211112262

VAIBHAV PATEL

In [ ]:
```
import numpy as np
# Define the data
x = np.array([3, 4, 5, 7, 5])
y = np.array([1, 2, 3, 4, 5])
# Define the learning rate and number of iterations
learning_rate = 0.02
iterations = 100
# Initialize the weights
m = 0
b = 0
# Implement gradient descent algorithm
for _ in range(iterations):
 # Calculate the predictions
 y_pred = m * x + b
 # Calculate the errors
 errors = y - y_pred
 # Calculate the gradients
 m_gradient = -2 * np.mean(x * errors)
 b_gradient = -2 * np.mean(errors)
 # Update the weights
 m -= learning_rate * m_gradient
 b -= learning_rate * b_gradient
# Calculate the RMSE
rmse = np.sqrt(np.mean(errors**2))
# Calculate the MAE
mae = np.mean(np.abs(errors))
# Calculate the coefficient of determination (R^2)
r_squared = 1 - np.var(errors) / np.var(y)
# Print the results
print(f"RMSE: {rmse:.4f}")
print(f"MAE: {mae:.4f}")
print(f"R^2: {r_squared:.4f}")
# Implement ridge regularization
lambda_value = 0.1
for _ in range(iterations):
 # Calculate the predictions
 y_pred = m * x + b
 # Calculate the errors
 errors = y - y_pred
 # Calculate the gradients with regularization
 m_gradient = -2 * np.mean(x * errors) - 2 * lambda_value * m
 b_gradient = -2 * np.mean(errors)
 # Update the weights
 m -= learning_rate * m_gradient
 b -= learning_rate * b_gradient
# Calculate the RMSE with regularization
rmse_reg = np.sqrt(np.mean(errors**2))
# Print the results with regularization
print(f"\nRMSE with Ridge Regularization (lambda={lambda_value}): {rmse_r
```

```
RMSE: 0.9605
MAE: 0.7768
R^2: 0.5399

RMSE with Ridge Regularization (lambda=0.1): 0.9509
```