In [6]:

```
import pandas as pd
from sklearn.datasets import load_boston
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV
```

In [7]:

```
X,y = load_boston(return_X_y=True)
```

In [8]:

```
pipe=Pipeline([("Scaling:",StandardScaler()),
          ("algo:",KNeighborsRegressor())])
```

In [9]:

```
pipe.get_params()
```

Out[9]:

```
{'memory': None,
 'steps': [('Scaling:', StandardScaler()), ('algo:', KNeighborsRegressor())],
 'verbose': False,
 'Scaling:': StandardScaler(),
 'algo:': KNeighborsRegressor(),
 'Scaling:__copy': True,
 'Scaling:__with_mean': True,
 'Scaling:__with_std': True,
 'algo:__algorithm': 'auto',
 'algo:__leaf_size': 30,
 'algo:__metric': 'minkowski',
 'algo:__metric_params': None,
 'algo:__n_jobs': None,
 'algo:__n_neighbors': 5,
 'algo:__p': 2,
 'algo:__weights': 'uniform'}
```

In [33]:

```
## it's very useful to use GridSearchCV to save your time while coding
model = GridSearchCV(
              estimator = pipe,
              param_grid={'algo:__n_neighbors':[1,2,3,4,5,6,7,8,9,10]},
              cv=5)
```

In [34]:

```
model.fit(X,y)
```

Out[34]:

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('Scaling:', StandardScaler()),
                                       ('algo:', KNeighborsRegressor())]),
             param_grid={'algo:__n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]})
```

In [35]:

```
model.cv_results_
```

Out[35]:

```
{'mean_fit_time': array([0.00279808, 0.0016036 , 0.00159907, 0.00199165, 0.00199795,
        0.00200248, 0.00199862, 0.00199881, 0.0023984 , 0.00199895]),
 'std_fit_time': array([7.48519919e-04, 4.91875268e-04, 4.89492795e-04, 8.89221397e-06,
        2.63341928e-06, 7.03557028e-06, 1.78416128e-07, 1.90734863e-07,
        7.99107713e-04, 1.78416128e-07]),
 'mean_score_time': array([0.00280218, 0.00179243, 0.00199509, 0.0010066 , 0.00160003,
        0.00179157, 0.00179911, 0.00199895, 0.0027976 , 0.00199862]),
 'std_score_time': array([7.48818876e-04, 3.96676444e-04, 6.38533153e-04, 9.06116075e-06,
        4.87744840e-04, 3.96261938e-04, 3.99708787e-04, 6.32485143e-04,
        3.99591701e-04, 1.78416128e-07]),
 'param_algo:__n_neighbors': masked_array(data=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
              mask=[False, False, False, False, False, False, False, False,
                    False, False],
        fill_value='?',
             dtype=object),
 'params': [{'algo:__n_neighbors': 1},
  {'algo:__n_neighbors': 2},
  {'algo:__n_neighbors': 3},
  {'algo:__n_neighbors': 4},
  {'algo:__n_neighbors': 5},
  {'algo:__n_neighbors': 6},
  {'algo:__n_neighbors': 7},
  {'algo:__n_neighbors': 8},
  {'algo:__n_neighbors': 9},
  {'algo:__n_neighbors': 10}],
 'split0_test_score': array([0.33931282, 0.44164945, 0.52030402, 0.54708785, 0.56089547,
        0.5824495 , 0.6024341 , 0.61508985, 0.62531412, 0.61444567]),
 'split1_test_score': array([0.42377859, 0.54796246, 0.59333945, 0.60692536, 0.61917359,
        0.62119411, 0.63618485, 0.63118482, 0.63062076, 0.65248907]),
 'split2_test_score': array([0.53456551, 0.47497978, 0.54774641, 0.50977006, 0.48661916,
        0.50911069, 0.51610185, 0.55133981, 0.56446366, 0.55555543]),
 'split3_test_score': array([0.48637285, 0.4967943 , 0.51389083, 0.49045195, 0.46986886,
        0.44685947, 0.44208773, 0.44011729, 0.42910655, 0.42064756]),
 'split4_test_score': array([-1.62392847, -0.54869909,  0.00297988,  0.21127771,  0.23133037,
         0.25041748,  0.24574919,  0.23907177,  0.27937626,  0.26112772]),
 'mean_test_score': array([0.03202026, 0.28253738, 0.43565212, 0.47310259, 0.47357749,
        0.48200625, 0.48851155, 0.49536071, 0.50577627, 0.50085309]),
 'std_test_score': array([0.83054892, 0.41705187, 0.21813892, 0.13680653, 0.13243123,
        0.13043421, 0.13902209, 0.14467396, 0.13450269, 0.1433808 ]),
 'rank_test_score': array([10,  9,  8,  7,  6,  5,  4,  3,  1,  2])}
```

In [36]:

```
pd.DataFrame(model.cv_results_)
```

Out[36]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_algo:__n_neighbors	params	split0_test_score	split1_test_scor
0	0.002798	7.485199e-04	0.002802	7.488189e-04	1	{'algo:__n_neighbors': 1}	0.339313	0.42377
1	0.001604	4.918753e-04	0.001792	3.966764e-04	2	{'algo:__n_neighbors': 2}	0.441649	0.54796
2	0.001599	4.894928e-04	0.001995	6.385332e-04	3	{'algo:__n_neighbors': 3}	0.520304	0.59333
3	0.001992	8.892214e-06	0.001007	9.061161e-06	4	{'algo:__n_neighbors': 4}	0.547088	0.60692
4	0.001998	2.633419e-06	0.001600	4.877448e-04	5	{'algo:__n_neighbors': 5}	0.560895	0.61917
5	0.002002	7.035570e-06	0.001792	3.962619e-04	6	{'algo:__n_neighbors': 6}	0.582450	0.62119
6	0.001999	1.784161e-07	0.001799	3.997088e-04	7	{'algo:__n_neighbors': 7}	0.602434	0.63618
7	0.001999	1.907349e-07	0.001999	6.324851e-04	8	{'algo:__n_neighbors': 8}	0.615090	0.63118
8	0.002398	7.991077e-04	0.002798	3.995917e-04	9	{'algo:__n_neighbors': 9}	0.625314	0.63062
9	0.001999	1.784161e-07	0.001999	1.784161e-07	10	{'algo:__n_neighbors': 10}	0.614446	0.65248

In [ ]:

In [ ]: