

How can we perform polynomial regression using Scikit-Learn?

Authored by
stats writer

June 25, 2024

RECOMMENDED CITATION

stats writer (2024). *How can we perform polynomial regression using Scikit-Learn?*. PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=152429>

Scikit-Learn is a popular machine learning library that provides a user-friendly interface for performing various regression tasks, including polynomial regression. This technique is used to model non-linear relationships between a dependent variable and one or more independent variables. To perform polynomial regression using Scikit-Learn, the first step is to import the necessary modules and load the dataset. Then, the data is split into training and testing sets. Next, a polynomial features object is created, which transforms the original features into their higher-order polynomial terms. The transformed data is then fed into a linear regression model, which can be fitted to the training data. Finally, the model can be evaluated on the testing data and used to make predictions on new data. Scikit-Learn offers a range of tools and functions to streamline the process of performing polynomial regression, making it a powerful and efficient tool for data analysis.

Perform Polynomial Regression Using Scikit-Learn

Polynomial regression is a technique we can use when the relationship between a predictor variable and a response variable is nonlinear.

This type of regression takes the form:

$$Y = \beta_0 + \beta_1X + \beta_2X^2 + \dots + \beta_hX^h + \varepsilon$$

where h is the "degree" of the polynomial.

The following step-by-step example shows how to perform polynomial regression in Python using sklearn.

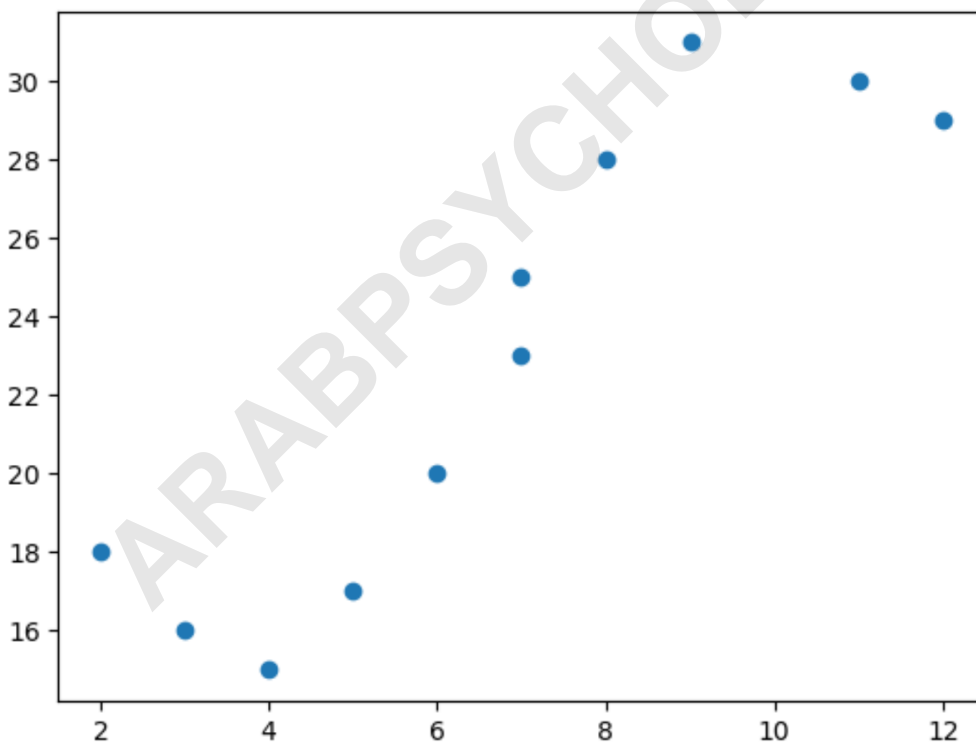
Step 1: Create the Data

First, let's create two NumPy arrays to hold the values for a predictor and response variable:

```
import matplotlib.pyplot as plt
import numpy as np

#define predictor and response variables
x = np.array()
y = np.array()

#create scatterplot to visualize relationship between x
and y
plt.scatter(x, y)
```



From the scatterplot we can see that the relationship between x and y is not linear.

Thus, it's a good idea to fit a polynomial regression model to the data to capture the non-linear relationship between the two variables.

Step 2: Fit the Polynomial Regression Model

The following code shows how to use functions from sklearn to fit a polynomial regression model with a degree of 3 to this dataset:

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression

#specify degree of 3 for polynomial regression model
#include bias=False means don't force y-intercept to
equal zero
poly      = PolynomialFeatures(degree=3,
include_bias=False)

#reshape data to work properly with sklearn
poly_features = poly.fit_transform(x.reshape(-1, 1))

#fit polynomial regression model
poly_reg_model = LinearRegression()
poly_reg_model.fit(poly_features, y)
```

```
#display model coefficients  
print(poly_reg_model.intercept_, poly_reg_model.coef_)  
  
33.62640037532282
```

Using the model coefficients displayed on the last line, we can write the fitted polynomial regression equation as:

$$y = -0.109x^3 + 2.256x^2 - 11.839x + 33.626$$

This equation can be used to find the expected value for the response variable based on a given value for the predicted variable.

$$y = -0.109(4)^3 + 2.256(4)^2 - 11.839(4) + 33.626 = 15.39$$

Note: To fit a polynomial regression model with a different degree, simply change the value for the degree argument within the `PolynomialFeatures()` function.

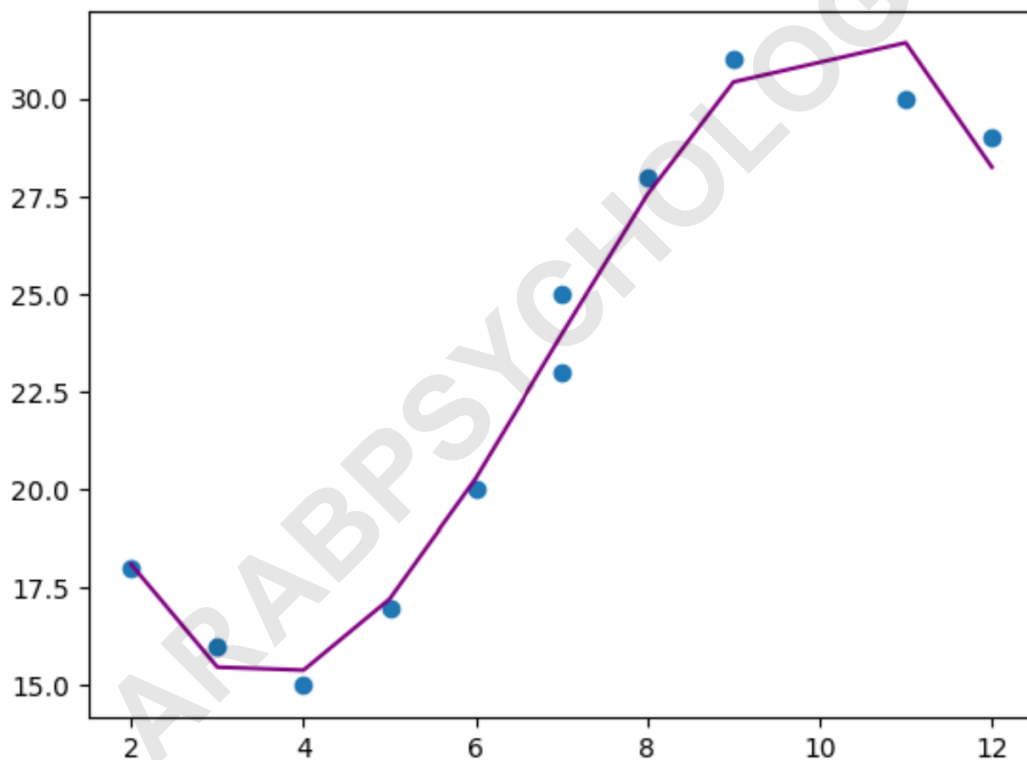
Step 3: Visualize the Polynomial Regression Model

Lastly, we can create a simple plot to visualize the fitted polynomial regression model over the original data points:

```
#use model to make predictions on response variable  
y_predicted = poly_reg_model.predict(poly_features)
```

```
#create scatterplot of x vs. y  
plt.scatter(x, y)
```

```
#add line to show fitted polynomial regression model  
plt.plot(x, y_predicted, color='purple')
```



From the plot we can see that the polynomial regression model seems to fit the data well without .

Note: You can find the complete documentation for the

sklearn PolynomialFeatures() function .

The following tutorials explain how to perform other common tasks using sklearn:

ARABPSYCHOLOGY.COM