In [1]:

```python
# To support both python 2 and python 3  :  California Housing Prices dataset \ a multivariate regression
from __future__ import division, print_function, unicode_literals

# Common imports
import numpy as np
import os

# to make this notebook's output stable across runs
np.random.seed(456)
```

In [2]:

```python
# 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)
```

In [3]:

```python
# Where to save the figures
PROJECT_ROOT_DIR = "."
CHAPTER_ID = "end_to_end_project"
IMAGES_PATH = os.path.join(PROJECT_ROOT_DIR, "images", CHAPTER_ID)

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")
```

In [4]:
```python
# Automating the process of fetching the data is also useful if you need to install the dataset on multiple machines.
import os
import tarfile    #  download a single compressed file, housing.tgz
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"  #  download a single compressed file, housing.tgz

# creates a datasets/housing directory in your workspace
def fetch_housing_data(housing_url=HOUSING_URL, housing_path=HOUSING_PATH):
    os.makedirs(housing_path, exist_ok=True)
    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 [5]:
```python
fetch_housing_data()
```

In [6]:
```python
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]:
```python
# Save into housing = load_housing_data
housing = load_housing_data()
housing.head()
```

Out[7]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

In [8]:
```
# The info() method is useful to get a quick description of the data

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]:
```
#  meaning that 207 districts are missing this feature
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["population"].value_counts()
```

Out[10]:

```
891.0      25
761.0      24
1227.0     24
850.0      24
1052.0     24
825.0      23
999.0      22
782.0      22
1005.0     22
781.0      21
1098.0     21
753.0      21
872.0      21
1056.0     20
1158.0     20
899.0      20
837.0      20
804.0      20
1011.0     20
926.0      20
1155.0     20
1203.0     20
1047.0     20
986.0      20
861.0      20
735.0      20
1301.0     20
1054.0     20
928.0      19
866.0      19
           ..
8738.0      1
8907.0      1
3663.0      1
4100.0      1
3297.0      1
5116.0      1
5595.0      1
5731.0      1
116.0       1
4089.0      1
7009.0      1
3886.0      1
3549.0      1
4531.0      1
3926.0      1
3303.0      1
3794.0      1
3147.0      1
3589.0      1
4625.0      1
3284.0      1
```

```
7305.0      1
4033.0      1
5110.0      1
11973.0     1
2994.0      1
4333.0      1
3891.0      1
5087.0      1
3591.0      1
Name: population, Length: 3888, dtype: int64
```

In [15]:
```
housing.describe()
```

Out[15]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	421.385070	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

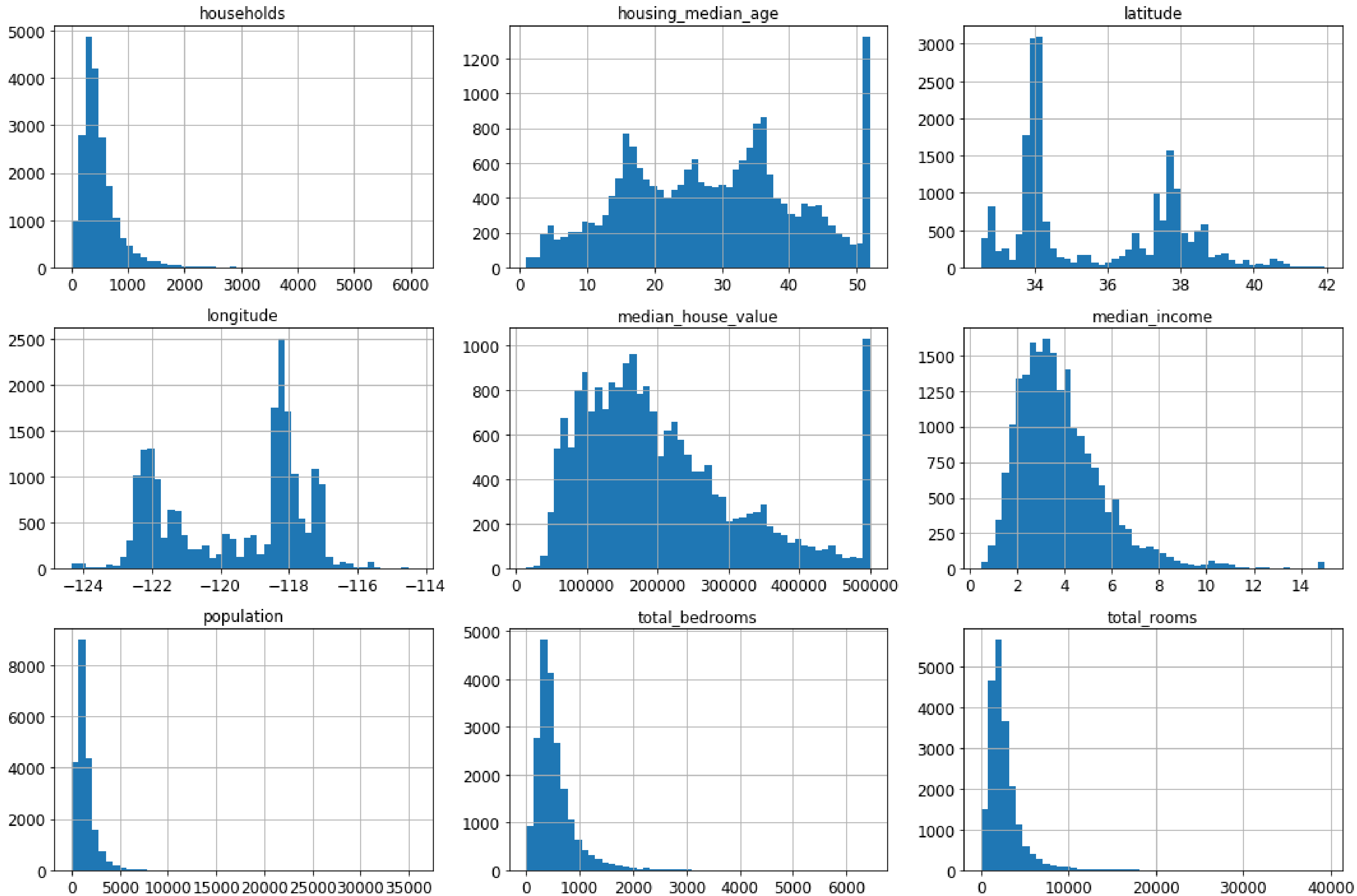
In [17]:

```
#  plot a histogram for each numerical attribute

%matplotlib inline
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(15,10))
save_fig("Attribute of histogram_plots")
plt.show()
```

```
Saving figure Attribute of histogram_plots
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-17-eb356ebb4b0d> in <module>
      2 import matplotlib.pyplot as plt
      3 housing.hist(bins=50, figsize=(15,10))
----> 4 save_fig("Attribute of histogram_plots")
      5 plt.show()

<ipython-input-3-f6435d47e026> in save_fig(fig_id, tight_layout, fig_extension, resolution)
      9     if tight_layout:
     10         plt.tight_layout()
---> 11     plt.savefig(path, format=fig_extension, dpi=resolution)
     12 
     13 # Ignore useless warnings (see SciPy issue #5998)

~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args, **kwargs)
    714 def savefig(*args, **kwargs):
    715     fig = gcf()
--> 716     res = fig.savefig(*args, **kwargs)
    717     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    718     return res

~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self, fname, transparent, **kwargs)
   2178             self.patch.set_visible(frameon)
   2179 
-> 2180         self.canvas.print_figure(fname, **kwargs)
   2181 
   2182         if frameon:

~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filename, dpi, facecolor, edgecolor, orientatio
n, format, bbox_inches, **kwargs)
   2080                     orientation=orientation,
   2081                     bbox_inches_restore=_bbox_inches_restore,
-> 2082                     **kwargs)
   2083             finally:
   2084                 if bbox_inches and restore_bbox:

~\Anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(self, filename_or_obj, metadata, pil_kwargs, *arg
s, **kwargs)
    528             renderer = self.get_renderer()
    529             with cbook._setattr_cm(renderer, dpi=self.figure.dpi), \
--> 530                     cbook.open_file_cm(filename_or_obj, "wb") as fh:
    531                 _png.write_png(renderer._renderer, fh,
    532                                self.figure.dpi, metadata=metadata)

~\Anaconda3\lib\contextlib.py in __enter__(self)
    110         del self.args, self.kwds, self.func
    111         try:
--> 112             return next(self.gen)
    113         except StopIteration:
    114             raise RuntimeError("generator didn't yield") from None
```

```
~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in open_file_cm(path_or_file, mode, encoding)
    445 def open_file_cm(path_or_file, mode="r", encoding=None):
    446     r"""Pass through file objects and context-manage `.PathLike`\s."""
--> 447     fh, opened = to_filehandle(path_or_file, mode, True, encoding)
    448     if opened:
    449         with fh:

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in to_filehandle(fname, flag, return_opened, encoding)
    430             fh = bz2.BZ2File(fname, flag)
    431         else:
--> 432             fh = open(fname, flag, encoding=encoding)
    433         opened = True
    434     elif hasattr(fname, 'seek'):

FileNotFoundError: [Errno 2] No such file or directory: '.\\images\\end_to_end_project\\Attribute of histogram_plots.png'
```



In [18]:
```python
# to make this notebook's output identical at every run
np.random.seed(456)
```

In [19]:
```python
# the housing dataset does not have an identifier column. The simplest solution is to use the row  index as the ID:

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 [20]:
```python
train_set, test_set = split_train_test(housing, 0.2)
print(len(train_set), "train +", len(test_set), "test")
```

```
16512 train + 4128 test
```

In [21]:
```python
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 [22]:
```python
import hashlib

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

In [24]:
```python
def test_set_check(identifier, test_ratio, hash=hashlib.md5):
    return bytearray(hash(np.int64(identifier)).digest())[-1] < 256 * test_ratio
```

In [25]:
```python
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 [26]:
```python
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 [27]:
```
test_set.head()
```

Out[27]:

	index	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximit
8	8	-122.26	37.84	42.0	2555.0	665.0	1206.0	595.0	2.0804	226700.0	NEAR BA
10	10	-122.26	37.85	52.0	2202.0	434.0	910.0	402.0	3.2031	281500.0	NEAR BA
11	11	-122.26	37.85	52.0	3503.0	752.0	1504.0	734.0	3.2705	241800.0	NEAR BA
12	12	-122.26	37.85	52.0	2491.0	474.0	1098.0	468.0	3.0750	213500.0	NEAR BA
13	13	-122.26	37.84	52.0	696.0	191.0	345.0	174.0	2.6736	191300.0	NEAR BA

In [30]:
```
from sklearn.model_selection import train_test_split

train_set, test_set = train_test_split(housing, test_size=0.2, random_state=456)
```

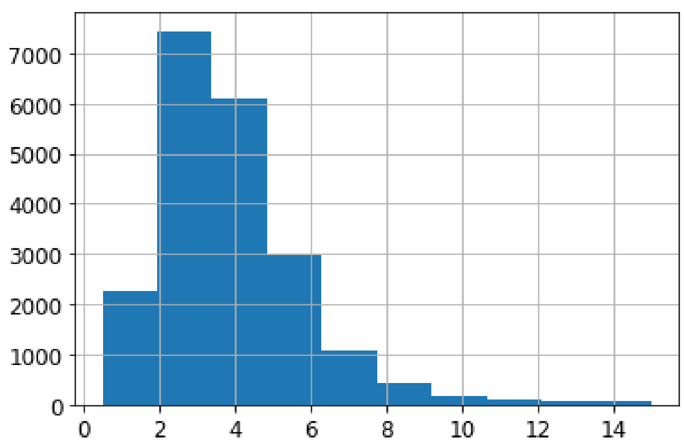
In [31]:
```
test_set.head()
```

Out[31]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
12165	-117.06	33.78	17.0	2813.0	565.0	1345.0	488.0	2.5847	145300.0	INLAND
11628	-118.07	33.80	22.0	1391.0	338.0	810.0	295.0	3.8792	218200.0	<1H OCEAN
8230	-118.20	33.77	24.0	2404.0	819.0	1566.0	753.0	1.5076	145800.0	NEAR OCEAN
6756	-118.11	34.11	50.0	2131.0	294.0	753.0	284.0	6.7099	352200.0	<1H OCEAN
17074	-122.21	37.48	20.0	505.0	216.0	326.0	216.0	2.9286	237500.0	NEAR BAY

In [32]:
```
housing["median_income"].hist()
```

Out[32]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7b08ed4198>
```



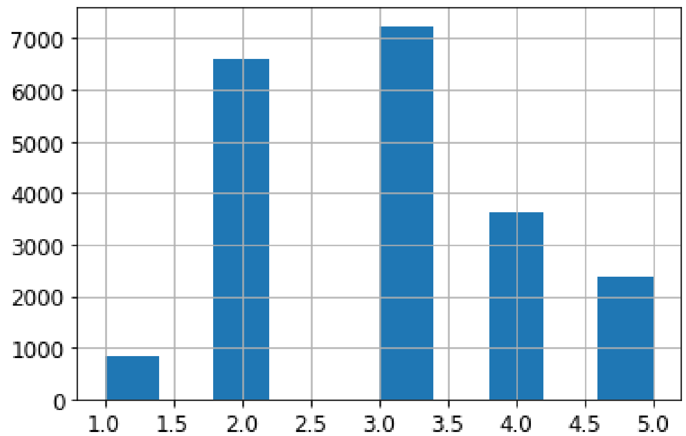
In [33]:
```
housing["income_cat"] = pd.cut(housing["median_income"],
                               bins=[0., 1.5, 3.0, 4.5, 6., np.inf],
                               labels=[1, 2, 3, 4, 5])
```

In [34]:
```
housing["income_cat"].value_counts()
```

Out[34]:
```
3    7236
2    6581
4    3639
5    2362
1     822
Name: income_cat, dtype: int64
```

In [35]:
```
housing["income_cat"].hist()
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7b09513208>
```



In [36]:
```
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 [37]:
```
strat_test_set["income_cat"].value_counts() / len(strat_test_set)
```

Out[37]:
```
3    0.350533
2    0.318798
4    0.176357
5    0.114583
1    0.039729
Name: income_cat, dtype: float64
```

In [38]:
```
housing["income_cat"].value_counts() / len(housing)
```

Out[38]:
```
3    0.350581
2    0.318847
4    0.176308
5    0.114438
1    0.039826
Name: income_cat, dtype: float64
```

In [39]:
```
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 [40]:
```
compare_props
```

Out[40]:

	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 [41]:
```
for set_ in (strat_train_set, strat_test_set):
    set_.drop("income_cat", axis=1, inplace=True)
```

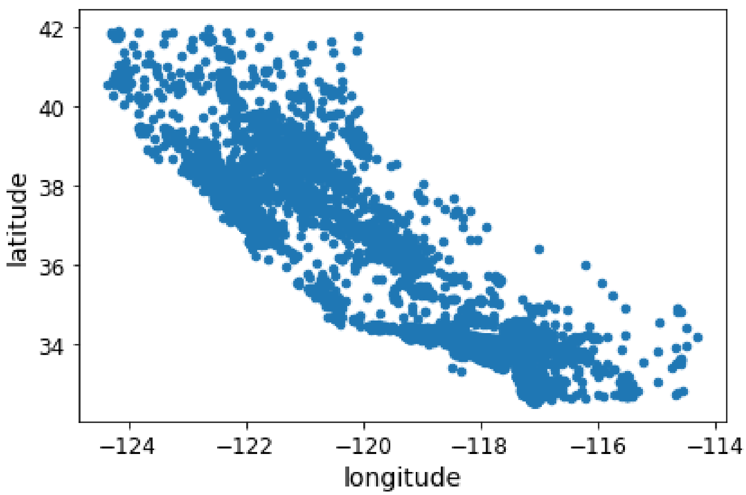
In [42]:
```
#Discover and visualize the data to gain insights

housing = strat_train_set.copy()
```

In [43]:
```
#  it is a good idea to create a scatterplot of all districts to visualize the data
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-43-04b7bb0c99fe> in <module>
      1 housing.plot(kind="scatter", x="longitude", y="latitude")
----> 2 save_fig("bad_visualization_plot")

<ipython-input-3-f6435d47e026> in save_fig(fig_id, tight_layout, fig_extension, resolution)
      9     if tight_layout:
     10         plt.tight_layout()
---> 11     plt.savefig(path, format=fig_extension, dpi=resolution)
     12 
     13 # Ignore useless warnings (see SciPy issue #5998)

~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args, **kwargs)
    714 def savefig(*args, **kwargs):
    715     fig = gcf()
--> 716     res = fig.savefig(*args, **kwargs)
    717     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    718     return res

~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self, fname, transparent, **kwargs)
   2178             self.patch.set_visible(frameon)
   2179 
-> 2180         self.canvas.print_figure(fname, **kwargs)
   2181 
   2182         if frameon:

~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filename, dpi, facecolor, edgecolor, orientatio
n, format, bbox_inches, **kwargs)
   2080                     orientation=orientation,
   2081                     bbox_inches_restore=_bbox_inches_restore,
-> 2082                     **kwargs)
   2083             finally:
   2084                 if bbox_inches and restore_bbox:

~\Anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(self, filename_or_obj, metadata, pil_kwargs, *arg
s, **kwargs)
    528             renderer = self.get_renderer()
    529             with cbook._setattr_cm(renderer, dpi=self.figure.dpi), \
--> 530                     cbook.open_file_cm(filename_or_obj, "wb") as fh:
    531                 _png.write_png(renderer._renderer, fh,
    532                                self.figure.dpi, metadata=metadata)

~\Anaconda3\lib\contextlib.py in __enter__(self)
    110         del self.args, self.kwds, self.func
    111         try:
--> 112             return next(self.gen)
    113         except StopIteration:
    114             raise RuntimeError("generator didn't yield") from None

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in open_file_cm(path_or_file, mode, encoding)
    445 def open_file_cm(path_or_file, mode="r", encoding=None):
```

```
    446     r"""Pass through file objects and context-manage `.PathLike`\s."""
--> 447     fh, opened = to_filehandle(path_or_file, mode, True, encoding)
    448     if opened:
    449         with fh:

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in to_filehandle(fname, flag, return_opened, encoding)
    430             fh = bz2.BZ2File(fname, flag)
    431         else:
--> 432             fh = open(fname, flag, encoding=encoding)
    433         opened = True
    434     elif hasattr(fname, 'seek'):

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

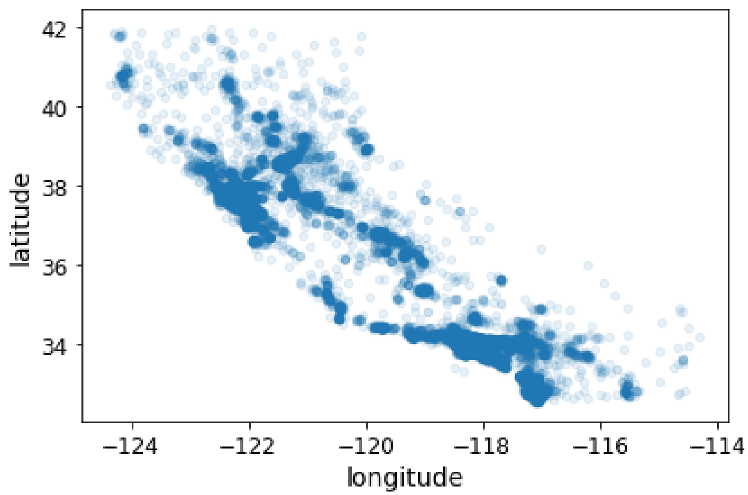


In [44]:
```
#  0.1 makes it much easier to visualize the places where there is a high density of data points
housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
save_fig("better_visualization_plot")
```

```
Saving figure better_visualization_plot
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-44-8737c00d5aba> in <module>
      1 housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.1)
----> 2 save_fig("better_visualization_plot")

<ipython-input-3-f6435d47e026> in save_fig(fig_id, tight_layout, fig_extension, resolution)
      9     if tight_layout:
     10         plt.tight_layout()
---> 11     plt.savefig(path, format=fig_extension, dpi=resolution)
     12 
     13 # Ignore useless warnings (see SciPy issue #5998)

~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args, **kwargs)
    714 def savefig(*args, **kwargs):
    715     fig = gcf()
--> 716     res = fig.savefig(*args, **kwargs)
    717     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    718     return res

~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self, fname, transparent, **kwargs)
   2178             self.patch.set_visible(frameon)
   2179 
-> 2180         self.canvas.print_figure(fname, **kwargs)
   2181 
   2182         if frameon:

~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filename, dpi, facecolor, edgecolor, orientatio
n, format, bbox_inches, **kwargs)
   2080                     orientation=orientation,
   2081                     bbox_inches_restore=_bbox_inches_restore,
-> 2082                     **kwargs)
   2083             finally:
   2084                 if bbox_inches and restore_bbox:

~\Anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(self, filename_or_obj, metadata, pil_kwargs, *arg
s, **kwargs)
    528             renderer = self.get_renderer()
    529             with cbook._setattr_cm(renderer, dpi=self.figure.dpi), \
--> 530                     cbook.open_file_cm(filename_or_obj, "wb") as fh:
    531                 _png.write_png(renderer._renderer, fh,
    532                                self.figure.dpi, metadata=metadata)

~\Anaconda3\lib\contextlib.py in __enter__(self)
    110         del self.args, self.kwds, self.func
    111         try:
--> 112             return next(self.gen)
    113         except StopIteration:
    114             raise RuntimeError("generator didn't yield") from None

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in open_file_cm(path_or_file, mode, encoding)
    445 def open_file_cm(path_or_file, mode="r", encoding=None):
```

```
    446         r"""Pass through file objects and context-manage `.PathLike`\s."""
--> 447         fh, opened = to_filehandle(path_or_file, mode, True, encoding)
    448         if opened:
    449             with fh:

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in to_filehandle(fname, flag, return_opened, encoding)
    430                 fh = bz2.BZ2File(fname, flag)
    431             else:
--> 432                 fh = open(fname, flag, encoding=encoding)
    433             opened = True
    434         elif hasattr(fname, 'seek'):

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



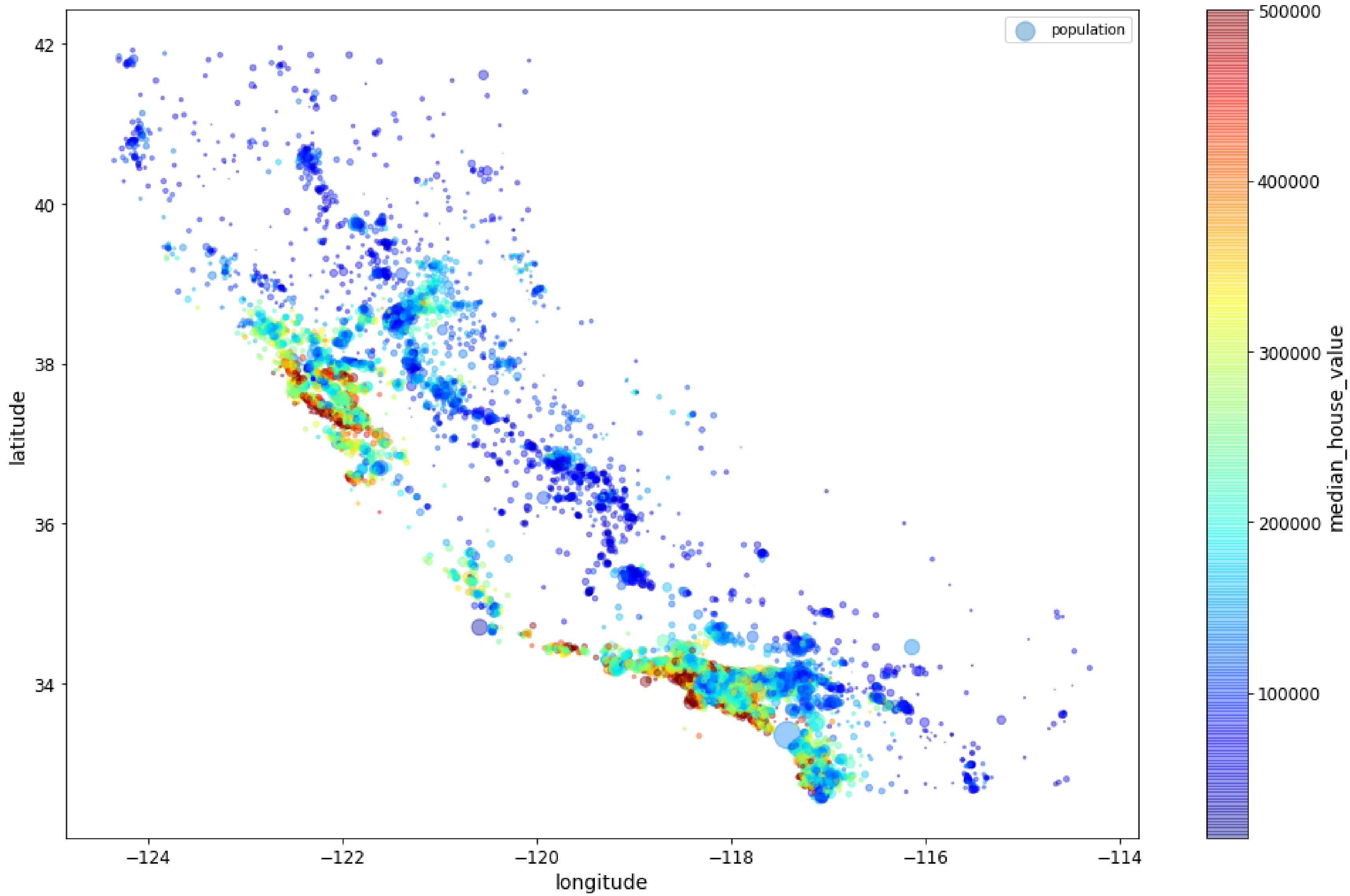
In [46]:

```
## This image tells you that the housing prices are very much related to the location and to the population density

housing.plot(kind="scatter", x="longitude", y="latitude", alpha=0.4,
    s=housing["population"]/100, label="population", figsize=(14,9),
    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-46-32cdb2663f27> in <module>
      4     sharex=False)
      5 plt.legend()
----> 6 save_fig("housing_prices_scatterplot")

<ipython-input-3-f6435d47e026> in save_fig(fig_id, tight_layout, fig_extension, resolution)
      9     if tight_layout:
     10         plt.tight_layout()
---> 11     plt.savefig(path, format=fig_extension, dpi=resolution)
     12 
     13 # Ignore useless warnings (see SciPy issue #5998)

~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args, **kwargs)
    714 def savefig(*args, **kwargs):
    715     fig = gcf()
--> 716     res = fig.savefig(*args, **kwargs)
    717     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    718     return res

~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self, fname, transparent, **kwargs)
   2178             self.patch.set_visible(frameon)
   2179 
-> 2180         self.canvas.print_figure(fname, **kwargs)
   2181 
   2182         if frameon:

~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filename, dpi, facecolor, edgecolor, orientatio
n, format, bbox_inches, **kwargs)
   2080                     orientation=orientation,
   2081                     bbox_inches_restore=_bbox_inches_restore,
-> 2082                     **kwargs)
   2083             finally:
   2084                 if bbox_inches and restore_bbox:

~\Anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(self, filename_or_obj, metadata, pil_kwargs, *arg
s, **kwargs)
    528             renderer = self.get_renderer()
    529             with cbook._setattr_cm(renderer, dpi=self.figure.dpi), \
--> 530                     cbook.open_file_cm(filename_or_obj, "wb") as fh:
    531                 _png.write_png(renderer._renderer, fh,
    532                                self.figure.dpi, metadata=metadata)

~\Anaconda3\lib\contextlib.py in __enter__(self)
    110         del self.args, self.kwds, self.func
    111         try:
--> 112             return next(self.gen)
    113         except StopIteration:
    114             raise RuntimeError("generator didn't yield") from None

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in open_file_cm(path_or_file, mode, encoding)
```

```
    445 def open_file_cm(path_or_file, mode="r", encoding=None):
    446     r"""Pass through file objects and context-manage `.PathLike`\s."""
--> 447     fh, opened = to_filehandle(path_or_file, mode, True, encoding)
    448     if opened:
    449         with fh:

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in to_filehandle(fname, flag, return_opened, encoding)
    430             fh = bz2.BZ2File(fname, flag)
    431         else:
--> 432             fh = open(fname, flag, encoding=encoding)
    433         opened = True
    434     elif hasattr(fname, 'seek'):

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



In [48]:
```
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-48-b4b47ed4e188> in <module>
      1 import matplotlib.image as mpimg
----> 2 california_img=mpimg.imread(PROJECT_ROOT_DIR + '/images/end_to_end_project/california.png')
      3 ax = housing.plot(kind="scatter", x="longitude", y="latitude", figsize=(10,7),
      4                        s=housing['population']/100, label="Population",
      5                        c="median_house_value", cmap=plt.get_cmap("jet"),

~\Anaconda3\lib\site-packages\matplotlib\image.py in imread(fname, format)
   1424             return handler(fd)
   1425         else:
-> 1426             with open(fname, 'rb') as fd:
   1427                 return handler(fd)
   1428     else:

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

In [49]:
```
corr_matrix = housing.corr()
```

In [50]:
```
corr_matrix["median_house_value"].sort_values(ascending=False)
```

Out[50]:
```
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
```

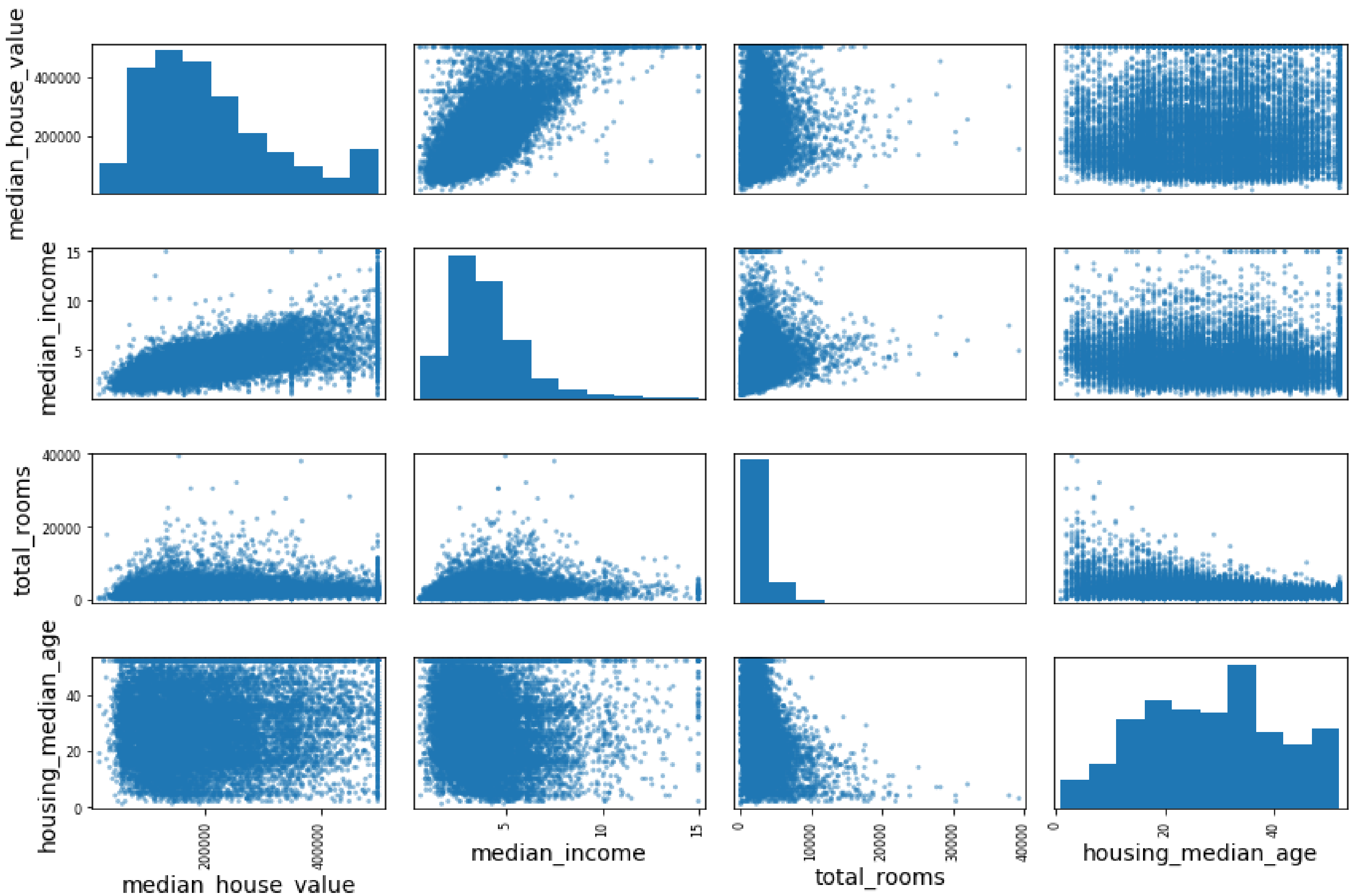
In [51]:

```python
# from pandas.tools.plotting import scatter_matrix # For older versions of Pandas
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-51-695aaa819555> in <module>
      5               "housing_median_age"]
      6 scatter_matrix(housing[attributes], figsize=(12, 8))
----> 7 save_fig("scatter_matrix_plot")

<ipython-input-3-f6435d47e026> in save_fig(fig_id, tight_layout, fig_extension, resolution)
      9     if tight_layout:
     10         plt.tight_layout()
---> 11     plt.savefig(path, format=fig_extension, dpi=resolution)
     12 
     13 # Ignore useless warnings (see SciPy issue #5998)

~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args, **kwargs)
    714 def savefig(*args, **kwargs):
    715     fig = gcf()
--> 716     res = fig.savefig(*args, **kwargs)
    717     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    718     return res

~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self, fname, transparent, **kwargs)
   2178             self.patch.set_visible(frameon)
   2179 
-> 2180         self.canvas.print_figure(fname, **kwargs)
   2181 
   2182         if frameon:

~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filename, dpi, facecolor, edgecolor, orientatio
n, format, bbox_inches, **kwargs)
   2080                     orientation=orientation,
   2081                     bbox_inches_restore=_bbox_inches_restore,
-> 2082                     **kwargs)
   2083             finally:
   2084                 if bbox_inches and restore_bbox:

~\Anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(self, filename_or_obj, metadata, pil_kwargs, *arg
s, **kwargs)
    528             renderer = self.get_renderer()
    529             with cbook._setattr_cm(renderer, dpi=self.figure.dpi), \
--> 530                     cbook.open_file_cm(filename_or_obj, "wb") as fh:
    531                 _png.write_png(renderer._renderer, fh,
    532                                self.figure.dpi, metadata=metadata)

~\Anaconda3\lib\contextlib.py in __enter__(self)
    110         del self.args, self.kwds, self.func
    111         try:
--> 112             return next(self.gen)
    113         except StopIteration:
    114             raise RuntimeError("generator didn't yield") from None

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in open_file_cm(path_or_file, mode, encoding)
```

```
    445 def open_file_cm(path_or_file, mode="r", encoding=None):
    446     r"""Pass through file objects and context-manage `.PathLike`\s."""
--> 447     fh, opened = to_filehandle(path_or_file, mode, True, encoding)
    448     if opened:
    449         with fh:

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in to_filehandle(fname, flag, return_opened, encoding)
    430             fh = bz2.BZ2File(fname, flag)
    431         else:
--> 432             fh = open(fname, flag, encoding=encoding)
    433         opened = True
    434     elif hasattr(fname, 'seek'):

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

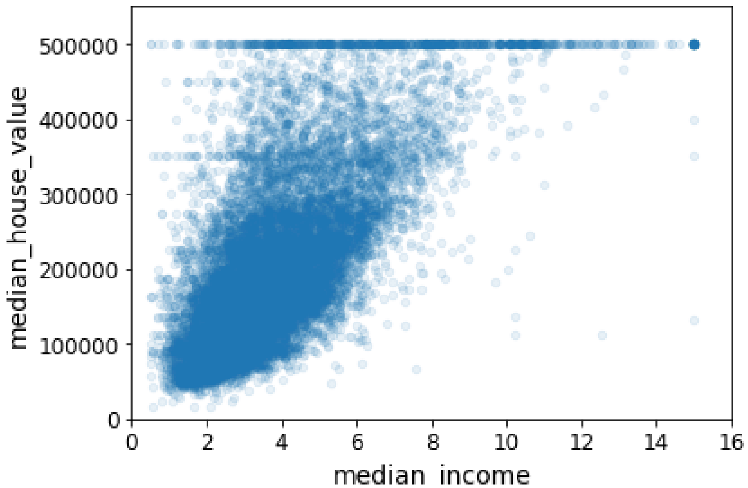


In [54]:
```
housing.plot(kind="scatter", x="median_income", y="median_house_value",
             alpha=0.1)
plt.axis([0, 16, 0, 550000])
save_fig("income_vs_house_value_scatterplot")
```

```
Saving figure income_vs_house_value_scatterplot
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-54-50ae954e87f8> in <module>
      2              alpha=0.1)
      3 plt.axis([0, 16, 0, 550000])
----> 4 save_fig("income_vs_house_value_scatterplot")

<ipython-input-3-f6435d47e026> in save_fig(fig_id, tight_layout, fig_extension, resolution)
      9     if tight_layout:
     10         plt.tight_layout()
---> 11     plt.savefig(path, format=fig_extension, dpi=resolution)
     12 
     13 # Ignore useless warnings (see SciPy issue #5998)

~\Anaconda3\lib\site-packages\matplotlib\pyplot.py in savefig(*args, **kwargs)
    714 def savefig(*args, **kwargs):
    715     fig = gcf()
--> 716     res = fig.savefig(*args, **kwargs)
    717     fig.canvas.draw_idle()   # need this if 'transparent=True' to reset colors
    718     return res

~\Anaconda3\lib\site-packages\matplotlib\figure.py in savefig(self, fname, transparent, **kwargs)
   2178             self.patch.set_visible(frameon)
   2179 
-> 2180         self.canvas.print_figure(fname, **kwargs)
   2181 
   2182         if frameon:

~\Anaconda3\lib\site-packages\matplotlib\backend_bases.py in print_figure(self, filename, dpi, facecolor, edgecolor, orientatio
n, format, bbox_inches, **kwargs)
   2080                     orientation=orientation,
   2081                     bbox_inches_restore=_bbox_inches_restore,
-> 2082                     **kwargs)
   2083             finally:
   2084                 if bbox_inches and restore_bbox:

~\Anaconda3\lib\site-packages\matplotlib\backends\backend_agg.py in print_png(self, filename_or_obj, metadata, pil_kwargs, *arg
s, **kwargs)
    528             renderer = self.get_renderer()
    529             with cbook._setattr_cm(renderer, dpi=self.figure.dpi), \
--> 530                     cbook.open_file_cm(filename_or_obj, "wb") as fh:
    531                 _png.write_png(renderer._renderer, fh,
    532                                self.figure.dpi, metadata=metadata)

~\Anaconda3\lib\contextlib.py in __enter__(self)
    110         del self.args, self.kwds, self.func
    111         try:
--> 112             return next(self.gen)
    113         except StopIteration:
    114             raise RuntimeError("generator didn't yield") from None

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in open_file_cm(path_or_file, mode, encoding)
```

```
    445 def open_file_cm(path_or_file, mode="r", encoding=None):
    446     r"""Pass through file objects and context-manage `.PathLike`\s."""
--> 447     fh, opened = to_filehandle(path_or_file, mode, True, encoding)
    448     if opened:
    449         with fh:

~\Anaconda3\lib\site-packages\matplotlib\cbook\__init__.py in to_filehandle(fname, flag, return_opened, encoding)
    430             fh = bz2.BZ2File(fname, flag)
    431         else:
--> 432             fh = open(fname, flag, encoding=encoding)
    433         opened = True
    434     elif hasattr(fname, 'seek'):

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

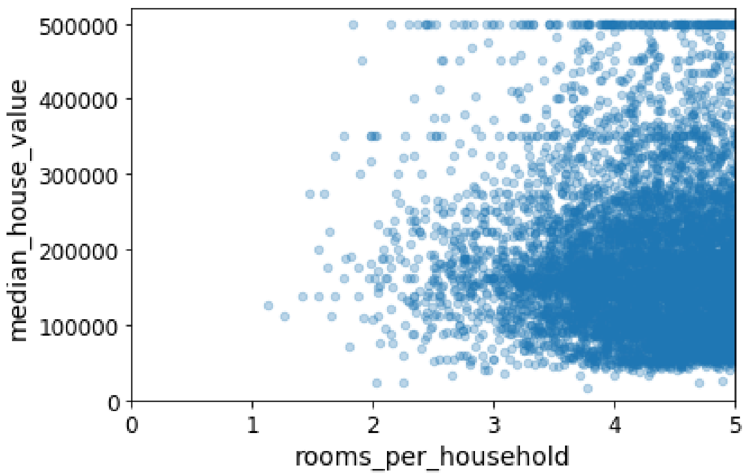


In [55]:
```
housing["rooms_per_household"] = housing["total_rooms"]/housing["households"]
housing["bedrooms_per_room"] = housing["total_bedrooms"]/housing["total_rooms"]
housing["population_per_household"]=housing["population"]/housing["households"]
```

In [56]:
```
corr_matrix = housing.corr()
corr_matrix["median_house_value"].sort_values(ascending=False)
```

Out[56]:
```
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 [63]:
```
housing.plot(kind="scatter", x="rooms_per_household", y="median_house_value",
             alpha=0.3)
plt.axis([0, 5, 0, 520000])
plt.show()
```



In [64]:
```
housing.describe()
```

Out[64]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	roor
count	16512.000000	16512.000000	16512.000000	16512.000000	16354.000000	16512.000000	16512.000000	16512.000000	16512.000000	
mean	-119.575834	35.639577	28.653101	2622.728319	534.973890	1419.790819	497.060380	3.875589	206990.920724	
std	2.001860	2.138058	12.574726	2138.458419	412.699041	1115.686241	375.720845	1.904950	115703.014830	
min	-124.350000	32.540000	1.000000	6.000000	2.000000	3.000000	2.000000	0.499900	14999.000000	
25%	-121.800000	33.940000	18.000000	1443.000000	295.000000	784.000000	279.000000	2.566775	119800.000000	
50%	-118.510000	34.260000	29.000000	2119.500000	433.000000	1164.000000	408.000000	3.540900	179500.000000	
75%	-118.010000	37.720000	37.000000	3141.000000	644.000000	1719.250000	602.000000	4.744475	263900.000000	
max	-114.310000	41.950000	52.000000	39320.000000	6210.000000	35682.000000	5358.000000	15.000100	500001.000000	

In [65]:
```
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 [66]:
```
sample_incomplete_rows = housing[housing.isnull().any(axis=1)].head()
sample_incomplete_rows
```

Out[66]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity
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

In [67]:
```
sample_incomplete_rows.dropna(subset=["total_bedrooms"])    # option 1
```

Out[67]:

longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity

In [69]:
```python
sample_incomplete_rows.drop("total_bedrooms", axis=1)       # option 2
```

Out[69]:

	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 [70]:
```python
median = housing["total_bedrooms"].median()
sample_incomplete_rows["total_bedrooms"].fillna(median, inplace=True) # option 3
sample_incomplete_rows
```

Out[70]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity
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

In [71]:
```python
try:
    from sklearn.impute import SimpleImputer # Scikit-Learn 0.20+
except ImportError:
    from sklearn.preprocessing import Imputer as SimpleImputer

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

In [72]:
```python
housing_num = housing.drop('ocean_proximity', axis=1)
# alternatively: housing_num = housing.select_dtypes(include=[np.number])
```

In [73]:
```python
imputer.fit(housing_num)
```

Out[73]:
```
SimpleImputer(add_indicator=False, copy=True, fill_value=None,
              missing_values=nan, strategy='median', verbose=0)
```

In [74]:
```python
imputer.statistics_
```

Out[74]:
```
array([-118.51  ,   34.26  ,   29.    , 2119.5   ,  433.    , 1164.    ,
        408.    ,    3.5409])
```

In [75]:
```
housing_num.median().values
```

Out[75]:
```
array([-118.51  ,   34.26  ,   29.    , 2119.5   ,  433.    , 1164.    ,
        408.    ,    3.5409])
```

In [76]:
```
X = imputer.transform(housing_num)
```

In [77]:
```
housing_tr = pd.DataFrame(X, columns=housing_num.columns,
                          index=housing.index)
```

In [78]:
```
housing_tr.loc[sample_incomplete_rows.index.values]
```

Out[78]:

	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 [79]:
```
imputer.strategy
```

Out[79]:
```
'median'
```

In [80]:
```
housing_tr = pd.DataFrame(X, columns=housing_num.columns,
                          index=housing_num.index)
housing_tr.head()
```

Out[80]:

	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

In [81]:
```python
# Now let's preprocess the categorical input feature, ocean_proximity:
housing_cat = housing[['ocean_proximity']]
housing_cat.head(10)
```

Out[81]:

	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 [82]:
```python
try:
    from sklearn.preprocessing import OrdinalEncoder
except ImportError:
    from future_encoders import OrdinalEncoder # Scikit-Learn < 0.20
```

In [83]:
```python
ordinal_encoder = OrdinalEncoder()
housing_cat_encoded = ordinal_encoder.fit_transform(housing_cat)
housing_cat_encoded[:10]
```

Out[83]:
```
array([[0.],
       [0.],
       [4.],
       [1.],
       [0.],
       [1.],
       [0.],
       [1.],
       [0.],
       [0.]])
```

In [84]:
```python
ordinal_encoder.categories_
```

Out[84]:
```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

In [85]:
```python
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[85]:
```
<16512x5 sparse matrix of type '<class 'numpy.float64'>'
	with 16512 stored elements in Compressed Sparse Row format>
```

In [86]:
```python
housing_cat_1hot.toarray()
```

Out[86]:
```
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 [87]:
```python
# Alternatively, you can set sparse=False when creating the OneHotEncoder:
cat_encoder = OneHotEncoder(sparse=False)
housing_cat_1hot = cat_encoder.fit_transform(housing_cat)
housing_cat_1hot
```

Out[87]:
```
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 [88]:
```python
cat_encoder.categories_
```

Out[88]:
```
[array(['<1H OCEAN', 'INLAND', 'ISLAND', 'NEAR BAY', 'NEAR OCEAN'],
       dtype=object)]
```

In [89]:
```python
housing.columns
```

Out[89]:
```
Index(['longitude', 'latitude', 'housing_median_age', 'total_rooms',
       'total_bedrooms', 'population', 'households', 'median_income',
       'ocean_proximity'],
      dtype='object')
```

In [90]:

```python
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 [91]:

```python
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 [92]:
```
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[92]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	ocean_proximity	rooms_per_household
17606	-121.89	37.29	38	1568	351	710	339	2.7042	<1H OCEAN	4.62537
18632	-121.93	37.05	14	679	108	306	113	6.4214	<1H OCEAN	6.00885
14650	-117.2	32.77	31	1952	471	936	462	2.8621	NEAR OCEAN	4.22511
3230	-119.61	36.31	25	1847	371	1460	353	1.8839	INLAND	5.23229
3555	-118.59	34.23	17	6592	1525	4459	1463	3.0347	<1H OCEAN	4.50581

In [93]:
```
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 [94]:
```
housing_num_tr
```

Out[94]:
```
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 [95]:
```
try:
    from sklearn.compose import ColumnTransformer
except ImportError:
    from future_encoders import ColumnTransformer # Scikit-Learn < 0.20
```

In [96]:
```
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 [97]:
```
housing_prepared
```

Out[97]:
```
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 [98]:
```
housing_prepared.shape
```

Out[98]:
```
(16512, 16)
```

In [99]:
```
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 [100]:
```
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 [101]:
```
from sklearn.pipeline import FeatureUnion

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

In [102]:
```
old_housing_prepared = old_full_pipeline.fit_transform(housing)
old_housing_prepared
```

Out[102]:
```
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 [103]:
```
np.allclose(housing_prepared, old_housing_prepared)
```

Out[103]:
```
True
```

```
In [104]: # Select and train a model
          #  working Linear Regression model

          from sklearn.linear_model import LinearRegression

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

Out[104]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
In [105]: # let's try the full preprocessing pipeline on a few training instances
          # It works, although the predictions are not exactly accurate (e.g., the second prediction is off by more than 50%!).
          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 [106]: print("Labels:", list(some_labels))

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

```
In [107]: some_data_prepared

Out[107]: 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 [108]:
```
# typical prediction error of $68,628 is not very satisfying. This is an example of a model underfitting the training data.
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[108]:
```
68628.19819848923
```

In [109]:
```
from sklearn.metrics import mean_absolute_error

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

Out[109]:
```
49439.89599001897
```

In [111]:
```
# select a more powerful model, to feed the training algorithm with better features, or to reduce the constraints on the model.
# Let’s train a DecisionTreeRegressor. This is a powerful model, capable of finding complex nonlinear relationships in the data
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(random_state=456)
tree_reg.fit(housing_prepared, housing_labels)
```

Out[111]:
```
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=456, splitter='best')
```

In [112]:
```
#  Could this model really be absolutely perfect
#$  the model has badly overfit the data
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[112]:
```
0.0
```

In [113]:
```
### Fine-tune your model ####
# The following code performs K-fold cross-validation: it randomly splits the training set into 10 distinct subsets called
# folds, then it trains and evaluates the Decision Tree model 10 times, picking a different fold for evaluation every time and
# training on the other 9 folds. The result is an array containing the 10 evaluation scores:


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 [114]:
```
def display_scores(scores):
    print("Scores:", scores)
    print("Mean:", scores.mean())
    print("Standard deviation:", scores.std())

display_scores(tree_rmse_scores)


## Now the Decision Tree doesn’t look as good as it did earlier. In fact,
## it seems to perform worse than the Linear Regression model!
```

```
Scores: [69685.73500968 66395.31605095 72034.76264608 68598.59209804
 71700.64818069 73674.45171191 70677.84500106 70836.72321577
 75232.90321974 70927.34637046]
Mean: 70976.43235043726
Standard deviation: 2354.9763744322877
```

In [115]:
```
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 [117]:
```
##  we specify n_estimators=10 to avoid a warning about the fact that the default value is going to change to 100 in Scikit-Lear
n 0.22.
# try one last model now: the RandomForestRegressor.
from sklearn.ensemble import RandomForestRegressor

forest_reg = RandomForestRegressor(n_estimators=10, random_state=456)
forest_reg.fit(housing_prepared, housing_labels)
```

Out[117]:
```
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=456, verbose=0,
                      warm_start=False)
```

In [118]:
```
housing_predictions = forest_reg.predict(housing_prepared)
forest_mse = mean_squared_error(housing_labels, housing_predictions)
forest_rmse = np.sqrt(forest_mse)
forest_rmse

## Random Forests look very promising
```

Out[118]:
```
22803.989217850758
```

In [119]:
```
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: [52172.27833732 49843.26720706 51931.98899916 55314.52433594
 51941.40376951 55218.3402264  52484.1762354  50145.52727709
 55579.43135802 52896.94021734]
Mean: 52752.78779632276
Standard deviation: 1940.5270472422412
```

In [120]:
```
scores = cross_val_score(lin_reg, housing_prepared, housing_labels, scoring="neg_mean_squared_error", cv=10)
pd.Series(np.sqrt(-scores)).describe()
```

Out[120]:
```
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 [121]:
```
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[121]:
```
111094.6308539982
```

In [123]:
```python
## evaluate all the possible combinations of hyperparameter values

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=456)
# 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[123]:
```
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=456,
                                             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 [124]:
```python
## The best hyperparameter combination found:
grid_search.best_params_
```

Out[124]:
```
{'max_features': 8, 'n_estimators': 30}
```

In [125]:
```
grid_search.best_estimator_
```

Out[125]:
```
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=456, verbose=0,
                      warm_start=False)
```

In [126]:
```
## Let's look at the score of each hyperparameter combination tested during the grid search:

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

```
64468.241451500224 {'max_features': 2, 'n_estimators': 3}
56020.18893461684 {'max_features': 2, 'n_estimators': 10}
53315.8660113502 {'max_features': 2, 'n_estimators': 30}
59943.83252456937 {'max_features': 4, 'n_estimators': 3}
53171.32152057854 {'max_features': 4, 'n_estimators': 10}
50645.705692833915 {'max_features': 4, 'n_estimators': 30}
59142.38014614101 {'max_features': 6, 'n_estimators': 3}
52676.33094596141 {'max_features': 6, 'n_estimators': 10}
50317.055877563944 {'max_features': 6, 'n_estimators': 30}
58386.66081229148 {'max_features': 8, 'n_estimators': 3}
52178.13689933393 {'max_features': 8, 'n_estimators': 10}
50190.581922030906 {'max_features': 8, 'n_estimators': 30}
62801.3232888978 {'bootstrap': False, 'max_features': 2, 'n_estimators': 3}
54724.442718324346 {'bootstrap': False, 'max_features': 2, 'n_estimators': 10}
60750.13269511117 {'bootstrap': False, 'max_features': 3, 'n_estimators': 3}
53126.6236793922 {'bootstrap': False, 'max_features': 3, 'n_estimators': 10}
58636.38654396156 {'bootstrap': False, 'max_features': 4, 'n_estimators': 3}
52012.67396220179 {'bootstrap': False, 'max_features': 4, 'n_estimators': 10}
```

In [127]:
```
pd.DataFrame(grid_search.cv_results_)
```

Out[127]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_features	param_n_estimators	param_bootstrap	params	split0_test_score
0	0.122098	0.003035	0.005807	0.000402	2	3	NaN	{'max_features': 2, 'n_estimators': 3}	-3.885432e+09
1	0.431626	0.027912	0.016717	0.000877	2	10	NaN	{'max_features': 2, 'n_estimators': 10}	-2.927840e+09
2	1.199080	0.012779	0.047028	0.001670	2	30	NaN	{'max_features': 2, 'n_estimators': 30}	-2.696168e+09
3	0.185539	0.002339	0.006004	0.000633	4	3	NaN	{'max_features': 4, 'n_estimators': 3}	-3.593934e+09
4	0.651067	0.028065	0.017216	0.001476	4	10	NaN	{'max_features': 4, 'n_estimators': 10}	-2.659265e+09
5	2.035062	0.099283	0.047837	0.003660	4	30	NaN	{'max_features': 4, 'n_estimators': 30}	-2.378088e+09
6	0.266990	0.015927	0.006605	0.000490	6	3	NaN	{'max_features': 6, 'n_estimators': 3}	-3.389611e+09
7	0.937065	0.077307	0.016815	0.000749	6	10	NaN	{'max_features': 6, 'n_estimators': 10}	-2.572952e+09
8	2.732968	0.070698	0.050836	0.009689	6	30	NaN	{'max_features': 6, 'n_estimators': 30}	-2.343883e+09
9	0.336639	0.016826	0.006004	0.000633	8	3	NaN	{'max_features': 8, 'n_estimators': 3}	-3.254609e+09
10	1.178857	0.065689	0.018010	0.002192	8	10	NaN	{'max_features': 8, 'n_estimators': 10}	-2.522807e+09
11	3.361637	0.094524	0.046236	0.000753	8	30	NaN	{'max_features': 8, 'n_estimators': 30}	-2.346582e+09

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_max_features	param_n_estimators	param_bootstrap	params	split0_test_score
12	0.191335	0.002137	0.006606	0.000490	2	3	False	{'bootstrap': False, 'max_features': 2, 'n_est...	-3.790663e+09
13	0.638260	0.007079	0.018216	0.000399	2	10	False	{'bootstrap': False, 'max_features': 2, 'n_est...	-2.862670e+09
14	0.247182	0.002687	0.006809	0.000402	3	3	False	{'bootstrap': False, 'max_features': 3, 'n_est...	-3.382557e+09
15	0.942389	0.101655	0.020215	0.002928	3	10	False	{'bootstrap': False, 'max_features': 3, 'n_est...	-2.590158e+09
16	0.302418	0.003192	0.006405	0.000490	4	3	False	{'bootstrap': False, 'max_features': 4, 'n_est...	-3.229346e+09
17	1.056156	0.061352	0.018813	0.001166	4	10	False	{'bootstrap': False, 'max_features': 4, 'n_est...	-2.537274e+09

18 rows × 23 columns

In [130]:
```
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=456)
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[130]:
```
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 0x0000007B15B615C
0>,
                                        'n_estimators': <scipy.stats._distn_infrastructure.rv_frozen object at 0x0000007B15B6132
0>},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring='neg_mean_squared_error',
                   verbose=0)
```

In [131]:
```
cvres = rnd_search.cv_results_
for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
    print(np.sqrt(-mean_score), params)
```

```
49138.34350627016 {'max_features': 7, 'n_estimators': 180}
51016.640434110166 {'max_features': 5, 'n_estimators': 15}
50745.93937668918 {'max_features': 3, 'n_estimators': 72}
50499.23542597587 {'max_features': 5, 'n_estimators': 21}
49199.76606594992 {'max_features': 7, 'n_estimators': 122}
50704.7725743003 {'max_features': 3, 'n_estimators': 75}
50578.41323880302 {'max_features': 3, 'n_estimators': 88}
49278.36148584473 {'max_features': 5, 'n_estimators': 100}
50309.221904590486 {'max_features': 3, 'n_estimators': 150}
63405.37600274506 {'max_features': 5, 'n_estimators': 2}
```

In [132]:
```
feature_importances = grid_search.best_estimator_.feature_importances_
feature_importances
```

Out[132]:
```
array([7.27757602e-02, 6.43307108e-02, 4.15876610e-02, 1.54680286e-02,
       1.47917922e-02, 1.54825018e-02, 1.36596396e-02, 3.77683075e-01,
       4.14196745e-02, 1.12041045e-01, 6.31432957e-02, 7.94461562e-03,
       1.54802112e-01, 9.28233619e-05, 2.07122723e-03, 2.70603698e-03])
```

In [133]:
```
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[133]:
```
[(0.3776830751371189, 'median_income'),
 (0.1548021123341523, 'INLAND'),
 (0.11204104488089804, 'pop_per_hhold'),
 (0.07277576023549523, 'longitude'),
 (0.0643307107957676, 'latitude'),
 (0.06314329569219225, 'bedrooms_per_room'),
 (0.04158766102908416, 'housing_median_age'),
 (0.041419674485049914, 'rooms_per_hhold'),
 (0.015482501822875196, 'population'),
 (0.015468028588720607, 'total_rooms'),
 (0.014791792176536374, 'total_bedrooms'),
 (0.013659639634435587, 'households'),
 (0.007944615618082284, '<1H OCEAN'),
 (0.0027060369796617053, 'NEAR OCEAN'),
 (0.002071227227994808, 'NEAR BAY'),
 (9.282336193511499e-05, 'ISLAND')]
```

In [134]:
```
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 [135]:
```
final_rmse
```

Out[135]:
```
47585.462194452804
```

In [136]:
```
###  We can compute a 95% confidence interval for the test RMSE:
from scipy import stats
```

In [137]:
```
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[137]:
```
array([45576.45099334, 49513.02393685])
```

In [138]:
```
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[138]:
```
(45576.450993336584, 49513.02393685269)
```

In [139]:
```
## Alternatively, we could use a z-scores rather than t-scores:
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[139]:
```
(45577.053174264045, 49512.46962604134)
```

In [140]:
```
###  A full pipeline with both preparation and prediction
full_pipeline_with_predictor = Pipeline([
        ("preparation", full_pipeline),
        ("linear", LinearRegression())
    ])

full_pipeline_with_predictor.fit(housing, housing_labels)
full_pipeline_with_predictor.predict(some_data)
```

Out[140]:
```
array([210644.60459286, 317768.80697211, 210956.43331178,  59218.98886849,
       189747.55849879])
```

In [141]:
```
##  Model persistence using joblib

my_model = full_pipeline_with_predictor
```

In [142]:
```
from sklearn.externals import joblib
joblib.dump(my_model, "my_model.pkl") # DIFF
#...
my_model_loaded = joblib.load("my_model.pkl") # DIFF
```

```
C:\Users\nilesh\Anaconda3\lib\site-packages\sklearn\externals\joblib\__init__.py:15: DeprecationWarning: sklearn.externals.jobli
b is deprecated in 0.21 and will be removed in 0.23. Please import this functionality directly from joblib, which can be install
ed with: pip install joblib. If this warning is raised when loading pickled models, you may need to re-serialize those models wi
th scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
```

In [ ]:
```
### Example SciPy distributions for RandomizedSearchCV
```

In [143]:
```
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()
```