# 1_Regresion Lineal Simple

July 6, 2025

*Creado por:*

*Isabel Maniega*

## 1 Regresión lineal Simple

[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
data
```

```
.. _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)
```

[3]:

| | 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]
```

[4]:
```
# Load the diabetes dataset
diabetes_X, diabetes_y = datasets.load_diabetes(return_X_y=True)
```

[5]:
```
# Use only one feature (bmi)
diabetes_X = diabetes_X[:, np.newaxis, 2]

from sklearn.model_selection import train_test_split
# Separo los datos de "train" entrenamiento y "test" prueba para probar los␣
 ↪algoritmos

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

[6]:
```
# Create linear regression object
regr = linear_model.LinearRegression()
```

[7]:
```
# Train the model using the training sets
regr.fit(X_train, y_train)

# Make predictions using the testing set
diabetes_y_pred = regr.predict(X_test)
```

[8]:
```
# The coefficients
print("Coefficients: \n", regr.coef_)
# The mean squared error
print("Mean squared error: %.2f" % mean_squared_error(y_test, diabetes_y_pred))
# The coefficient of determination: 1 is perfect prediction
print("Coefficient of determination: %.2f" % r2_score(y_test, diabetes_y_pred))
print('Valor de la intersección o coeficiente "b":')
print(regr.intercept_)
print()
print('La ecuación del modelo es igual a:')
print('y = ', regr.coef_, 'x ', regr.intercept_)
```

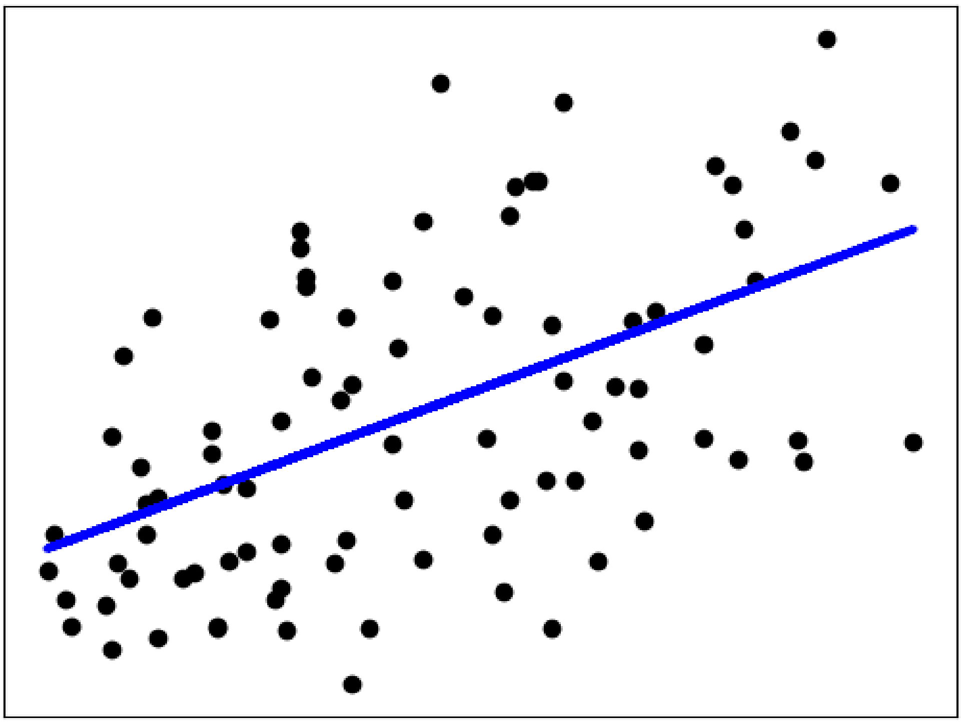
```
Coefficients:
 [953.08807169]
Mean squared error: 3997.57
Coefficient of determination: 0.27
Valor de la intersección o coeficiente "b":
152.82921485401943
```

```
La ecuación del modelo es igual a:
y =  [953.08807169] x  152.82921485401943
```

[9]:
```
# Plot outputs
plt.scatter(X_test, y_test, color="black")
plt.plot(X_test, diabetes_y_pred, color="blue", linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```



[10]:
```
print("Precisión del modelo:")
print(regr.score(X_train, y_train))
```

```
Precisión del modelo:
0.36022625235922356
```

*Creado por:*

*Isabel Maniega*