Lab 3 :

1. Using matrix method

import numpy as np

import matplotlib.pyplot as plt

# Sample data (X and y)

X = np.array([1, 2, 3, 4, 5])  # Input features

y = np.array([1.2, 1.8, 2.6, 3.2, 3.8])  # Target variable

# Add a column of ones to X for the bias term (intercept)

X\_b = np.c\_[np.ones((X.shape[0], 1)), X]

# Calculate the coefficients (intercept and slope) using the Normal Equation

theta = np.linalg.inv(X\_b.T.dot(X\_b)).dot(X\_b.T).dot(y)

# Intercept and Slope

intercept, slope = theta

# Print intercept and slope

print(f"Intercept (theta0): {intercept}")

print(f"Slope (theta1): {slope}")

# Make predictions for a new input value of X

x\_input = float(input("Enter a value for X to predict y: "))

x\_input\_b = np.array([1, x\_input])  # Add 1 for the bias term

y\_pred = x\_input\_b.dot(theta)  # Calculate the predicted y

# Display the predicted value

print(f"Predicted y for X = {x\_input}: {y\_pred}")

# Visualize the data and the regression line

plt.scatter(X, y, color='blue', label='Data points')

plt.plot(X, X\_b.dot(theta), color='red', label='Regression line')

plt.xlabel('X')

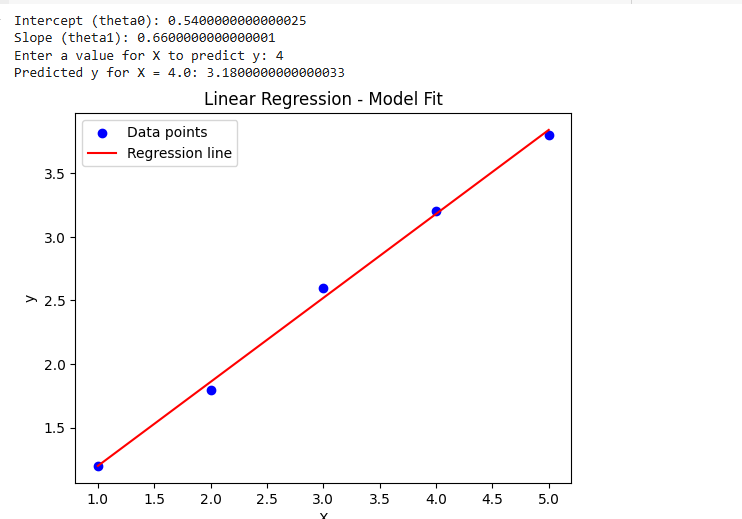
plt.ylabel('y')

plt.title('Linear Regression - Model Fit')

plt.legend()

plt.show()

Output :



1. Using LinearRegresssion model :

import numpy as np

import pandas as pd

from sklearn.linear\_model import LinearRegression

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

# Create xi values (1, 2, 3, 4, 5)

xi = np.array([1, 2, 3, 4, 5]).reshape(-1, 1)

# Create random yi values

yi = np.array([1.2, 1.8, 2.6, 3.2, 3.8]) # Random values between 0 and 10

# Create a pandas DataFrame

data = pd.DataFrame({'xi': xi.flatten(), 'yi': yi.flatten()})

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

    data[['xi']], data['yi'], test\_size=0.2, random\_state=42

)

model = LinearRegression()

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

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

from sklearn.metrics import mean\_squared\_error, r2\_score

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

r2 = r2\_score(y\_test, y\_pred)

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

print(f"R-squared: {r2}")

new\_x = np.array([[6]])  # Reshape to a column vector

predicted\_y = model.predict(new\_x)

print(f"Predicted y for x = 6: {predicted\_y[0]}")

# Display intercept and coefficient

print(f"Intercept: {model.intercept\_}")

print(f"Coefficient (slope): {model.coef\_[0]}")

import matplotlib.pyplot as plt

# Scatter plot of the original data points

plt.scatter(X\_train, y\_train, color='blue', label='Training Data')

plt.scatter(X\_test, y\_test, color='green', label='Testing Data')

# Plot the regression line

plt.plot(X\_train, model.predict(X\_train), color='red', label='Regression Line')

# Add labels and title

plt.xlabel('xi')

plt.ylabel('yi')

plt.title('Linear Regression Model')

# Display the legend

plt.legend()

# Show the plot

plt.show()

Output :

