2. To implement linear regression and evaluate using MSE/R2 score in machine learning

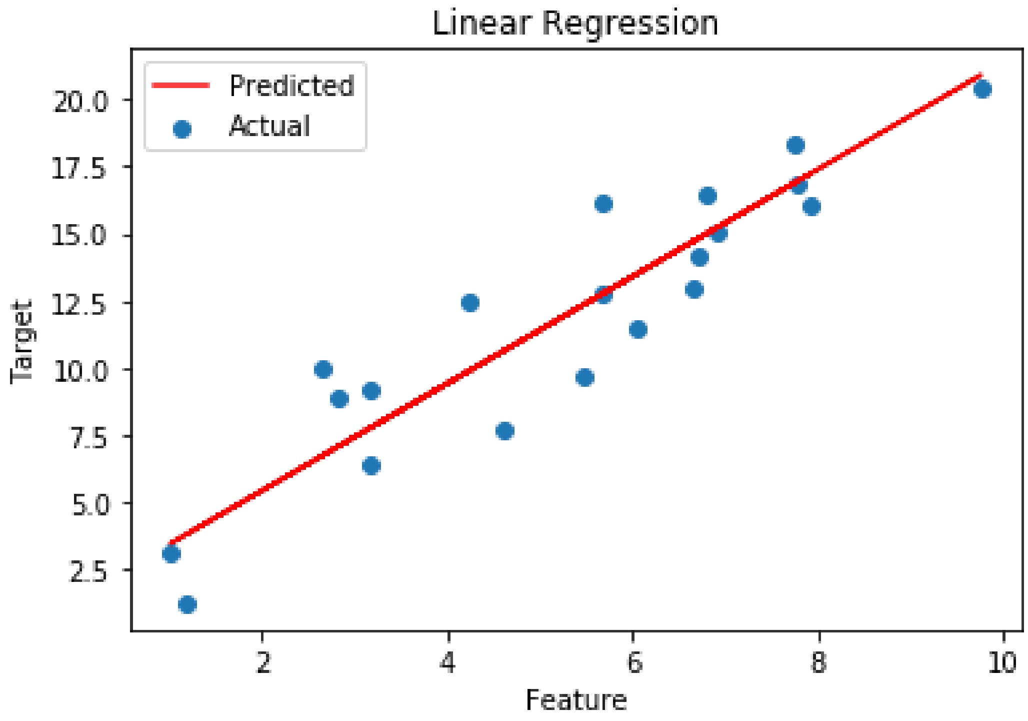
```
import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error, r2_score

import matplotlib.pyplot as plt


# 1. Sample Data (replace with your actual data)

# Assuming you have a DataFrame 'df' with features in columns and target in a column named 'target'

# For demonstration, let's create some sample data:

np.random.seed(0)

X = np.random.rand(100, 1) * 10  # 100 samples, 1 feature

y = 2 * X + 1 + np.random.randn(100, 1) * 2  # Linear relationship with some noise


# Convert to Pandas DataFrame for easier handling

df = pd.DataFrame({'feature': X.flatten(), 'target': y.flatten()})
```

```
# 2. Split Data into Training and Testing Sets
X = df[['feature']]
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Train the Linear Regression Model
model = LinearRegression()
model.fit(X_train, y_train)

# 4. Make Predictions
y_pred = model.predict(X_test)

# 5. Evaluate the Model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse:.2f}")
print(f"R-squared: {r2:.2f}")

# 6. (Optional) Visualize the Results
```

```
plt.scatter(X_test, y_test, label="Actual")
plt.plot(X_test, y_pred, color='red', label="Predicted")
plt.xlabel("Feature")
plt.ylabel("Target")
plt.title("Linear Regression")
plt.legend()
plt.show()

# 7. Get Model Coefficients (optional)
print(f"Intercept: {model.intercept_}")
print(f"Coefficient: {model.coef_}")
```

output:

```
Mean Squared Error: 3.67
R-squared: 0.85
```



```
Intercept: 1.412680377422868

Coefficient: [1.99610364]
```