18 Parameter Tuning in Supervised Learning

Nandi Christmas

In [9]:

```python
import pandas as pd
data_path = "D:\\kc.csv"
df = pd.read_csv(data_path)
df.rename(columns={'price': 'Target'}, inplace=True)
```

In [10]:

```python
import numpy as np
df['Partition'] = np.random.choice(
    a = ['Train','Validation','Test'],
    size = df.shape[0]
)
list_predictors = [
   'id',
 'bedrooms',
 'bathrooms',
 'sqft_living',
 'sqft_lot',
 'floors',
 'waterfront',
 'view',
 'condition',
 'grade',
 'sqft_above',
 'sqft_basement',
 'yr_built',
 'yr_renovated',
 'zipcode',
 'lat',
 'long',
 'sqft_living15',
 'sqft_lot15'
]
X = df[list_predictors]
y = df['Target']
X_Train = X.loc[df['Partition'] == 'Train']
y_Train = y.loc[df['Partition'] == 'Train']
```

In [11]:

```python
X
```

Out[11]:

	id	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
0	7129300520	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257	1340	5650
1	6414100192	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	1991	98125	47.7210	-122.319	1690	7639
2	5631500400	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933	0	98028	47.7379	-122.233	2720	8062
3	2487200875	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	0	98136	47.5208	-122.393	1360	5000
4	1954400510	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	0	98074	47.6168	-122.045	1800	7503
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
21608	263000018	3	2.50	1530	1131	3.0	0	0	3	8	1530	0	2009	0	98103	47.6993	-122.346	1530	1509
21609	6600060120	4	2.50	2310	5813	2.0	0	0	3	8	2310	0	2014	0	98146	47.5107	-122.362	1830	7200
21610	1523300141	2	0.75	1020	1350	2.0	0	0	3	7	1020	0	2009	0	98144	47.5944	-122.299	1020	2007
21611	291310100	3	2.50	1600	2388	2.0	0	0	3	8	1600	0	2004	0	98027	47.5345	-122.069	1410	1287
21612	1523300157	2	0.75	1020	1076	2.0	0	0	3	7	1020	0	2008	0	98144	47.5941	-122.299	1020	1357

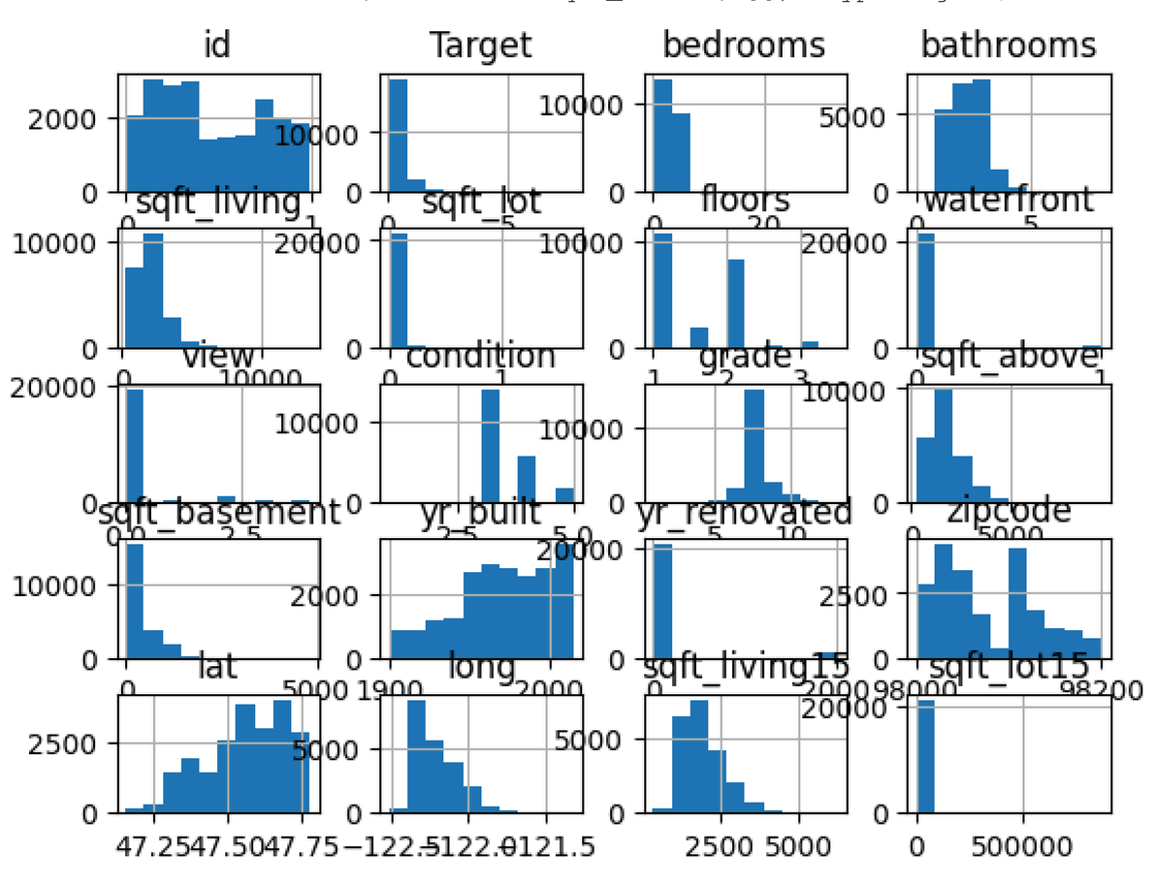
21613 rows × 19 columns

In [12]:

```python
df.hist()
```

Out[12]:

```
array([[<Axes: title={'center': 'id'}>,
        <Axes: title={'center': 'Target'}>,
        <Axes: title={'center': 'bedrooms'}>,
        <Axes: title={'center': 'bathrooms'}>],
       [<Axes: title={'center': 'sqft_living'}>,
        <Axes: title={'center': 'sqft_lot'}>,
        <Axes: title={'center': 'floors'}>,
        <Axes: title={'center': 'waterfront'}>],
       [<Axes: title={'center': 'view'}>,
        <Axes: title={'center': 'condition'}>,
        <Axes: title={'center': 'grade'}>,
        <Axes: title={'center': 'sqft_above'}>],
       [<Axes: title={'center': 'sqft_basement'}>,
        <Axes: title={'center': 'yr_built'}>,
        <Axes: title={'center': 'yr_renovated'}>,
        <Axes: title={'center': 'zipcode'}>],
       [<Axes: title={'center': 'lat'}>,
        <Axes: title={'center': 'long'}>,
        <Axes: title={'center': 'sqft_living15'}>,
        <Axes: title={'center': 'sqft_lot15'}>]], dtype=object)
```



In [13]:

```python
from sklearn.ensemble import RandomForestRegressor
```

In [14]:

```python
from sklearn.model_selection import ParameterGrid
ParameterGrid_RandomForestRegressor = ParameterGrid(
    param_grid = {
        'n_estimators': [50,51],
        'min_samples_leaf': [10, 20],
    }
)
pd.DataFrame(
    data = ParameterGrid_RandomForestRegressor
)
```

Out[14]:

	min_samples_leaf	n_estimators
0	10	50
1	10	51
2	20	50
3	20	51

In [15]:

```python
from sklearn.model_selection import ParameterSampler
from scipy.stats import randint
pd.DataFrame(ParameterSampler(
    param_distributions = {
        'n_estimators': randint(50,51),
        'min_samples_leaf': randint(10, 20),
    },
    n_iter = 30,
))
```

Out[15]:

	min_samples_leaf	n_estimators
0	15	50
1	13	50
2	10	50
3	12	50
4	13	50
5	19	50
6	10	50
7	13	50
8	19	50
9	10	50
10	13	50
11	15	50
12	15	50
13	17	50
14	10	50
15	13	50
16	12	50
17	14	50
18	15	50
19	17	50
20	18	50
21	12	50
22	10	50
23	18	50
24	14	50
25	17	50
26	19	50
27	13	50
28	18	50
29	12	50

In [16]:

```python
from sklearn.model_selection import GridSearchCV
GridSearchCV_RandomForestRegressor = GridSearchCV(
    estimator = RandomForestRegressor(),
    param_grid = {
        'n_estimators': [50,51],
        'min_samples_leaf': [10, 20],
    }
)
GridSearchCV_RandomForestRegressor = GridSearchCV_RandomForestRegressor.fit(
    X = X_Train,
    y = y_Train
)
pd.DataFrame(
    data = GridSearchCV_RandomForestRegressor.cv_results_
).sort_values('rank_test_score')
```

Out[16]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_min_samples_leaf	param_n_estimators	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
0	1.359001	0.016248	0.008415	0.000351	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	0.822985	0.725300	0.785817	0.855072	0.797400	0.797314	0.043158	1
1	1.408053	0.004803	0.008885	0.000610	10	51	{'min_samples_leaf': 10, 'n_estimators': 51}	0.822364	0.730896	0.780319	0.849763	0.798821	0.796433	0.040192	2
2	1.154449	0.012879	0.007528	0.000334	20	50	{'min_samples_leaf': 20, 'n_estimators': 50}	0.801055	0.702954	0.753779	0.837605	0.807000	0.780479	0.047145	3
3	1.180491	0.009839	0.007472	0.000218	20	51	{'min_samples_leaf': 20, 'n_estimators': 51}	0.796373	0.702525	0.754337	0.829220	0.811797	0.778850	0.045510	4

In [17]:

```python
from sklearn.experimental import enable_halving_search_cv
from sklearn.model_selection import HalvingGridSearchCV
HalvingGridSearchCV_RandomForestRegressor = HalvingGridSearchCV(
    estimator = RandomForestRegressor(),
    param_grid = {
        'n_estimators': [50,51],
        'min_samples_leaf': [10, 20],
    },
)
HalvingGridSearchCV_RandomForestRegressor = HalvingGridSearchCV_RandomForestRegressor.fit(
    X = X_Train,
    y = y_Train
)
pd.DataFrame(
    data = HalvingGridSearchCV_RandomForestRegressor.cv_results_
).sort_values('rank_test_score')
```

Out[17]:

	iter	n_resources	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_min_samples_leaf	param_n_estimators	params	split0_test_score	...	mean_test_score	std_test_score	rank_test_score	split0_train_score	split1_train_score	split2_train_score	split3_train_score	split4_train_score	mean_train_score	std_train_score
5	1	7134	1.398344	0.014521	0.008441	0.000156	10	51	{'min_samples_leaf': 10, 'n_estimators': 51}	0.825323	...	0.797061	0.043640	1	0.855732	0.876451	0.869639	0.854417	0.862806	0.863809	0.008346
4	1	7134	1.374625	0.020757	0.008456	0.000170	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	0.818020	...	0.792152	0.042364	2	0.848913	0.875175	0.871096	0.855756	0.861884	0.862565	0.009644
1	0	2378	0.420301	0.012243	0.004461	0.000500	10	51	{'min_samples_leaf': 10, 'n_estimators': 51}	0.744141	...	0.775141	0.028122	3	0.860754	0.856574	0.872299	0.775869	0.748307	0.822761	0.050563
0	0	2378	0.404225	0.006834	0.004151	0.000069	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	0.732781	...	0.774357	0.030554	4	0.858457	0.851802	0.871954	0.782630	0.750260	0.823021	0.047758
2	0	2378	0.339178	0.005131	0.003902	0.000217	20	50	{'min_samples_leaf': 20, 'n_estimators': 50}	0.680559	...	0.740928	0.041435	5	0.791358	0.776875	0.815978	0.712080	0.689861	0.757230	0.048123
3	0	2378	0.347638	0.007157	0.004088	0.000226	20	51	{'min_samples_leaf': 20, 'n_estimators': 51}	0.687805	...	0.739723	0.039498	6	0.794324	0.782417	0.818241	0.710646	0.688938	0.758913	0.050105

6 rows × 24 columns

In [18]:

```python
from sklearn.model_selection import RandomizedSearchCV
RandomizedSearchCV_RandomForestRegressor = RandomizedSearchCV(
    estimator = RandomForestRegressor(),
    param_distributions = {
        'n_estimators': randint(10,11),
        'min_samples_leaf': randint(10, 20),
    },
)
RandomizedSearchCV_RandomForestRegressor = RandomizedSearchCV_RandomForestRegressor.fit(
    X = X_Train,
    y = y_Train
)
pd.DataFrame(
    data = RandomizedSearchCV_RandomForestRegressor.cv_results_
).sort_values('rank_test_score')
```

Out[18]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_min_samples_leaf	param_n_estimators	params	split0_test_score	split1_test_score	split2_test_score	split3_test_score	split4_test_score	mean_test_score	std_test_score	rank_test_score
6	0.280038	0.012800	0.003244	0.000375	10	10	{'min_samples_leaf': 10, 'n_estimators': 10}	0.819460	0.722049	0.789525	0.854931	0.794878	0.796169	0.043669	1
7	0.259365	0.002725	0.002842	0.000115	13	10	{'min_samples_leaf': 13, 'n_estimators': 10}	0.814670	0.710787	0.772800	0.836375	0.803088	0.787544	0.043522	2
0	0.277313	0.007606	0.003036	0.000154	10	10	{'min_samples_leaf': 10, 'n_estimators': 10}	0.817746	0.709341	0.774593	0.849628	0.772086	0.784679	0.047430	3
4	0.251613	0.003333	0.002784	0.000049	14	10	{'min_samples_leaf': 14, 'n_estimators': 10}	0.804764	0.708871	0.757655	0.830456	0.817200	0.783789	0.044788	4
2	0.247544	0.001815	0.002784	0.000055	15	10	{'min_samples_leaf': 15, 'n_estimators': 10}	0.801473	0.701550	0.771419	0.830658	0.800018	0.781024	0.043934	5
5	0.257432	0.002119	0.003054	0.000352	13	10	{'min_samples_leaf': 13, 'n_estimators': 10}	0.808882	0.698085	0.759844	0.823282	0.802824	0.778584	0.045464	6
8	0.247792	0.003191	0.002975	0.000328	16	10	{'min_samples_leaf': 16, 'n_estimators': 10}	0.782910	0.712815	0.760684	0.831579	0.796628	0.776923	0.039459	7
1	0.242979	0.004538	0.002756	0.000085	17	10	{'min_samples_leaf': 17, 'n_estimators': 10}	0.800863	0.702126	0.757994	0.833376	0.790188	0.776909	0.044472	8
9	0.231550	0.000519	0.002765	0.000128	19	10	{'min_samples_leaf': 19, 'n_estimators': 10}	0.788575	0.707602	0.744442	0.838003	0.802249	0.776174	0.045531	9
3	0.234544	0.001707	0.002934	0.000362	19	10	{'min_samples_leaf': 19, 'n_estimators': 10}	0.790020	0.695806	0.755300	0.826413	0.802391	0.773986	0.045329	10

In [19]:

```python
from sklearn.model_selection import HalvingRandomSearchCV
from sklearn.experimental import enable_halving_search_cv
from scipy.stats import randint
HalvingRandomSearchCV_RandomForestRegressor = HalvingRandomSearchCV(
    estimator = RandomForestRegressor(),
    param_distributions = {
        'n_estimators': randint(50,51),
        'min_samples_leaf': randint(10, 11),
    },
)
HalvingRandomSearchCV_RandomForestRegressor = HalvingRandomSearchCV_RandomForestRegressor.fit(
    X = X_Train,
    y = y_Train
)
pd.DataFrame(
    data = HalvingRandomSearchCV_RandomForestRegressor.cv_results_
).sort_values('rank_test_score')
```

```
d:\python\Lib\site-packages\numpy\ma\core.py:2881: RuntimeWarning: invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
d:\python\Lib\site-packages\numpy\ma\core.py:2881: RuntimeWarning: invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
```

Out[19]:

	iter	n_resources	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_min_samples_leaf	param_n_estimators	params	split0_test_score	...	mean_test_score	std_test_score	rank_test_score	split0_train_score	split1_train_score	split2_train_score	split3_train_score	split4_train_score	mean_train_score	std_train_score
1067	5	2430	0.400324	0.005217	0.004015	0.000049	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	0.759994	...	0.794861	0.043984	1	0.732767	0.867663	0.770389	0.865797	0.856797	0.818683	0.056188
1069	5	2430	0.401000	0.002007	0.003969	0.000058	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	0.748075	...	0.794085	0.042766	2	0.728473	0.864179	0.777613	0.864510	0.855148	0.817984	0.055357
1068	5	2430	0.403342	0.003462	0.003980	0.000036	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	0.721722	...	0.786336	0.053755	3	0.725632	0.871050	0.776870	0.863083	0.852525	0.817832	0.057030
1055	3	270	0.056464	0.000643	0.002499	0.000104	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	0.720699	...	0.624430	0.069749	4	0.649021	0.672496	0.777839	0.765994	0.661801	0.705430	0.054920
1036	3	270	0.057275	0.001129	0.002518	0.000083	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	0.725007	...	0.622292	0.060070	5	0.647132	0.669798	0.778767	0.770143	0.664224	0.706013	0.056446
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
188	0	10	0.036254	0.001214	0.002505	0.000469	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	-3.277361	...	-4.014584	6.145554	1066	-0.003744	-0.000717	-0.001835	-0.001220	-0.010715	-0.003646	0.003680
615	0	10	0.035286	0.000325	0.002344	0.000034	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	-3.248647	...	-4.037927	6.110986	1067	-0.003090	-0.001223	-0.001886	-0.007601	-0.009929	-0.004746	0.003416
645	0	10	0.035268	0.000258	0.002337	0.000050	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	-3.370331	...	-4.063893	6.192554	1068	-0.006268	-0.001195	-0.000456	-0.001493	-0.013934	-0.004669	0.005066
543	0	10	0.037712	0.004620	0.002511	0.000306	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	-3.275850	...	-4.071454	6.233073	1069	-0.003708	-0.000064	-0.000065	-0.001984	-0.016807	-0.004526	0.006289
2	0	10	0.036840	0.000557	0.002750	0.000434	10	50	{'min_samples_leaf': 10, 'n_estimators': 50}	-3.154588	...	-4.148786	6.415595	1070	-0.001372	-0.000915	-0.000566	-0.001707	-0.032600	-0.007432	0.012590

1070 rows × 24 columns