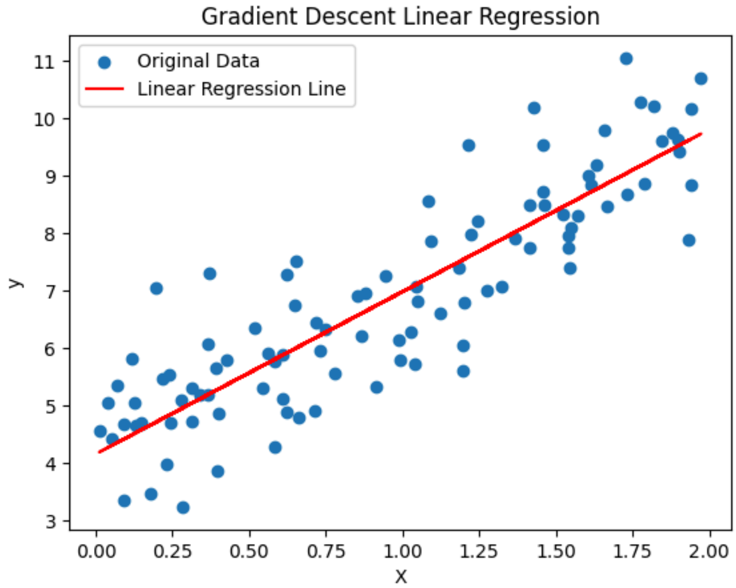
```
import numpy as np
import matplotlib.pyplot as plt

# Generate random data for demonstration
np.random.seed(42)
X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)

# Function to compute Mean Squared Error (MSE)
def compute_mse(y_true, y_pred):
    return np.mean((y_true - y_pred) ** 2)

# Gradient Descent function
def gradient_descent(X, y, learning_rate=0.01, n_iterations=1000):
    m = len(X)
    theta = np.random.randn(2, 1)  # Random initialization of parameters (intercept and slope)

    for iteration in range(n_iterations):
        gradients = 2/m * X.T.dot(X.dot(theta) - y)
        theta -= learning_rate * gradients

    return theta

# Add a bias term to the input features (X)
X_b = np.c_[np.ones((100, 1)), X]

# Run Gradient Descent
theta = gradient_descent(X_b, y)

# Predictions using the learned parameters
y_pred = X_b.dot(theta)

# Plot the original data and the linear regression line
plt.scatter(X, y, label='Original Data')
plt.plot(X, y_pred, color='red', label='Linear Regression Line')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.title('Gradient Descent Linear Regression')
plt.show()

# Evaluate performance using Mean Squared Error (MSE)
mse = compute_mse(y, y_pred)
print(f'Mean Squared Error: {mse}')
```




```
Mean Squared Error: 0.8075659033465308
```