How can I perform curve fitting in Python and what are some examples of its use?

Authored by

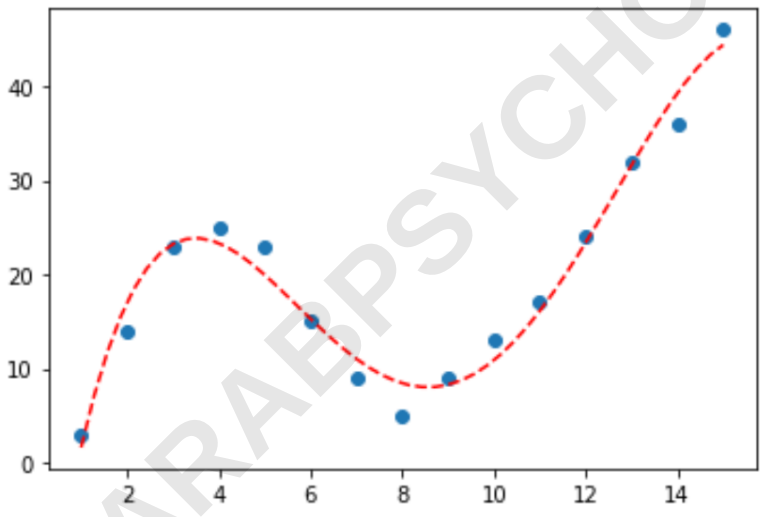
stats writer

April 28, 2024

Curve fitting in Python is a method used to find the mathematical relationship between a set of data points by fitting a curve that best represents the data. This process is commonly used in data analysis and visualization to understand the trend and patterns in the data. In Python, the curve fitting process can be performed using the library "scipy.optimize.curve_fit" which uses the least-squares method to find the best-fit parameters for the curve. Some examples of its use include predicting future values based on past data, analyzing the growth rate of a population, and identifying the relationship between variables in a scientific experiment. Curve fitting in Python is a powerful tool that can help in making data-driven decisions and understanding complex relationships between variables.

Curve Fitting in Python (With Examples)

Often you may want to fit a curve to some dataset in Python.



The following step-by-step example explains how to fit curves to data in Python using the function and how to determine which curve fits the data best.

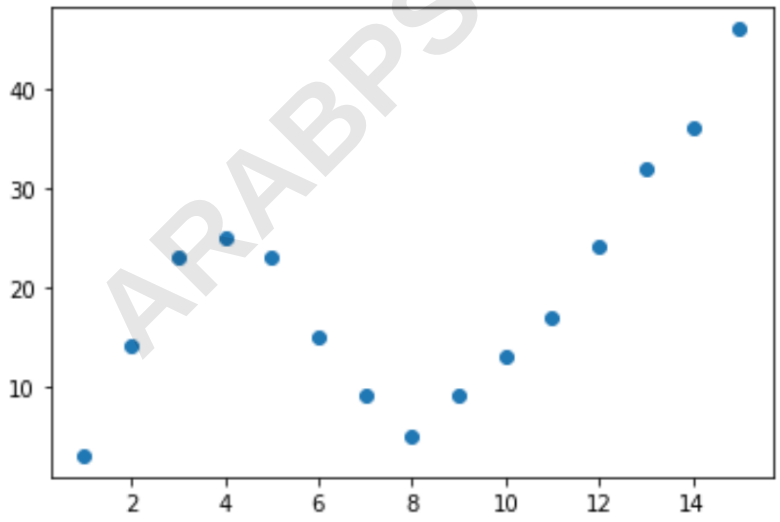
Step 1: Create & Visualize Data

First, let's create a fake dataset and then create a scatterplot to visualize the data:

```
import pandas as pd
import matplotlib.pyplotas plt

#create DataFrame
df = pd.DataFrame({'x': ,
'y': })

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



Step 2: Fit Several Curves

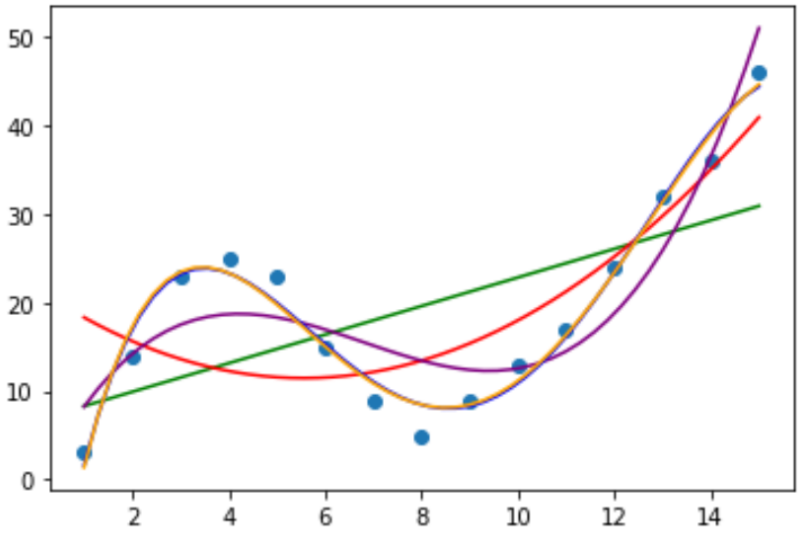
Next, let's fit several polynomial regression models to the data and visualize the curve of each model in the same plot:

```
import numpy as np

#fit polynomial models up to degree 5
model1 = np.poly1d(np.polyfit(df.x, df.y, 1))
model2 = np.poly1d(np.polyfit(df.x, df.y, 2))
model3 = np.poly1d(np.polyfit(df.x, df.y, 3))
model4 = np.poly1d(np.polyfit(df.x, df.y, 4))
model5 = np.poly1d(np.polyfit(df.x, df.y, 5))

#create scatterplot
polyline = np.linspace(1, 15, 50)
plt.scatter(df.x, df.y)

#add fitted polynomial lines to scatterplot
plt.plot(polyline, model1(polyline), color='green')
plt.plot(polyline, model2(polyline), color='red')
plt.plot(polyline, model3(polyline), color='purple')
plt.plot(polyline, model4(polyline), color='blue')
plt.plot(polyline, model5(polyline), color='orange')
plt.show()
```



To determine which curve best fits the data, we can look at the of each model.

This value tells us the percentage of the variation in the response variable that can be explained by the predictor variable(s) in the model, adjusted for the number of predictor variables.

```
#define function to calculate adjusted r-squared
def adjR(x, y, degree):
results = {}
coeffs = np.polyfit(x, y, degree)
p = np.poly1d(coeffs)
yhat = p(x)
ybar = np.sum(y)/len(y)
```

```
ssreg = np.sum((yhat-ybar)**2)
sstot = np.sum((y - ybar)**2)
results = 1- (((1-(ssreg/sstot))*(len(y)-1))/(len(y)-degree-1))

return results

#calculated adjusted R-squared of each model
adjR(df.x, df.y, 1)
adjR(df.x, df.y, 2)
adjR(df.x, df.y, 3)
adjR(df.x, df.y, 4)
adjR(df.x, df.y, 5)

{'r_squared': 0.3144819}
{'r_squared': 0.5186706}
{'r_squared': 0.7842864}
{'r_squared': 0.9590276}
{'r_squared': 0.9549709}
```

From the output we can see that the model with the highest adjusted R-squared is the fourth-degree polynomial, which has an adjusted R-squared of 0.959.

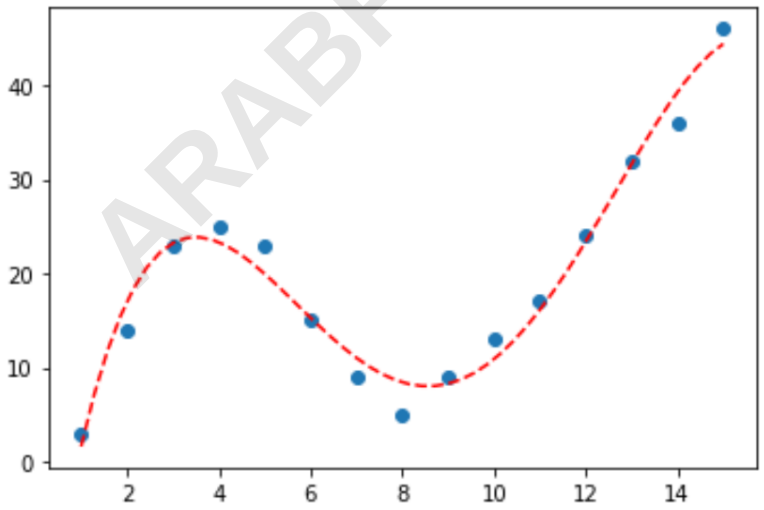
Step 3: Visualize the Final Curve

Lastly, we can create a scatterplot with the curve of the fourth-degree polynomial model:

```
#fit fourth-degree polynomial
model4 = np.poly1d(np.polyfit(df.x, df.y, 4))

#define scatterplot
polyline = np.linspace(1, 15, 50)
plt.scatter(df.x, df.y)

#add fitted polynomial curve to scatterplot
plt.plot(polyline, model4(polyline), '--', color='red')
plt.show()
```



We can also get the equation for this line using the

print() function:

print(model4)

4 3 2
-0.01924 x + 0.7081 x - 8.365 x + 35.82 x - 26.52

The equation of the curve is as follows:

y = -0.01924x4 + 0.7081x3 - 8.365x2 + 35.82x - 26.52

We can use this equation to predict the value of the based on the predictor variables in the model. For example if *x* = 4 then we would predict that *y* = 23.32:

y = -0.0192(4)4 + 0.7081(4)3 - 8.365(4)2 + 35.82(4) - 26.52 = 23.32