Creado por:

Isabel Maniega

Regresión Líneal Polinomial

In [1]:
```python
import warnings
warnings.filterwarnings("ignore")
```

https://scikit-learn.org/stable/install.html

In [2]:
```python
# pip install scikit-learn
```

In [3]:
```python
import numpy as np
from sklearn import datasets, linear_model
import matplotlib.pyplot as plt
import pandas as pd
```

In [4]:
```python
boston = datasets.load_boston()
```

In [5]:
```python
boston.data
```

Out[5]:
```
array([[6.3200e-03, 1.8000e+01, 2.3100e+00, ..., 1.5300e+01, 3.9690e+02,
        4.9800e+00],
       [2.7310e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9690e+02,
        9.1400e+00],
       [2.7290e-02, 0.0000e+00, 7.0700e+00, ..., 1.7800e+01, 3.9283e+02,
        4.0300e+00],
       ...,
       [6.0760e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        5.6400e+00],
       [1.0959e-01, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9345e+02,
        6.4800e+00],
       [4.7410e-02, 0.0000e+00, 1.1930e+01, ..., 2.1000e+01, 3.9690e+02,
        7.8800e+00]])
```

In [6]:
```python
print('Nombre de columnas:')
print(boston.feature_names)
```

```
Nombre de columnas:
['CRIM' 'ZN' 'INDUS' 'CHAS' 'NOX' 'RM' 'AGE' 'DIS' 'RAD' 'TAX' 'PTRATIO'
 'B' 'LSTAT']
```

In [7]:
```python
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df
```

Out[7]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0.0	0.573	6.593	69.1	2.4786	1.0	273.0	21.0	391.99	9.67
502	0.04527	0.0	11.93	0.0	0.573	6.120	76.7	2.2875	1.0	273.0	21.0	396.90	9.08
503	0.06076	0.0	11.93	0.0	0.573	6.976	91.0	2.1675	1.0	273.0	21.0	396.90	5.64
504	0.10959	0.0	11.93	0.0	0.573	6.794	89.3	2.3889	1.0	273.0	21.0	393.45	6.48
505	0.04741	0.0	11.93	0.0	0.573	6.030	80.8	2.5050	1.0	273.0	21.0	396.90	7.88

506 rows × 13 columns

In [8]:
```python
print("Informacion en el dataset:")
print(boston.keys())
```

```
Informacion en el dataset:
dict_keys(['data', 'target', 'feature_names', 'DESCR', 'filename', 'data_module'])
```

In [9]:
```python
print("Características del dataset:")
print(boston.DESCR)
```

```
Características del dataset:
.. _boston_dataset:

Boston house prices dataset
---------------------------

**Data Set Characteristics:**

    :Number of Instances: 506

    :Number of Attributes: 13 numeric/categorical predictive. Median Value (attribute 14) is usually the targe
t.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,000 sq.ft.
        - INDUS    proportion of non-retail business acres per town
        - CHAS     Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
        - NOX      nitric oxides concentration (parts per 10 million)
        - RM       average number of rooms per dwelling
        - AGE      proportion of owner-occupied units built prior to 1940
        - DIS      weighted distances to five Boston employment centres
        - RAD      index of accessibility to radial highways
        - TAX      full-value property-tax rate per $10,000
        - PTRATIO  pupil-teacher ratio by town
        - B        1000(Bk - 0.63)^2 where Bk is the proportion of black people by town
        - LSTAT    % lower status of the population
        - MEDV     Median value of owner-occupied homes in $1000's

    :Missing Attribute Values: None

    :Creator: Harrison, D. and Rubinfeld, D.L.

This is a copy of UCI ML housing dataset.
https://archive.ics.uci.edu/ml/machine-learning-databases/housing/


This dataset was taken from the StatLib library which is maintained at Carnegie Mellon University.

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic
prices and the demand for clean air', J. Environ. Economics & Management,
vol.5, 81-102, 1978.   Used in Belsley, Kuh & Welsch, 'Regression diagnostics
...', Wiley, 1980.   N.B. Various transformations are used in the table on
pages 244-261 of the latter.

The Boston house-price data has been used in many machine learning papers that address regression
problems.

.. topic:: References

   - Belsley, Kuh & Welsch, 'Regression diagnostics: Identifying Influential Data and Sources of Collinearity',
Wiley, 1980. 244-261.
   - Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth Internat
ional Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.
```

In [10]:
```python
print("Cantidad de datos:")
print(boston.data.shape)
```

```
Cantidad de datos:
(506, 13)
```

In [11]:
```python
# Seleccionamos como valor de la X la columna numero 6 (RM)

X = boston.data[:,np.newaxis, 5]
```

In [12]:
```python
y = boston.target
```

In [13]:
```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [15]:
```python
## Cargar el modelo:
```

In [14]:
```python
from sklearn.preprocessing import PolynomialFeatures
```

In [16]:
```python
# se define el grado de polinomio

poli_reg = PolynomialFeatures(degree= 2)
```

In [18]:
```python
# 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)
```

In [19]:
```python
pr = linear_model.LinearRegression()
```

In [20]:
```python
pr.fit(X_train_poli, y_train)
```

Out[20]:
```
▾ LinearRegression
LinearRegression()
```

In [21]:
```python
y_pred = pr.predict(X_test_poli)
y_pred
```

Out[21]:
```
array([22.88664164, 27.31542458, 33.32875042, 16.84202821, 25.90514569,
       21.26908508, 23.38011269, 26.60066722, 17.5608785 , 19.59605916,
       22.95893593, 20.91304696, 23.40790531, 24.79859834, 19.72692656,
       21.98966286, 22.87762473, 22.08418569, 20.97714527, 46.99142435,
       26.01973012, 17.79870369, 19.22603479, 23.26933964, 19.41667245,
       34.04661294, 17.89581211, 18.08093738, 23.26013726, 20.12056693,
       16.5730405 , 14.45903673, 29.82492985, 20.31055189, 41.57128975,
       55.46432656, 17.60198787, 28.61827943, 24.89793056, 49.41839045,
       24.25049859, 16.05252743, 19.3882267 , 15.58352185, 30.74531105,
       19.17016779, 31.75432239, 18.52180815, 16.62894595, 16.06169847,
       21.10618974, 21.8361417 , 22.57367712, 41.47719311, 15.75122195,
       19.61052933, 22.77873012, 22.71607539, 19.31036521, 16.12174079,
       18.04980717, 39.88665639, 17.58434307, 23.50083442, 30.50063254,
       17.83498687, 18.39831138, 21.96397664, 20.4800217 , 39.88665639,
       37.96070505, 23.52879926, 19.531162  , 23.04060483, 15.02101762,
       19.20505155, 40.05428507, 24.28918931, 20.84127418, 18.0996685 ,
       14.24028785, 26.92317233, 47.1116055 , 19.70501597, 17.08444382,
       20.64378212, 18.71337671, 23.88646062, 19.0523888 , 15.77268125,
       27.11858307, 15.47431851, 21.24453817, 24.90788807, 22.77873012,
       26.26104239, 30.84367703, 23.22337188, 22.92275346, 28.66426529,
       25.09791977, 26.94481398])
```

In [22]:
```python
y_test
```

Out[22]:
```
array([13. , 29.9, 31.5, 23.1, 30.1, 21.4, 13.1, 13.3, 13.1, 22.9, 16.1,
       14.1, 19.8, 23.9, 11.9, 22.3, 23.6, 24.4, 11. , 42.8, 30.5, 18.7,
       21.7, 11.8,  5.6, 31.7, 15.6, 19.4, 19.2, 22.2, 12.3, 50. , 28.7,
       21.9, 50. , 50. , 18.5, 23.9, 13.4, 48.8, 32.5, 23.7, 13.6, 12. ,
       37.9, 19. , 36.1, 17.3,  8.5,  7. , 21.4, 16.1, 23.1, 50. , 20. ,
       17.5, 22. , 25. , 20.3, 13.8, 19.8, 45.4, 16.2, 16.7, 32.9,  8.5,
       10.2, 23. , 13.8, 43.8, 35.2, 24.4, 19.1,  7.2, 16.3,  8.8, 50. ,
       23.8, 21.2, 20.3, 10.2,  8.4, 44.8, 16.8, 17.2, 17.8, 20.9, 14.1,
       23.1,  8.3, 30.1,  7.2, 20.5, 27.9, 18.6, 25. , 31.6, 17.1, 21.6,
       37.2, 13.9, 33.1])
```

In [23]:
```python
print("Valor de pendiente o coeficiente 'a':")
print(pr.coef_)
```

```
Valor de pendiente o coeficiente 'a':
[  0.         -19.31795108   2.20796895]
```

In [24]:
```python
print("Precisión del modelo: ")
print(pr.score(X_train_poli, y_train))
```

```
Precisión del modelo:
0.5253512895692117
```

Creado por:

Isabel Maniega