prac3-1

October 18, 2024

Aim: Implementing polynomial regression

Theory: ### Polynomial Regression: Theoretical Overview

Polynomial Regression is an extension of linear regression where the relationship between the independent variable(s) and the dependent variable is modeled as an (n)-degree polynomial. Instead of fitting a straight line (as in linear regression), polynomial regression fits a curve to capture more complex patterns in the data.

0.0.1 Key Concepts

1. **Purpose of Polynomial Regression**
   - **Capturing Non-Linearity**: Linear regression can only model linear relationships between variables, but many real-world relationships are non-linear. Polynomial regression allows for the modeling of such non-linear relationships by adding powers of the independent variable(s) as new features.
   - **Flexibility**: The model can fit more complex patterns in the data by using higher-degree polynomials.
2. **How Polynomial Regression Works**
   - **Features Transformation**: In polynomial regression, the input features are transformed to include powers of the original features. For example, if the original feature is (x), a polynomial regression model of degree 2 will include (x), (x^2) as the input features.
   - **Fit a Curve**: Instead of a straight line, the model fits a polynomial curve that can capture more intricate patterns in the data. This is useful when the data shows curvature or higher-order relationships.
   - **Higher Degrees for More Complexity**: The degree of the polynomial determines how flexible the model is. A higher-degree polynomial can fit more complex data but risks overfitting, while a lower-degree polynomial might underfit if the relationship is too complex.
3. **Training the Model**
   - **Feature Expansion**: The original features are transformed by adding powers of the feature values. For example, if the original feature is (x), a degree 3 polynomial model would use (x), (x^2), and (x^3).
   - **Linear Regression on Transformed Features**: Once the feature set is expanded to include polynomial terms, the model is trained using the same method as linear regression. The coefficients (weights) of the polynomial terms are learned by minimizing the error between the predicted and actual values.

    - **Learning the Coefficients**: The goal of training is to find the best-fitting polynomial by adjusting the coefficients of the polynomial terms to minimize the error.
4. **Prediction**
    - **Using the Fitted Polynomial**: Once the model is trained, new data points are fed into the polynomial equation with the learned coefficients. The model calculates a predicted value for the dependent variable based on the polynomial relationship between the input features and the target variable.
5. **Overfitting and Underfitting**
    - **Overfitting**: When using higher-degree polynomials, there is a risk that the model will overfit the training data. This means the model becomes too complex and captures the noise in the data rather than the underlying relationship, leading to poor generalization on new data.
    - **Underfitting**: If the degree of the polynomial is too low, the model may not be complex enough to capture the relationship in the data, leading to poor performance (underfitting).
6. **Choosing the Right Degree**
    - **Cross-Validation**: To avoid overfitting or underfitting, cross-validation is often used to choose the appropriate degree for the polynomial. By testing different polynomial degrees on validation sets, you can find the one that balances model complexity with generalization.

0.0.2 Applications

- **Predicting Curved Trends**: Polynomial regression is used in situations where the data has a curvilinear relationship. For example, it can model the growth of a population, pricing models that exhibit diminishing returns, or economic variables that follow non-linear patterns.
- **Engineering and Physics**: In fields like engineering and physics, polynomial regression is frequently used to model complex systems, like the trajectory of objects or chemical reactions.

[12]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

[13]:
```
# Set parameters
n_samples = 300
noise_level = 1000

# Generate x values
X = np.linspace(-4, 4, n_samples)

# Generate y values with polynomial relationship and noise
y = 7*X**5 - 4*X**3 - 2*X**2 + X + np.random.normal(0, noise_level, n_samples)

# Create a DataFrame
```

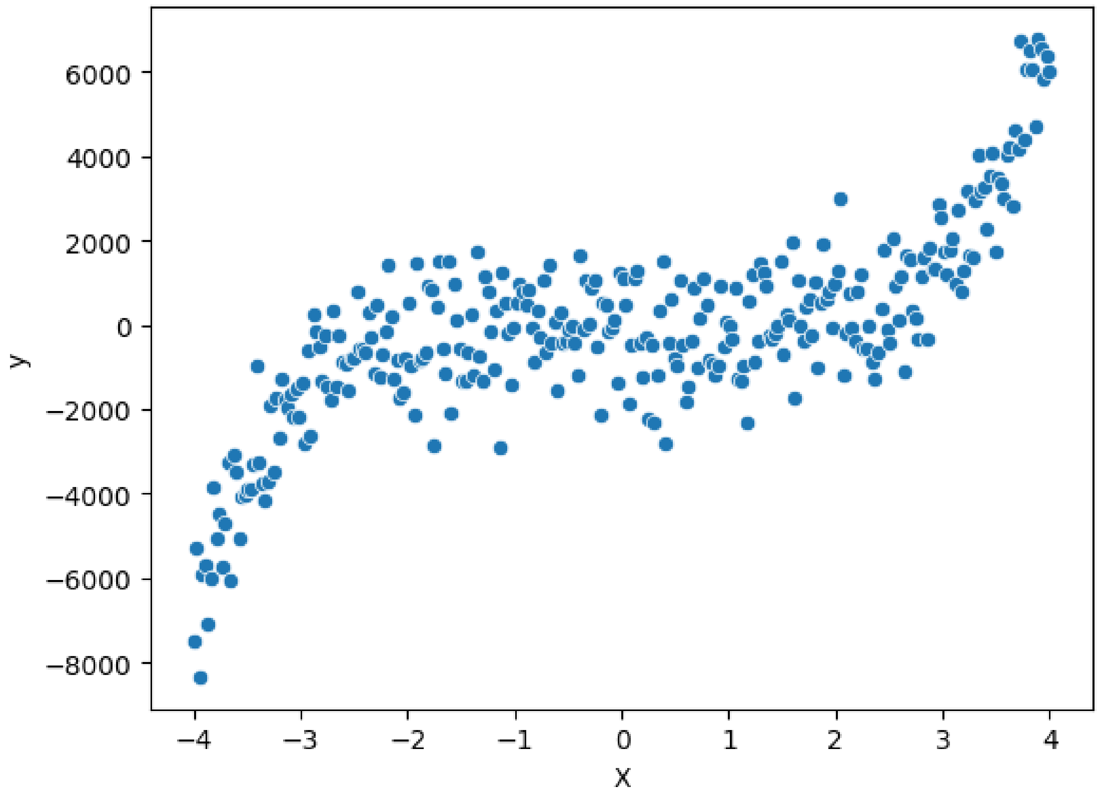
```
df = pd.DataFrame({'X': X, 'y': y})

print(df.head())
```

```
          X            y
0 -4.000000 -7477.113057
1 -3.973244 -5284.777623
2 -3.946488 -8345.460038
3 -3.919732 -5908.101560
4 -3.892977 -5705.565371
```

[14]:
```
sns.scatterplot(data=df, x='X', y='y')
```

[14]:
```
<Axes: xlabel='X', ylabel='y'>
```



[15]:
```
X = df.iloc[:, :-1].values
y = df.iloc[:,-1].values
```

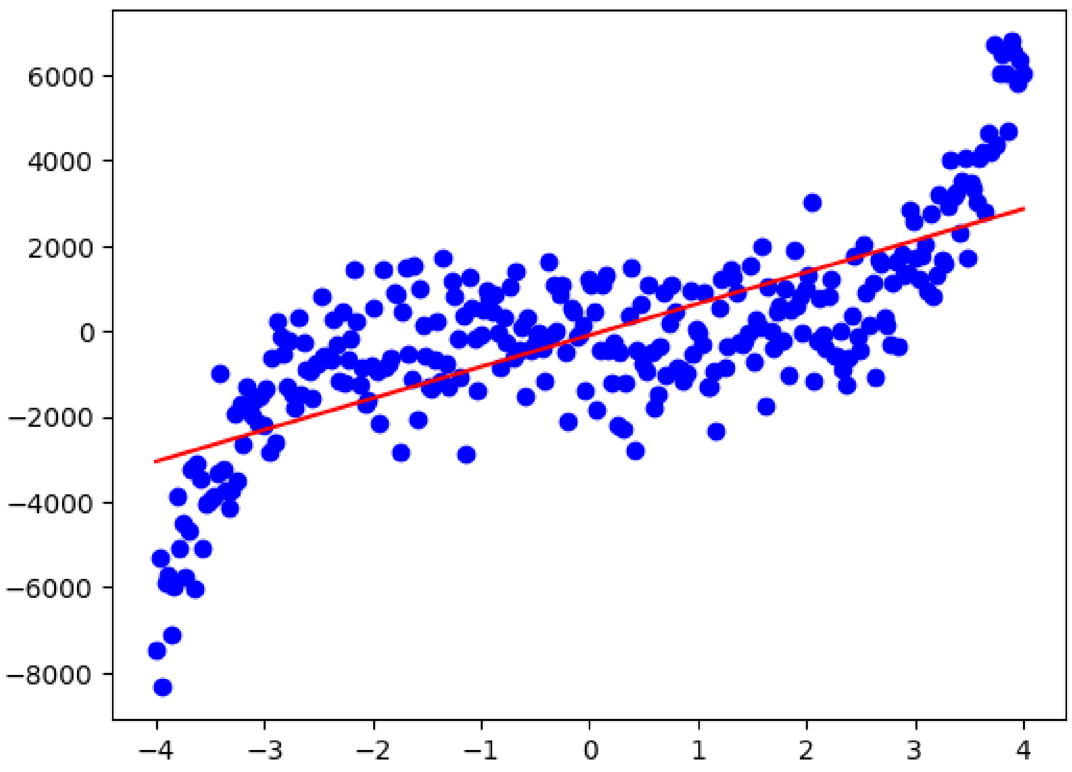
[16]:
```
linear_reg = LinearRegression()
linear_reg.fit(X, y)
y_pred = linear_reg.predict(X)
```

[17]:
```
print("Linear Regression Metrics:")
print("Mean Squared Error:", mean_squared_error(y, y_pred))
print("R-squared:", r2_score(y, y_pred))
```

```
Linear Regression Metrics:
Mean Squared Error: 2433704.9672165522
R-squared: 0.546195773748521
```

[18]:
```
plt.scatter(X, y, color='blue')
plt.plot(X, y_pred, color='red')
```

[18]:
```
[<matplotlib.lines.Line2D at 0x7accf4692c80>]
```



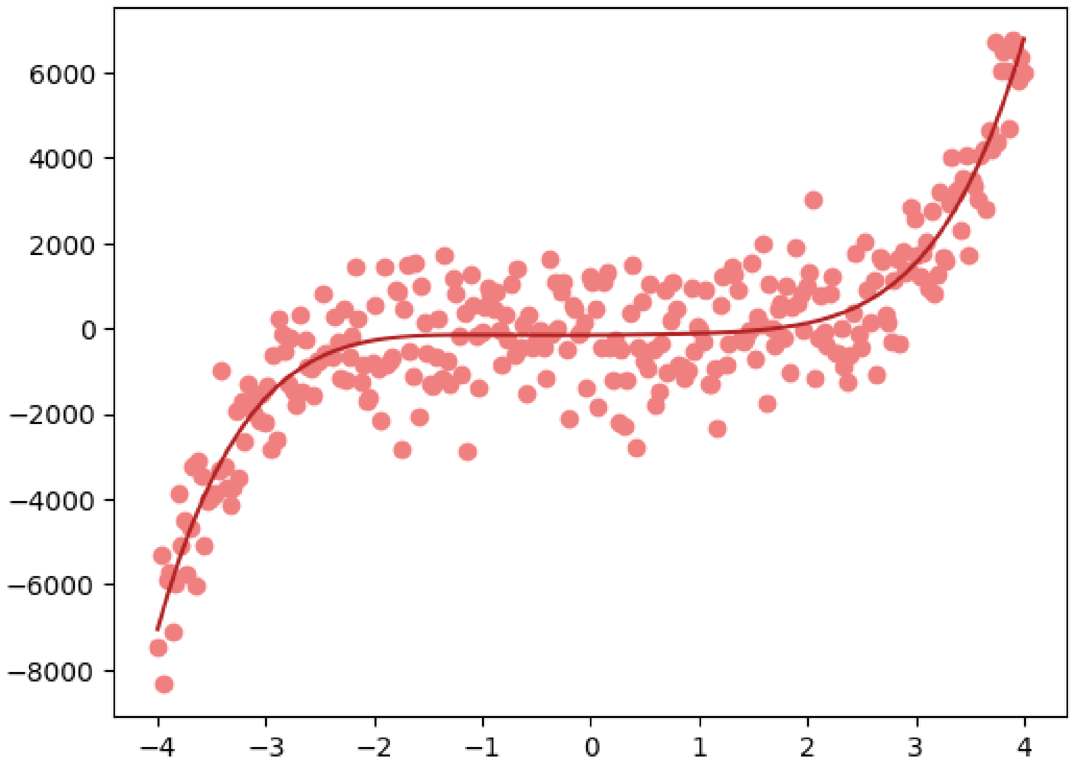
[19]:
```
poly_regression = PolynomialFeatures(degree=5)
X_poly = poly_regression.fit_transform(X)
poly_regression.fit(X_poly, y)
lin_reg2 = LinearRegression()
lin_reg2.fit(X_poly, y)
y_pred_poly = lin_reg2.predict(X_poly)
```

[20]:
```
print("Polynoimal Regression Metrics:")
print("Mean Squared Error:", mean_squared_error(y, y_pred_poly))
print("R-squared:", r2_score(y, y_pred_poly))
```

```
Polynoimal Regression Metrics:
Mean Squared Error: 961086.0225652136
R-squared: 0.820789740454801
```

[21]:
```
plt.scatter(X, y, color = 'lightcoral')
plt.plot(X, y_pred_poly, color = 'firebrick')
```

[21]:
```
[<matplotlib.lines.Line2D at 0x7accf2060190>]
```



0.0.3 Conclusion

Polynomial regression is a powerful extension of linear regression, allowing for the modeling of non-linear relationships between variables. By adding powers of the input features, it can fit more complex patterns, but care must be taken to avoid overfitting by choosing an appropriate polynomial degree. It strikes a balance between simplicity and flexibility, making it useful in a wide range of real-world applications.