```
# Python ≥3.5 is required
import sys
assert sys.version_info >= (3, 5)

# Scikit-Learn ≥0.20 is required
import sklearn
assert sklearn.__version__ >= "0.20"

# Common imports
import numpy as np
import os

# To plot pretty figures
%matplotlib inline
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

# Where to save the figures
IMAGES_PATH = r"E:\Jupyter\handson-ml2\images\end_to_end_project"
os.makedirs(IMAGES_PATH, exist_ok=True)

def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
    print("Saving figure", fig_id)
    if tight_layout:
        plt.tight_layout()
    plt.savefig(path, format=fig_extension, dpi=resolution)

# Ignore useless warnings (see SciPy issue #5998)
import warnings
warnings.filterwarnings(action="ignore", message="^internal gelsd")
```

一、获取数据

```
import pandas as pd
HOUSING_PATH=r'E:\Jupyter\handson-ml2\datasets\housing'
def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

1.快速查看数据结构

```
housing = load_housing_data()
housing.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
```

```
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

```
housing["ocean_proximity"].value_counts()
```

```
<1H OCEAN     9136
INLAND        6551
NEAR OCEAN    2658
NEAR BAY      2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

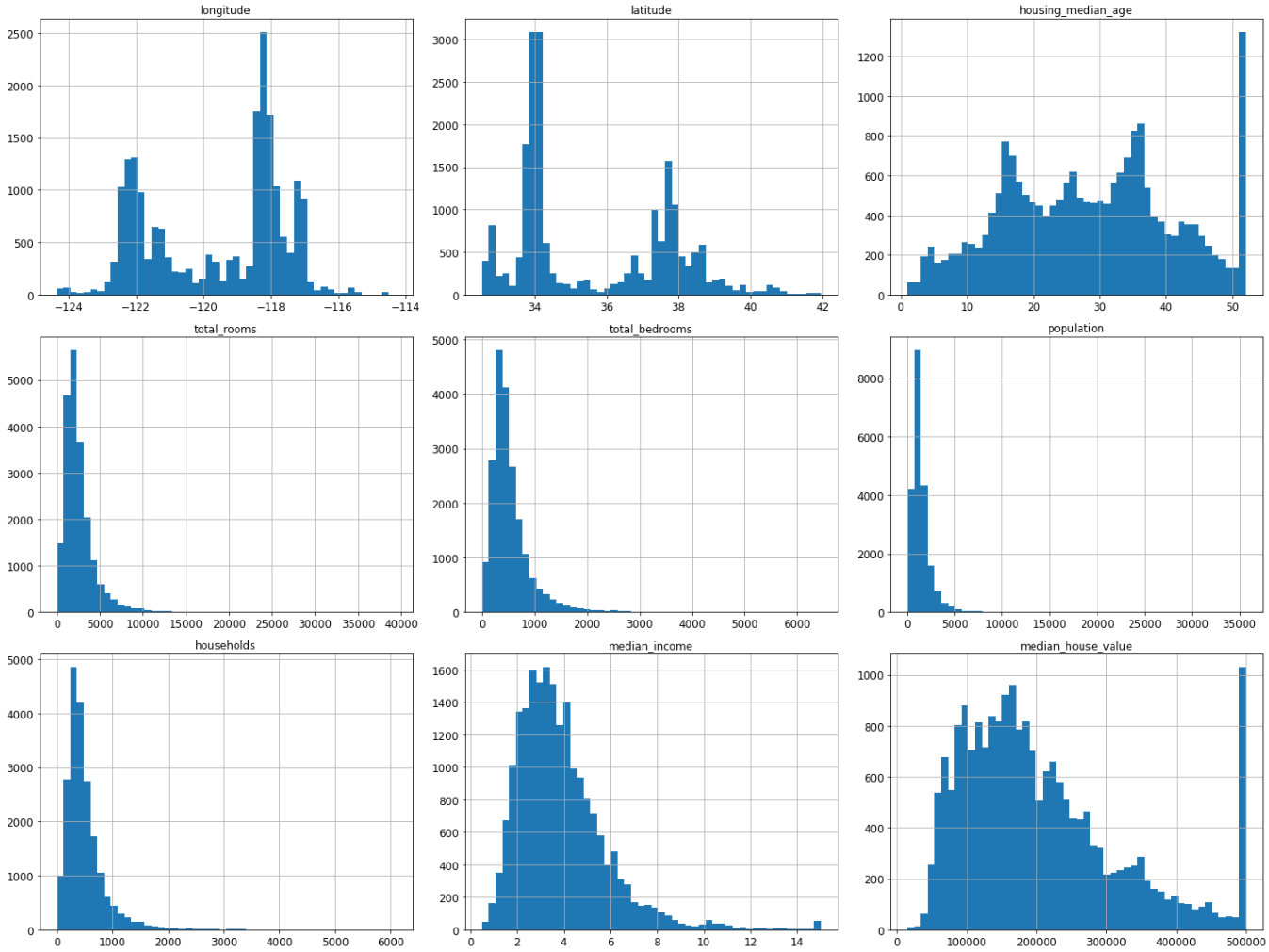
```
housing.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100

```
%matplotlib inline
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
save_fig("attribute_histogram_plots")
plt.show()
```

```
Saving figure attribute_histogram_plots
```



2.创建测试集

①自己构造train_test_split与随机数

```
import numpy as np
np.random.seed(42)

def split_train_test(data,test_ratio):
    shuffled_indices = np.random.permutation(len(data))
    test_set_size = int(len(data) * test_ratio)
    test_indices = shuffled_indices[:test_set_size]
    train_indices = shuffled_indices[test_set_size:]
    return data.iloc[train_indices],data.iloc[test_indices]

train_set,test_set = split_train_test(housing,0.2)
print(len(train_set))
print(len(test_set))
```

```
16512
4128
```

目前虽然可以正确分割，但若再运行一遍且删掉seed，会产生一个不同的数据集。

为了避免这种情况，解决方案有如下几点：

1.第一次运行程序后即保存测试集。

2.调用`np.random.premutation()`前设置随机数生成的种子。

但是这两种解决方案在下一次获取更新的数据时都会中断。为了即使在更新数据集之后也有一个稳定的训练测试分割，常用方法是每个实例都使用一个标识符来决定是否进入测试集（假设每个实例都有一个**唯一且不变**的标识符）。例如，先计算每个实例标识符的hash值。若hash值小于等于最大哈希值的20%，则将该实例放入测试集。

至于标识符列，可以选择使用行索引，也可以使用最稳定的特征，例如经纬度。

```
from zlib import crc32

def test_set_check(identifier, test_ratio):
    return crc32(np.int64(identifier)) & 0xffffffff < test_ratio * 2**32

def split_train_test_by_id(data, test_ratio, id_column):
    ids = data[id_column]
    in_test_set = ids.apply(lambda id_: test_set_check(id_, test_ratio))
    return data.loc[~in_test_set], data.loc[in_test_set]

housing_with_id= housing.reset_index()   # adds an `index` column
train_set1, test_set1 = split_train_test_by_id(housing_with_id, 0.2, "index")
```

```
test_set.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_ho
20046	-119.01	36.06	25.0	1505.0	NaN	1392.0	359.0	1.6812	47700.0
3024	-119.46	35.14	30.0	2943.0	NaN	1565.0	584.0	2.5313	45800.0
15663	-122.44	37.80	52.0	3830.0	NaN	1310.0	963.0	3.4801	500001.0
20484	-118.72	34.28	17.0	3051.0	NaN	1705.0	495.0	5.7376	218600.0
9814	-121.93	36.62	34.0	2351.0	NaN	1063.0	428.0	3.7250	278000.0

```
test_set1.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	index	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	med
2	2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	3521
5	5	-122.25	37.85	52.0	919.0	213.0	413.0	193.0	4.0368	2697
12	12	-122.26	37.85	52.0	2491.0	474.0	1098.0	468.0	3.0750	2135
16	16	-122.27	37.85	52.0	1966.0	347.0	793.0	331.0	2.7750	1525
23	23	-122.27	37.84	52.0	1688.0	337.0	853.0	325.0	2.1806	9970

```
housing_with_id["id"] = housing["longitude"] * 1000 + housing["latitude"]
train_set2, test_set2 = split_train_test_by_id(housing_with_id, 0.2, "id")
test_set2.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	index	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	med
59	59	-122.29	37.82	2.0	158.0	43.0	94.0	57.0	2.5625	6000
60	60	-122.29	37.83	52.0	1121.0	211.0	554.0	187.0	3.3929	7570
61	61	-122.29	37.82	49.0	135.0	29.0	86.0	23.0	6.1183	7500
62	62	-122.29	37.81	50.0	760.0	190.0	377.0	122.0	0.9011	8610
67	67	-122.29	37.80	52.0	1027.0	244.0	492.0	147.0	2.6094	8130

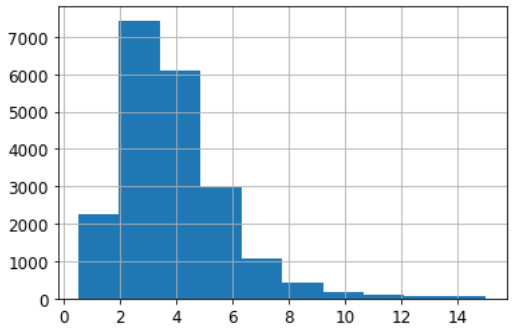
②sklearn中的train_test_split

```
from sklearn.model_selection import train_test_split

train_set,test_set = train_test_split(housing,test_size=0.2,random_state=42)
test_set.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_ho
20046	-119.01	36.06	25.0	1505.0	NaN	1392.0	359.0	1.6812	47700.0
3024	-119.46	35.14	30.0	2943.0	NaN	1565.0	584.0	2.5313	45800.0
15663	-122.44	37.80	52.0	3830.0	NaN	1310.0	963.0	3.4801	500001.0
20484	-118.72	34.28	17.0	3051.0	NaN	1705.0	495.0	5.7376	218600.0
9814	-121.93	36.62	34.0	2351.0	NaN	1063.0	428.0	3.7250	278000.0

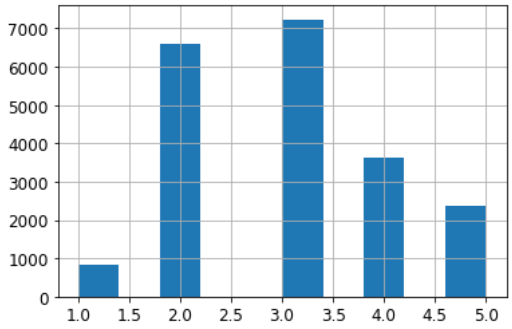
```
housing['median_income'].hist()
plt.show()
```



目前为止，以及思考过纯随机的抽样方法，但如果数据集不够大，会有可能导致明显的抽样偏差。我们可以尝试分层抽样。

```
housing["income_cat"] = pd.cut(housing["median_income"],
                               bins=[0.,1.5,3.0,4.5,6.,np.inf],
                               labels=[1,2,3,4,5])

housing["income_cat"].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x22b4dda8910>
```




```
housing["income_cat"].value_counts()
```

```
3    7236
2    6581
4    3639
5    2362
1     822
Name: income_cat, dtype: int64
```

现在就可以根据收入类别进行分层抽样了。

```
from sklearn.model_selection import StratifiedShuffleSplit
#分层随机分割交叉验证器可以将数据分割为训练集和测试集，
#不过它只提供训练集/测试集数据在原始数据集中的位置索引。

split = StratifiedShuffleSplit(n_splits=1,test_size=0.2,random_state=42)
for train_index,test_index in split.split(housing,housing["income_cat"]):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]

strat_train_set["income_cat"].value_counts()/len(strat_train_set)
```

```
3    0.350594
2    0.318859
4    0.176296
5    0.114402
1    0.039850
Name: income_cat, dtype: float64
```

```
housing["income_cat"].value_counts() / len(housing)
```

```
3    0.350581
2    0.318847
4    0.176308
5    0.114438
1    0.039826
Name: income_cat, dtype: float64
```

```
def income_cat_proportions(data):
    return data["income_cat"].value_counts() / len(data)

train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)

compare_props = pd.DataFrame({
    "Overall":income_cat_proportions(housing),
    "Stratified":income_cat_proportions(strat_train_set),
    "Random":income_cat_proportions(test_set),
}).sort_index()

compare_props["Rand. %error"] = 100 * compare_props["Random"] / compare_props["Overall"] - 100
compare_props["Strat. %error"] = 100 * compare_props["Stratified"] / compare_props["Overall"] - 100

compare_props
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	Overall	Stratified	Random	Rand. %error	Strat. %error
1	0.039826	0.039850	0.040213	0.973236	0.060827
2	0.318847	0.318859	0.324370	1.732260	0.003799
3	0.350581	0.350594	0.358527	2.266446	0.003455
4	0.176308	0.176296	0.167393	-5.056334	-0.006870
5	0.114438	0.114402	0.109496	-4.318374	-0.031753

由此可见，分层抽样的测试集中的比例分布与完整数据集中的分布几乎一致，而纯随机抽样的测试集结果则是有偏的。

现在就可以删除 `income_cat` 属性，将数据恢复原样了。

```
for set_ in (strat_train_set, strat_test_set):
    set_.drop("income_cat", axis=1, inplace=True)
```

二、从数据探索和可视化中获得洞见

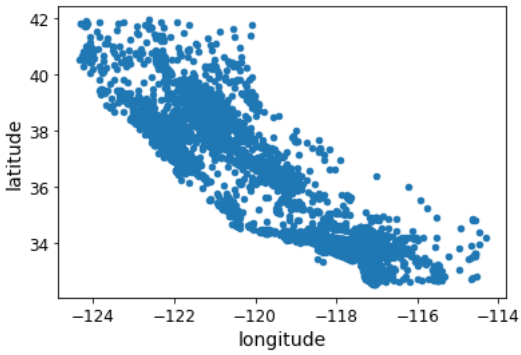
首先创建一个副本，这样可以随便尝试而不损害训练集

```
housing = strat_train_set.copy()
```

1.将地理数据可视化

```
housing.plot(kind="scatter",x="longitude",y="latitude")
```

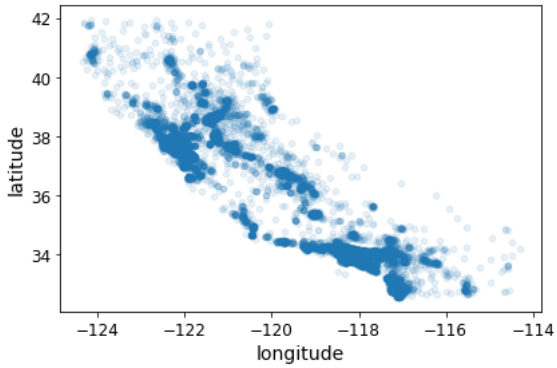
```
<matplotlib.axes._subplots.AxesSubplot at 0x22b4de9cc10>
```



将 `alpha` 设为0.1,可以更清楚地看出高密度数据点的位置

```
housing.plot(kind="scatter",x="longitude",y="latitude",alpha=0.1)
save_fig("better_visualization_plot")
```

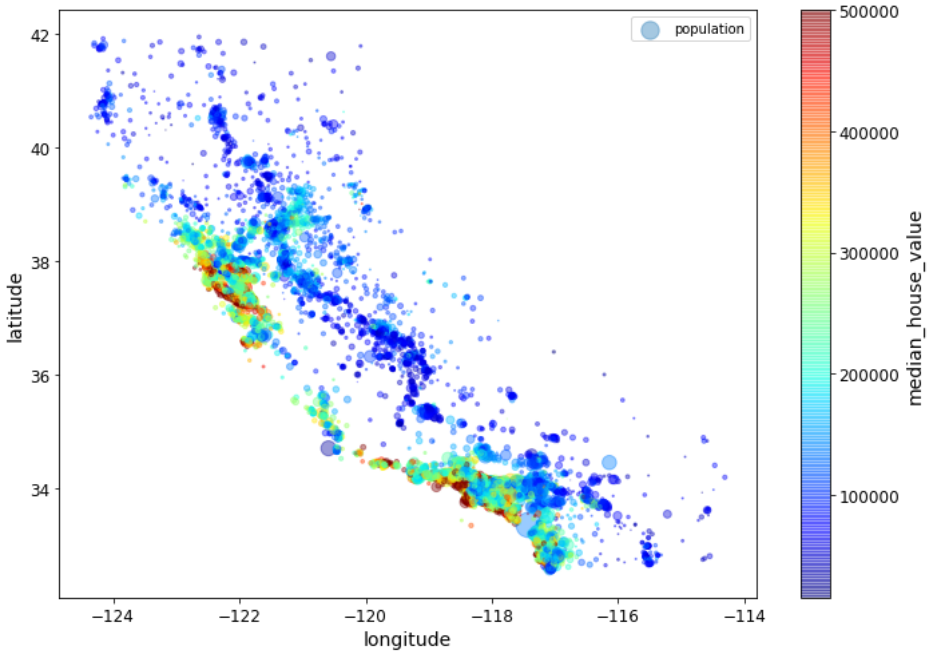
```
Saving figure better_visualization_plot
```



每个圆的半径大小代表了每个区域的人口数量（选项s），颜色代表价格（选项c）。我们使用一个名为jet的预定义颜色表（选项cmap）来进行可视化，颜色范围从蓝（低）到红（高）。

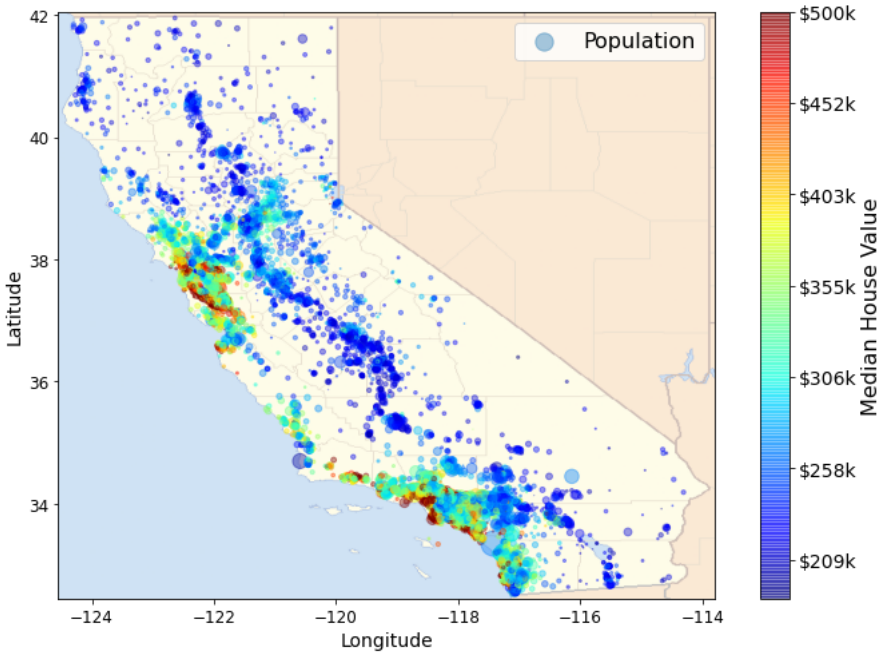
```
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
    s=housing["population"]/100, label="population", figsize=(10,7),
    c="median_house_value", cmap=plt.get_cmap("jet"), colorbar=True,
    sharex=False)
plt.legend()
save_fig("housing_prices_scatterplot")
```

```
Saving figure housing_prices_scatterplot
```




```
import matplotlib.image as mpimg
california_img=mpimg.imread(r"E:\Jupyter\handson-ml2\images\end_to_end_project\california.png")
ax = housing.plot(kind="scatter", x="longitude", y="latitude", figsize=(10,7),
                       s=housing['population']/100, label="Population",
                       c="median_house_value", cmap=plt.get_cmap("jet"),
                       colorbar=False, alpha=0.4,
                      )
plt.imshow(california_img, extent=[-124.55, -113.80, 32.45, 42.05], alpha=0.5,
           cmap=plt.get_cmap("jet"))
plt.ylabel("Latitude", fontsize=14)
plt.xlabel("Longitude", fontsize=14)

prices = housing["median_house_value"]
tick_values = np.linspace(prices.min(), prices.max(), 11)
cbar = plt.colorbar(ticks=tick_values/prices.max())
cbar.ax.set_yticklabels(["$%dk"%(round(v/1000)) for v in tick_values], fontsize=14)
cbar.set_label('Median House Value', fontsize=16)

plt.legend(fontsize=16)
save_fig("california_housing_prices_plot")
plt.show()
```

```
Saving figure california_housing_prices_plot
```



2.寻找相关性

由于数据集不大，可以使用`corr()`方法轻松计算出每对属性之间的标准相关系数

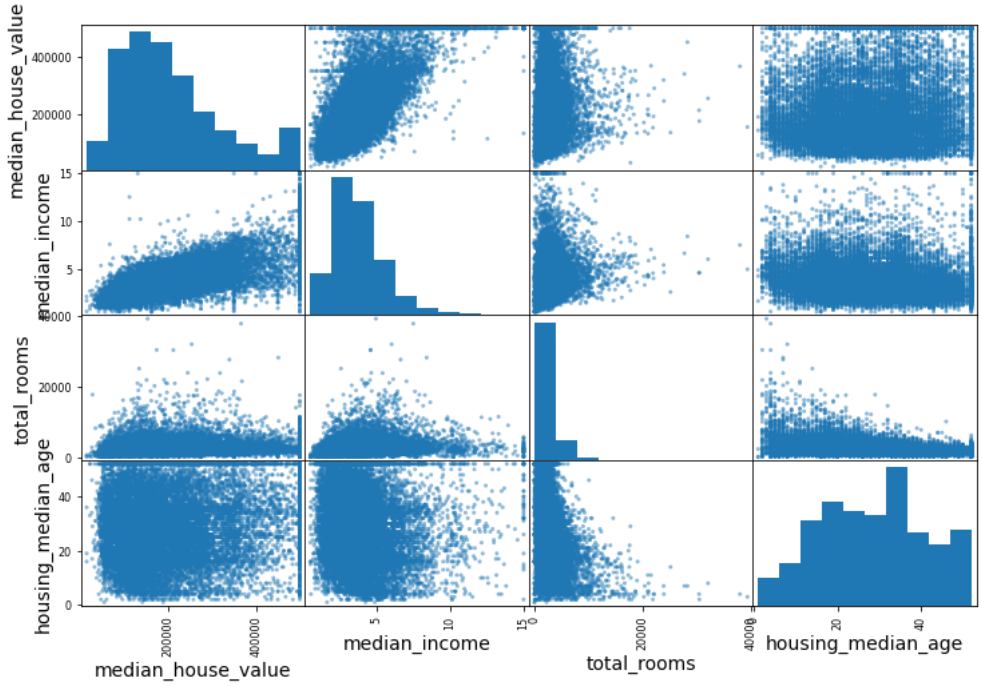
```
corr_matrix = housing.corr()
corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
median_house_value    1.000000
median_income         0.687160
total_rooms           0.135097
housing_median_age    0.114110
households            0.064506
total_bedrooms        0.047689
population           -0.026920
longitude            -0.047432
latitude             -0.142724
Name: median_house_value, dtype: float64
```

相关系数仅测量线性相关性，所以它有可能遗漏非线性相关性。

还有一种方法可以检测属性之间的相关性，就是使用pandas的`scatter__matrix`函数，它会绘制出每个属性相对其他数值属性的相关性。

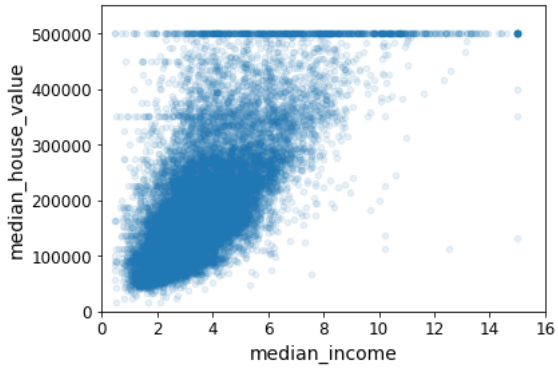
```
from pandas.plotting import scatter_matrix

attributes=['median_house_value',"median_income","total_rooms","housing_median_age"]
scatter_matrix(housing[attributes],figsize=(12,8))
plt.show()
```



最有潜力能够预测房价中位数的属性是收入中位数，所以放大其相关性的散点图。

```
housing.plot(kind="scatter", x="median_income", y="median_house_value",
             alpha=0.1)
plt.axis([0, 16, 0, 550000])
save_fig("income_vs_house_value_scatterplot")
```

```
Saving figure income_vs_house_value_scatterplot
```

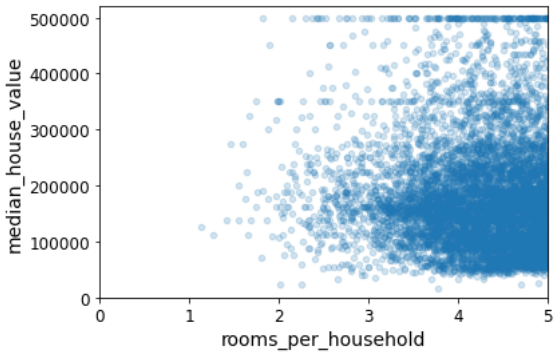


3.试验不同属性的组合

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
#每个家庭房间数
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
#卧室占房间数目比
housing["population_per_household"]=housing["population"]/housing["households"]
#每个家庭人口数

corr_matrix = housing.corr()
corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
median_house_value          1.000000
median_income               0.687160
rooms_per_household         0.146285
total_rooms                 0.135097
housing_median_age          0.114110
households                  0.064506
total_bedrooms              0.047689
population_per_household   -0.021985
population                 -0.026920
longitude                  -0.047432
latitude                   -0.142724
bedrooms_per_room          -0.259984
Name: median_house_value, dtype: float64
```

```
housing.plot(kind="scatter", x="rooms_per_household", y="median_house_value",
             alpha=0.2)
plt.axis([0, 5, 0, 520000])
plt.show()
```




```
housing.describe()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
count	16512.000000	16512.000000	16512.000000	16512.000000	16354.000000	16512.000000	16512.000000	16512.000000
mean	-119.575834	35.639577	28.653101	2622.728319	534.973890	1419.790819	497.060380	3.875589
std	2.001860	2.138058	12.574726	2138.458419	412.699041	1115.686241	375.720845	1.904950
min	-124.350000	32.540000	1.000000	6.000000	2.000000	3.000000	2.000000	0.499900
25%	-121.800000	33.940000	18.000000	1443.000000	295.000000	784.000000	279.000000	2.566775
50%	-118.510000	34.260000	29.000000	2119.500000	433.000000	1164.000000	408.000000	3.540900
75%	-118.010000	37.720000	37.000000	3141.000000	644.000000	1719.250000	602.000000	4.744475
max	-114.310000	41.950000	52.000000	39320.000000	6210.000000	35682.000000	5358.000000	15.000100

三、机器学习算法的数据准备

```
housing = strat_train_set.drop("median_house_value",axis=1)
housing_labels = strat_train_set["median_house_value"].copy()
```

1.数据清洗

```
housing.isnull().any(axis=1)
```

```
17606    False
18632    False
14650    False
3230     False
3555     False
         ...
6563     False
12053    False
13908    False
11159    False
15775    False
Length: 16512, dtype: bool
```

```
sample_imcomplete_rows = housing[housing.isnull().any(axis=1)].head()
#显示哪些列存在缺失值
sample_imcomplete_rows
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_pro
4629	-118.30	34.07	18.0	3759.0	NaN	3296.0	1462.0	2.2708	<1H OCEAN
6068	-117.86	34.01	16.0	4632.0	NaN	3038.0	727.0	5.1762	<1H OCEAN
17923	-121.97	37.35	30.0	1955.0	NaN	999.0	386.0	4.6328	<1H OCEAN
13656	-117.30	34.05	6.0	2155.0	NaN	1039.0	391.0	1.6675	INLAND
19252	-122.79	38.48	7.0	6837.0	NaN	3468.0	1405.0	3.1662	<1H OCEAN

有如下三种方法可以解决部分值缺省：

1.放弃这些相应的区域

2.放弃整个属性

3.将缺省的值设为某个值(0，平均数或中位数等)。

```
sample_imcomplete_rows.dropna(subset=['total_bedrooms'])#option 1
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity

```
sample_imcomplete_rows.drop("total_bedrooms",axis=1)#option 2
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	population	households	median_income	ocean_proximity
4629	-118.30	34.07	18.0	3759.0	3296.0	1462.0	2.2708	<1H OCEAN
6068	-117.86	34.01	16.0	4632.0	3038.0	727.0	5.1762	<1H OCEAN
17923	-121.97	37.35	30.0	1955.0	999.0	386.0	4.6328	<1H OCEAN
13656	-117.30	34.05	6.0	2155.0	1039.0	391.0	1.6675	INLAND
19252	-122.79	38.48	7.0	6837.0	3468.0	1405.0	3.1662	<1H OCEAN

```
median = housing['total_bedrooms'].median()
sample_imcomplete_rows["total_bedrooms"].fillna(median,inplace=True)#option 3
sample_imcomplete_rows
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_prox
4629	-118.30	34.07	18.0	3759.0	433.0	3296.0	1462.0	2.2708	<1H OCEAN
6068	-117.86	34.01	16.0	4632.0	433.0	3038.0	727.0	5.1762	<1H OCEAN
17923	-121.97	37.35	30.0	1955.0	433.0	999.0	386.0	4.6328	<1H OCEAN
13656	-117.30	34.05	6.0	2155.0	433.0	1039.0	391.0	1.6675	INLAND
19252	-122.79	38.48	7.0	6837.0	433.0	3468.0	1405.0	3.1662	<1H OCEAN

Scikit-Learn提供了一个非常容易上手的类来处理缺失值：`SimpleImputer`

```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy='median')
```

由于中位数值只能在数值属性上计算，所以我们要创建一个没有文本属性`ocean_proximity`的数据副本

```
housing_num = housing.drop("ocean_proximity",axis=1)
# alternatively: housing_num = housing.select_dtypes(include=[np.number])

imputer.fit(housing_num)#目前仅仅计算了中位数的值
print(imputer.statistics_)
print(housing_num.median().values)

X = imputer.transform(housing_num)#包含转换后特征的NumPy数组
```

```
[-118.51     34.26     29.     2119.5     433.     1164.      408.
    3.5409]
[-118.51     34.26     29.     2119.5     433.     1164.      408.
    3.5409]
```

```
housing_tr = pd.DataFrame(X,columns=housing_num.columns,index=housing_num.index)
housing_tr.loc[sample_imcomplete_rows.index.values]
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
4629	-118.30	34.07	18.0	3759.0	433.0	3296.0	1462.0	2.2708
6068	-117.86	34.01	16.0	4632.0	433.0	3038.0	727.0	5.1762
17923	-121.97	37.35	30.0	1955.0	433.0	999.0	386.0	4.6328
13656	-117.30	34.05	6.0	2155.0	433.0	1039.0	391.0	1.6675
19252	-122.79	38.48	7.0	6837.0	433.0	3468.0	1405.0	3.1662

```
housing_tr.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
17606	-121.89	37.29	38.0	1568.0	351.0	710.0	339.0	2.7042
18632	-121.93	37.05	14.0	679.0	108.0	306.0	113.0	6.4214
14650	-117.20	32.77	31.0	1952.0	471.0	936.0	462.0	2.8621
3230	-119.61	36.31	25.0	1847.0	371.0	1460.0	353.0	1.8839
3555	-118.59	34.23	17.0	6592.0	1525.0	4459.0	1463.0	3.0347

2.处理文本和分类属性

```
housing_cat = housing[['ocean_proximity']]
housing_cat.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	ocean_proximity
17606	<1H OCEAN
18632	<1H OCEAN
14650	NEAR OCEAN
3230	INLAND
3555	<1H OCEAN

```
housing_cat.value_counts()
```

```
ocean_proximity
<1H OCEAN          7276
INLAND             5263
NEAR OCEAN         2124
NEAR BAY           1847
ISLAND                2
dtype: int64
```

a.OrdinalEncoder

```
from sklearn.preprocessing import OrdinalEncoder

ordinal_encoder = OrdinalEncoder()
housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
housing_cat_encoded[:10]
```

```
array([[0.],
       [0.],
       [4.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.]])
```

```
ordinal_encoder.categories_
```

```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

这会让算法认为两个相近的值比两个离得较远的值更为相似一些。

b.OneHotEncoder

```
from sklearn.preprocessing import OneHotEncoder

cat_encoder = OneHotEncoder()
housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
housing_cat_1hot.toarray()
```

```
array([[1., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       ...,
       [0., 1., 0., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0.]])
```

```
cat_encoder.categories_
```

```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

3.自定义转换器

```
from sklearn.base import BaseEstimator,TransformerMixin

rooms_ix,bedrooms_ix,population_ix,households_ix=3,4,5,6

class CombinedAttributesAdder(BaseEstimator,TransformerMixin):
    def __init__(self,add_bedrooms_per_room = True):#no *args or ** kargs
        self.add_bedrooms_per_room = add_bedrooms_per_room

    def fit(self,X,y=None):
        return self

    def transform(self,X):
        rooms_per_household = X[:, rooms_ix] / X[:, households_ix]
        population_per_household = X[:, population_ix] / X[:, households_ix]
        if self.add_bedrooms_per_room:
```

```
            bedrooms_per_room = X[:, bedrooms_ix] / X[:, rooms_ix]
            return np.c_[X, rooms_per_household, population_per_household,
                         bedrooms_per_room]
        else:
            return np.c_[X, rooms_per_household, population_per_household]

attr_adder = CombinedAttributesAdder(add_bedrooms_per_room=False)
housing_extra_attribs = attr_adder.transform(housing.values)

housing_extra_attribs = pd.DataFrame(
    housing_extra_attribs,
    columns=list(housing.columns)+["rooms_per_household", "population_per_household"],
    index=housing.index)
housing_extra_attribs.head()
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_prox
17606	-121.89	37.29	38	1568	351	710	339	2.7042	<1H OCEAN
18632	-121.93	37.05	14	679	108	306	113	6.4214	<1H OCEAN
14650	-117.2	32.77	31	1952	471	936	462	2.8621	NEAR OCEA
3230	-119.61	36.31	25	1847	371	1460	353	1.8839	INLAND
3555	-118.59	34.23	17	6592	1525	4459	1463	3.0347	<1H OCEAN

4.转换流水线

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

num_pipeline = Pipeline([
    ('imputer',SimpleImputer(strategy='median')),
    ('attribs_adder',CombinedAttributesAdder()),
    ('std_scaler',StandardScaler()),
])

housing_num_tr = num_pipeline.fit_transform(housing_num)
housing_num_tr
```

```
array([[-1.15604281,  0.77194962,  0.74333089, ..., -0.31205452,
        -0.08649871,  0.15531753],
       [-1.17602483,  0.6596948 , -1.1653172 , ...,  0.21768338,
        -0.03353391, -0.83628902],
       [ 1.18684903, -1.34218285,  0.18664186, ..., -0.46531516,
        -0.09240499,  0.4222004 ],
       ...,
       [ 1.58648943, -0.72478134, -1.56295222, ...,  0.3469342 ,
        -0.03055414, -0.52177644],
       [ 0.78221312, -0.85106801,  0.18664186, ...,  0.02499488,
         0.06150916, -0.30340741],
       [-1.43579109,  0.99645926,  1.85670895, ..., -0.22852947,
        -0.09586294,  0.10180567]])
```

目前为止，我们分别处理了类别列和数值列。拥有一个能够处理所有列的转换器会更方便，将适当的转换应用于每个列。

```
from sklearn.compose import ColumnTransformer
num_attribs = list(housing_num)
cat_attribs = ['ocean_proximity']

full_pipeline = ColumnTransformer([
    ("num",num_pipeline,num_attribs),
    ('cat',OneHotEncoder(),cat_attribs)
])


#若我们只想转换数字列并保持其余列不变，可以将代码改成如下:
```

```
"""
full_pipeline = ColumnTransformer([
    ("num",num_pipeline,num_attribs),
],remainder='passthrough')
"""

#若我们只想转换数字列并删除其余列，可以将代码改成如下：
"""
full_pipeline = ColumnTransformer([
    ("num",num_pipeline,num_attribs),
])
"""

housing_prepared = full_pipeline.fit_transform(housing)
print(housing_prepared)
print(housing_prepared.shape)
```

```
[[-1.15604281  0.77194962  0.74333089 ...  0.          0.
   0.        ]
 [-1.17602483  0.6596948  -1.1653172  ...  0.          0.
   0.        ]
 [ 1.18684903 -1.34218285  0.18664186 ...  0.          0.
   1.        ]
 ...
 [ 1.58648943 -0.72478134 -1.56295222 ...  0.          0.
   0.        ]
 [ 0.78221312 -0.85106801  0.18664186 ...  0.          0.
   0.        ]
 [-1.43579109  0.99645926  1.85670895 ...  0.          1.
   0.        ]]
(16512, 16)
```

四、选择和训练模型

1.训练和评估训练集

```
from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()
lin_reg.fit(housing_prepared,housing_labels)
```

```
LinearRegression()
```

```
some_data = housing.iloc[:5]
some_labels = housing_labels.iloc[:5]
some_data_prepared = full_pipeline.transform(some_data)

print("Predictions:", lin_reg.predict(some_data_prepared))
print("Labels:", list(some_labels))
```

```
Predictions: [210644.60459286 317768.80697211 210956.43331178  59218.98886849
 189747.55849879]
Labels: [286600.0, 340600.0, 196900.0, 46300.0, 254500.0]
```

```
from sklearn.metrics import mean_squared_error

housing_predictions = lin_reg.predict(housing_prepared)
lin_mse = mean_squared_error(housing_labels,housing_predictions)
lin_rmse = np.sqrt(lin_mse)
lin_rmse
```

```
68628.19819848922
```

```
from sklearn.metrics import mean_absolute_error

lin_mae = mean_absolute_error(housing_labels, housing_predictions)
lin_mae
```

```
49439.895990018966
```

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(random_state=42)
tree_reg.fit(housing_prepared, housing_labels)
```

```
DecisionTreeRegressor(random_state=42)
```

```
housing_predictions = tree_reg.predict(housing_prepared)
tree_mse = mean_squared_error(housing_labels, housing_predictions)
tree_rmse = np.sqrt(tree_mse)
tree_rmse
```

```
0.0
```

2.使用交叉验证来更好地进行评估

a.决策树

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(tree_reg,housing_prepared,housing_labels,
                         scoring='neg_mean_squared_error',cv=10)
tree_rmse_scores = np.sqrt(-scores)

def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard deviation:", scores.std())

display_scores(tree_rmse_scores)
```

```
Scores: [70194.33680785 66855.16363941 72432.58244769 70758.73896782
 71115.88230639 75585.14172901 70262.86139133 70273.6325285
 75366.87952553 71231.65726027]
Mean: 71407.68766037929
Standard deviation: 2439.4345041191004
```

b.线性回归

```
lin_scores = cross_val_score(lin_reg, housing_prepared, housing_labels,
                             scoring="neg_mean_squared_error", cv=10)
lin_rmse_scores = np.sqrt(-lin_scores)
display_scores(lin_rmse_scores)
```

```
Scores: [66773.97885472 66961.35334671 70348.53369788 74742.90821745
 68031.13388938 71193.84183426 64968.92806931 68281.61137997
 71548.40088036 67665.10082067]
Mean: 69051.57909907165
Standard deviation: 2732.723069207482
```

c.随机森林

```
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators=100, random_state=42)
forest_reg.fit(housing_prepared, housing_labels)

housing_predictions = forest_reg.predict(housing_prepared)
forest_mse = mean_squared_error(housing_labels, housing_predictions)
forest_rmse = np.sqrt(forest_mse)
forest_rmse
```

```
18603.515021376355
```

```
forest_scores = cross_val_score(forest_reg, housing_prepared, housing_labels,
                                scoring="neg_mean_squared_error", cv=10)
forest_rmse_scores = np.sqrt(-forest_scores)
display_scores(forest_rmse_scores)
```

```
Scores: [49519.80364233 47461.9115823  50029.02762854 52325.28068953
 49308.39426421 53446.37892622 48634.8036574  47585.73832311
 53490.10699751 50021.5852922 ]
Mean: 50182.303100336096
Standard deviation: 2097.0810550985693
```

d.支持向量机

```
from sklearn.svm import SVR

svm_reg = SVR(kernel="linear")
svm_reg.fit(housing_prepared, housing_labels)
housing_predictions = svm_reg.predict(housing_prepared)
svm_mse = mean_squared_error(housing_labels, housing_predictions)
svm_rmse = np.sqrt(svm_mse)
svm_rmse
```

```
111094.6308539982
```

```
svm_scores = cross_val_score(svm_reg, housing_prepared, housing_labels,
                                scoring="neg_mean_squared_error", cv=10)
svm_rmse_scores = np.sqrt(-svm_scores)
display_scores(svm_rmse_scores)
```

```
Scores: [105342.09141998 112489.24624123 110092.35042753 113403.22892482
 110638.90119657 115675.8320024  110703.56887243 114476.89008206
 113756.17971227 111520.1120808 ]
Mean: 111809.84009600841
Standard deviation: 2762.393664321567
```

3.保存模型

```
import joblib

joblib.dump(svm_reg,'my_model.pkl')
```

```
['my_model.pkl']
```

五、微调模型

1.网格搜索

```
from sklearn.model_selection import GridSearchCV


param_grid = [
    # try 12 (3×4) combinations of hyperparameters
    {'n_estimators': [3, 10, 30], 'max_features': [2, 4, 6, 8]},
    # then try 6 (2×3) combinations with bootstrap set as False
    {'bootstrap': [False], 'n_estimators': [3, 10], 'max_features': [2, 3, 4]},
  ]

forest_reg = RandomForestRegressor(random_state=42)
# train across 5 folds, that's a total of (12+6)*5=90 rounds of training
grid_search = GridSearchCV(forest_reg, param_grid, cv=5,
                           scoring='neg_mean_squared_error',
                           return_train_score=True)
grid_search.fit(housing_prepared, housing_labels)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegressor(random_state=42),
             param_grid=[{'max_features': [2, 4, 6, 8],
                          'n_estimators': [3, 10, 30]},
                         {'bootstrap': [False], 'max_features': [2, 3, 4],
                          'n_estimators': [3, 10]}],
             return_train_score=True, scoring='neg_mean_squared_error')
```

```
grid_search.best_params_
```

```
{'max_features': 8, 'n_estimators': 30}
```

```
grid_search.best_estimator_
```

```
RandomForestRegressor(max_features=8, n_estimators=30, random_state=42)
```

```
cvres = grid_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
63669.11631261028 {'max_features': 2, 'n_estimators': 3}
55627.099719926795 {'max_features': 2, 'n_estimators': 10}
53384.57275149205 {'max_features': 2, 'n_estimators': 30}
60965.950449450494 {'max_features': 4, 'n_estimators': 3}
52741.04704299915 {'max_features': 4, 'n_estimators': 10}
50377.40461678399 {'max_features': 4, 'n_estimators': 30}
58663.93866579625 {'max_features': 6, 'n_estimators': 3}
52006.19873526564 {'max_features': 6, 'n_estimators': 10}
50146.51167415009 {'max_features': 6, 'n_estimators': 30}
57869.25276169646 {'max_features': 8, 'n_estimators': 3}
51711.127883959234 {'max_features': 8, 'n_estimators': 10}
49682.273345071546 {'max_features': 8, 'n_estimators': 30}
62895.06951262424 {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
54658.176157539405 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
59470.40652318466 {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
52724.9822587892 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
57490.5691951261 {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
51009.495668875716 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
```

```
pd.DataFrame(grid_search.cv_results_)
```

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_features	param_n_estimators	param_bootstrap	
0	0.051135	0.000647	0.002893	4.902012e-04	2	3	NaN	{'max_ 2, 'n_esti 3}
1	0.169416	0.003771	0.007607	4.841330e-04	2	10	NaN	{'max_ 2, 'n_esti 10}
2	0.505160	0.002457	0.021343	4.780561e-04	2	30	NaN	{'max_ 2, 'n_esti 30}
3	0.081666	0.003038	0.002806	3.714818e-04	4	3	NaN	{'max_ 4, 'n_esti 3}
4	0.264048	0.001362	0.007585	4.824197e-04	4	10	NaN	{'max_ 4, 'n_esti 10}
5	0.787531	0.003521	0.020944	5.917394e-07	4	30	NaN	{'max_ 4, 'n_esti 30}
6	0.106178	0.002664	0.002793	3.992319e-04	6	3	NaN	{'max_ 6, 'n_esti 3}
7	0.352271	0.001775	0.007894	1.836714e-04	6	10	NaN	{'max_ 6, 'n_esti 10}
8	1.064749	0.007041	0.021943	6.304534e-04	6	30	NaN	{'max_ 6, 'n_esti 30}
9	0.133162	0.001340	0.002195	3.985194e-04	8	3	NaN	{'max_ 8, 'n_esti 3}
10	0.448987	0.004617	0.007594	4.909205e-04	8	10	NaN	{'max_ 8, 'n_esti 10}
11	1.358745	0.008240	0.021559	8.057724e-04	8	30	NaN	{'max_ 8, 'n_esti 30}
12	0.078891	0.001271	0.002992	9.536743e-08	2	3	False	{'boots False, 'max_f 2, 'n_e
13	0.262122	0.001872	0.008782	4.016270e-04	2	10	False	{'boots False, 'max_f 2, 'n_e
14	0.103066	0.002724	0.003599	4.866179e-04	3	3	False	{'boots False, 'max_f 3, 'n_e
15	0.338739	0.002383	0.008401	4.761108e-04	3	10	False	{'boots False, 'max_f 3, 'n_e

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_features	param_n_estimators	param_bootstrap	
16	0.125841	0.001473	0.003192	3.988505e-04	4	3	False	{'boots False, 'max_f 4, 'n_e
17	0.414752	0.002206	0.008481	4.401121e-04	4	10	False	{'boots False, 'max_f 4, 'n_e

18 rows × 23 columns

2.随机搜索

```
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import randint

param_distribs = {
        'n_estimators': randint(low=1, high=200),
        'max_features': randint(low=1, high=8),
    }

forest_reg = RandomForestRegressor(random_state=42)
rnd_search = RandomizedSearchCV(forest_reg, param_distributions=param_distribs,
                                n_iter=10, cv=5, scoring='neg_mean_squared_error', random_state=42)
rnd_search.fit(housing_prepared, housing_labels)
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(random_state=42),
                   param_distributions={'max_features': <scipy.stats._distn_infrastructure.rv_frozen object at 0x0000022B8870C2E0>,
                                        'n_estimators': <scipy.stats._distn_infrastructure.rv_frozen object at 0x0000022B851CFA90>},
                   random_state=42, scoring='neg_mean_squared_error')
```

```
cvres = rnd_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
print(rnd_search.best_estimator_)
```

```
49150.70756927707 {'max_features': 7, 'n_estimators': 180}
51389.889203389284 {'max_features': 5, 'n_estimators': 15}
50796.155224308866 {'max_features': 3, 'n_estimators': 72}
50835.13360315349 {'max_features': 5, 'n_estimators': 21}
49280.9449827171 {'max_features': 7, 'n_estimators': 122}
50774.90662363929 {'max_features': 3, 'n_estimators': 75}
50682.78888164288 {'max_features': 3, 'n_estimators': 88}
49608.99608105296 {'max_features': 5, 'n_estimators': 100}
50473.61930350219 {'max_features': 3, 'n_estimators': 150}
64429.84143294435 {'max_features': 5, 'n_estimators': 2}
RandomForestRegressor(max_features=7, n_estimators=180, random_state=42)
```

3.分析最佳模型及其误差

```
feature_importances = grid_search.best_estimator_.feature_importances_
feature_importances
```

```
array([7.33442355e-02, 6.29090705e-02, 4.11437985e-02, 1.46726854e-02,
       1.41064835e-02, 1.48742809e-02, 1.42575993e-02, 3.66158981e-01,
       5.64191792e-02, 1.08792957e-01, 5.33510773e-02, 1.03114883e-02,
       1.64780994e-01, 6.02803867e-05, 1.96041560e-03, 2.85647464e-03])
```

```
extra_attribs = ["rooms_per_hhold", "pop_per_hhold", "bedrooms_per_room"]
cat_encoder = full_pipeline.named_transformers_["cat"]
cat_one_hot_attribs = list(cat_encoder.categories_[0])
attributes = num_attribs + extra_attribs + cat_one_hot_attribs
sorted(zip(feature_importances, attributes), reverse=True)
```

```
[(0.36615898061813423, 'median_income'),
 (0.16478099356159054, 'INLAND'),
 (0.10879295677551575, 'pop_per_hhold'),
 (0.07334423551601243, 'longitude'),
 (0.06290907048262032, 'latitude'),
 (0.056419179181954014, 'rooms_per_hhold'),
 (0.053351077347675815, 'bedrooms_per_room'),
 (0.04114379847872964, 'housing_median_age'),
 (0.014874280890402769, 'population'),
 (0.014672685420543239, 'total_rooms'),
 (0.014257599323407808, 'households'),
 (0.014106483453584104, 'total_bedrooms'),
 (0.010311488326303788, '<1H OCEAN'),
 (0.0028564746373201584, 'NEAR OCEAN'),
 (0.0019604155994780706, 'NEAR BAY'),
 (6.0280386727366e-05, 'ISLAND')]
```

4.通过测试集评估系统

```
final_model = grid_search.best_estimator_

X_test = strat_test_set.drop("median_house_value", axis=1)
y_test = strat_test_set["median_house_value"].copy()

X_test_prepared = full_pipeline.transform(X_test)
final_predictions = final_model.predict(X_test_prepared)

final_mse = mean_squared_error(y_test, final_predictions)
final_rmse = np.sqrt(final_mse)

final_rmse
```

```
47730.22690385927
```

```
from scipy import stats

confidence = 0.95
squared_errors = (final_predictions - y_test) ** 2
np.sqrt(stats.t.interval(confidence, len(squared_errors) - 1,
                         loc=squared_errors.mean(),
                         scale=stats.sem(squared_errors)))
```

```
array([45685.10470776, 49691.25001878])
```

```
m = len(squared_errors)
mean = squared_errors.mean()
tscore = stats.t.ppf((1 + confidence) / 2, df=m - 1)
tmargin = tscore * squared_errors.std(ddof=1) / np.sqrt(m)
np.sqrt(mean - tmargin), np.sqrt(mean + tmargin)
```

```
(45685.10470776, 49691.25001877858)
```

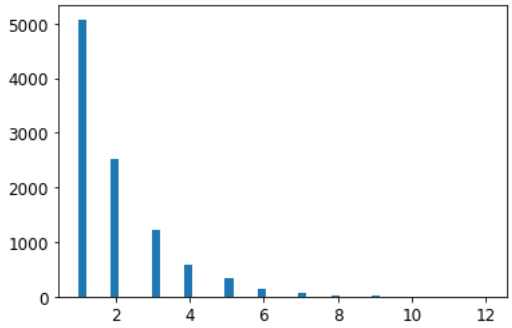
```
zscore = stats.norm.ppf((1 + confidence) / 2)
zmargin = zscore * squared_errors.std(ddof=1) / np.sqrt(m)
np.sqrt(mean - zmargin), np.sqrt(mean + zmargin)
```

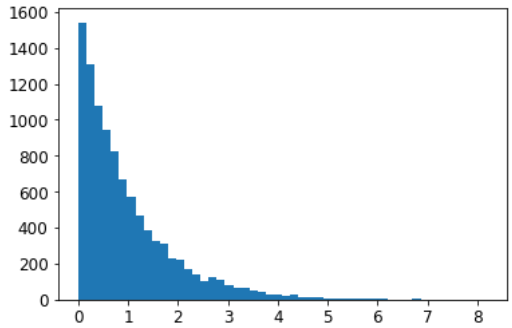
```
(45685.717918136455, 49690.68623889413)
```

六、额外示例

Example SciPy distributions for `RandomizedSearchCV`

```
from scipy.stats import geom, expon
geom_distrib=geom(0.5).rvs(10000, random_state=42)
expon_distrib=expon(scale=1).rvs(10000, random_state=42)
plt.hist(geom_distrib, bins=50)
plt.show()
plt.hist(expon_distrib, bins=50)
plt.show()
```





七、课后习题

1.

使用不同的超参数，如`kernel="linear"`(具有C超参数的多种值)或`kernel="rbf"`(C超参数和gamma超参数的多种值)，尝试一个SVM。

```
from sklearn.model_selection import GridSearchCV

param_grid = [
        {'kernel': ['linear'], 'C': [10., 30., 100., 300., 1000., 3000., 10000., 30000.0]},
        {'kernel': ['rbf'], 'C': [1.0, 3.0, 10., 30., 100., 300., 1000.0],
         'gamma': [0.01, 0.03, 0.1, 0.3, 1.0, 3.0]},
    ]

svm_reg = SVR()
grid_search = GridSearchCV(svm_reg,param_grid,cv=5,scoring='neg_mean_squared_error',verbose=2)
grid_search.fit(housing_prepared,housing_labels)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
[CV] C=10.0, kernel=linear ..........................................
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
[CV] ............................ C=10.0, kernel=linear, total=   4.3s
[CV] C=10.0, kernel=linear ..........................................
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.2s remaining:    0.0s
```

```
[CV] ............................ C=10.0, kernel=linear, total=   4.3s
[CV] C=10.0, kernel=linear ..........................................
[CV] ............................ C=10.0, kernel=linear, total=   4.4s
[CV] C=10.0, kernel=linear ..........................................
[CV] ............................ C=10.0, kernel=linear, total=   4.3s
[CV] C=10.0, kernel=linear ..........................................
[CV] ............................ C=10.0, kernel=linear, total=   4.3s
[CV] C=30.0, kernel=linear ..........................................
[CV] ............................ C=30.0, kernel=linear, total=   4.2s
[CV] C=30.0, kernel=linear ..........................................
```

```
[CV] ........................... C=30.0, kernel=linear, total=   4.2s
[CV] C=30.0, kernel=linear .....................................
[CV] ........................... C=30.0, kernel=linear, total=   4.3s
[CV] C=30.0, kernel=linear .....................................
[CV] ........................... C=30.0, kernel=linear, total=   4.3s
[CV] C=30.0, kernel=linear .....................................
[CV] ........................... C=30.0, kernel=linear, total=   4.2s
[CV] C=100.0, kernel=linear ....................................
[CV] ........................... C=100.0, kernel=linear, total=   4.2s
[CV] C=100.0, kernel=linear ....................................
[CV] ........................... C=100.0, kernel=linear, total=   4.2s
[CV] C=100.0, kernel=linear ....................................
[CV] ........................... C=100.0, kernel=linear, total=   4.3s
[CV] C=100.0, kernel=linear ....................................
[CV] ........................... C=100.0, kernel=linear, total=   4.2s
[CV] C=100.0, kernel=linear ....................................
[CV] ........................... C=100.0, kernel=linear, total=   4.2s
[CV] C=300.0, kernel=linear ....................................
[CV] ........................... C=300.0, kernel=linear, total=   4.3s
[CV] C=300.0, kernel=linear ....................................
[CV] ........................... C=300.0, kernel=linear, total=   4.3s
[CV] C=300.0, kernel=linear ....................................
[CV] ........................... C=300.0, kernel=linear, total=   4.3s
[CV] C=300.0, kernel=linear ....................................
[CV] ........................... C=300.0, kernel=linear, total=   4.3s
[CV] C=300.0, kernel=linear ....................................
[CV] ........................... C=300.0, kernel=linear, total=   4.3s
[CV] C=1000.0, kernel=linear ...................................
[CV] ........................... C=1000.0, kernel=linear, total=   4.4s
[CV] C=1000.0, kernel=linear ...................................
[CV] ........................... C=1000.0, kernel=linear, total=   4.4s
[CV] C=1000.0, kernel=linear ...................................
[CV] ........................... C=1000.0, kernel=linear, total=   4.4s
[CV] C=1000.0, kernel=linear ...................................
[CV] ........................... C=1000.0, kernel=linear, total=   4.5s
[CV] C=1000.0, kernel=linear ...................................
[CV] ........................... C=1000.0, kernel=linear, total=   4.4s
[CV] C=3000.0, kernel=linear ...................................
[CV] ........................... C=3000.0, kernel=linear, total=   4.8s
[CV] C=3000.0, kernel=linear ...................................
[CV] ........................... C=3000.0, kernel=linear, total=   4.7s
[CV] C=3000.0, kernel=linear ...................................
[CV] ........................... C=3000.0, kernel=linear, total=   4.8s
[CV] C=3000.0, kernel=linear ...................................
[CV] ........................... C=3000.0, kernel=linear, total=   4.8s
[CV] C=3000.0, kernel=linear ...................................
[CV] ........................... C=3000.0, kernel=linear, total=   4.7s
[CV] C=10000.0, kernel=linear ..................................
[CV] ........................... C=10000.0, kernel=linear, total=   6.3s
[CV] C=10000.0, kernel=linear ..................................
[CV] ........................... C=10000.0, kernel=linear, total=   6.4s
[CV] C=10000.0, kernel=linear ..................................
[CV] ........................... C=10000.0, kernel=linear, total=   6.5s
[CV] C=10000.0, kernel=linear ..................................
[CV] ........................... C=10000.0, kernel=linear, total=   5.9s
[CV] C=10000.0, kernel=linear ..................................
[CV] ........................... C=10000.0, kernel=linear, total=   5.8s
[CV] C=30000.0, kernel=linear ..................................
[CV] ........................... C=30000.0, kernel=linear, total=  10.0s
[CV] C=30000.0, kernel=linear ..................................
[CV] ........................... C=30000.0, kernel=linear, total=  10.2s
[CV] C=30000.0, kernel=linear ..................................
[CV] ........................... C=30000.0, kernel=linear, total=  10.7s
[CV] C=30000.0, kernel=linear ..................................
[CV] ........................... C=30000.0, kernel=linear, total=  10.3s
[CV] C=30000.0, kernel=linear ..................................
[CV] ........................... C=30000.0, kernel=linear, total=   9.2s
[CV] C=1.0, gamma=0.01, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=1.0, gamma=0.01, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=1.0, gamma=0.01, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.01, kernel=rbf, total=   7.4s
[CV] C=1.0, gamma=0.01, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=1.0, gamma=0.01, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=1.0, gamma=0.03, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=1.0, gamma=0.03, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=1.0, gamma=0.03, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=1.0, gamma=0.03, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.03, kernel=rbf, total=   7.4s
[CV] C=1.0, gamma=0.03, kernel=rbf .............................
[CV] ..................... C=1.0, gamma=0.03, kernel=rbf, total=   7.7s
```

```
[CV] C=1.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=1.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=1.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=1.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=1.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.1, kernel=rbf, total=   7.3s
[CV] C=1.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=1.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=1.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=1.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=1.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=1.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=1.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=1.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=1.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=1.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=1.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=1.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=3.0, kernel=rbf, total=   7.6s
[CV] C=1.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=3.0, kernel=rbf, total=   7.6s
[CV] C=1.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=1.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=1.0, gamma=3.0, kernel=rbf, total=   7.6s
[CV] C=3.0, gamma=0.01, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.01, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.01, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.01, kernel=rbf, total=   7.4s
[CV] C=3.0, gamma=0.01, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.01, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.03, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.03, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.03, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.03, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.03, kernel=rbf ...................................
[CV] .................... C=3.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=3.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=3.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=3.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=3.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=3.0, gamma=0.1, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=3.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=3.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=3.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=3.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=3.0, gamma=0.3, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=3.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=3.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=3.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=3.0, gamma=1.0, kernel=rbf ....................................
```

```
[CV] ..................... C=3.0, gamma=1.0, kernel=rbf, total=   6.9s
[CV] C=3.0, gamma=1.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=3.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=3.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=3.0, kernel=rbf, total=   7.6s
[CV] C=3.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=3.0, kernel=rbf, total=   7.6s
[CV] C=3.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=3.0, gamma=3.0, kernel=rbf ....................................
[CV] ..................... C=3.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=10.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.01, kernel=rbf, total=   7.4s
[CV] C=10.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=10.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=10.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.01, kernel=rbf, total=   7.2s
[CV] C=10.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.01, kernel=rbf, total=   7.2s
[CV] C=10.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.03, kernel=rbf, total=   7.2s
[CV] C=10.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.03, kernel=rbf, total=   7.2s
[CV] C=10.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=10.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.03, kernel=rbf, total=   7.2s
[CV] C=10.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=10.0, gamma=0.03, kernel=rbf, total=   7.2s
[CV] C=10.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.1, kernel=rbf, total=   7.1s
[CV] C=10.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.1, kernel=rbf, total=   7.1s
[CV] C=10.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.1, kernel=rbf, total=   7.1s
[CV] C=10.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=10.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.1, kernel=rbf, total=   7.2s
[CV] C=10.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=10.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=10.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=10.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.3, kernel=rbf, total=   7.1s
[CV] C=10.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=10.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=10.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=10.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=10.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=10.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=10.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=10.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=10.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=10.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=10.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=10.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=30.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.01, kernel=rbf, total=   7.2s
[CV] C=30.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=30.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.01, kernel=rbf, total=   7.2s
[CV] C=30.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=30.0, gamma=0.01, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.01, kernel=rbf, total=   7.3s
[CV] C=30.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.03, kernel=rbf, total=   7.3s
[CV] C=30.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.03, kernel=rbf, total=   7.0s
```

```
[CV] C=30.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.03, kernel=rbf, total=   7.1s
[CV] C=30.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.03, kernel=rbf, total=   6.9s
[CV] C=30.0, gamma=0.03, kernel=rbf ..................................
[CV] ................... C=30.0, gamma=0.03, kernel=rbf, total=   7.0s
[CV] C=30.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.1, kernel=rbf, total=   6.9s
[CV] C=30.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.1, kernel=rbf, total=   6.9s
[CV] C=30.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.1, kernel=rbf, total=   6.9s
[CV] C=30.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.1, kernel=rbf, total=   6.9s
[CV] C=30.0, gamma=0.1, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.1, kernel=rbf, total=   6.9s
[CV] C=30.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.3, kernel=rbf, total=   6.7s
[CV] C=30.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.3, kernel=rbf, total=   6.7s
[CV] C=30.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.3, kernel=rbf, total=   6.7s
[CV] C=30.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.3, kernel=rbf, total=   6.7s
[CV] C=30.0, gamma=0.3, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=0.3, kernel=rbf, total=   6.7s
[CV] C=30.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=1.0, kernel=rbf, total=   6.5s
[CV] C=30.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=1.0, kernel=rbf, total=   6.6s
[CV] C=30.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=1.0, kernel=rbf, total=   6.6s
[CV] C=30.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=1.0, kernel=rbf, total=   6.6s
[CV] C=30.0, gamma=1.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=1.0, kernel=rbf, total=   6.5s
[CV] C=30.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=3.0, kernel=rbf, total=   7.2s
[CV] C=30.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=3.0, kernel=rbf, total=   7.3s
[CV] C=30.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=3.0, kernel=rbf, total=   7.3s
[CV] C=30.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=3.0, kernel=rbf, total=   7.3s
[CV] C=30.0, gamma=3.0, kernel=rbf ...................................
[CV] .................... C=30.0, gamma=3.0, kernel=rbf, total=   7.3s
[CV] C=100.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.01, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.01, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.01, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.01, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.01, kernel=rbf, total=   7.1s
[CV] C=100.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.03, kernel=rbf, total=   7.1s
[CV] C=100.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.03, kernel=rbf, total=   7.1s
[CV] C=100.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.03, kernel=rbf, total=   7.1s
[CV] C=100.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.03, kernel=rbf, total=   7.1s
[CV] C=100.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=100.0, gamma=0.03, kernel=rbf, total=   7.1s
[CV] C=100.0, gamma=0.1, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.1, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.1, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.1, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.1, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.1, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.1, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.1, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.1, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.1, kernel=rbf, total=   7.4s
[CV] C=100.0, gamma=0.3, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.3, kernel=rbf, total=   7.2s
[CV] C=100.0, gamma=0.3, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=100.0, gamma=0.3, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.3, kernel=rbf, total=   6.9s
[CV] C=100.0, gamma=0.3, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.3, kernel=rbf, total=   6.9s
[CV] C=100.0, gamma=0.3, kernel=rbf ..................................
[CV] ................... C=100.0, gamma=0.3, kernel=rbf, total=   6.9s
[CV] C=100.0, gamma=1.0, kernel=rbf ..................................
```

```
[CV] .................. C=100.0, gamma=1.0, kernel=rbf, total=   6.7s
[CV] C=100.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=100.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=1.0, kernel=rbf, total=   6.7s
[CV] C=100.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=1.0, kernel=rbf, total=   6.7s
[CV] C=100.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=100.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=100.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=100.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=100.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=100.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=100.0, gamma=3.0, kernel=rbf, total=   8.5s
[CV] C=300.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.01, kernel=rbf, total=   7.2s
[CV] C=300.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.01, kernel=rbf, total=   7.2s
[CV] C=300.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.01, kernel=rbf, total=   7.1s
[CV] C=300.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.01, kernel=rbf, total=   7.1s
[CV] C=300.0, gamma=0.01, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.01, kernel=rbf, total=   7.2s
[CV] C=300.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.03, kernel=rbf, total=   7.0s
[CV] C=300.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.03, kernel=rbf, total=   7.0s
[CV] C=300.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.03, kernel=rbf, total=   6.9s
[CV] C=300.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.03, kernel=rbf, total=   7.0s
[CV] C=300.0, gamma=0.03, kernel=rbf .................................
[CV] .................. C=300.0, gamma=0.03, kernel=rbf, total=   6.9s
[CV] C=300.0, gamma=0.1, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.1, kernel=rbf, total=   6.9s
[CV] C=300.0, gamma=0.1, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.1, kernel=rbf, total=   6.8s
[CV] C=300.0, gamma=0.1, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.1, kernel=rbf, total=   6.9s
[CV] C=300.0, gamma=0.1, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.1, kernel=rbf, total=   6.8s
[CV] C=300.0, gamma=0.1, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.1, kernel=rbf, total=   6.8s
[CV] C=300.0, gamma=0.3, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.3, kernel=rbf, total=   6.8s
[CV] C=300.0, gamma=0.3, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=300.0, gamma=0.3, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.3, kernel=rbf, total=   7.0s
[CV] C=300.0, gamma=0.3, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.3, kernel=rbf, total=   7.5s
[CV] C=300.0, gamma=0.3, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=0.3, kernel=rbf, total=   6.8s
[CV] C=300.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=1.0, kernel=rbf, total=   6.8s
[CV] C=300.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=1.0, kernel=rbf, total=   6.6s
[CV] C=300.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=1.0, kernel=rbf, total=   7.8s
[CV] C=300.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=1.0, kernel=rbf, total=   6.9s
[CV] C=300.0, gamma=1.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=1.0, kernel=rbf, total=   6.7s
[CV] C=300.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=300.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=300.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=300.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=3.0, kernel=rbf, total=   7.9s
[CV] C=300.0, gamma=3.0, kernel=rbf ..................................
[CV] .................. C=300.0, gamma=3.0, kernel=rbf, total=   7.7s
[CV] C=1000.0, gamma=0.01, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.01, kernel=rbf, total=   6.8s
[CV] C=1000.0, gamma=0.01, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.01, kernel=rbf, total=   6.8s
[CV] C=1000.0, gamma=0.01, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.01, kernel=rbf, total=   6.8s
[CV] C=1000.0, gamma=0.01, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.01, kernel=rbf, total=   6.8s
```

```
[CV] C=1000.0, gamma=0.01, kernel=rbf ...............................
[CV] ................ C=1000.0, gamma=0.01, kernel=rbf, total=   6.8s
[CV] C=1000.0, gamma=0.03, kernel=rbf ...............................
[CV] ................ C=1000.0, gamma=0.03, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.03, kernel=rbf ...............................
[CV] ................ C=1000.0, gamma=0.03, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.03, kernel=rbf ...............................
[CV] ................ C=1000.0, gamma=0.03, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.03, kernel=rbf ...............................
[CV] ................ C=1000.0, gamma=0.03, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.03, kernel=rbf ...............................
[CV] ................ C=1000.0, gamma=0.03, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.1, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.1, kernel=rbf, total=   6.6s
[CV] C=1000.0, gamma=0.1, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.1, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.1, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.1, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.1, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.1, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.1, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.1, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.3, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.3, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.3, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.3, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.3, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.3, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=0.3, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.3, kernel=rbf, total=   6.6s
[CV] C=1000.0, gamma=0.3, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=0.3, kernel=rbf, total=   6.6s
[CV] C=1000.0, gamma=1.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=1.0, kernel=rbf, total=   6.6s
[CV] C=1000.0, gamma=1.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=1.0, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=1.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=1.0, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=1.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=1.0, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=1.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=1.0, kernel=rbf, total=   6.7s
[CV] C=1000.0, gamma=3.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=1000.0, gamma=3.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=3.0, kernel=rbf, total=   7.5s
[CV] C=1000.0, gamma=3.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=3.0, kernel=rbf, total=   7.4s
[CV] C=1000.0, gamma=3.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=3.0, kernel=rbf, total=   7.4s
[CV] C=1000.0, gamma=3.0, kernel=rbf ................................
[CV] ................. C=1000.0, gamma=3.0, kernel=rbf, total=   7.4s
```

```
[Parallel(n_jobs=1)]: Done 250 out of 250 | elapsed: 28.3min finished
```

```
GridSearchCV(cv=5, estimator=SVR(),
             param_grid=[{'C': [10.0, 30.0, 100.0, 300.0, 1000.0, 3000.0,
                                10000.0, 30000.0],
                          'kernel': ['linear']},
                         {'C': [1.0, 3.0, 10.0, 30.0, 100.0, 300.0, 1000.0],
                          'gamma': [0.01, 0.03, 0.1, 0.3, 1.0, 3.0],
                          'kernel': ['rbf']}],
             scoring='neg_mean_squared_error', verbose=2)
```

```
negative_mse = grid_search.best_score_
rmse = np.sqrt(-negative_mse)
rmse
```

```
70363.84006944533
```

```
grid_search.best_params_
```

```
{'C': 30000.0, 'kernel': 'linear'}
```

2.

尝试用 `RandomizedSearchCV` 替换 `GridSearchCV`

3.

尝试在准备流水线中添加一个转换器，从而只选出最重要的属性。

```
from sklearn.base import BaseEstimator, TransformerMixin

def indices_of_top_k(arr,k):
    return np.sort(np.argpartition(np.array(arr),-k)[-k:])

class TopFeatureSelector(BaseEstimator, TransformerMixin):
    def __init__(self, feature_importances, k):
        self.feature_importances = feature_importances
        self.k = k
    def fit(self, X, y=None):
        self.feature_indices_ = indices_of_top_k(self.feature_importances, self.k)
        return self
    def transform(self, X):
        return X[:, self.feature_indices_]
```

```
k = 5

top_k_feature_indices = indices_of_top_k(feature_importances, k)
top_k_feature_indices
```

```
array([ 0,  1,  7,  9, 12], dtype=int64)
```

```
np.array(attributes)[top_k_feature_indices]
```

```
array(['longitude', 'latitude', 'median_income', 'pop_per_hhold',
       'INLAND'], dtype='<U18')
```

```
sorted(zip(feature_importances, attributes), reverse=True)[:k]
```

```
[(0.36615898061813423, 'median_income'),
 (0.16478099356159054, 'INLAND'),
 (0.10879295677551575, 'pop_per_hhold'),
 (0.07334423551601243, 'longitude'),
 (0.06290907048262032, 'latitude')]
```

```
preparation_and_feature_selection_pipeline = Pipeline([
    ('preparation', full_pipeline),
    ('feature_selection', TopFeatureSelector(feature_importances, k))
])
```

```
housing_prepared_top_k_features = preparation_and_feature_selection_pipeline.fit_transform(housing)
housing_prepared_top_k_features[0:3]
```

```
array([[-1.15604281,  0.77194962, -0.61493744, -0.08649871,  0.        ],
       [-1.17602483,  0.6596948 ,  1.33645936, -0.03353391,  0.        ],
       [ 1.18684903, -1.34218285, -0.5320456 , -0.09240499,  0.        ]])
```

```
housing_prepared[0:3, top_k_feature_indices]
```

```
array([[-1.15604281,  0.77194962, -0.61493744, -0.08649871,  0.        ],
       [-1.17602483,  0.6596948 ,  1.33645936, -0.03353391,  0.        ],
       [ 1.18684903, -1.34218285, -0.5320456 , -0.09240499,  0.        ]])
```

4.

尝试创建一个覆盖完整的数据准备和最终预测的流水线

```
prepare_select_and_predict_pipeline = Pipeline([
    ('preparation', full_pipeline),
    ('feature_selection', TopFeatureSelector(feature_importances, k)),
    ('svm_reg', SVR())
])
```

```
prepare_select_and_predict_pipeline.fit(housing, housing_labels)
```

```
Pipeline(steps=[('preparation',
                 ColumnTransformer(transformers=[('num',
                                                  Pipeline(steps=[('imputer',
                                                                   SimpleImputer(strategy='median')),
                                                                  ('attribs_adder',
                                                                   CombinedAttributesAdder()),
                                                                  ('std_scaler',
                                                                   StandardScaler())]),
                                                  ['longitude', 'latitude',
                                                   'housing_median_age',
                                                   'total_rooms',
                                                   'total_bedrooms',
                                                   'population', 'households',
                                                   'median_income']),
                                                 ('cat', OneHotEncoder(),
                                                  ['ocean_proximity'])])),
                ('feature_selection',
                 TopFeatureSelector(feature_importances=array([7.33442355e-02, 6.29090705e-02, 4.11437985e-02, 1.46726854e-02,
       1.41064835e-02, 1.48742809e-02, 1.42575993e-02, 3.66158981e-01,
       5.64191792e-02, 1.08792957e-01, 5.33510773e-02, 1.03114883e-02,
       1.64780994e-01, 6.02803867e-05, 1.96041560e-03, 2.85647464e-03]),
                                    k=5)),
                ('svm_reg', SVR())])
```

```
some_data = housing.iloc[:4]
some_labels = housing_labels.iloc[:4]

print("Predictions:\t", prepare_select_and_predict_pipeline.predict(some_data))
print("Labels:\t\t", list(some_labels))
```

```
Predictions:     [179092.11181607 180910.02890706 179209.8619433  177815.51957279]
Labels:      [286600.0, 340600.0, 196900.0, 46300.0]
```

5.

使用`GridSearchCV`自动探索一些准备选项

```
param_grid = [{
    'preparation__num__imputer__strategy': ['mean', 'median', 'most_frequent'],
    'feature_selection__k': list(range(1, len(feature_importances) + 1))
}]

grid_search_prep = GridSearchCV(prepare_select_and_predict_pipeline, param_grid, cv=5,
                                scoring='neg_mean_squared_error', verbose=2)
grid_search_prep.fit(housing, housing_labels)
```

```
Fitting 5 folds for each of 48 candidates, totalling 240 fits
[CV] feature_selection__k=1, preparation__num__imputer__strategy=mean
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
```

```
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=mean, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=mean
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.9s remaining:    0.0s
```

```
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=mean, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=mean, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=mean, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=mean, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=median, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=median, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=median, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=median, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=median, total=   5.1s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=most_frequent, total=   5.2s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=most_frequent, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=most_frequent, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=most_frequent, total=   5.0s
[CV] feature_selection__k=1, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=1, preparation__num__imputer__strategy=most_frequent, total=   5.0s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=mean, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=mean, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=mean, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=mean, total=   5.4s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=mean, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=median, total=   5.4s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=median, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=median, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=median, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=median, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=most_frequent, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=most_frequent, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=most_frequent, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=most_frequent, total=   5.3s
[CV] feature_selection__k=2, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=2, preparation__num__imputer__strategy=most_frequent, total=   5.3s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=mean, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=mean, total=   5.5s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=mean, total=   5.5s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=mean, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=mean, total=   5.7s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=median, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=median, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=median, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=median, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=median, total=   5.5s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=most_frequent, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=most_frequent
```

```
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=most_frequent, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=most_frequent, total=   5.7s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=most_frequent, total=   5.6s
[CV] feature_selection__k=3, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=3, preparation__num__imputer__strategy=most_frequent, total=   5.6s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=mean, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=mean, total=   5.6s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=mean, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=mean, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=mean, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=median, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=median, total=   5.6s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=median, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=median, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=median, total=   5.6s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=most_frequent, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=most_frequent, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=most_frequent, total=   5.6s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=most_frequent, total=   5.7s
[CV] feature_selection__k=4, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=4, preparation__num__imputer__strategy=most_frequent, total=   7.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=mean, total=   7.0s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=mean, total=   6.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=median, total=   6.4s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=median, total=   6.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=median, total=   6.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=median, total=   6.6s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=median, total=   6.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=most_frequent, total=   6.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=most_frequent, total=   6.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=most_frequent, total=   6.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=most_frequent, total=   6.5s
[CV] feature_selection__k=5, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=5, preparation__num__imputer__strategy=most_frequent, total=   6.4s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=mean, total=   6.6s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=median, total=   6.5s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=median, total=   6.1s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=median, total=   6.1s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=median, total=   6.0s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=median, total=   5.9s
```

```
[CV] feature_selection__k=6, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=most_frequent, total=   5.9s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=most_frequent, total=   6.2s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=most_frequent, total=   6.4s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=most_frequent, total=   6.0s
[CV] feature_selection__k=6, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=6, preparation__num__imputer__strategy=most_frequent, total=   7.5s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=mean, total=   7.0s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=mean, total=   6.7s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=mean, total=   6.5s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=median, total=   6.7s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=median, total=   6.6s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=median, total=   6.7s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=median, total=   6.6s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=median, total=   6.7s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=most_frequent, total=   6.8s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=most_frequent, total=   6.9s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=most_frequent, total=   6.7s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=7, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=7, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=mean, total=   6.8s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=median, total=   6.3s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=median, total=   6.4s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=median, total=   6.5s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=median, total=   6.2s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=median, total=   6.0s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=most_frequent, total=   6.1s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=most_frequent, total=   6.2s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=most_frequent, total=   6.2s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=most_frequent, total=   6.2s
[CV] feature_selection__k=8, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=8, preparation__num__imputer__strategy=most_frequent, total=   6.3s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=mean, total=   6.8s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=mean, total=   7.0s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=median, total=   7.0s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=median, total=   6.9s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=median, total=   7.0s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=median
```

```
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=median, total=   7.0s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=median, total=   7.0s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=most_frequent, total=   7.1s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=most_frequent, total=   7.1s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=most_frequent, total=   7.1s
[CV] feature_selection__k=9, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=9, preparation__num__imputer__strategy=most_frequent, total=   7.4s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=mean, total=   7.6s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=mean, total=   7.3s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=mean, total=   7.3s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=mean, total=   7.5s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=mean, total=   7.3s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=median, total=   7.4s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=median, total=   7.2s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=median, total=   6.6s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=median, total=   6.6s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=median, total=   6.8s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=most_frequent, total=   6.6s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=most_frequent, total=   6.6s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=most_frequent, total=   6.7s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=most_frequent, total=   6.6s
[CV] feature_selection__k=10, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=10, preparation__num__imputer__strategy=most_frequent, total=   6.2s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=mean, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=median, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=median, total=   6.5s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=median, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=median, total=   6.4s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=median, total=   6.4s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=most_frequent, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=most_frequent, total=   6.2s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=most_frequent, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=most_frequent, total=   6.3s
[CV] feature_selection__k=11, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=11, preparation__num__imputer__strategy=most_frequent, total=   6.3s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=mean, total=   6.5s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=mean, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=mean, total=   6.5s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=median, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=median, total=   6.4s
```

```
[CV] feature_selection__k=12, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=median, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=median, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=median, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=most_frequent, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=most_frequent, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=most_frequent, total=   6.4s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=most_frequent, total=   6.6s
[CV] feature_selection__k=12, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=12, preparation__num__imputer__strategy=most_frequent, total=   6.4s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=mean, total=   6.7s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=mean, total=   6.8s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=mean, total=   7.5s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=mean, total=   8.7s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=mean, total=   8.8s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=median, total=   7.5s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=median, total=   8.3s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=median, total=   6.8s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=median, total=   6.8s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=median, total=   6.7s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=most_frequent, total=   6.8s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=most_frequent, total=   6.9s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=most_frequent, total=   9.2s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=most_frequent, total=   8.0s
[CV] feature_selection__k=13, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=13, preparation__num__imputer__strategy=most_frequent, total=   7.4s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=mean, total=   7.7s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=mean, total=   7.2s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=mean, total=   7.2s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=mean, total=   7.1s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=mean, total=   7.1s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=median, total=   7.1s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=median, total=   7.1s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=median, total=   7.0s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=median, total=   7.1s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=median, total=   7.1s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=most_frequent, total=   7.4s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=most_frequent, total=   7.1s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=14, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=14, preparation__num__imputer__strategy=most_frequent, total=   7.4s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=mean, total=   7.6s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=mean, total=   7.4s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=mean, total=   7.2s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=mean, total=   7.2s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=mean, total=   7.2s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=median
```

```
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=median, total=   7.5s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=median, total=   7.4s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=median, total=   7.3s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=median, total=   7.2s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=median, total=   7.3s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=most_frequent, total=   7.2s
[CV] feature_selection__k=15, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=15, preparation__num__imputer__strategy=most_frequent, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=mean, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=mean, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=mean, total=   7.3s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=mean, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=mean
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=mean, total=   7.3s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=median, total=   7.5s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=median, total=   7.6s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=median, total=   7.5s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=median, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=median
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=median, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=most_frequent, total=   7.5s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=most_frequent, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=most_frequent, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=most_frequent, total=   7.4s
[CV] feature_selection__k=16, preparation__num__imputer__strategy=most_frequent
[CV]  feature_selection__k=16, preparation__num__imputer__strategy=most_frequent, total=   7.4s
```

```
[Parallel(n_jobs=1)]: Done 240 out of 240 | elapsed: 25.9min finished
```

```
GridSearchCV(cv=5,
             estimator=Pipeline(steps=[('preparation',
                                        ColumnTransformer(transformers=[('num',
                                                                         Pipeline(steps=[('imputer',
                                                                                          SimpleImputer(strategy='median')),
                                                                                         ('attribs_adder',
                                                                                          CombinedAttributesAdder()),
                                                                                         ('std_scaler',
                                                                                          StandardScaler())]),
                                                                         ['longitude',
                                                                          'latitude',
                                                                          'housing_median_age',
                                                                          'total_rooms',
                                                                          'total_bedrooms',
                                                                          'population',
                                                                          'households',
                                                                          'median_inc...
       1.41064835e-02, 1.48742809e-02, 1.42575993e-02, 3.66158981e-01,
       5.64191792e-02, 1.08792957e-01, 5.33510773e-02, 1.03114883e-02,
       1.64780994e-01, 6.02803867e-05, 1.96041560e-03, 2.85647464e-03]),
                                                                  k=5)),
                                       ('svm_reg', SVR())]),
             param_grid=[{'feature_selection__k': [1, 2, 3, 4, 5, 6, 7, 8, 9,
                                                   10, 11, 12, 13, 14, 15, 16],
                          'preparation__num__imputer__strategy': ['mean',
                                                                  'median',
                                                                  'most_frequent']}],
             scoring='neg_mean_squared_error', verbose=2)
```

```
grid_search_prep.best_params_
```

```
{'feature_selection__k': 3, 'preparation__num__imputer__strategy': 'mean'}
```