In [7]: ▶
```python
from sklearn.datasets import load_boston
```

In [13]: ▶
```python
boston = load_boston()
```

In [14]: ▶
```python
import pandas as pd
```

In [15]: ▶
```python
data = pd.DataFrame(boston.data, columns = boston.feature_names)
```

In [16]: ▶
```python
data['MEDV'] = pd.DataFrame(boston.target)
```

In [17]: ▶
```python
pd.DataFrame(data.corr().round(2))
```

Out[17]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B
CRIM	1.00	-0.20	0.41	-0.06	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29	-0.39
ZN	-0.20	1.00	-0.53	-0.04	-0.52	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18
INDUS	0.41	-0.53	1.00	0.06	0.76	-0.39	0.64	-0.71	0.60	0.72	0.38	-0.36
CHAS	-0.06	-0.04	0.06	1.00	0.09	0.09	0.09	-0.10	-0.01	-0.04	-0.12	0.05
NOX	0.42	-0.52	0.76	0.09	1.00	-0.30	0.73	-0.77	0.61	0.67	0.19	-0.38
RM	-0.22	0.31	-0.39	0.09	-0.30	1.00	-0.24	0.21	-0.21	-0.29	-0.36	0.13
AGE	0.35	-0.57	0.64	0.09	0.73	-0.24	1.00	-0.75	0.46	0.51	0.26	-0.27
DIS	-0.38	0.66	-0.71	-0.10	-0.77	0.21	-0.75	1.00	-0.49	-0.53	-0.23	0.29
RAD	0.63	-0.31	0.60	-0.01	0.61	-0.21	0.46	-0.49	1.00	0.91	0.46	-0.44
TAX	0.58	-0.31	0.72	-0.04	0.67	-0.29	0.51	-0.53	0.91	1.00	0.46	-0.44
PTRATIO	0.29	-0.39	0.38	-0.12	0.19	-0.36	0.26	-0.23	0.46	0.46	1.00	-0.18
B	-0.39	0.18	-0.36	0.05	-0.38	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1.00
LSTAT	0.46	-0.41	0.60	-0.05	0.59	-0.61	0.60	-0.50	0.49	0.54	0.37	-0.37
MEDV	-0.39	0.36	-0.48	0.18	-0.43	0.70	-0.38	0.25	-0.38	-0.47	-0.51	0.33

In [18]: ▶
```python
x=data[['RM','ZN']]
```

In [19]: ▶
```python
y=data['MEDV']
```

In [21]: ▶
```python
from sklearn.model_selection import train_test_split
```

In [22]: ▶
```python
x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(x,y,test_size=0.2
```

In [24]: ▶
```python
from sklearn.linear_model import LinearRegression
```

In [25]:   ▶| ```mlr = LinearRegression()```

In [26]:   ▶| ```mlr.fit(x_train_1, y_train_1)```

Out[26]:   ```LinearRegression()```

In [27]:   ▶| ```yTestPredict = mlr.predict(x_test_1)```

In [28]:   ▶| ```import numpy as np```

In [29]:   ▶| ```from sklearn.metrics import mean_squared_error```

In [31]:   ▶| ```np.sqrt(mean_squared_error(y_test_1, yTestPredict))```

Out[31]:   6.807807495048596

In [32]:   ▶| ```print(boston.data)```

```
[[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 [33]:   ▶| ```data.head()```

Out[33]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
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	
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	
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	
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	
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	

In [34]:   ▶| `print(boston.DESCR)`

```
.. _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 target.

    :Attribute Information (in order):
        - CRIM     per capita crime rate by town
        - ZN       proportion of residential land zoned for lots over 25,00
0 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 blacks b
y 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/ (http
s://archive.ics.uci.edu/ml/machine-learning-databases/housing/)


This dataset was taken from the StatLib library which is maintained at Carn
egie 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 diagnosti
cs
...', 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 t
hat address regression
problems.
```

.. topic:: References

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

In [35]: ► `print(boston.target)`

```
[24.   21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 15.   18.9 21.7 20.4
 18.2 19.9 23.1 17.5 20.2 18.2 13.6 19.6 15.2 14.5 15.6 13.9 16.6 14.8
 18.4 21.   12.7 14.5 13.2 13.1 13.5 18.9 20.   21.   24.7 30.8 34.9 26.6
 25.3 24.7 21.2 19.3 20.   16.6 14.4 19.4 19.7 20.5 25.   23.4 18.9 35.4
 24.7 31.6 23.3 19.6 18.7 16.   22.2 25.   33.   23.5 19.4 22.   17.4 20.9
 24.2 21.7 22.8 23.4 24.1 21.4 20.   20.8 21.2 20.3 28.   23.9 24.8 22.9
 23.9 26.6 22.5 22.2 23.6 28.7 22.6 22.   22.9 25.   20.6 28.4 21.4 38.7
 43.8 33.2 27.5 26.5 18.6 19.3 20.1 19.5 19.5 20.4 19.8 19.4 21.7 22.8
 18.8 18.7 18.5 18.3 21.2 19.2 20.4 19.3 22.   20.3 20.5 17.3 18.8 21.4
 15.7 16.2 18.   14.3 19.2 19.6 23.   18.4 15.6 18.1 17.4 17.1 13.3 17.8
 14.   14.4 13.4 15.6 11.8 13.8 15.6 14.6 17.8 15.4 21.5 19.6 15.3 19.4
 17.   15.6 13.1 41.3 24.3 23.3 27.   50.   50.   50.   22.7 25.   50.   23.8
 23.8 22.3 17.4 19.1 23.1 23.6 22.6 29.4 23.2 24.6 29.9 37.2 39.8 36.2
 37.9 32.5 26.4 29.6 50.   32.   29.8 34.9 37.   30.5 36.4 31.1 29.1 50.
 33.3 30.3 34.6 34.9 32.9 24.1 42.3 48.5 50.   22.6 24.4 22.5 24.4 20.
 21.7 19.3 22.4 28.1 23.7 25.   23.3 28.7 21.5 23.   26.7 21.7 27.5 30.1
 44.8 50.   37.6 31.6 46.7 31.5 24.3 31.7 41.7 48.3 29.   24.   25.1 31.5
 23.7 23.3 22.   20.1 22.2 23.7 17.6 18.5 24.3 20.5 24.5 26.2 24.4 24.8
 29.6 42.8 21.9 20.9 44.   50.   36.   30.1 33.8 43.1 48.8 31.   36.5 22.8
 30.7 50.   43.5 20.7 21.1 25.2 24.4 35.2 32.4 32.   33.2 33.1 29.1 35.1
 45.4 35.4 46.   50.   32.2 22.   20.1 23.2 22.3 24.8 28.5 37.3 27.9 23.9
 21.7 28.6 27.1 20.3 22.5 29.   24.8 22.   26.4 33.1 36.1 28.4 33.4 28.2
 22.8 20.3 16.1 22.1 19.4 21.6 23.8 16.2 17.8 19.8 23.1 21.   23.8 23.1
 20.4 18.5 25.   24.6 23.   22.2 19.3 22.6 19.8 17.1 19.4 22.2 20.7 21.1
 19.5 18.5 20.6 19.   18.7 32.7 16.5 23.9 31.2 17.5 17.2 23.1 24.5 26.6
 22.9 24.1 18.6 30.1 18.2 20.6 17.8 21.7 22.7 22.6 25.   19.9 20.8 16.8
 21.9 27.5 21.9 23.1 50.   50.   50.   50.   50.   13.8 13.8 15.   13.9 13.3
 13.1 10.2 10.4 10.9 11.3 12.3  8.8  7.2 10.5  7.4 10.2 11.5 15.1 23.2
  9.7 13.8 12.7 13.1 12.5  8.5  5.    6.3  5.6  7.2 12.1  8.3  8.5  5.
 11.9 27.9 17.2 27.5 15.   17.2 17.9 16.3  7.    7.2  7.5 10.4  8.8  8.4
 16.7 14.2 20.8 13.4 11.7  8.3 10.2 10.9 11.    9.5 14.5 14.1 16.1 14.3
 11.7 13.4  9.6  8.7  8.4 12.8 10.5 17.1 18.4 15.4 10.8 11.8 14.9 12.6
 14.1 13.   13.4 15.2 16.1 17.8 14.9 14.1 12.7 13.5 14.9 20.   16.4 17.7
 19.5 20.2 21.4 19.9 19.   19.1 19.1 20.1 19.9 19.6 23.2 29.8 13.8 13.3
 16.7 12.   14.6 21.4 23.   23.7 25.   21.8 20.6 21.2 19.1 20.6 15.2  7.
  8.1 13.6 20.1 21.8 24.5 23.1 19.7 18.3 21.2 17.5 16.8 22.4 20.6 23.9
 22.   11.9]
```

In [36]: ► `type(data)`

Out[36]: pandas.core.frame.DataFrame

In [37]: ► `x2 = data[['RM', 'ZN', 'LSTAT']]`

In [38]: ▶|
```python
mlr2 = LinearRegression()
```

In [39]: ▶|
```python
x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(x2,y,test_size=0.
```

In [40]: ▶|
```python
mlr2.fit(x_train_1, y_train_1)
```

Out[40]: LinearRegression()

In [41]: ▶|
```python
yTestPredict = mlr2.predict(x_test_1)
```

In [42]: ▶|
```python
np.sqrt(mean_squared_error(y_test_1, yTestPredict))
```

Out[42]: 5.2328871493133535

In [ ]: ▶|