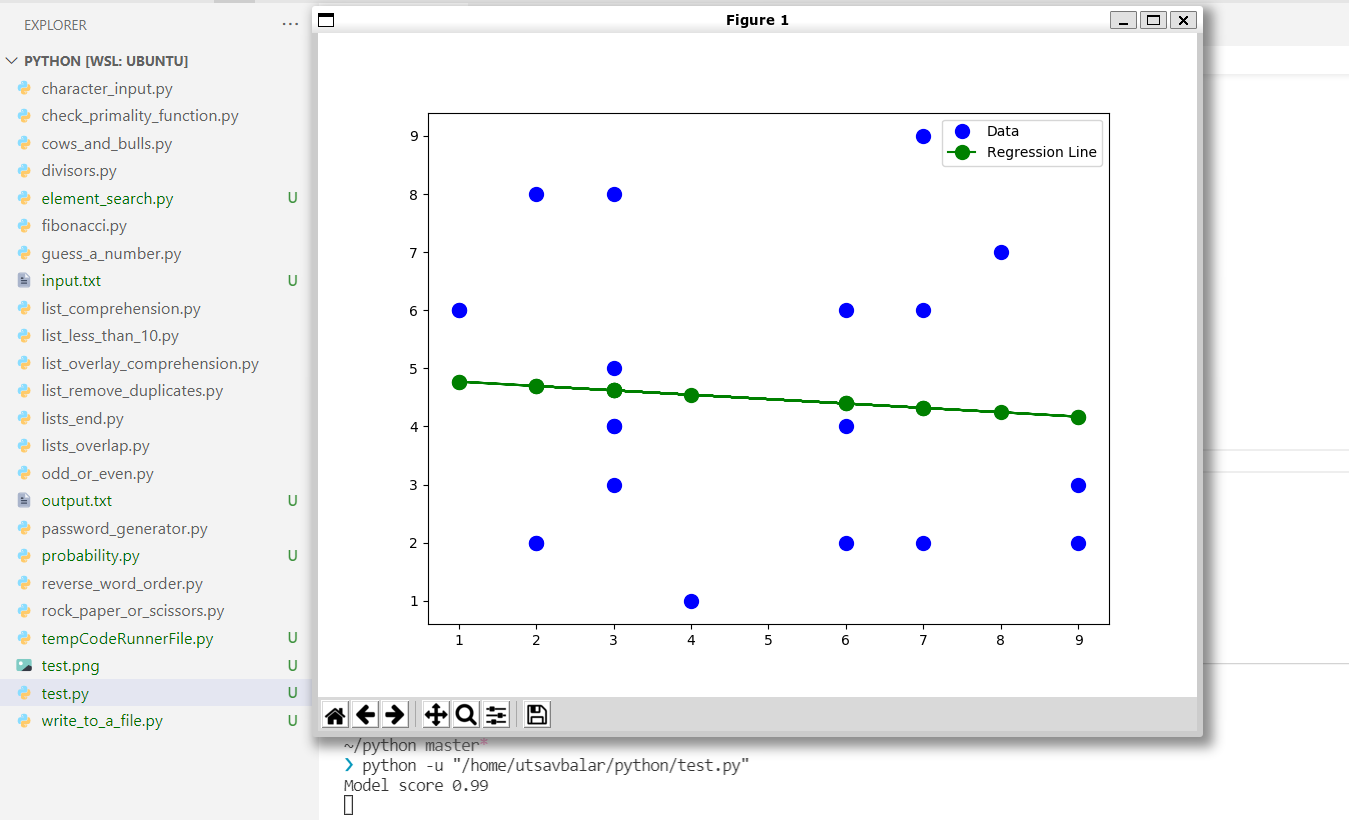
print("Model score", score)

plt.legend(['Data', 'Regression Line'])

plt.show()

**Output:**

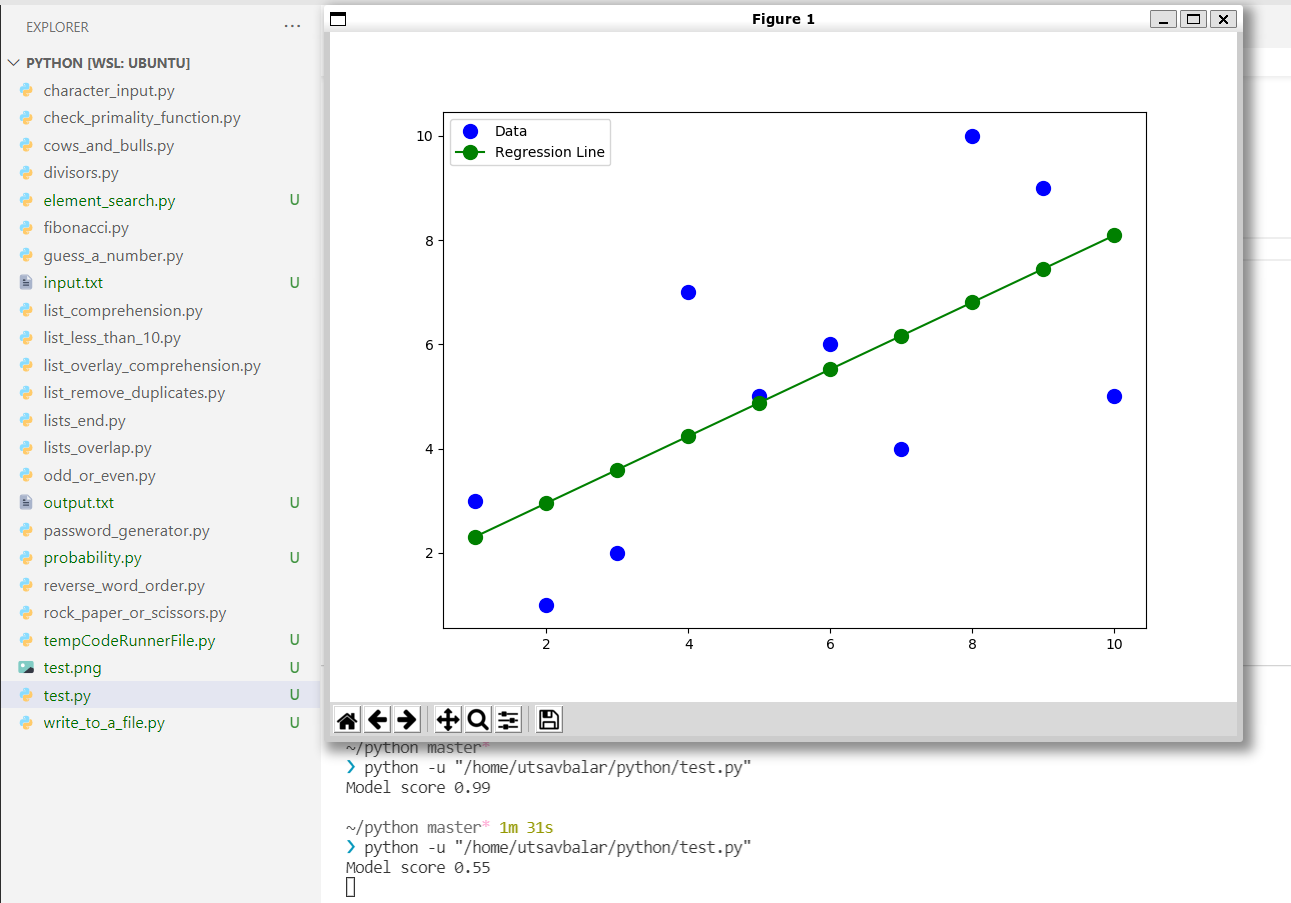
**When x and y has array of size 20 with random values in range 0, 10**

****

**When**

**x = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]**

**y = [3, 1, 2, 7, 5, 6, 4, 10, 9, 5]**

****