Cómo hacer una regresión linear en Python

Creador: David Rubio G.

Entrada: https://soceame.wordpress.com/2023/08/02/como-hacer-una-regresion-linear-en-python/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

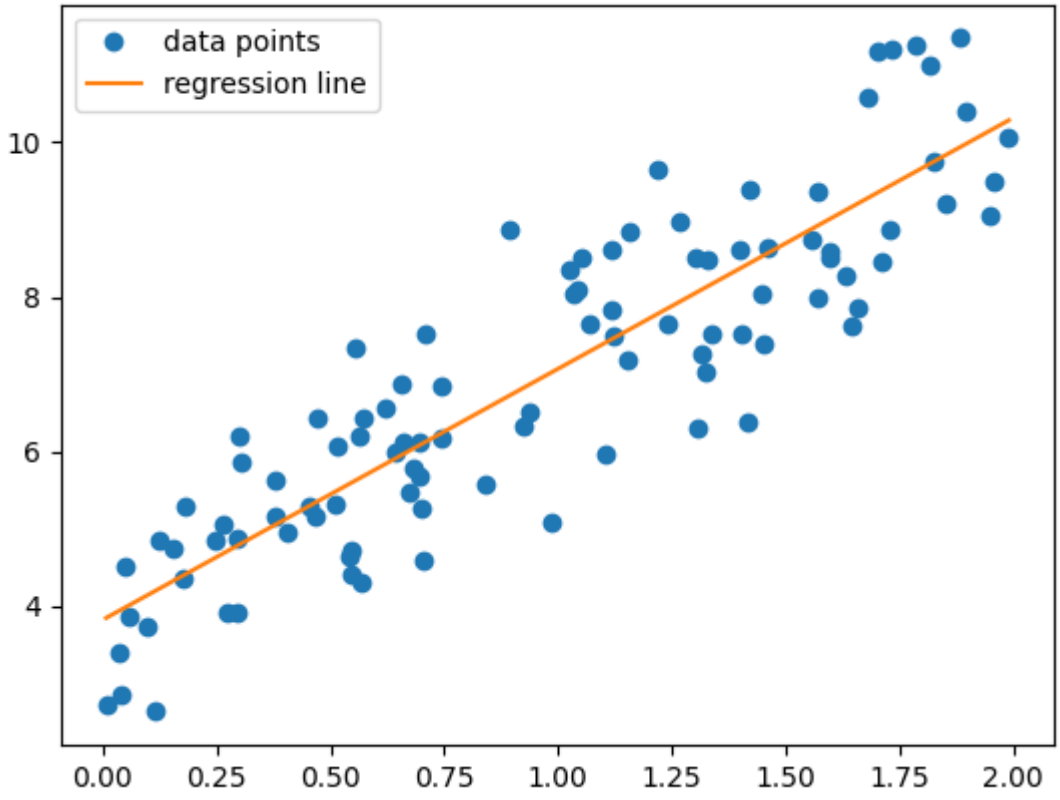
Fecha última modificación: 22/02/2025

Si has llegado aquí, es porque tienes o quieres hacer una regresión linear en Python y no te ha quedado claro cómo hacerla.

Bien, pues empecemos.

Mini-explicación.

Una regresión lineal es la forma de obtener una recta que mantenga una forma de aproximación igual a todos los puntos.



Desarrollo

Si has intentado hacer o has pensado como hacer una regresión lineal, te habrás dado cuenta que para poder hacerla requieres de realizar varios bucles «infinitos»(estos bucles necesitan una parada en base a la precisión deseada).

Bien, pues existe otra forma, basada en sistemas matriciales.

Regresión lineal = $(X'\cdot X)^{-1}\cdot X'\cdot y$

Seguro, que no te has enterado de cómo se aplica, pues se hace con este código.

```
import numpy as np

X_b = np.c_[np.ones((len(x),1)), x]
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

```
X_new = np.array([[0],[2]])
X_new_b = np.c_[np.ones((2,1)), X_new]
y_predict = X_new_b.dot(theta_best)

x1, y1 = X_new[0], y_predict[0]
x2, y2 = X_new[1], y_predict[1]

m = (y2 - y1)/(x2 - x1)
n = ((x2 - x1)*y1 - (y2 - y1)*x1)/(x2 - x1)
```

La primera parte arranca de generar una matriz con todas los valores de ‘x’, y después se realiza la regresión lineal. Después, se obtienen los valores de la diagonal que representan los valores predichos. Al final, el código anterior, entrega una predicción de la que se puede obtener la pendiente de la recta «m» y el origen de la recta «n».

Para facilitar el uso, se provee de una función que permite el cálculo de la pendiente y el punto inicial de la regresión lineal de un conjunto de puntos.

```
def LinearRegression(x, y):
    """
    This function calculates the linear regression of input data

    Inputs:
        - X: list of x values
        - y: list of y values
    Return:
        - m: slope of the data
        - n: cross zero
    """
    import numpy as np

    X_b = np.c_[np.ones((len(x),1)), x]
    theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)

    X_new = np.array([[0],[2]])
    X_new_b = np.c_[np.ones((2,1)), X_new]
    y_predict = X_new_b.dot(theta_best)

    x1, y1 = X_new[0], y_predict[0]
    x2, y2 = X_new[1], y_predict[1]

    m = (y2 - y1)/(x2 - x1)
    n = ((x2 - x1)*y1 - (y2 - y1)*x1)/(x2 - x1)

    return m, n
```