Polynomial Regression

Polynomial Regression

- **Polynomial Regression** is a type of regression analysis used to model the relationship between an independent variable (or input feature) and a dependent variable (or output) by fitting a polynomial function to the data. Unlike simple linear regression, which models the relationship as a straight line, polynomial regression allows for curved or non-linear relationships between variables.

Significance

- **Capturing Non-Linear Patterns:** Polynomial regression is used when there is a non-linear relationship between the variables, and fitting a straight line wouldn't adequately capture the data's behavior.
- **Flexibility:** It offers flexibility by allowing the use of higher-degree polynomial functions to model complex relationships.
- **Balance of Bias and Variance:** The choice of the degree of the polynomial balances the bias-variance trade-off. Lower-degree polynomials are simpler but may underfit, while higher-degree polynomials may overfit.

Implementation

- **Model Form:** The model equation for polynomial regression is often expressed as

  $$h_\Theta(x) = g[\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \Theta_3 x_1^2 + \Theta_4 x_2^2]$$

  where Ø represents coefficients, x is the input variable with the degree of the polynomial.
- **Degree Selection:** You need to choose an appropriate degree (n) for the polynomial, which is a hyperparameter. Cross-validation or other model selection techniques can help in making this choice.

Decision Boundary

- For degree 1 (simple linear regression), the decision boundary is a straight line.
- For degree 2, the decision boundary can be a parabola.
- For higher degrees, the decision boundary can be more complex, possibly resembling curves, loops, or intricate shapes.

Uses

- **Regression Problems:** Polynomial regression is used for regression tasks where you need to model the relationship between variables with curves. It's valuable in fields such as economics, physics, and environmental science.
- **Curve Fitting:** It's often used for curve fitting in experimental data analysis.
- **Engineering:** In engineering applications, polynomial regression is used to model non-linear behavior in materials, systems, and processes.
- **Financial Modeling:** In finance, it's applied to model financial time series data, where the relationship between variables may not be linear.
- **Predictive Modeling:** In machine learning, polynomial regression can be used for predictive modeling when the data shows non-linear relationships between features and the target variable.

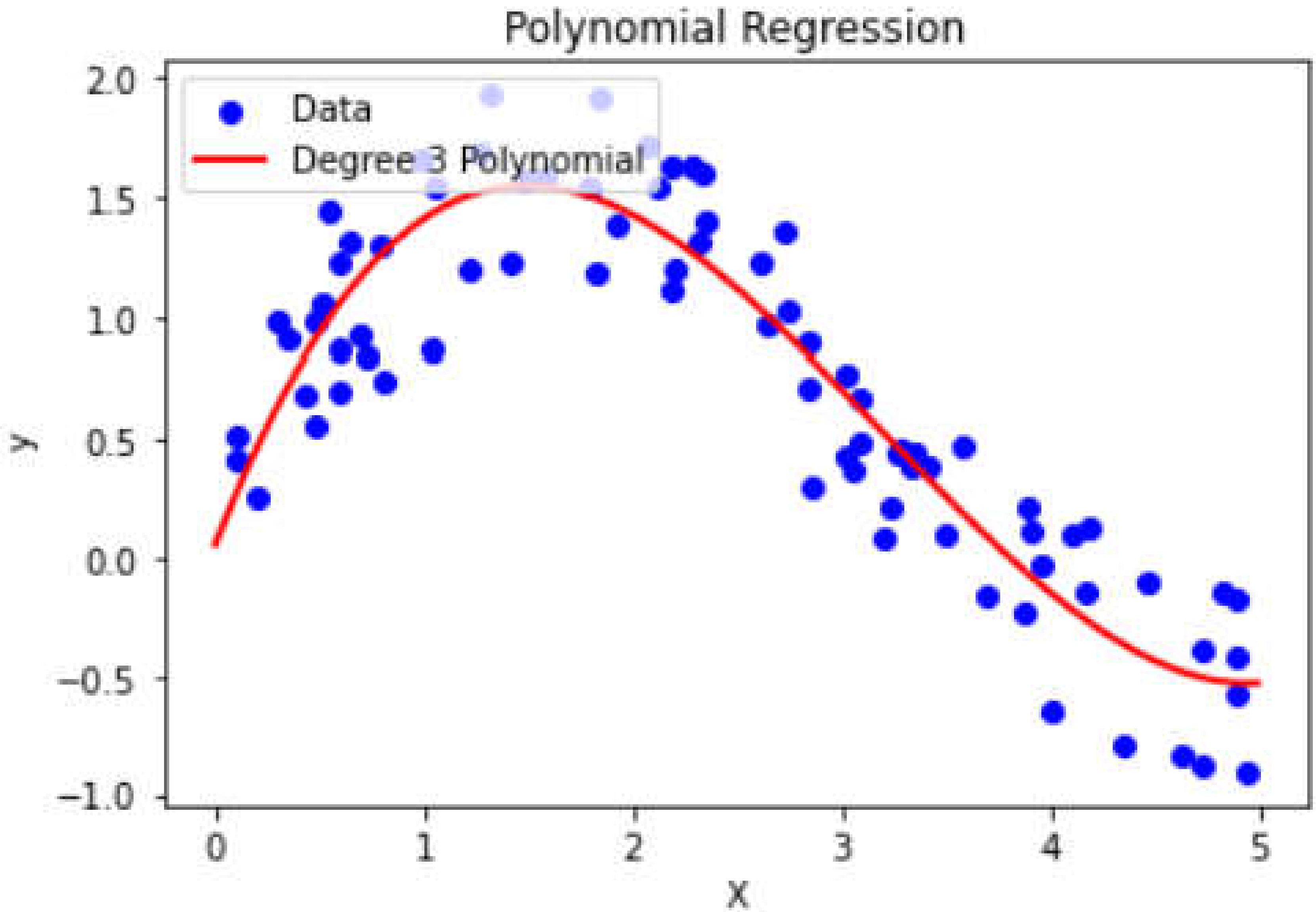
Examples

- Imagine you're analyzing the relationship between temperature and ice cream sales. Simple linear regression might not capture the pattern accurately. By using polynomial regression, you can model the temperature-sales relationship with a curve, which better reflects the real-world scenario where ice cream sales increase as temperature rises but then saturate at high temperatures.

Examples

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import
    LinearRegression
from sklearn.preprocessing import
    PolynomialFeatures

# Generate synthetic data
np.random.seed(0)
X = np.sort(5 * np.random.rand(80, 1), axis=0)
y = np.sin(X).ravel() + np.random.rand(80)

# Fit polynomial regression model
degree = 3
poly_features =
    PolynomialFeatures(degree=degree)
X_poly = poly_features.fit_transform(X)
model = LinearRegression()
model.fit(X_poly, y)

# Predict the values
X_test = np.arange(0, 5, 0.01)[:, np.newaxis]
X_test_poly = poly_features.transform(X_test)
y_pred = model.predict(X_test_poly)

# Plot the data and the regression curve
plt.scatter(X, y, color='blue', label='Data')
plt.plot(X_test, y_pred, color='red', linewidth=2,
    label=f'Degree {degree} Polynomial')
plt.xlabel('X')
plt.ylabel('y')
plt.legend(loc='upper left')
plt.title('Polynomial Regression')
plt.show()
```

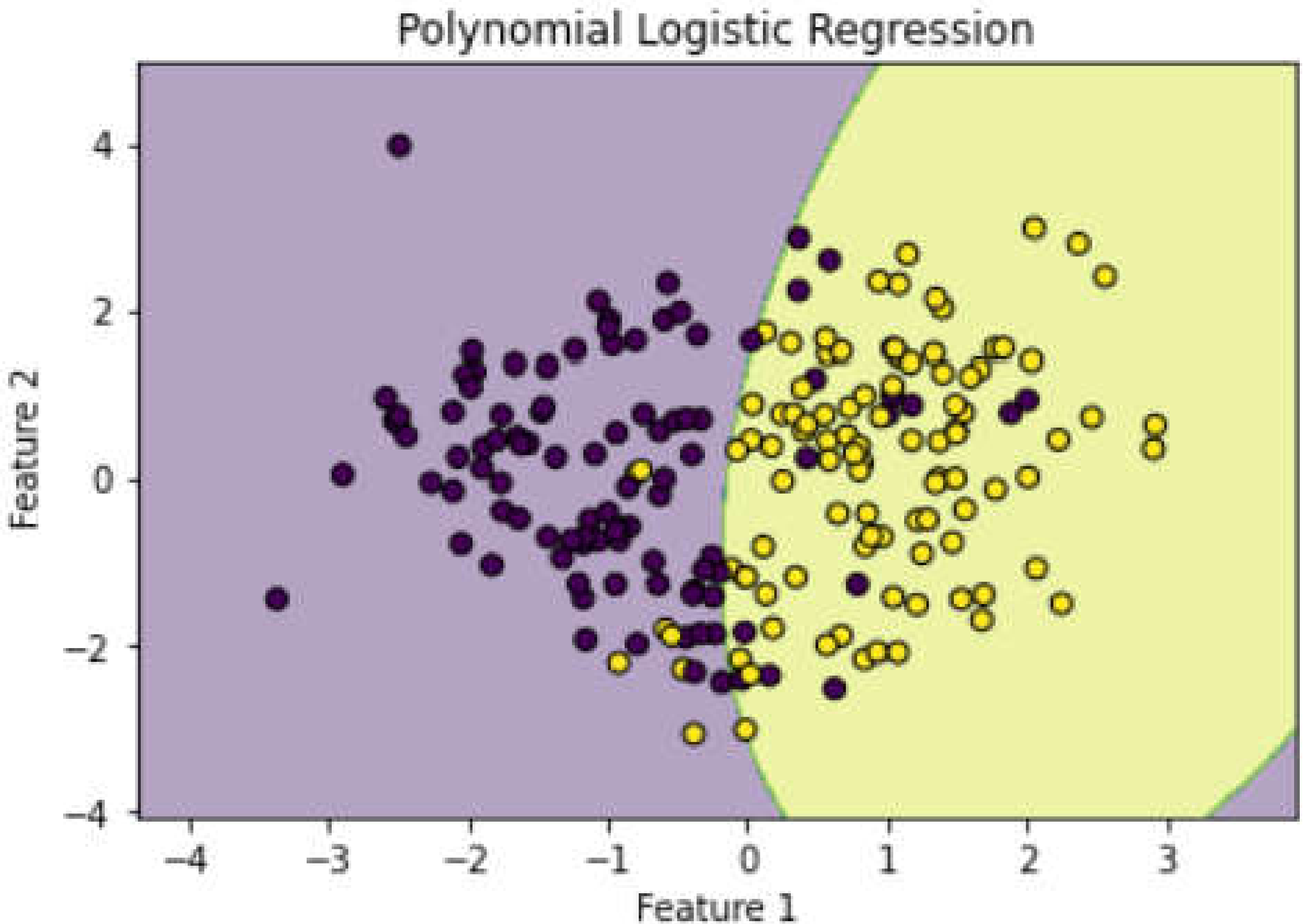
Examples


Polynomial Regression
Data
Degree 3 Polynomial
y
X
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5

Examples

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split

# Generate synthetic data for classification
X, y = make_classification(n_samples=200, n_features=2,
    n_informative=2, n_redundant=0, random_state=42)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.3, random_state=42)
# Apply polynomial features
degree = 2
poly_features = PolynomialFeatures(degree=degree)
X_train_poly = poly_features.fit_transform(X_train)
X_test_poly = poly_features.transform(X_test)

# Fit logistic regression with polynomial features
model = LogisticRegression()
model.fit(X_train_poly, y_train)

# Make predictions
y_pred = model.predict(X_test_poly)

# Plot the decision boundary
xx, yy = np.meshgrid(np.arange(X[:, 0].min() - 1, X[:, 0].max()
    + 1, 0.01),
        np.arange(X[:, 1].min() - 1, X[:, 1].max() + 1,
    0.01))
Z = model.predict(poly_features.transform(np.c_[xx.ravel(),
    yy.ravel()]))
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, marker='o', edgecolors='k')
plt.title('Polynomial Logistic Regression')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

Examples


Polynomial Logistic Regression
Feature 2
Feature 1
4
2
0
−2
−4
−4
−3
−2
−1
0
1
2
3