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)

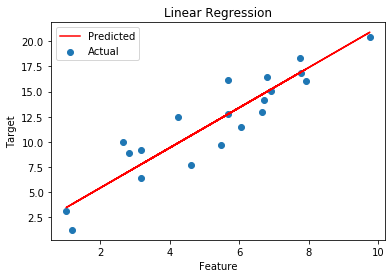
print(f"Intercept: {model.intercept\_}")

print(f"Coefficient: {model.coef\_}")

output:

Mean Squared Error: 3.67

R-squared: 0.85



Intercept: 1.412680377422868

Coefficient: [1.99610364]

​