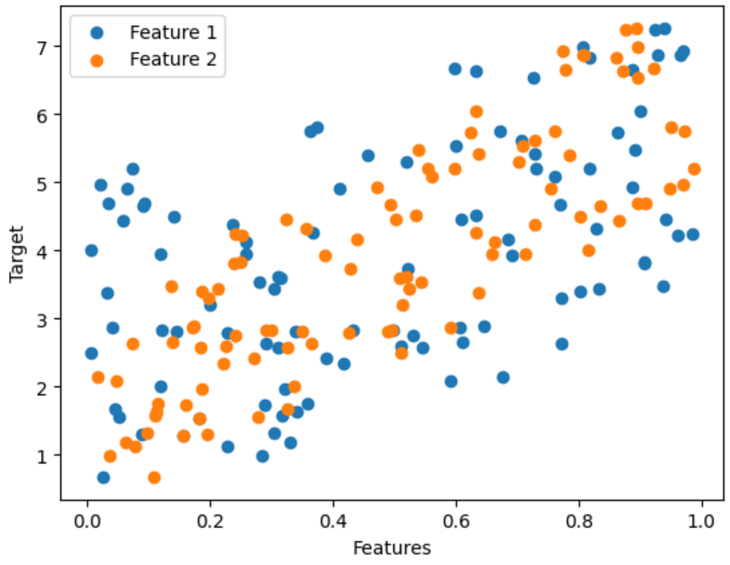
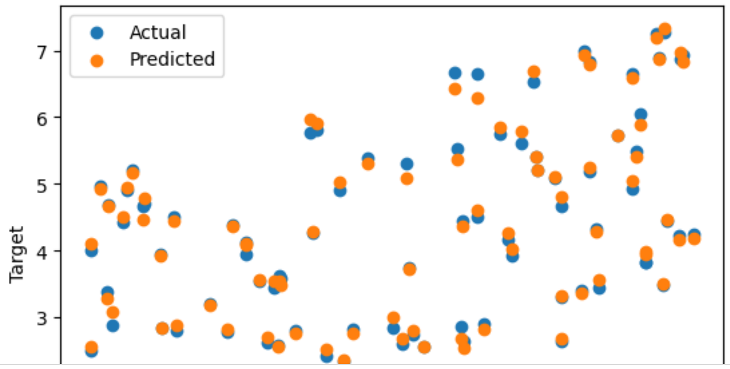
```python
import numpy as np
import matplotlib.pyplot as plt

# Generate synthetic data
np.random.seed(42)
num_samples = 100
num_features = 2

# Generate random features
X = np.random.rand(num_samples, num_features)

# Generate target variable (perfectly linear with coefficients [3, 5])
coefficients = np.array([3, 5])
noise = np.random.randn(num_samples) * 0.1  # Adding small noise
y = X.dot(coefficients) + noise

# Visualize the synthetic data
plt.scatter(X[:, 0], y, label='Feature 1')
plt.scatter(X[:, 1], y, label='Feature 2')
plt.xlabel('Features')
plt.ylabel('Target')
plt.legend()
plt.show()

# Add a column of ones to X for the bias term
X_bias = np.c_[np.ones(num_samples), X]

# Compute the optimal weights using closed-form solution (Normal Equation)
weights = np.linalg.inv(X_bias.T.dot(X_bias)).dot(X_bias.T).dot(y)

# Extract bias (intercept) and feature coefficients
bias = weights[0]
coefficients = weights[1:]

print("Bias (Intercept): {:.2f}".format(bias))
print("Coefficients (Feature Weights):", coefficients)

# Make predictions using the trained model
predictions = X_bias.dot(weights)

# Visualize the results
plt.scatter(X[:, 0], y, label='Actual')
plt.scatter(X[:, 0], predictions, label='Predicted')
plt.xlabel('Feature 1')
plt.ylabel('Target')
plt.legend()
plt.show()
```



```
Bias (Intercept): -0.02
Coefficients (Feature Weights): [3.03386668 5.0354946 ]
```



acts here

✓ 1s completed at 7:05 PM