```
import numpy as np
import pandas as pd

df =
pd.read_csv('https://raw.githubusercontent.com/selva86/datasets/master
/BostonHousing.csv')

df.head()
```

```
      crim    zn  indus  chas    nox     rm   age     dis  rad  tax
ptratio  \
0  0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
15.3
1  0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
17.8
2  0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
17.8
3  0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
18.7
4  0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
18.7

        b  lstat  medv
0  396.90   4.98  24.0
1  396.90   9.14  21.6
2  392.83   4.03  34.7
3  394.63   2.94  33.4
4  396.90   5.33  36.2
```

```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]

from sklearn.model_selection import cross_val_score,KFold
from sklearn.neighbors import KNeighborsRegressor

knn = KNeighborsRegressor()

kfold = KFold(n_splits=5, shuffle=True, random_state=1)
scores = cross_val_score(knn, X, y, cv=kfold, scoring='r2')

scores.mean()
```

```
0.4761976351913221
```

GridSearchCV

```
from sklearn.model_selection import GridSearchCV

knn = KNeighborsRegressor()

param_grid = {
    'n_neighbors':[1,3,5,7,10,12,15,17,20],
```

```
    'weights':['uniform','distance'],
    'algorithm':['ball_tree', 'kd_tree', 'brute'],
    'p':[1,2]
}

gcv = GridSearchCV(knn, param_grid, scoring='r2', refit=True,
cv=kfold, verbose=2)

gcv.fit(X,y)
```

```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=1, p=2, weights=distance;
```

```
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=uniform;
total time=   0.0s
```

```
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=distance;
```

```
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
```

```
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=distance;
```

```
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
```

```
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=distance;
```

```
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
```

```
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=distance;
```

```
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=3, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=5, p=2, weights=distance;
total time=   0.0s
```

```
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=7, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=uniform;
total time=   0.0s
```

```
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=10, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=distance;
```

```
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
```

```
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=distance;
```

```
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=17, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
```

```
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=1, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=distance; total
```

```
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=3, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
```

```
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=distance; total
```

```
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=7, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=uniform; total
time=   0.0s
```

```
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=12, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=distance; total
```

```
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=15, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=1, weights=distance; total
time=   0.0s
```

```
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=17, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=1, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=uniform; total
time=   0.0s
```

```
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=20, p=2, weights=distance; total
time=   0.0s

GridSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True),
             estimator=KNeighborsRegressor(),
             param_grid={'algorithm': ['ball_tree', 'kd_tree',
'brute'],
                         'n_neighbors': [1, 3, 5, 7, 10, 12, 15, 17,
20],
                         'p': [1, 2], 'weights': ['uniform',
'distance']},
             scoring='r2', verbose=2)

gcv.best_params_

{'algorithm': 'ball_tree', 'n_neighbors': 5, 'p': 1, 'weights':
'distance'}

gcv.best_score_

0.6117139367845081

gcv.cv_results_

{'mean_fit_time': array([0.00324793, 0.00282245, 0.00226316,
0.00210419, 0.00318089,
        0.00204415, 0.0021935 , 0.00203738, 0.00242333, 0.0022748 ,
        0.00211887, 0.00205107, 0.00216346, 0.00240793, 0.0022934 ,
        0.00280962, 0.0021884 , 0.00222678, 0.00252113, 0.00254021,
        0.00235338, 0.00218024, 0.00220795, 0.00261083, 0.00229049,
        0.00213299, 0.00251431, 0.00299511, 0.00238109, 0.00259953,
        0.00234065, 0.00246878, 0.00232348, 0.00237899, 0.0027041 ,
        0.00211935, 0.00167446, 0.00198393, 0.00227852, 0.00216284,
        0.00225735, 0.00240941, 0.00235553, 0.00323544, 0.00231743,
        0.00233212, 0.00216718, 0.00216064, 0.00220675, 0.00218887,
        0.00210042, 0.00217166, 0.00262518, 0.00224423, 0.00266161,
        0.00236797, 0.00230169, 0.00211105, 0.00214224, 0.00220013,
        0.0022799 , 0.00209951, 0.00219088, 0.00214963, 0.00234199,
        0.00213561, 0.00221457, 0.0026598 , 0.00332079, 0.00300508,
        0.00282698, 0.00253181, 0.00181017, 0.00161614, 0.00159106,
        0.00162644, 0.00179343, 0.00262713, 0.00179825, 0.00161548,
        0.00171423, 0.0020658 , 0.0016253 , 0.00270987, 0.00175343,
        0.00256972, 0.00164256, 0.001613  , 0.00157485, 0.00165358,
```

```
        0.00159431, 0.00189114, 0.00163531, 0.00188155, 0.00160174,
        0.00157638, 0.00169582, 0.00163164, 0.00160079, 0.00160151,
        0.00242987, 0.00171027, 0.00225854, 0.00160985, 0.001652  ,
        0.00165215, 0.0016768 , 0.00172195]),
 'std_fit_time': array([1.65904178e-03, 1.42407883e-03, 3.08938692e-
04, 7.39286153e-05,
        1.43718456e-03, 3.59164434e-05, 3.94862784e-04, 1.80100580e-
05,
        2.89650429e-04, 2.28626158e-04, 2.38360989e-04, 5.15629223e-
05,
        1.14639765e-04, 4.73579547e-04, 5.09461808e-04, 4.20568146e-
04,
        2.29578963e-04, 4.03615302e-04, 9.65592302e-06, 3.12054259e-
04,
        1.71526445e-04, 1.25431795e-04, 1.26613456e-04, 5.27766085e-
04,
        5.98856975e-05, 1.07400687e-04, 3.87667448e-04, 1.48935421e-
03,
        1.81605483e-04, 3.59468093e-04, 1.36985332e-04, 2.00026362e-
04,
        1.69610220e-04, 1.84986479e-04, 1.17861152e-03, 2.86180700e-
04,
        3.53446678e-05, 4.02044350e-04, 1.18476849e-04, 1.10704357e-
04,
        1.32652721e-04, 2.67554617e-04, 1.25522380e-04, 1.67277518e-
03,
        1.25719091e-04, 2.38668959e-04, 1.13689359e-04, 7.77537928e-
05,
        1.07447972e-04, 1.20687042e-04, 1.01694778e-04, 1.26424054e-
04,
        5.42827605e-04, 9.32223352e-05, 8.49517357e-04, 3.38935671e-
04,
        1.07410573e-04, 8.36592551e-05, 1.15887951e-04, 1.09913499e-
04,
        8.05460534e-05, 1.70665142e-04, 9.03521989e-05, 1.24985323e-
04,
        5.58339459e-05, 1.02060040e-04, 1.47218915e-04, 3.23115956e-
04,
        8.50990298e-04, 1.03370261e-03, 6.80604918e-04, 1.52590305e-
04,
        2.56237737e-04, 5.23141869e-05, 5.75415424e-05, 7.53247873e-
05,
        7.04708858e-05, 1.77638881e-03, 7.51980816e-05, 2.96170772e-
05,
        1.06682231e-04, 3.97828690e-04, 3.09422862e-05, 1.80556423e-
03,
        9.82938937e-05, 1.75640133e-03, 3.78629722e-05, 6.44772357e-
05,
        2.04293422e-05, 4.93563545e-05, 4.27066178e-05, 4.24476062e-
```

```
04,
        1.45376694e-05, 3.14616470e-04, 2.29107229e-05, 6.41838402e-
05,
        2.90214862e-05, 9.29597189e-05, 3.85759198e-05, 8.22904618e-
05,
        1.44452204e-03, 1.18438479e-04, 6.80107174e-04, 4.07776351e-
05,
        1.24394834e-04, 9.86293141e-05, 5.44862356e-05, 5.38174903e-
05]),
 'mean_score_time': array([0.00318284, 0.00276599, 0.00275011,
0.00244274, 0.00298781,
        0.00267   , 0.00237589, 0.00240645, 0.00342822, 0.00331349,
        0.00252318, 0.0025075 , 0.00328217, 0.00306826, 0.0029295 ,
        0.0033082 , 0.00314717, 0.00313759, 0.00310698, 0.00307536,
        0.00295343, 0.00233712, 0.00205622, 0.00263119, 0.00215554,
        0.00222425, 0.00255775, 0.00232067, 0.00252113, 0.00270538,
        0.00247564, 0.00229802, 0.00258727, 0.00313072, 0.00247521,
        0.00303264, 0.00174913, 0.00211644, 0.00184507, 0.00191441,
        0.00196571, 0.00259748, 0.00176377, 0.00245938, 0.00191007,
        0.00197129, 0.00179286, 0.00184813, 0.00197282, 0.002002  ,
        0.0019527 , 0.00197067, 0.00314374, 0.00207238, 0.00216298,
        0.00227761, 0.00229173, 0.00239487, 0.00211887, 0.00194592,
        0.00206895, 0.00327392, 0.00200014, 0.0021626 , 0.00216246,
        0.00237246, 0.00204325, 0.00254693, 0.00357165, 0.00322509,
        0.00241861, 0.00231457, 0.00305076, 0.00251179, 0.001931  ,
        0.00203123, 0.002736  , 0.00282669, 0.00208206, 0.00199895,
        0.00255542, 0.00361013, 0.00196438, 0.00275521, 0.00288677,
        0.00271101, 0.00200953, 0.00203533, 0.00260391, 0.00263968,
        0.00197554, 0.00231142, 0.00257998, 0.00382829, 0.00201192,
        0.00207281, 0.00276065, 0.00269256, 0.00226426, 0.00216284,
        0.00297971, 0.0036521 , 0.003369  , 0.00216751, 0.00307288,
        0.00279055, 0.00232382, 0.00255008]),
 'std_score_time': array([9.22926633e-04, 9.44475393e-05, 5.10252109e-
04, 1.59255088e-04,
        3.57178673e-04, 5.44042570e-05, 8.03927368e-05, 6.33292167e-
05,
        5.25282459e-04, 4.51336418e-04, 2.43138093e-04, 4.12480404e-
05,
        7.95730408e-04, 3.39996376e-04, 5.03275686e-04, 6.63622782e-
04,
        3.25141388e-04, 2.88166704e-04, 9.42254356e-05, 3.89497538e-
04,
        4.14262764e-04, 1.81714477e-04, 1.24286268e-04, 3.09812261e-
04,
        7.20054108e-05, 1.25512960e-04, 4.13150382e-04, 2.59472358e-
04,
        2.72254960e-04, 5.04516234e-04, 3.76916582e-04, 2.57545157e-
04,
        2.63335081e-04, 1.06376514e-03, 7.02122636e-04, 8.24557320e-
```

```
04,
        8.34212401e-05, 4.42897791e-04, 2.33363949e-04, 1.89477567e-
04,
        2.08907763e-04, 4.95153032e-04, 5.16417945e-05, 5.54105718e-
04,
        1.31918239e-04, 1.49444097e-04, 1.34131453e-04, 1.53361579e-
04,
        1.60243509e-04, 2.57289197e-04, 1.66290107e-04, 2.13329111e-
04,
        1.79759890e-03, 1.57183306e-04, 1.95745307e-04, 2.72763025e-
04,
        3.05890773e-04, 1.93386916e-04, 1.26137841e-04, 1.23056456e-
04,
        1.04492355e-04, 1.87633922e-03, 1.40770648e-04, 1.30749201e-
04,
        7.51739489e-05, 2.69664605e-04, 1.27904668e-04, 4.67349206e-
04,
        8.67126590e-04, 1.59725401e-03, 3.69536695e-04, 1.83841597e-
04,
        6.53877780e-04, 9.57966207e-05, 1.25217147e-04, 1.95666838e-
04,
        2.76893824e-04, 2.34459623e-04, 8.12940091e-05, 4.11746608e-
05,
        4.55636307e-05, 1.14544090e-03, 2.86798425e-05, 1.18604144e-
03,
        4.61907457e-04, 1.79721905e-04, 5.44832309e-05, 5.84459988e-
05,
        6.36779560e-05, 4.92982749e-05, 4.48100092e-05, 3.98211830e-
04,
        5.14647144e-05, 2.15697737e-03, 6.85826449e-05, 3.00123444e-
05,
        1.60856476e-04, 3.16591147e-05, 4.76287979e-04, 1.28714381e-
04,
        4.67396196e-04, 1.71492439e-03, 1.90486929e-03, 5.69460967e-
05,
        8.23404962e-04, 6.52573210e-05, 5.68999613e-05, 3.93004615e-
04]),
 'param_algorithm': masked_array(data=['ball_tree', 'ball_tree',
'ball_tree', 'ball_tree',
                    'ball_tree', 'ball_tree', 'ball_tree',
'ball_tree',
                    'ball_tree', 'ball_tree', 'ball_tree',
'ball_tree',
                    'ball_tree', 'ball_tree', 'ball_tree',
'ball_tree',
                    'ball_tree', 'ball_tree', 'ball_tree',
'ball_tree',
                    'ball_tree', 'ball_tree', 'ball_tree',
'ball_tree',
```

```
                   'ball_tree', 'ball_tree', 'ball_tree',
'ball_tree',
                   'ball_tree', 'ball_tree', 'ball_tree',
'ball_tree',
                   'ball_tree', 'ball_tree', 'ball_tree',
'ball_tree',
                   'kd_tree', 'kd_tree', 'kd_tree', 'kd_tree',
'kd_tree',
                   'kd_tree', 'kd_tree', 'kd_tree', 'kd_tree',
'kd_tree',
                   'kd_tree', 'kd_tree', 'kd_tree', 'kd_tree',
'kd_tree',
                   'kd_tree', 'kd_tree', 'kd_tree', 'kd_tree',
'kd_tree',
                   'kd_tree', 'kd_tree', 'kd_tree', 'kd_tree',
'kd_tree',
                   'kd_tree', 'kd_tree', 'kd_tree', 'kd_tree',
'kd_tree',
                   'kd_tree', 'brute', 'brute', 'brute', 'brute',
'brute',
                   'brute', 'brute', 'brute', 'brute', 'brute',
'brute',
                   'brute', 'brute', 'brute', 'brute', 'brute',
'brute',
                   'brute', 'brute', 'brute', 'brute', 'brute',
'brute',
                   'brute', 'brute', 'brute', 'brute', 'brute',
'brute',
                   'brute', 'brute', 'brute', 'brute', 'brute',
'brute',
                   'brute'],
               mask=[False, False, False, False, False, False, False,
False,
                     False, False, False, False, False, False, False,
False,
                     False, False, False, False, False, False, False,
False,
                     False, False, False, False, False, False, False,
False,
                     False, False, False, False, False, False, False,
False,
                     False, False, False, False, False, False, False,
False,
                     False, False, False, False, False, False, False,
False,
                     False, False, False, False, False, False, False,
False,
```

```
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False],
        fill_value='?',
             dtype=object),
 'param_n_neighbors': masked_array(data=[1, 1, 1, 1, 3, 3, 3, 3, 5, 5,
5, 5, 7, 7, 7, 7, 10, 10,
                    10, 10, 12, 12, 12, 12, 15, 15, 15, 15, 17, 17,
17, 17,
                    20, 20, 20, 20, 1, 1, 1, 1, 3, 3, 3, 3, 5, 5, 5,
5, 7,
                    7, 7, 7, 10, 10, 10, 10, 12, 12, 12, 12, 15, 15,
15,
                    15, 17, 17, 17, 17, 20, 20, 20, 20, 1, 1, 1, 1, 3,
3,
                    3, 3, 5, 5, 5, 5, 7, 7, 7, 7, 10, 10, 10, 10, 12,
12,
                    12, 12, 15, 15, 15, 15, 17, 17, 17, 17, 20, 20,
20, 20],
              mask=[False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
```

```
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False],
             fill_value='?',
                  dtype=object),
 'param_p': masked_array(data=[1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1,
1, 2, 2, 1, 1,
                    2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2,
2,
                    1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1,
1,
                    2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2,
2,
                    1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1,
1,
                    2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2, 2, 1, 1, 2,
2],
                mask=[False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False],
             fill_value='?',
                  dtype=object),
 'param_weights': masked_array(data=['uniform', 'distance', 'uniform',
```

```
'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance',
                        'uniform', 'distance', 'uniform', 'distance'],
                 mask=[False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
                        False, False, False, False, False, False, False,
False,
```

```
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False, False, False, False,
False,
                    False, False, False, False],
       fill_value='?',
            dtype=object),
 'params': [{'algorithm': 'ball_tree',
   'n_neighbors': 1,
   'p': 1,
   'weights': 'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 1, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 1, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 1, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 3, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 3, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 3, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 3, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 5, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 5, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 5, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 5, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 7, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 7, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 7, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 7, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 10, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 10, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 10, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 10, 'p': 2, 'weights':
'distance'},
```

```
  {'algorithm': 'ball_tree', 'n_neighbors': 12, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 12, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 12, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 12, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 15, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 15, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 15, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 15, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 17, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 17, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 17, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 17, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 20, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 20, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'ball_tree', 'n_neighbors': 20, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'ball_tree', 'n_neighbors': 20, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 1, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 1, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 1, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 1, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 3, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 3, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 3, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 3, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 5, 'p': 1, 'weights':
```

```
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 5, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 5, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 5, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 7, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 7, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 7, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 7, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 10, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 10, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 10, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 10, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 12, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 12, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 12, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 12, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 15, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 15, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 15, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 15, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 17, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 17, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 17, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 17, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 20, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 20, 'p': 1, 'weights':
```

```
'distance'},
  {'algorithm': 'kd_tree', 'n_neighbors': 20, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'kd_tree', 'n_neighbors': 20, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 1, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 1, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 1, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 1, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 3, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 3, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 3, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 3, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 5, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 5, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 5, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 5, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 7, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 7, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 7, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 7, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 10, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 10, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 10, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 10, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 12, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 12, 'p': 1, 'weights':
'distance'},
```

```
  {'algorithm': 'brute', 'n_neighbors': 12, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 12, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 15, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 15, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 15, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 15, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 17, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 17, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 17, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 17, 'p': 2, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 20, 'p': 1, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 20, 'p': 1, 'weights':
'distance'},
  {'algorithm': 'brute', 'n_neighbors': 20, 'p': 2, 'weights':
'uniform'},
  {'algorithm': 'brute', 'n_neighbors': 20, 'p': 2, 'weights':
'distance'}],
 'split0_test_score': array([0.57368927, 0.57368927, 0.36092576,
0.36092576, 0.63059578,
        0.63732158, 0.59758726, 0.5827671 , 0.62405806, 0.65088308,
        0.54016122, 0.58189902, 0.60742085, 0.65003815, 0.49922842,
        0.5535314 , 0.56133446, 0.63302822, 0.50035679, 0.55707972,
        0.53049772, 0.61517589, 0.4868698 , 0.55586852, 0.48590179,
        0.59022763, 0.42373362, 0.51943974, 0.47958182, 0.5849959 ,
        0.398799  , 0.50051613, 0.43549943, 0.55794027, 0.36972368,
        0.47832487, 0.57368927, 0.57368927, 0.36092576, 0.36092576,
        0.63059578, 0.63732158, 0.59758726, 0.5827671 , 0.62405806,
        0.65088308, 0.54016122, 0.58189902, 0.60742085, 0.65003815,
        0.49922842, 0.5535314 , 0.56133446, 0.63302822, 0.50035679,
        0.55707972, 0.53049772, 0.61517589, 0.4868698 , 0.55586852,
        0.48590179, 0.59022763, 0.42373362, 0.51943974, 0.47958182,
        0.5849959 , 0.398799  , 0.50051613, 0.43549943, 0.55794027,
        0.36972368, 0.47832487, 0.57368927, 0.57368927, 0.36092576,
        0.36092576, 0.63059578, 0.63732158, 0.59758726, 0.5827671 ,
        0.62405806, 0.65088308, 0.54016122, 0.58189902, 0.60742085,
        0.65003815, 0.49922842, 0.5535314 , 0.56133446, 0.63302822,
        0.50035679, 0.55707972, 0.53049772, 0.61517589, 0.4868698 ,
        0.55586852, 0.48590179, 0.59022763, 0.42373362, 0.51943974,
```

```
       0.47958182, 0.5849959 , 0.398799  , 0.50051613, 0.43549943,
       0.55794027, 0.36972368, 0.47832487]),
 'split1_test_score': array([0.28694756, 0.28694756, 0.20650436,
0.20650436, 0.48731247,
       0.5014786 , 0.26209867, 0.2874217 , 0.58170572, 0.59245496,
       0.32531094, 0.34793909, 0.51871668, 0.56873777, 0.31445506,
       0.37068565, 0.45846088, 0.53880666, 0.40287651, 0.4409148 ,
       0.44868274, 0.53450988, 0.41114174, 0.45366621, 0.41222369,
       0.51433902, 0.3628422 , 0.43145049, 0.39784353, 0.50303833,
       0.34599592, 0.42065167, 0.3910486 , 0.49595413, 0.31813665,
       0.40662694, 0.28694756, 0.28694756, 0.20650436, 0.20650436,
       0.48731247, 0.5014786 , 0.26209867, 0.2874217 , 0.58170572,
       0.59245496, 0.32531094, 0.34793909, 0.51871668, 0.56873777,
       0.31445506, 0.37068565, 0.45846088, 0.53880666, 0.40287651,
       0.4409148 , 0.44868274, 0.53450988, 0.41114174, 0.45366621,
       0.41222369, 0.51433902, 0.3628422 , 0.43145049, 0.39784353,
       0.50303833, 0.34599592, 0.42065167, 0.3910486 , 0.49595413,
       0.31813665, 0.40662694, 0.28694756, 0.28694756, 0.20650436,
       0.20650436, 0.48731247, 0.5014786 , 0.26209867, 0.2874217 ,
       0.58170572, 0.59245496, 0.32531094, 0.34793909, 0.51871668,
       0.56873777, 0.31445506, 0.37068565, 0.45846088, 0.53880666,
       0.40287651, 0.4409148 , 0.44868274, 0.53450988, 0.41114174,
       0.45366621, 0.41222369, 0.51433902, 0.3628422 , 0.43145049,
       0.39784353, 0.50303833, 0.34599592, 0.42065167, 0.3910486 ,
       0.49595413, 0.31813665, 0.40662694]),
 'split2_test_score': array([0.52222195, 0.52222195, 0.51996861,
0.51996861, 0.56601077,
       0.61340095, 0.54604255, 0.59096178, 0.6421808 , 0.66038327,
       0.54378871, 0.58219007, 0.65394475, 0.68076718, 0.53552682,
       0.59227347, 0.60839674, 0.66474016, 0.5274634 , 0.59219986,
       0.609265  , 0.67191249, 0.5216754 , 0.59006049, 0.59268536,
       0.66963739, 0.48305239, 0.57596132, 0.57879249, 0.66135721,
       0.47582827, 0.56686223, 0.57361804, 0.65664193, 0.46469845,
       0.55731068, 0.52222195, 0.52222195, 0.51996861, 0.51996861,
       0.56601077, 0.61340095, 0.54604255, 0.59096178, 0.6421808 ,
       0.66038327, 0.54378871, 0.58219007, 0.65394475, 0.68076718,
       0.53552682, 0.59227347, 0.60839674, 0.66474016, 0.5274634 ,
       0.59219986, 0.609265  , 0.67191249, 0.5216754 , 0.59006049,
       0.59268536, 0.66963739, 0.48305239, 0.57596132, 0.57879249,
       0.66135721, 0.47582827, 0.56686223, 0.57361804, 0.65664193,
       0.46469845, 0.55731068, 0.52222195, 0.52222195, 0.51996861,
       0.51996861, 0.56601077, 0.61340095, 0.54604255, 0.59096178,
       0.6421808 , 0.66038327, 0.54378871, 0.58219007, 0.65394475,
       0.68076718, 0.53552682, 0.59227347, 0.60839674, 0.66474016,
       0.5274634 , 0.59219986, 0.609265  , 0.67191249, 0.5216754 ,
       0.59006049, 0.59268536, 0.66963739, 0.48305239, 0.57596132,
       0.57879249, 0.66135721, 0.47582827, 0.56686223, 0.57361804,
       0.65664193, 0.46469845, 0.55731068]),
 'split3_test_score': array([0.30892171, 0.30892171, 0.27253947,
```

```
0.27253947, 0.50341955,
       0.50498489, 0.37452435, 0.39378087, 0.51143253, 0.5238355 ,
       0.39244609, 0.42909381, 0.46915763, 0.49982763, 0.38872478,
       0.42987275, 0.39752207, 0.46346408, 0.33617132, 0.39164531,
       0.38898424, 0.45745644, 0.33276124, 0.39608896, 0.37105088,
       0.45170755, 0.27181668, 0.37034213, 0.34042827, 0.43393662,
       0.29494753, 0.38210198, 0.33168596, 0.42554071, 0.29753534,
       0.38098985, 0.30892171, 0.30892171, 0.27253947, 0.27253947,
       0.50341955, 0.50498489, 0.37452435, 0.39378087, 0.51143253,
       0.5238355 , 0.39244609, 0.42909381, 0.46915763, 0.49982763,
       0.38872478, 0.42987275, 0.39752207, 0.46346408, 0.33617132,
       0.39164531, 0.38898424, 0.45745644, 0.33276124, 0.39608896,
       0.37105088, 0.45170755, 0.27181668, 0.37034213, 0.34042827,
       0.43393662, 0.29494753, 0.38210198, 0.33168596, 0.42554071,
       0.29753534, 0.38098985, 0.30892171, 0.30892171, 0.27253947,
       0.27253947, 0.50341955, 0.50498489, 0.37452435, 0.39378087,
       0.51143253, 0.5238355 , 0.39244609, 0.42909381, 0.46915763,
       0.49982763, 0.38872478, 0.42987275, 0.39752207, 0.46346408,
       0.33617132, 0.39164531, 0.38898424, 0.45745644, 0.33276124,
       0.39608896, 0.37105088, 0.45170755, 0.27181668, 0.37034213,
       0.34042827, 0.43393662, 0.29494753, 0.38210198, 0.33168596,
       0.42554071, 0.29753534, 0.38098985]),
 'split4_test_score': array([0.4409745 , 0.4409745 , 0.2976715 ,
0.2976715 , 0.62276084,
       0.64733061, 0.58663289, 0.59214646, 0.58941767, 0.63101287,
       0.57928121, 0.6073807 , 0.56931059, 0.6292083 , 0.55741285,
       0.60806719, 0.51524244, 0.58782237, 0.49100337, 0.56524814,
       0.51599399, 0.58990904, 0.45691689, 0.54722598, 0.50760441,
       0.5928878 , 0.43576931, 0.53374073, 0.48310784, 0.58239611,
       0.42943441, 0.53134402, 0.47030239, 0.57702367, 0.39856523,
       0.51520454, 0.4409745 , 0.4409745 , 0.2976715 , 0.2976715 ,
       0.62276084, 0.64733061, 0.58663289, 0.59214646, 0.58941767,
       0.63101287, 0.57928121, 0.6073807 , 0.56931059, 0.6292083 ,
       0.55741285, 0.60806719, 0.51524244, 0.58782237, 0.49100337,
       0.56524814, 0.51599399, 0.58990904, 0.45691689, 0.54722598,
       0.50760441, 0.5928878 , 0.43576931, 0.53374073, 0.48310784,
       0.58239611, 0.42943441, 0.53134402, 0.47030239, 0.57702367,
       0.39856523, 0.51520454, 0.4409745 , 0.4409745 , 0.2976715 ,
       0.2976715 , 0.62276084, 0.64733061, 0.58663289, 0.59214646,
       0.58941767, 0.63101287, 0.57928121, 0.6073807 , 0.56931059,
       0.6292083 , 0.55741285, 0.60806719, 0.51524244, 0.58782237,
       0.49100337, 0.56524814, 0.51599399, 0.58990904, 0.45691689,
       0.54722598, 0.50760441, 0.5928878 , 0.43576931, 0.53374073,
       0.48310784, 0.58239611, 0.42943441, 0.53134402, 0.47030239,
       0.57702367, 0.39856523, 0.51520454]),
 'mean_test_score': array([0.426551  , 0.426551  , 0.33152194,
0.33152194, 0.56201988,
       0.58090333, 0.47337714, 0.48941558, 0.58975896, 0.61171394,
       0.47619764, 0.50970054, 0.5637101 , 0.60571581, 0.45906959,
```

```
        0.51088609, 0.50819132, 0.5775723 , 0.45157428, 0.50941757,
        0.49868474, 0.57379275, 0.44187302, 0.50858203, 0.47389323,
        0.56375988, 0.39544284, 0.48618688, 0.45595079, 0.55314484,
        0.38900103, 0.48029521, 0.44043088, 0.54262014, 0.36973187,
        0.46769137, 0.426551  , 0.426551  , 0.33152194, 0.33152194,
        0.56201988, 0.58090333, 0.47337714, 0.48941558, 0.58975896,
        0.61171394, 0.47619764, 0.50970054, 0.5637101 , 0.60571581,
        0.45906959, 0.51088609, 0.50819132, 0.5775723 , 0.45157428,
        0.50941757, 0.49868474, 0.57379275, 0.44187302, 0.50858203,
        0.47389323, 0.56375988, 0.39544284, 0.48618688, 0.45595079,
        0.55314484, 0.38900103, 0.48029521, 0.44043088, 0.54262014,
        0.36973187, 0.46769137, 0.426551  , 0.426551  , 0.33152194,
        0.33152194, 0.56201988, 0.58090333, 0.47337714, 0.48941558,
        0.58975896, 0.61171394, 0.47619764, 0.50970054, 0.5637101 ,
        0.60571581, 0.45906959, 0.51088609, 0.50819132, 0.5775723 ,
        0.45157428, 0.50941757, 0.49868474, 0.57379275, 0.44187302,
        0.50858203, 0.47389323, 0.56375988, 0.39544284, 0.48618688,
        0.45595079, 0.55314484, 0.38900103, 0.48029521, 0.44043088,
        0.54262014, 0.36973187, 0.46769137]),
 'std_test_score': array([0.11343404, 0.11343404, 0.10642454,
0.10642454, 0.05903091,
        0.06437949, 0.13262409, 0.12611685, 0.04499884, 0.04973711,
        0.09906284, 0.10263818, 0.06488276, 0.06438751, 0.09272712,
        0.09392126, 0.07434492, 0.07113666, 0.07125007, 0.0784915 ,
        0.07493206, 0.07304391, 0.06552746, 0.07216446, 0.07716463,
        0.07450795, 0.07272999, 0.07462258, 0.08137311, 0.07784768,
        0.06319071, 0.06885924, 0.08114125, 0.0778519 , 0.05953893,
        0.06580002, 0.11343404, 0.11343404, 0.10642454, 0.10642454,
        0.05903091, 0.06437949, 0.13262409, 0.12611685, 0.04499884,
        0.04973711, 0.09906284, 0.10263818, 0.06488276, 0.06438751,
        0.09272712, 0.09392126, 0.07434492, 0.07113666, 0.07125007,
        0.0784915 , 0.07493206, 0.07304391, 0.06552746, 0.07216446,
        0.07716463, 0.07450795, 0.07272999, 0.07462258, 0.08137311,
        0.07784768, 0.06319071, 0.06885924, 0.08114125, 0.0778519 ,
        0.05953893, 0.06580002, 0.11343404, 0.11343404, 0.10642454,
        0.10642454, 0.05903091, 0.06437949, 0.13262409, 0.12611685,
        0.04499884, 0.04973711, 0.09906284, 0.10263818, 0.06488276,
        0.06438751, 0.09272712, 0.09392126, 0.07434492, 0.07113666,
        0.07125007, 0.0784915 , 0.07493206, 0.07304391, 0.06552746,
        0.07216446, 0.07716463, 0.07450795, 0.07272999, 0.07462258,
        0.08137311, 0.07784768, 0.06319071, 0.06885924, 0.08114125,
        0.0778519 , 0.05953893, 0.06580002]),
 'rank_test_score': array([ 88,  91, 103, 107,  25,  10,  67,  52,
7,   1,  61,  37,  22,
         4,  73,  34,  46,  13,  79,  40,  49,  16,  82,  43,  64,
19,
        94,  55,  76,  28,  97,  58,  85,  31, 100,  70,  88,  91,
103,
        107,  25,  10,  67,  52,   7,   1,  61,  37,  22,   4,  73,
```

```
34,
         46,  13,  79,  40,  49,  16,  82,  43,  64,  19,  94,  55,
76,
         28,  97,  58,  85,  31, 100,  70,  88,  91, 103, 103,  25,
10,
         67,  54,   7,   1,  61,  39,  22,   4,  73,  36,  46,  13,
79,
         42,  49,  16,  82,  45,  64,  19,  94,  57,  76,  28,  97,
60,
         85,  31, 100,  72], dtype=int32)}
```

```
pd.DataFrame(gcv.cv_results_)[['param_algorithm',
     'param_n_neighbors',  'param_p', 'param_weights',
'mean_test_score']].sort_values('mean_test_score',ascending=False)
```

```
   param_algorithm param_n_neighbors param_p param_weights
mean_test_score
81           brute                 5       1      distance
0.611714
45         kd_tree                 5       1      distance
0.611714
9        ball_tree                 5       1      distance
0.611714
49         kd_tree                 7       1      distance
0.605716
85           brute                 7       1      distance
0.605716
..             ...               ...     ...           ...
...
38         kd_tree                 1       2       uniform
0.331522
2        ball_tree                 1       2       uniform
0.331522
75           brute                 1       2      distance
0.331522
39         kd_tree                 1       2      distance
0.331522
3        ball_tree                 1       2      distance
0.331522

[108 rows x 5 columns]
```

```
gcv.predict(new_data)
```

RandomizedSearchCV

```
from sklearn.model_selection import RandomizedSearchCV

rcv = RandomizedSearchCV(knn, param_grid, scoring='r2', refit=True,
cv=kfold, verbose=2)
```

```
rcv.fit(X,y)

Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=15, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=2, weights=distance; total
time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=17, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=ball_tree, n_neighbors=3, p=1, weights=distance;
total time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
```

```
time=   0.0s
[CV] END algorithm=brute, n_neighbors=5, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=1, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=20, p=2, weights=uniform;
total time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=brute, n_neighbors=10, p=1, weights=uniform; total
time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=15, p=2, weights=distance;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
```

```
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s
[CV] END algorithm=kd_tree, n_neighbors=12, p=1, weights=uniform;
total time=   0.0s

RandomizedSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True),
                   estimator=KNeighborsRegressor(),
                   param_distributions={'algorithm': ['ball_tree',
'kd_tree',
                                                      'brute'],
                                        'n_neighbors': [1, 3, 5, 7,
10, 12, 15,
                                                        17, 20],
                                        'p': [1, 2],
                                        'weights': ['uniform',
'distance']},
                   scoring='r2', verbose=2)

rcv.best_score_

0.589758956010885

rcv.best_params_

{'weights': 'uniform', 'p': 1, 'n_neighbors': 5, 'algorithm': 'brute'}
```