In [131]:

```
pwd
```

Out[131]:

```
'/home/ajith'
```

In [130]:

```
import os
import tarfile
from six.moves import urllib

DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml/master/"
HOUSING_PATH = os.path.join("datasets", "housing")
HOUSING_URL = DOWNLOAD_ROOT + "datasets/housing/housing.tgz"

def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    if not os.path.isdir(housing_path):
        os.makedirs(housing_path)
    tgz_path = os.path.join(housing_path, "housing.tgz")
    urllib.request.urlretrieve(housing_url, tgz_path)
    housing_tgz = tarfile.open(tgz_path)
    housing_tgz.extractall(path=housing_path)
    housing_tgz.close()
```

In [3]:

```
fetch_housing_data()
```

In [4]:

```
import pandas as pd

def load_housing_data(housing_path=HOUSING_PATH):
    csv_path = os.path.join(housing_path, "housing.csv")
    return pd.read_csv(csv_path)
```

In [7]:

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

Out[7]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	media
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200

In [8]:

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
longitude             20640 non-null float64
latitude              20640 non-null float64
housing_median_age    20640 non-null float64
```

```
total_rooms           20640 non-null float64
total_bedrooms        20433 non-null float64
population            20640 non-null float64
households            20640 non-null float64
median_income         20640 non-null float64
median_house_value    20640 non-null float64
ocean_proximity       20640 non-null object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

In [9]:

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

Out[9]:

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

In [10]:

```
housing.describe()
```

Out[10]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	med
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	206
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.87
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.89
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.49
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.56
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.53
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.74
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.0

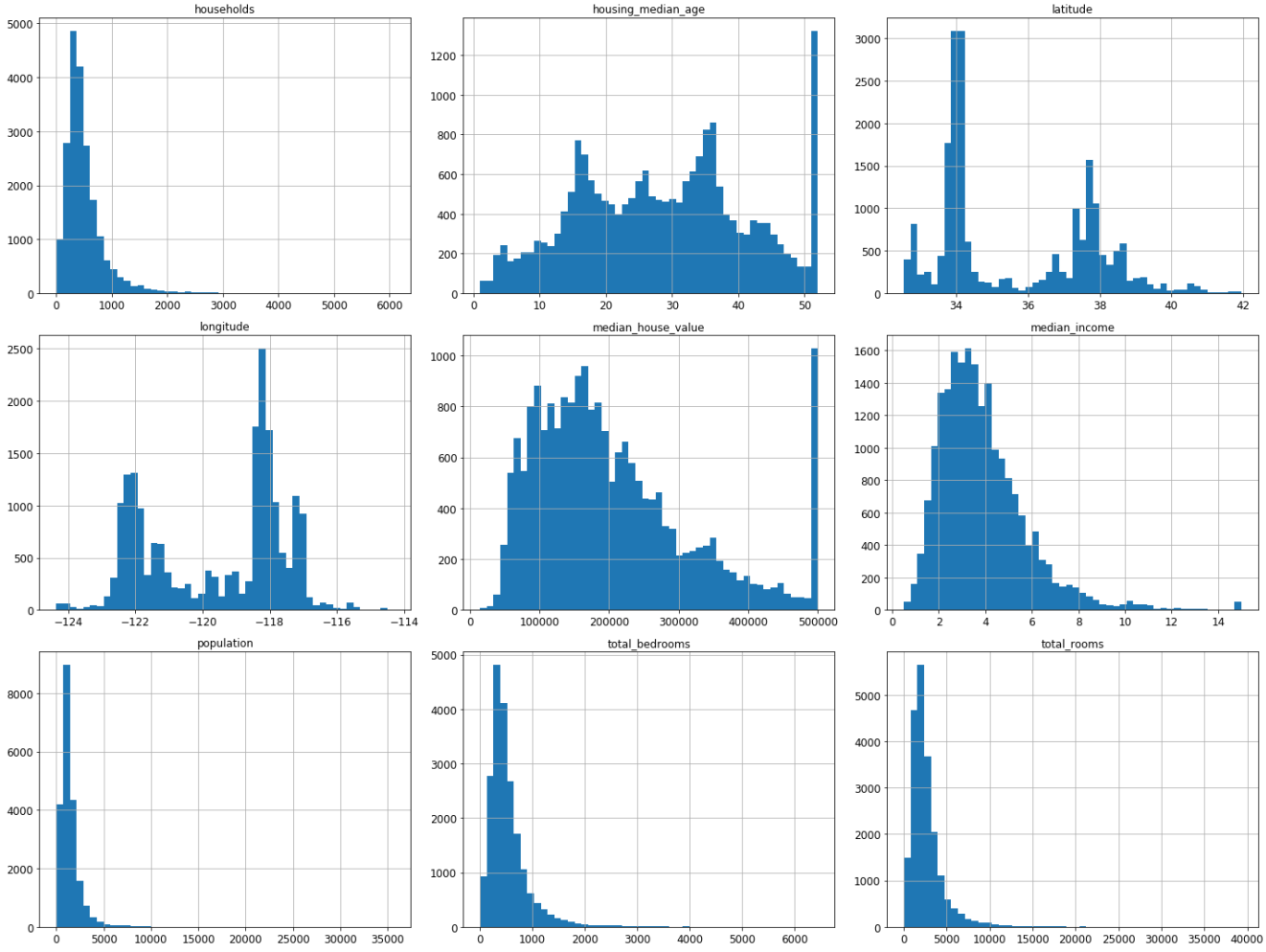
In [11]:

```
%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
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-11-f63abbc91a70> in <module>()
      2 import matplotlib.pyplot as plt
      3 housing.hist(bins=50, figsize=(20,15))
----> 4 save_fig("attribute_histogram_plots")
      5 plt.show()

<ipython-input-1-1c273385e6c6> in save_fig(fig_id, tight_layout, fig_extension, resolution)
     29     if tight_layout:
     30         plt.tight_layout()
---> 31     plt.savefig(path, format=fig_extension, dpi=resolution)
     32
     33 # Ignore useless warnings (see SciPy issue #5998)

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py in savefig(*args, **kwargs)
    695 def savefig(*args, **kwargs):
```

```
    696     fig = gcf()
--> 697     res = fig.savefig(*args, **kwargs)
    698     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    699     return res

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/figure.py in savefig(self, *args,
**kwargs)
   1570             self.set_frameon(frameon)
   1571
-> 1572         self.canvas.print_figure(*args, **kwargs)
   1573
   1574         if frameon:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backend_bases.py in
print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, **kwargs)
   2242                 orientation=orientation,
   2243                 bbox_inches_restore=_bbox_inches_restore,
-> 2244                 **kwargs)
   2245         finally:
   2246             if bbox_inches and restore_bbox:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backends/backend_agg.py in print_png(
self, filename_or_obj, *args, **kwargs)
    548         renderer.dpi = self.figure.dpi
    549         if is_string_like(filename_or_obj):
--> 550             filename_or_obj = open(filename_or_obj, 'wb')
    551             close = True
    552         else:

FileNotFoundError: [Errno 2] No such file or directory:
'./images/end_to_end_project/attribute_histogram_plots.png'
```



In [12]:

```
np.random.seed(42)
```

In [13]:

```
import numpy as np

# For illustration only. Sklearn has train_test_split()
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]
```

In [15]:

```
train_set, test_set = split_train_test(housing, 0.2)
print(len(train_set), "train +", len(test_set), "test")
```

```
16512 train + 4128 test
```

In [16]:

```
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]
```

In [17]:

```
import hashlib

def test_set_check(identifier, test_ratio, hash=hashlib.md5):
    return hash(np.int64(identifier)).digest()[-1] < 256 * test_ratio
```

In [18]:

```
def test_set_check(identifier, test_ratio, hash=hashlib.md5):
    return bytearray(hash(np.int64(identifier)).digest())[-1] < 256 * test_ratio
```

In [19]:

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

In [20]:

```
housing_with_id["id"] = housing["longitude"] * 1000 + housing["latitude"]
train_set, test_set = split_train_test_by_id(housing_with_id, 0.2, "id")
```

In [21]:

```
test_set.head()
```

Out[21]:

	index	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
8	8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804
10	10	-122.26	37.85	52.0	2202.0	434.0	910.0	402.0	3.2031
11	11	-122.26	37.85	52.0	3503.0	752.0	1504.0	734.0	3.2705

	index	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income
12	12	-122.26	37.85	52.0	2491.0	474.0	1098.0	468.0	3.0750
13	13	-122.26	37.84	52.0	696.0	191.0	345.0	174.0	2.6736

In [23]:

```
from sklearn.model_selection import train_test_split

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

In [24]:

```
test_set.head()
```

Out[24]:

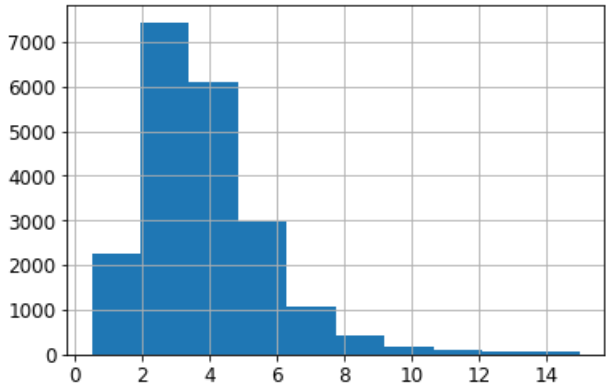
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	m
20046	-119.01	36.06	25.0	1505.0	NaN	1392.0	359.0	1.6812	47
3024	-119.46	35.14	30.0	2943.0	NaN	1565.0	584.0	2.5313	45
15663	-122.44	37.80	52.0	3830.0	NaN	1310.0	963.0	3.4801	50
20484	-118.72	34.28	17.0	3051.0	NaN	1705.0	495.0	5.7376	21
9814	-121.93	36.62	34.0	2351.0	NaN	1063.0	428.0	3.7250	27

In [25]:

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

Out[25]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb5426ef208>
```



In [26]:

```
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])
```

In [27]:

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

Out[27]:

```
3    7236
2    6581
4    3639
5    2362
```

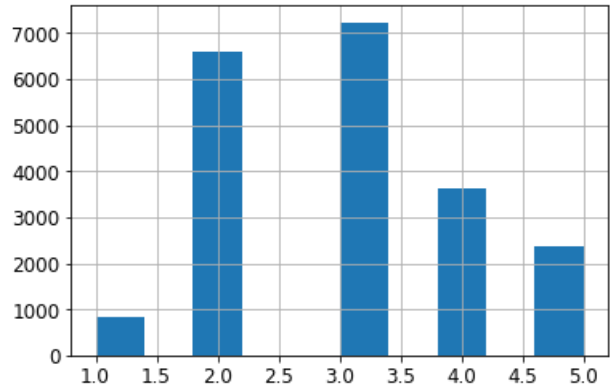
```
1     822
Name: income_cat, dtype: int64
```

In [28]:

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

Out[28]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb542401d30>
```



In [29]:

```
from sklearn.model_selection import StratifiedShuffleSplit

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]
```

In [30]:

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

Out[30]:

```
3    0.350533
2    0.318798
4    0.176357
5    0.114583
1    0.039729
Name: income_cat, dtype: float64
```

In [31]:

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

Out[31]:

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

In [32]:

```
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_test_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
```

In [33]:

```
compare_props
```

Out[33]:

	Overall	Stratified	Random	Rand. %error	Strat. %error
1	0.039826	0.039729	0.040213	0.973236	-0.243309
2	0.318847	0.318798	0.324370	1.732260	-0.015195
3	0.350581	0.350533	0.358527	2.266446	-0.013820
4	0.176308	0.176357	0.167393	-5.056334	0.027480
5	0.114438	0.114583	0.109496	-4.318374	0.127011

In [34]:

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

In [35]:

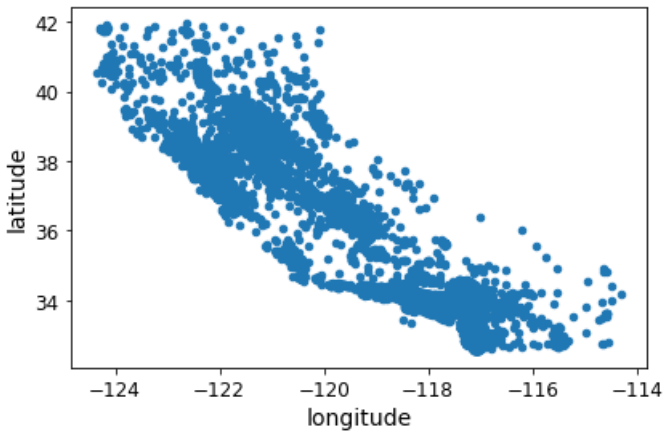
```
housing = strat_train_set.copy()
```

In [36]:

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

```
Saving figure bad_visualization_plot


---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-36-1e92da1f9d36> in <module>()
      1 housing.plot(kind="scatter", x="longitude", y="latitude")
----> 2 save_fig("bad_visualization_plot")

<ipython-input-1-1c273385e6c6> in save_fig(fig_id, tight_layout, fig_extension, resolution)
     29     if tight_layout:
     30         plt.tight_layout()
---> 31     plt.savefig(path, format=fig_extension, dpi=resolution)
     32
     33 # Ignore useless warnings (see SciPy issue #5998)

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py in savefig(*args, **kwargs)
    695 def savefig(*args, **kwargs):
    696     fig = gcf()
--> 697     res = fig.savefig(*args, **kwargs)
    698     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    699     return res

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/figure.py in savefig(self, *args,
**kwargs)
   1570             self.set_frameon(frameon)
   1571
-> 1572         self.canvas.print_figure(*args, **kwargs)
   1573
   1574         if frameon:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backend_bases.py in
print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, **kwargs)
   2242                 orientation=orientation,
```

```
   2243                 bbox_inches_restore=_bbox_inches_restore,
-> 2244                 **kwargs)
   2245         finally:
   2246             if bbox_inches and restore_bbox:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backends/backend_agg.py in print_png(
self, filename_or_obj, *args, **kwargs)
    548         renderer.dpi = self.figure.dpi
    549         if is_string_like(filename_or_obj):
--> 550             filename_or_obj = open(filename_or_obj, 'wb')
    551             close = True
    552         else:

FileNotFoundError: [Errno 2] No such file or directory:
'./images/end_to_end_project/bad_visualization_plot.png'
```

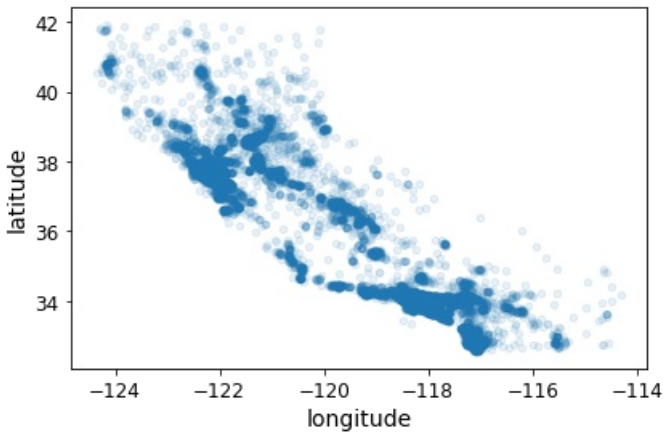


In [37]:

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

```
Saving figure better_visualization_plot


---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-37-2d6c3e8517f5> in <module>()
      1 housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
----> 2 save_fig("better_visualization_plot")

<ipython-input-1-1c273385e6c6> in save_fig(fig_id, tight_layout, fig_extension, resolution)
     29     if tight_layout:
     30         plt.tight_layout()
---> 31     plt.savefig(path, format=fig_extension, dpi=resolution)
     32
     33 # Ignore useless warnings (see SciPy issue #5998)

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py in savefig(*args, **kwargs)
    695 def savefig(*args, **kwargs):
    696     fig = gcf()
--> 697     res = fig.savefig(*args, **kwargs)
    698     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    699     return res

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/figure.py in savefig(self, *args,
**kwargs)
   1570             self.set_frameon(frameon)
   1571
-> 1572         self.canvas.print_figure(*args, **kwargs)
   1573
   1574         if frameon:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backend_bases.py in
print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, **kwargs)
   2242                 orientation=orientation,
   2243                 bbox_inches_restore=_bbox_inches_restore,
-> 2244                 **kwargs)
   2245         finally:
```

```
   2246             if bbox_inches and restore_bbox:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backends/backend_agg.py in print_png(
self, filename_or_obj, *args, **kwargs)
    548         renderer.dpi = self.figure.dpi
    549         if is_string_like(filename_or_obj):
--> 550             filename_or_obj = open(filename_or_obj, 'wb')
    551             close = True
    552         else:

FileNotFoundError: [Errno 2] No such file or directory:
'./images/end_to_end_project/better_visualization_plot.png'
```

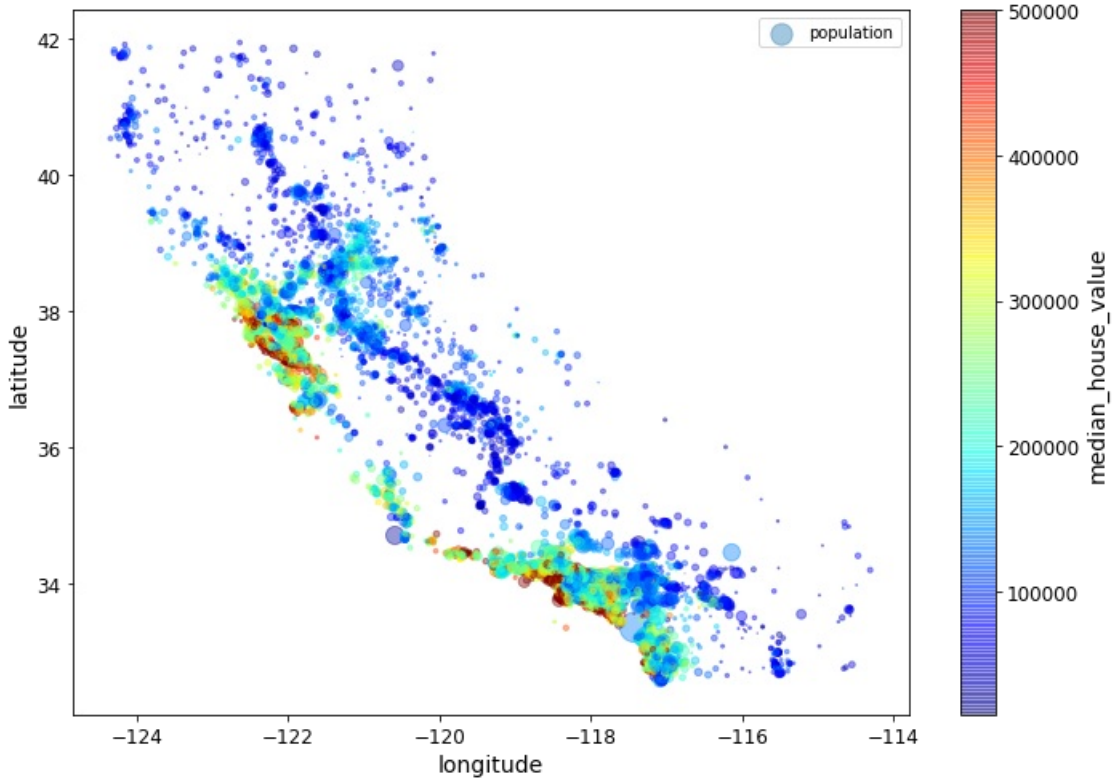


In [38]:

```
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


---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-38-eba3dc532b87> in <module>()
      4     sharex=False)
      5 plt.legend()
----> 6 save_fig("housing_prices_scatterplot")

<ipython-input-1-1c273385e6c6> in save_fig(fig_id, tight_layout, fig_extension, resolution)
     29     if tight_layout:
     30         plt.tight_layout()
---> 31     plt.savefig(path, format=fig_extension, dpi=resolution)
     32
     33 # Ignore useless warnings (see SciPy issue #5998)

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py in savefig(*args, **kwargs)
    695 def savefig(*args, **kwargs):
    696     fig = gcf()
--> 697     res = fig.savefig(*args, **kwargs)
    698     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    699     return res

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/figure.py in savefig(self, *args,
**kwargs)
   1570             self.set_frameon(frameon)
   1571
-> 1572         self.canvas.print_figure(*args, **kwargs)
   1573
   1574         if frameon:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backend_bases.py in
print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, **kwargs)
   2242                 orientation=orientation,
   2243                 bbox_inches_restore=_bbox_inches_restore,
```

```
-> 2244                 **kwargs)
   2245         finally:
   2246             if bbox_inches and restore_bbox:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backends/backend_agg.py in print_png(
self, filename_or_obj, *args, **kwargs)
    548         renderer.dpi = self.figure.dpi
    549         if is_string_like(filename_or_obj):
--> 550             filename_or_obj = open(filename_or_obj, 'wb')
    551             close = True
    552         else:

FileNotFoundError: [Errno 2] No such file or directory:
'./images/end_to_end_project/housing_prices_scatterplot.png'
```



In [39]:

```
import matplotlib.image as mpimg
california_img=mpimg.imread(PROJECT_ROOT_DIR + '/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()
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()
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-39-b02ff44a3a2c> in <module>()
      2
      3 import matplotlib.image as mpimg
----> 4 california_img=mpimg.imread(PROJECT_ROOT_DIR + '/images/end_to_end_project/california.png')
```

```
      5 ax = housing.plot(kind="scatter", x="longitude", y="latitude", figsize=(10,7),
      6                       s=housing['population']/100, label="Population",

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/image.py in imread(fname, format)
   1244             return handler(fd)
   1245         else:
-> 1246             with open(fname, 'rb') as fd:
   1247                 return handler(fd)
   1248     else:

FileNotFoundError: [Errno 2] No such file or directory:
'./images/end_to_end_project/california.png'
```

In [40]:

```
corr_matrix = housing.corr()
```

In [41]:

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

Out[41]:

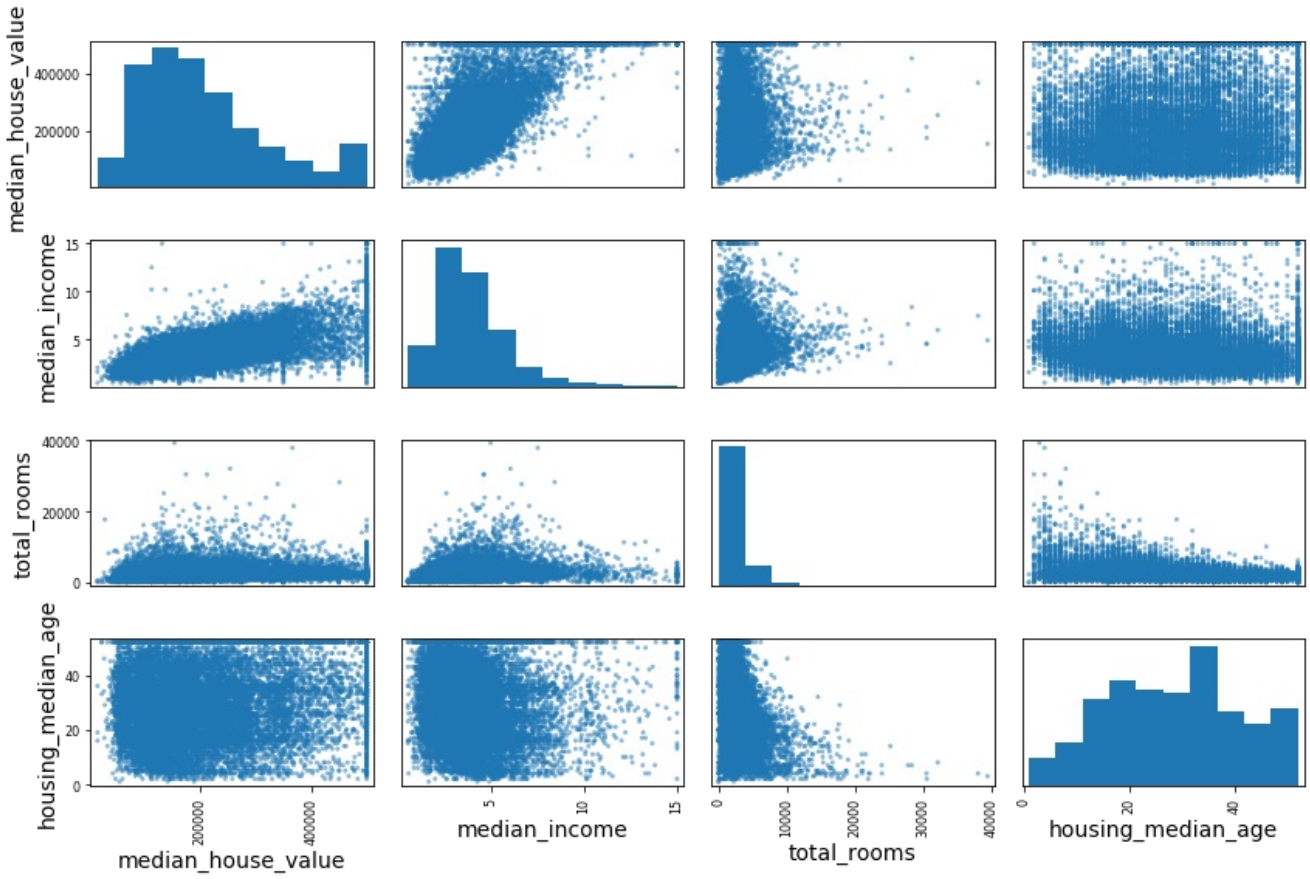
```
median_house_value    1.000000
median_income         0.687160
total_rooms           0.135097
housing_median_age    0.114110
households            0.064506
total_bedrooms        0.047689
population           -0.026920
longitude            -0.047432
latitude             -0.142724
Name: median_house_value,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

In [42]:

```
from pandas.plotting import scatter_matrix

attributes = ["median_house_value", "median_income", "total_rooms",
              "housing_median_age"]
scatter_matrix(housing[attributes], figsize=(12, 8))
save_fig("scatter_matrix_plot")
```

```
Saving figure scatter_matrix_plot

---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-42-7dd461c0e565> in <module>()
      4                 "housing_median_age"]
      5 scatter_matrix(housing[attributes], figsize=(12, 8))
----> 6 save_fig("scatter_matrix_plot")

<ipython-input-1-1c273385e6c6> in save_fig(fig_id, tight_layout, fig_extension, resolution)
     29     if tight_layout:
     30         plt.tight_layout()
---> 31     plt.savefig(path, format=fig_extension, dpi=resolution)
     32
     33 # Ignore useless warnings (see SciPy issue #5998)

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py in savefig(*args, **kwargs)
    695 def savefig(*args, **kwargs):
    696     fig = gcf()
--> 697     res = fig.savefig(*args, **kwargs)
    698     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    699     return res

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/figure.py in savefig(self, *args,
**kwargs)
   1570             self.set_frameon(frameon)
   1571
-> 1572         self.canvas.print_figure(*args, **kwargs)
   1573
   1574         if frameon:
```

```
/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backend_bases.py in
print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, **kwargs)
   2242                 orientation=orientation,
   2243                 bbox_inches_restore=_bbox_inches_restore,
-> 2244                 **kwargs)
   2245         finally:
   2246             if bbox_inches and restore_bbox:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backends/backend_agg.py in print_png(
self, filename_or_obj, *args, **kwargs)
    548         renderer.dpi = self.figure.dpi
    549         if is_string_like(filename_or_obj):
--> 550             filename_or_obj = open(filename_or_obj, 'wb')
    551             close = True
    552         else:

FileNotFoundError: [Errno 2] No such file or directory:
'./images/end_to_end_project/scatter_matrix_plot.png'
```



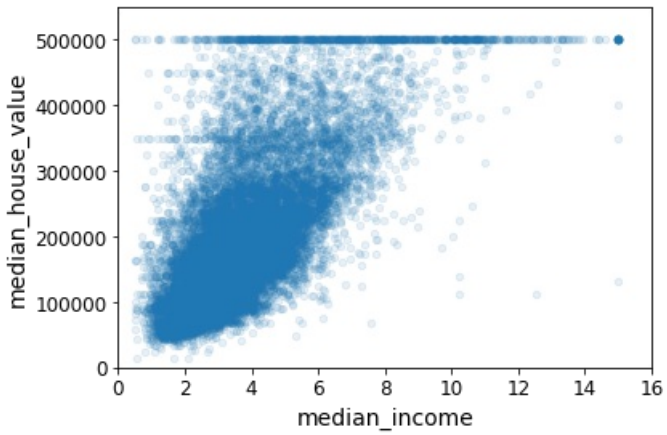
In [43]:

```
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
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-43-59fb660562c4> in <module>()
      2              alpha=0.1)
      3 plt.axis([0, 16, 0, 550000])
----> 4 save_fig("income_vs_house_value_scatterplot")

<ipython-input-1-1c273385e6c6> in save_fig(fig_id, tight_layout, fig_extension, resolution)
     29     if tight_layout:
     30         plt.tight_layout()
---> 31     plt.savefig(path, format=fig_extension, dpi=resolution)
```

```
     32
     33 # Ignore useless warnings (see SciPy issue #5998)

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/pyplot.py in savefig(*args, **kwargs)
    695 def savefig(*args, **kwargs):
    696     fig = gcf()
--> 697     res = fig.savefig(*args, **kwargs)
    698     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    699     return res

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/figure.py in savefig(self, *args,
**kwargs)
   1570             self.set_frameon(frameon)
   1571
-> 1572         self.canvas.print_figure(*args, **kwargs)
   1573
   1574         if frameon:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backend_bases.py in
print_figure(self, filename, dpi, facecolor, edgecolor, orientation, format, **kwargs)
   2242                 orientation=orientation,
   2243                 bbox_inches_restore=_bbox_inches_restore,
-> 2244                 **kwargs)
   2245         finally:
   2246             if bbox_inches and restore_bbox:

/home/ajith/anaconda3/lib/python3.6/site-packages/matplotlib/backends/backend_agg.py in print_png(
self, filename_or_obj, *args, **kwargs)
    548         renderer.dpi = self.figure.dpi
    549         if is_string_like(filename_or_obj):
--> 550             filename_or_obj = open(filename_or_obj, 'wb')
    551             close = True
    552         else:

FileNotFoundError: [Errno 2] No such file or directory:
'./images/end_to_end_project/income_vs_house_value_scatterplot.png'
```



In [44]:

```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
housing["population_per_household"]=housing["population"]/housing["households"]
```

In [45]:

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

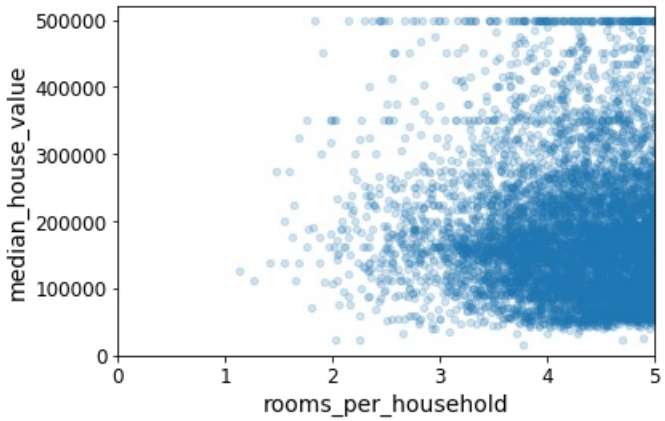
Out[45]:

```
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
```

In [46]:

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



In [47]:

```
housing.describe()
```

Out[47]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	med
count	16512.000000	16512.000000	16512.000000	16512.000000	16354.000000	16512.000000	16512.000000	165
mean	-119.575834	35.639577	28.653101	2622.728319	534.973890	1419.790819	497.060380	3.87
std	2.001860	2.138058	12.574726	2138.458419	412.699041	1115.686241	375.720845	1.90
min	-124.350000	32.540000	1.000000	6.000000	2.000000	3.000000	2.000000	0.49
25%	-121.800000	33.940000	18.000000	1443.000000	295.000000	784.000000	279.000000	2.56
50%	-118.510000	34.260000	29.000000	2119.500000	433.000000	1164.000000	408.000000	3.54
75%	-118.010000	37.720000	37.000000	3141.000000	644.000000	1719.250000	602.000000	4.74
max	-114.310000	41.950000	52.000000	39320.000000	6210.000000	35682.000000	5358.000000	15.0

In [48]:

```
housing = strat_train_set.drop("median_house_value", axis=1) # drop labels for training set
housing_labels = strat_train_set["median_house_value"].copy()
```

In [49]:

```
sample_incomplete_rows = housing[housing.isnull().any(axis=1)].head()
sample_incomplete_rows
```

Out[49]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	oc
4629	-118.30	34.07	18.0	3759.0	NaN	3296.0	1462.0	2.2708	<
6068	-117.86	34.01	16.0	4632.0	NaN	3038.0	727.0	5.1762	<

17923	-121.97	37.35	30.0	1955.0	NaN	999.0	386.0	4.6328	<1
13656	-117.30	34.05	6.0	2155.0	NaN	1039.0	391.0	1.6675	IN
19252	-122.79	38.48	7.0	6837.0	NaN	3468.0	1405.0	3.1662	<1

In [50]:

```
sample_incomplete_rows.dropna(subset=["total_bedrooms"])
```

Out[50]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_

In [51]:

```
sample_incomplete_rows.drop("total_bedrooms", axis=1)
```

Out[51]:

	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

In [52]:

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

Out[52]:

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

In [53]:

```
try:
    from sklearn.impute import SimpleImputer # Scikit-Learn 0.20+
except ImportError:
    from sklearn.preprocessing import Imputer as SimpleImputer

imputer = SimpleImputer(strategy="median")
```

In [54]:

```
housing_num = housing.drop('ocean_proximity', axis=1)
```

In [55]:

```
imputer.fit(housing_num)
```

Out[55]:

```
SimpleImputer(add_indicator=False, copy=True, fill_value=None,
              missing_values=nan, strategy='median', verbose=0)
```

In [56]:

```
imputer.fit(housing_num)
```

Out[56]:

```
SimpleImputer(add_indicator=False, copy=True, fill_value=None,
              missing_values=nan, strategy='median', verbose=0)
```

In [57]:

```
imputer.statistics_
```

Out[57]:

```
array([-118.51  ,   34.26  ,   29.    , 2119.5   ,  433.    , 1164.    ,
        408.    ,    3.5409])
```

In [58]:

```
housing_num.median().values
```

Out[58]:

```
array([-118.51  ,   34.26  ,   29.    , 2119.5   ,  433.    , 1164.    ,
        408.    ,    3.5409])
```

In [59]:

```
X = imputer.transform(housing_num)
```

In [60]:

```
housing_tr = pd.DataFrame(X, columns=housing_num.columns,
                          index=housing.index)
```

In [61]:

```
housing_tr.loc[sample_incomplete_rows.index.values]
```

Out[61]:

	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

In [62]:

```
housing_cat = housing[['ocean_proximity']]
housing_cat.head(10)
```

Out[62]:

	ocean_proximity

	ocean_proximity
17606	<1H OCEAN
18632	<1H OCEAN
14650	NEAR OCEAN
3230	INLAND
3555	<1H OCEAN
19480	INLAND
8879	<1H OCEAN
13685	INLAND
4937	<1H OCEAN
4861	<1H OCEAN

In [64]:

```
try:
    from sklearn.preprocessing import OrdinalEncoder
except ImportError:
    from future_encoders import OrdinalEncoder # Scikit-Learn < 0.20
ordinal_encoder = OrdinalEncoder()
housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
housing_cat_encoded[:10]
```

Out[64]:

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

In [65]:

```
ordinal_encoder.categories_
```

Out[65]:

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

In [71]:

```
try:
    from sklearn.preprocessing import OrdinalEncoder # just to raise an ImportError if Scikit-Learn
< 0.20
    from sklearn.preprocessing import OneHotEncoder
except ImportError:
    from future_encoders import OneHotEncoder # Scikit-Learn < 0.20

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

Out[71]:

```
<16512x5 sparse matrix of type '<class 'numpy.float64'>'
 with 16512 stored elements in Compressed Sparse Row format>
```

In [72]:

```
housing_cat_1hot.toarray()
```

Out[72]:

```
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.]])
```

In [73]:

```
cat_encoder = OneHotEncoder(sparse=False)
housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
housing_cat_1hot
```

Out[73]:

```
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.]])
```

In [74]:

```
cat_encoder.categories_
```

Out[74]:

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

In [75]:

```
housing.columns
```

Out[75]:

```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity'],
      dtype='object')
```

In [76]:

```
from sklearn.base import BaseEstimator, TransformerMixin

# get the right column indices: safer than hard-coding indices 3, 4, 5, 6
rooms_ix, bedrooms_ix, population_ix, household_ix = [
    list(housing.columns).index(col)
    for col in ("total_rooms", "total_bedrooms", "population", "households")]

class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self, add_bedrooms_per_room = True): # no *args or **kwargs
        self.add_bedrooms_per_room = add_bedrooms_per_room
    def fit(self, X, y=None):
        return self  # nothing else to do
    def transform(self, X, y=None):
        rooms_per_household = X[:, rooms_ix] / X[:, household_ix]
        population_per_household = X[:, population_ix] / X[:, household_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)
```

In [77]:

```
from sklearn.preprocessing import FunctionTransformer

def add_extra_features(X, add_bedrooms_per_room=True):
    rooms_per_household = X[:, rooms_ix] / X[:, household_ix]
    population_per_household = X[:, population_ix] / X[:, household_ix]
    if 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 = FunctionTransformer(add_extra_features, validate=False,
                                 kw_args={"add_bedrooms_per_room": False})
housing_extra_attribs = attr_adder.fit_transform(housing.values)
```

In [78]:

```
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()
```

Out[78]:

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

In [79]:

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

num_pipeline = Pipeline([
        ('imputer', SimpleImputer(strategy="median")),
        ('attribs_adder', FunctionTransformer(add_extra_features, validate=False)),
        ('std_scaler', StandardScaler()),
    ])

housing_num_tr = num_pipeline.fit_transform(housing_num)
```

In [81]:

```
housing_num_tr
```

Out[81]:

```
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]])
```

In [83]:

```
try:
    from sklearn.compose import ColumnTransformer
except ImportError:
    from future_encoders import ColumnTransformer # Scikit-Learn < 0.20
```

In [84]:

```
num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]

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

housing_prepared = full_pipeline.fit_transform(housing)
```

In [85]:

```
housing_prepared
```

Out[85]:

```
array([[-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.        ]])
```

In [86]:

```
from sklearn.base import BaseEstimator, TransformerMixin

# Create a class to select numerical or categorical columns
class OldDataFrameSelector(BaseEstimator, TransformerMixin):
    def __init__(self, attribute_names):
        self.attribute_names = attribute_names
    def fit(self, X, y=None):
        return self
    def transform(self, X):
        return X[self.attribute_names].values
```

In [87]:

```
num_attribs = list(housing_num)
cat_attribs = ["ocean_proximity"]

old_num_pipeline = Pipeline([
        ('selector', OldDataFrameSelector(num_attribs)),
        ('imputer', SimpleImputer(strategy="median")),
        ('attribs_adder', FunctionTransformer(add_extra_features, validate=False)),
        ('std_scaler', StandardScaler()),
    ])

old_cat_pipeline = Pipeline([
        ('selector', OldDataFrameSelector(cat_attribs)),
        ('cat_encoder', OneHotEncoder(sparse=False)),
    ])
```

In [88]:

```
from sklearn.pipeline import FeatureUnion

old_full_pipeline = FeatureUnion(transformer_list=[
        ("num_pipeline", old_num_pipeline),
        ("cat_pipeline", old_cat_pipeline),
    ])
```

In [89]:

```
old_housing_prepared = old_full_pipeline.fit_transform(housing)
old_housing_prepared
```

Out[89]:

```
array([[-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.        ]])
```

In [90]:

```
np.allclose(housing_prepared, old_housing_prepared)
```

Out[90]:

```
True
```

In [91]:

```
from sklearn.linear_model import LinearRegression

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

Out[91]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [92]:

```
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))
```

```
Predictions: [210644.60459286 317768.80697211 210956.43331178  59218.98886849
 189747.55849879]
```

In [93]:

```
print("Labels:", list(some_labels))
```

```
Labels: [286600.0, 340600.0, 196900.0, 46300.0, 254500.0]
```

In [94]:

```
some_data_prepared
```

Out[94]:

Out[94]:

```
array([[-1.15604281,  0.77194962,  0.74333089, -0.49323393, -0.44543821,
        -0.63621141, -0.42069842, -0.61493744, -0.31205452, -0.08649871,
         0.15531753,  1.        ,  0.        ,  0.        ,  0.        ,
         0.        ],
       [-1.17602483,  0.6596948 , -1.1653172 , -0.90896655, -1.0369278 ,
        -0.99833135, -1.02222705,  1.33645936,  0.21768338, -0.03353391,
        -0.83628902,  1.        ,  0.        ,  0.        ,  0.        ,
         0.        ],
       [ 1.18684903, -1.34218285,  0.18664186, -0.31365989, -0.15334458,
        -0.43363936, -0.0933178 , -0.5320456 , -0.46531516, -0.09240499,
         0.4222004 ,  0.        ,  0.        ,  0.        ,  0.        ,
         1.        ],
       [-0.01706767,  0.31357576, -0.29052016, -0.36276217, -0.39675594,
         0.03604096, -0.38343559, -1.04556555, -0.07966124,  0.08973561,
        -0.19645314,  0.        ,  1.        ,  0.        ,  0.        ,
         0.        ],
       [ 0.49247384, -0.65929936, -0.92673619,  1.85619316,  2.41221109,
         2.72415407,  2.57097492, -0.44143679, -0.35783383, -0.00419445,
         0.2699277 ,  1.        ,  0.        ,  0.        ,  0.        ,
         0.        ]])
```

In [95]:

```
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
```

Out[95]:

```
68628.19819848923
```

In [97]:

```
from sklearn.metrics import mean_absolute_error

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

Out[97]:

```
49439.89599001897
```

In [98]:

```
from sklearn.tree import DecisionTreeRegressor

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

Out[98]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=42, splitter='best')
```

In [99]:

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

Out[99]:

```
0.0
```

In [101]:

```
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)
```

In [102]:

```
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
```

In [103]:

```
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: [66782.73843989 66960.118071   70347.95244419 74739.57052552
 68031.13388938 71193.84183426 64969.63056405 68281.61137997
 71552.91566558 67665.10082067]
Mean: 69052.46136345083
Standard deviation: 2731.6740017983493
```

In [104]:

```
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators=10, random_state=42)
forest_reg.fit(housing_prepared, housing_labels)
```

Out[104]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=42, verbose=0,
                      warm_start=False)
```

In [105]:

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

Out[105]:

```
21933.31414779769
```

In [106]:

```
from sklearn.model_selection import cross_val_score

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: [51646.44545909 48940.60114882 53050.86323649 54408.98730149
 50922.14870785 56482.50703987 51864.52025526 49760.85037653
 55434.21627933 53326.10093303]
Mean: 52583.72407377466
Standard deviation: 2298.353351147122
```

In [107]:

```
scores = cross_val_score(lin_reg, housing_prepared, housing_labels,
scoring="neg_mean_squared_error", cv=10)
pd.Series(np.sqrt(-scores)).describe()
```

Out[107]:

```
count       10.000000
mean     69052.461363
std       2879.437224
min      64969.630564
25%      67136.363758
50%      68156.372635
75%      70982.369487
max      74739.570526
dtype: float64
```

In [108]:

```
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
```

Out[108]:

```
111094.6308539982
```

In [111]:

```
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)
```

Out[111]:

```
GridSearchCV(cv=5, error_score='raise-deprecating',
             estimator=RandomForestRegressor(bootstrap=True, criterion='mse',
                                             max_depth=None,
                                             max_features='auto',
                                             max_leaf_nodes=None,
                                             min_impurity_decrease=0.0,
                                             min_impurity_split=None,
                                             min_samples_leaf=1,
                                             min_samples_split=2,
                                             min_weight_fraction_leaf=0.0,
                                             n_estimators='warn', n_jobs=None,
                                             oob_score=False, random_state=42,
```

```
                                       verbose=0, warm_start=False),
             iid='warn', n_jobs=None,
             param_grid=[{'max_features': [2, 4, 6, 8],
                          'n_estimators': [3, 10, 30]},
                         {'bootstrap': [False], 'max_features': [2, 3, 4],
                          'n_estimators': [3, 10]}],
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring='neg_mean_squared_error', verbose=0)
```

In [110]:

```
grid_search.best_params_
```

Out[110]:

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

In [112]:

```
grid_search.best_estimator_
```

Out[112]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features=8, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=30,
                      n_jobs=None, oob_score=False, random_state=42, verbose=0,
                      warm_start=False)
```

In [113]:

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

```
63669.05791727153 {'max_features': 2, 'n_estimators': 3}
55627.16171305252 {'max_features': 2, 'n_estimators': 10}
53384.57867637289 {'max_features': 2, 'n_estimators': 30}
60965.99185930139 {'max_features': 4, 'n_estimators': 3}
52740.98248528835 {'max_features': 4, 'n_estimators': 10}
50377.344409590376 {'max_features': 4, 'n_estimators': 30}
58663.84733372485 {'max_features': 6, 'n_estimators': 3}
52006.15355973719 {'max_features': 6, 'n_estimators': 10}
50146.465964159885 {'max_features': 6, 'n_estimators': 30}
57869.25504027614 {'max_features': 8, 'n_estimators': 3}
51711.09443660957 {'max_features': 8, 'n_estimators': 10}
49682.25345942335 {'max_features': 8, 'n_estimators': 30}
62895.088889905004 {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
54658.14484390074 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
59470.399594730654 {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
52725.01091081235 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
57490.612956065226 {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
51009.51445842374 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
```

In [114]:

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

Out[114]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_features	param_n_estimators	param_boot
0	0.077848	0.000838	0.004669	0.000872	2	3	NaN

1	0.252692 **mean_fit_time**	0.002219 **std_fit_time**	0.011163 **mean_score_time**	0.000065 **std_score_time**	2 **param_max_features**	10 **param_n_estimators**	NaN **param_boot**
2	0.764095	0.006926	0.031640	0.000114	2	30	NaN
3	0.129226	0.001305	0.004253	0.000114	4	3	NaN
4	0.423487	0.002185	0.011162	0.000052	4	10	NaN
5	1.259430	0.003446	0.031774	0.000283	4	30	NaN
6	0.175151	0.004262	0.004184	0.000036	6	3	NaN
7	0.582112	0.004192	0.011197	0.000076	6	10	NaN
8	1.763947	0.008319	0.031661	0.000086	6	30	NaN
9	0.223684	0.000815	0.004168	0.000034	8	3	NaN
10	0.751385	0.005223	0.011166	0.000124	8	10	NaN
11	2.270840	0.009550	0.031819	0.000450	8	30	NaN
12	0.123007	0.002236	0.004699	0.000058	2	3	False
13	0.407523	0.002964	0.012824	0.000299	2	10	False
14	0.164722	0.002719	0.004675	0.000038	3	3	False
15	0.559217	0.018301	0.013073	0.000489	3	10	False

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_features	param_n_estimators	param_boot
16	0.209876	0.003828	0.004723	0.000048	4	3	False
17	0.702184	0.025057	0.013364	0.000751	4	10	False

18 rows × 23 columns

In [115]:

```
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)
```

Out[115]:

```
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   criterion='mse',
                                                   max_depth=None,
                                                   max_features='auto',
                                                   max_leaf_nodes=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0.0,
                                                   n_estimators='warn',
                                                   n_jobs=None, oob_score=False,
                                                   random_sta...
                                                   warm_start=False),
                   iid='warn', n_iter=10, n_jobs=None,
                   param_distributions={'max_features':
<scipy.stats._distn_infrastructure.rv_frozen object at 0x7fb56168f2e8>,
                                        'n_estimators': <scipy.stats._distn_infrastructure.rv_froze
object at 0x7fb5616bc1d0>},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring='neg_mean_squared_error',
                   verbose=0)
```

In [116]:

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

```
49150.657232934034 {'max_features': 7, 'n_estimators': 180}
51389.85295710133 {'max_features': 5, 'n_estimators': 15}
50796.12045980556 {'max_features': 3, 'n_estimators': 72}
50835.09932039744 {'max_features': 5, 'n_estimators': 21}
49280.90117886215 {'max_features': 7, 'n_estimators': 122}
50774.86679035961 {'max_features': 3, 'n_estimators': 75}
50682.75001237282 {'max_features': 3, 'n_estimators': 88}
49608.94061293652 {'max_features': 5, 'n_estimators': 100}
50473.57642831875 {'max_features': 3, 'n_estimators': 150}
64429.763804893395 {'max_features': 5, 'n_estimators': 2}
```

In [117]:

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

```
feature_importances
```

Out[117]:

```
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])
```

In [118]:

```
extra_attribs = ["rooms_per_hhold", "pop_per_hhold", "bedrooms_per_room"]
#cat_encoder = cat_pipeline.named_steps["cat_encoder"] # old solution
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)
```

Out[118]:

```
[(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'),
 (0.002856746373201584, 'NEAR OCEAN'),
 (0.0019604155994780706, 'NEAR BAY'),
 (6.0280386727366e-05, 'ISLAND')]
```

In [119]:

```
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)
```

In [120]:

```
final_rmse
```

Out[120]:

```
47730.22690385927
```

In [121]:

```
from scipy import stats
```

In [122]:

```
confidence = 0.95
squared_errors = (final_predictions - y_test) ** 2
mean = squared_errors.mean()
m = len(squared_errors)

np.sqrt(stats.t.interval(confidence, m - 1,
                         loc=np.mean(squared_errors),
```

```
                          scale=stats.sem(squared_errors)))
```

Out[122]:

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

In [123]:

```
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)
```

Out[123]:

```
(45685.10470776, 49691.25001877858)
```

In [124]:

```
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)
```

Out[124]:

```
(45685.717918136455, 49690.68623889413)
```

In [ ]:

```

```