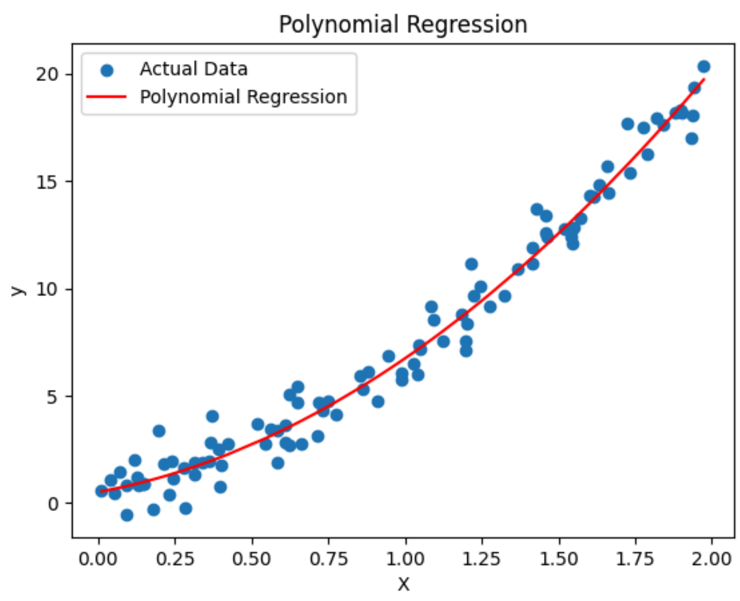
```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

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

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Polynomial features transformation
degree = 2  # Set the degree of the polynomial
poly_features = PolynomialFeatures(degree=degree, include_bias=False)
X_poly_train = poly_features.fit_transform(X_train)
X_poly_test = poly_features.transform(X_test)

# Train a linear regression model on the polynomial features
model = LinearRegression()
model.fit(X_poly_train, y_train)

# Make predictions on the training set
y_train_pred = model.predict(X_poly_train)

# Make predictions on the testing set
y_test_pred = model.predict(X_poly_test)

# Evaluate performance
train_rmse = np.sqrt(mean_squared_error(y_train, y_train_pred))
test_rmse = np.sqrt(mean_squared_error(y_test, y_test_pred))
r2_train = r2_score(y_train, y_train_pred)
r2_test = r2_score(y_test, y_test_pred)

# Plot the results
plt.scatter(X, y, label='Actual Data')
plt.plot(np.sort(X_train, axis=0), np.sort(y_train_pred, axis=0), label='Polynomial Regression', color='r')
plt.xlabel('X')
plt.ylabel('y')
plt.title('Polynomial Regression')
plt.legend()
plt.show()

# Display performance metrics
print(f'Degree of Polynomial: {degree}')
print(f'Training RMSE: {train_rmse:.2f}')
print(f'Testing RMSE: {test_rmse:.2f}')
print(f'R-squared (Training): {r2_train:.2f}')
print(f'R-squared (Testing): {r2_test:.2f}')
```


Polynomial Regression
Actual Data
Polynomial Regression
y
X
0
5
10
15
20
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00