Additional Examples of Linear Regression, Polynomial

Example 1: Predicting Student Scores Based on Study Hours

Dataset:

Study Hours	Score
1	50
2	55
3	65
4	70
5	85

Linear Regression

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Dataset
data = {'Study Hours': [1, 2, 3, 4, 5],
        'Score': [50, 55, 65, 70, 85]}
df = pd.DataFrame(data)

# Features (X) and Target (y)
X = df[['Study Hours']]
y = df['Score']

# Train the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predictions
y_pred = model.predict(X)

# Visualize the data and regression line
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.title('Linear Regression: Study Hours vs Score')
plt.xlabel('Study Hours')
plt.ylabel('Score')
plt.legend()
plt.show()
```

Explanation:

- The **red line** shows the predicted scores for given study hours.
- Linear regression assumes that more hours of study proportionally increase the score.

Activity: Evaluation of Model

Use the **Mean Absolute Error (MAE)** to evaluate how well the model predicts scores.

```
from sklearn.metrics import mean_absolute_error

# Evaluate the model
mae = mean_absolute_error(y, y_pred)
print("Mean Absolute Error (MAE):", mae)
```

Expected Output:

- MAE will show the average error between the actual and predicted scores.

Example 2: Predicting Car Prices Based on Mileage

Dataset:

Mileage (MPG)	Price ($)
30	20000
35	18000
40	16000
45	14000
50	12000

Linear Regression

```
# Dataset
data = {'Mileage': [30, 35, 40, 45, 50],
        'Price': [20000, 18000, 16000, 14000, 12000]}
df = pd.DataFrame(data)

# Features (X) and Target (y)
X = df[['Mileage']]
y = df['Price']

# Train the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predictions
y_pred = model.predict(X)

# Visualize the data and regression line
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.title('Linear Regression: Mileage vs Price')
plt.xlabel('Mileage (MPG)')
plt.ylabel('Price ($)')
plt.legend()
plt.show()
```

Explanation:

- As mileage increases, the price decreases. The regression line represents this trend.

Polynomial Regression for Car Prices

Sometimes, the relationship may not be perfectly linear. Let's try **polynomial regression**.

```
from sklearn.preprocessing import PolynomialFeatures

# Create polynomial features
poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(X)

# Train the polynomial regression model
poly_model = LinearRegression()
poly_model.fit(X_poly, y)

# Predictions
y_poly_pred = poly_model.predict(X_poly)

# Visualize polynomial regression
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, y_poly_pred, color='green', label='Polynomial Regression Curve')
plt.title('Polynomial Regression: Mileage vs Price')
plt.xlabel('Mileage (MPG)')
plt.ylabel('Price ($)')
plt.legend()
plt.show()
```

Explanation:

- Polynomial regression captures the **non-linear** trend that price drops faster initially as mileage increases.

Example 3: Predicting House Rent Based on Area

Dataset:

Area (sq ft)	Rent ($)
500	1500
1000	2000
1500	2500
2000	3000
2500	4000

Linear Regression

```
# Dataset
data = {'Area': [500, 1000, 1500, 2000, 2500],
        'Rent': [1500, 2000, 2500, 3000, 4000]}
df = pd.DataFrame(data)

# Features (X) and Target (y)
X = df[['Area']]
y = df['Rent']

# Train the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predictions
y_pred = model.predict(X)

# Visualize the data and regression line
plt.scatter(X, y, color='blue', label='Actual Data')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.title('Linear Regression: Area vs Rent')
plt.xlabel('Area (sq ft)')
plt.ylabel('Rent ($)')
plt.legend()
plt.show()
```

Explanation:

- The linear regression line shows a proportional increase in rent with area.

Evaluation Activity

1. **Calculate MAE, MSE, and RMSE for Each Model**
   Use the following code to evaluate the models.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np

# Evaluate the model
mae = mean_absolute_error(y, y_pred)
mse = mean_squared_error(y, y_pred)
rmse = np.sqrt(mse)

print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
```

Task:

- Compare the performance metrics of linear and polynomial regression for different datasets.
- Analyze which method (linear or polynomial) works better based on the nature of the data.

Example 4: Predicting Student Performance with Multiple Features

Dataset:

Study Hours	Sleep Hours	Score
2	8	50
3	7	60
5	6	70
6	5	80
8	4	90

Linear Regression with Multiple Features

```
# Dataset
data = {'Study Hours': [2, 3, 5, 6, 8],
        'Sleep Hours': [8, 7, 6, 5, 4],
        'Score': [50, 60, 70, 80, 90]}
df = pd.DataFrame(data)

# Features (X) and Target (y)
X = df[['Study Hours', 'Sleep Hours']]
y = df['Score']

# Train the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predictions
y_pred = model.predict(X)

print("Predicted Scores:", y_pred)
```

Explanation:

- Both `Study Hours` and `Sleep Hours` influence the predicted score.
- Students can experiment by changing the dataset values to see how predictions change.

Summary

- Regression is a versatile tool for predicting numerical values.
- Linear regression is simple but assumes a straight-line relationship.
- Polynomial regression captures non-linear relationships.
- Evaluation metrics like MAE, MSE, and RMSE help measure the model's accuracy.