3_Regresion lineal Polinomial

July 6, 2025

Creado por:

Isabel Maniega

1 Regresión Lineal Polinomial

[1]:
```
# pip install scikit-learn
```

[2]:
```
import numpy as np
from sklearn import datasets, linear_model
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.metrics import mean_squared_error, r2_score
```

[3]:
```
dataset = datasets.load_diabetes()
print(dataset.DESCR)
# Crear un DataFrame con los datos
data = pd.DataFrame(dataset.data, columns=dataset.feature_names)
data['level'] = dataset.target
```

```
.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

:Number of Instances: 442

:Number of Attributes: First 10 columns are numeric predictive values

:Target: Column 11 is a quantitative measure of disease progression one year
after baseline
```

```
:Attribute Information:
    - age     age in years
    - sex
    - bmi     body mass index
    - bp      average blood pressure
    - s1      tc, total serum cholesterol
    - s2      ldl, low-density lipoproteins
    - s3      hdl, high-density lipoproteins
    - s4      tch, total cholesterol / HDL
    - s5      ltg, possibly log of serum triglycerides level
    - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by
the standard deviation times the square root of `n_samples` (i.e. the sum of
squares of each column totals 1).

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least
Angle Regression," Annals of Statistics (with discussion), 407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```

[4]:
```
data
```

[4]:
```
          age       sex       bmi        bp        s1        s2        s3  \
0    0.038076  0.050680  0.061696  0.021872 -0.044223 -0.034821 -0.043401
1   -0.001882 -0.044642 -0.051474 -0.026328 -0.008449 -0.019163  0.074412
2    0.085299  0.050680  0.044451 -0.005670 -0.045599 -0.034194 -0.032356
3   -0.089063 -0.044642 -0.011595 -0.036656  0.012191  0.024991 -0.036038
4    0.005383 -0.044642 -0.036385  0.021872  0.003935  0.015596  0.008142
..        …         …         …         …         …         …         …
437  0.041708  0.050680  0.019662  0.059744 -0.005697 -0.002566 -0.028674
438 -0.005515  0.050680 -0.015906 -0.067642  0.049341  0.079165 -0.028674
439  0.041708  0.050680 -0.015906  0.017293 -0.037344 -0.013840 -0.024993
440 -0.045472 -0.044642  0.039062  0.001215  0.016318  0.015283 -0.028674
441 -0.045472 -0.044642 -0.073030 -0.081413  0.083740  0.027809  0.173816

           s4        s5        s6  level
0   -0.002592  0.019907 -0.017646  151.0
1   -0.039493 -0.068332 -0.092204   75.0
2   -0.002592  0.002861 -0.025930  141.0
3    0.034309  0.022688 -0.009362  206.0
4   -0.002592 -0.031988 -0.046641  135.0
```

```
..        ...       ...       ...     ...
437 -0.002592  0.031193  0.007207  178.0
438  0.034309 -0.018114  0.044485  104.0
439 -0.011080 -0.046883  0.015491  132.0
440  0.026560  0.044529 -0.025930  220.0
441 -0.039493 -0.004222  0.003064   57.0

[442 rows x 11 columns]
```

[5]:
```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
```

[6]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(diabetes_X, diabetes_y,␣
 ↪test_size=0.2)
```

[7]:
```
## Cargar el modelo:
```

[8]:
```
from sklearn.preprocessing import PolynomialFeatures
```

[9]:
```
# se define el grado de polinomio

poli_reg = PolynomialFeatures(degree= 2)
```

[10]:
```
# se transforman las características existentes en características de mayor␣
 ↪grado

X_train_poli = poli_reg.fit_transform(X_train)
X_test_poli = poli_reg.fit_transform(X_test)
```

[11]:
```
pr = linear_model.LinearRegression()
```

[12]:
```
pr.fit(X_train_poli, y_train)
```

[12]:
```
LinearRegression()
```

[13]:
```
y_pred = pr.predict(X_test_poli)
y_pred
```

[13]:
```
array([161.46059595,  21.19936311, 152.74422811, 105.16707649,
       139.93819504, 175.45239976, 167.73925153,  61.19310734,
        99.7067457 , 329.96540926, 208.73098939, 145.85781872,
       101.84935228,  91.47737904, 126.49933018, 109.02292978,
       162.55596257,  83.88771044, 161.46222907, 114.17831306,
        70.31573091, 288.1296595 , 191.64076517,  95.8587377 ,
       135.9189689 , 296.09446472, 280.85069079,  48.05196724,
       160.76955882,  79.56568853, 234.6009112 , 292.88958301,
```

```
        153.28144867, 122.91009697, 107.96772411, 141.5990241 ,
        141.3580112 , 155.33588991,  68.98602788, 188.18457916,
        179.40786372, 249.12219555, 183.86621881, 173.92400854,
        199.98428069, 138.95042811, 109.60561323, 242.21953144,
        200.99016698, 203.02891751, 112.18385329,  75.58419171,
        112.10517323, 142.43070884,  36.22492391,  30.65043378,
         95.34806061, 267.90642687, 122.25895593, 248.87283901,
        216.0861312 , 276.84057744, 181.970074  , 268.07601353,
         89.17969346,  64.21123689,  70.59124122, 288.50299697,
        258.81244519,  98.59898665, 149.70720783, 124.50334556,
        149.31339025, 104.26934431, 121.26443581, 119.72020029,
        252.46139606, 214.12324645, 167.40854488, 210.23100741,
        106.41389898, 239.55226936, 164.90659358, 113.93801248,
        122.86451019, 120.15538876, 120.09450332, 116.41404309,
        155.15514822])
```

[14]:
```
y_test
```

[14]:
```
array([185.,  45., 190.,  67., 104., 206., 101.,  77.,  59., 310.,  52.,
       210., 143.,  72.,  68., 252.,  95., 114., 129., 131.,  70., 341.,
       122.,  42.,  63., 263., 243.,  39., 122.,  84., 150., 270., 174.,
       113.,  90., 150., 102., 137., 135., 129., 110., 281.,  78.,  85.,
       296., 103.,  55., 346., 139., 265., 102.,  43.,  97., 216.,  72.,
        51., 178., 242.,  44., 303., 166., 220., 311., 275.,  49.,  72.,
        96., 245., 295.,  31., 214.,  86., 113., 230.,  89., 160., 163.,
       197.,  52., 275., 178., 261., 242., 152., 202.,  97., 191., 139.,
       197.])
```

[15]:
```
print("Valor de pendiente o coeficiente 'a':")
print(pr.coef_)
```

```
Valor de pendiente o coeficiente 'a':
[-2.25705732e-08  1.83812644e+01 -2.55753841e+02  5.14060728e+02
  3.51324805e+02  1.16410228e+04 -1.03847845e+04 -4.59614861e+03
  1.12033159e+02 -3.21809840e+03  9.44338067e+01  1.40773877e+03
  4.35129792e+03 -3.83906232e+02  8.58064919e+02 -4.72340834e+03
 -3.56810997e+02  5.11979355e+03  4.04764811e+03  2.11124494e+03
  6.58114600e+02 -1.54436503e+00  1.90933245e+03  2.76396241e+03
  1.38696239e+04 -1.02197133e+04 -4.24139963e+03 -3.58801611e+03
 -4.38982696e+03  1.20438559e+02  1.06046933e+02  4.43098578e+03
 -4.73363803e+03  3.55556733e+03  2.94488146e+03 -1.67412646e+03
  2.18643118e+02  1.61746191e+03 -6.89655561e+02  1.03396466e+04
 -7.72807547e+03 -3.91040351e+03 -5.74567960e+02 -3.24471166e+03
 -3.65461478e+03  8.61686874e+04 -1.23398131e+05 -9.02902516e+04
 -4.00266993e+04 -1.32037848e+05  2.75072939e+02  4.47022081e+04
  6.57548790e+04  2.77565588e+04  1.06648672e+05 -2.39192495e+03
  2.23618495e+04  1.48820958e+04  6.04627355e+04  3.16280162e+03
  2.17860628e+03  1.61277137e+04  4.94700421e+03  1.51823911e+04
```

```
 -7.71537942e+00  1.45514859e+03]
```

[16]:
```
print("Precisión del modelo: ")
print(pr.score(X_train_poli, y_train))
```

```
Precisión del modelo:
0.6019131040426905
```

Creado por:

Isabel Maniega