

Advanced Data Analysis

DATA 71200

Class 4

Weekly Schedule

7-Jun **Inspecting Data**

8-Jun Representing Data

9-Jun Evaluation Methods

10-Jun Async

Assignments for today

- ▶ **DataCamp**
 - *Introduction to Python (If Needed)*
 - AI Fundamentals
 - Introduction to AI
 - Data Manipulation with pandas
 - Transforming Data
 - Aggregating Data
 - Slicing and Indexing
 - *Creating and Visualizing DataFrames (Optional)*
 - *Writing Efficient Code with pandas (Optional)*
- ▶ **Reading**
 - Ch 2: End-to-End Machine Learning Project. in Géron, Aurélien. (2019). Hands-On Machine Learning with Scikit-Learn, Keras and TensorFlow' O'Reilly Media, Inc. 33–66

Frame the Problem

- ▶ **“what exactly is the business (research) objective”**
 - “how does the company (researcher) expect to use and benefit from this model?”
 - will determine
 - “how you frame the problem”
 - “what algorithms you will select”
 - “what performance measure you will use to evaluate your model”
 - “how much effort you should spend tweaking it”

Pipeline

- ▶ **“sequence of data processing components is called a data pipeline”**
 - “components typically run asynchronously” and “is fairly self-contained”
 - each component
 - “pulls in a large amount of data”
 - “processes it”
 - “spits out the result in another data store”
 - a later “component in the pipeline pulls this data and spits out its own output”

Pipeline

- ▶ “**data flow graph show how the components are connected**”
- ▶ “**if a component breaks down, the downstream components can often continue to run normally (at least for a while) by just using the last output from the broken component**”
 - “but a broken component can go unnoticed for some time if proper monitoring is not implemented”

Creating an Isolated Environment

If you would like to work in an isolated environment (which is strongly recommended so you can work on different projects without having conflicting library versions), install virtualenv by running the following pip command:

```
$ pip3 install --user --upgrade virtualenv  
Collecting virtualenv  
[...]  
Successfully installed virtualenv
```

Now you can create an isolated Python environment by typing:

```
$ cd $ML_PATH  
$ virtualenv env  
Using base prefix '[...]'  
New python executable in [...]/ml/env/bin/python3.5  
Also creating executable in [...]/ml/env/bin/python  
Installing setuptools, pip, wheel...done.
```

Now every time you want to activate this environment, just open a terminal and type:

```
$ cd $ML_PATH  
$ source env/bin/activate
```

While the environment is active, any package you install using pip will be installed in this isolated environment, and Python will only have access to these packages (if you also want access to the system's site packages, you should create the environment using virtualenv's `--system-site-packages` option). Check out virtualenv's documentation for more information.

Inspecting Data to Gain Insights

- ▶ **Data size and type**
- ▶ **Summary statistics**
- ▶ **Histograms**
- ▶ **Visualizing Geographic Data**

```
In [6]: 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
```

Figure 2-6. Housing info

```
In [8]: housing.describe()
```

Out[8]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms
count	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000
mean	-119.569704	35.631861	28.639486	2635.763081	537.870553
std	2.003532	2.135952	12.585558	2181.615252	421.385070
min	-124.350000	32.540000	1.000000	2.000000	1.000000
25%	-121.800000	33.930000	18.000000	1447.750000	296.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000
75%	-118.010000	37.710000	37.000000	3148.000000	647.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000

Figure 2-7. Summary of each numerical attribute

```
%matplotlib inline # only in a Jupyter notebook
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
plt.show()
```

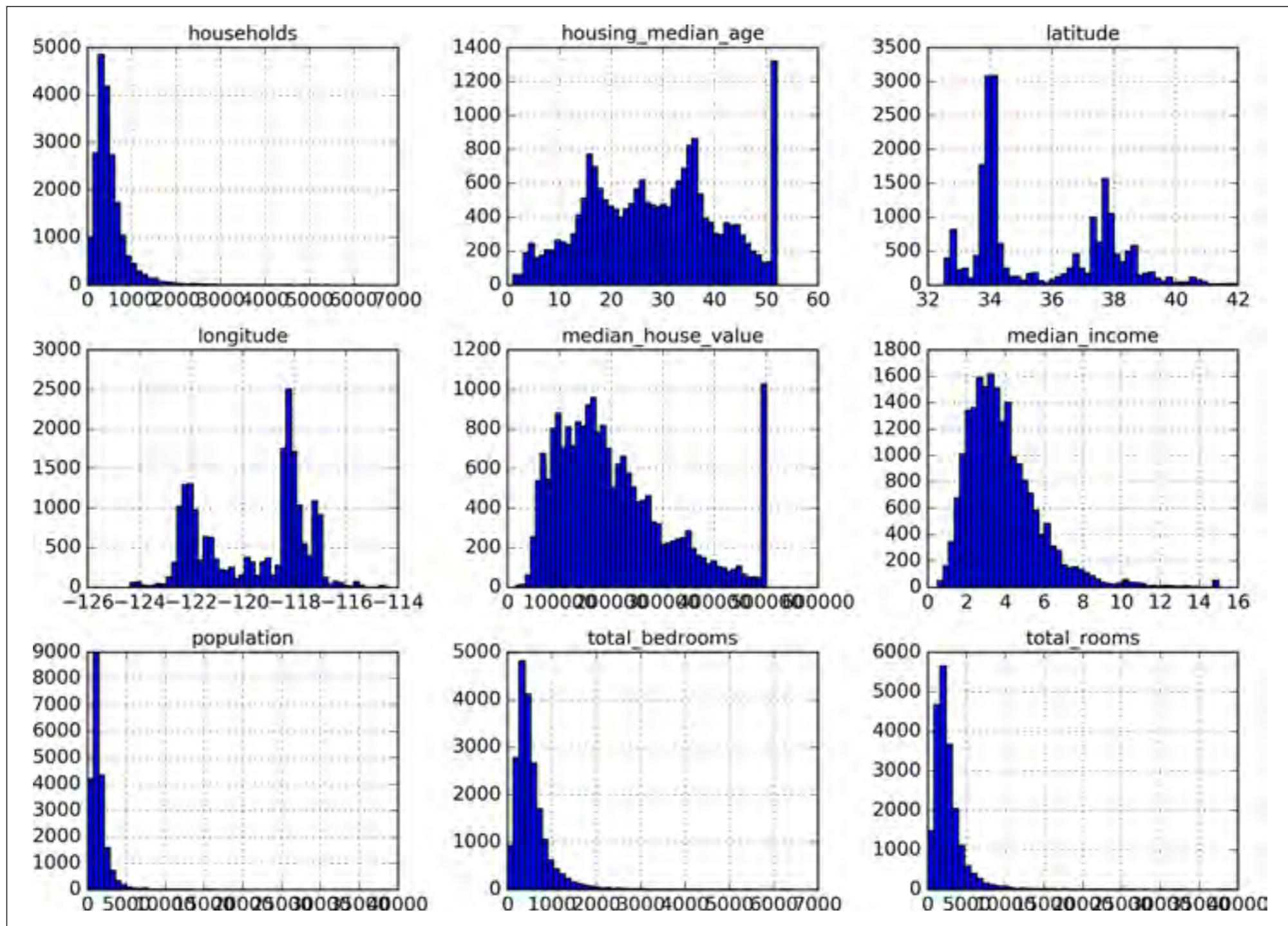


Figure 2-8. A histogram for each numerical attribute

```
%matplotlib inline # only in a Jupyter notebook
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
plt.show()
```

First, the median income attribute does not look like it is expressed in US dollars (USD). After checking with the team that collected the data, you are told that the data has been scaled and capped at 15 (actually 15.0001) for higher median incomes, and at 0.5 (actually 0.4999) for lower median incomes. Working with preprocessed attributes is common in Machine Learning, and it is not necessarily a problem, but you should try to understand how the data was computed.

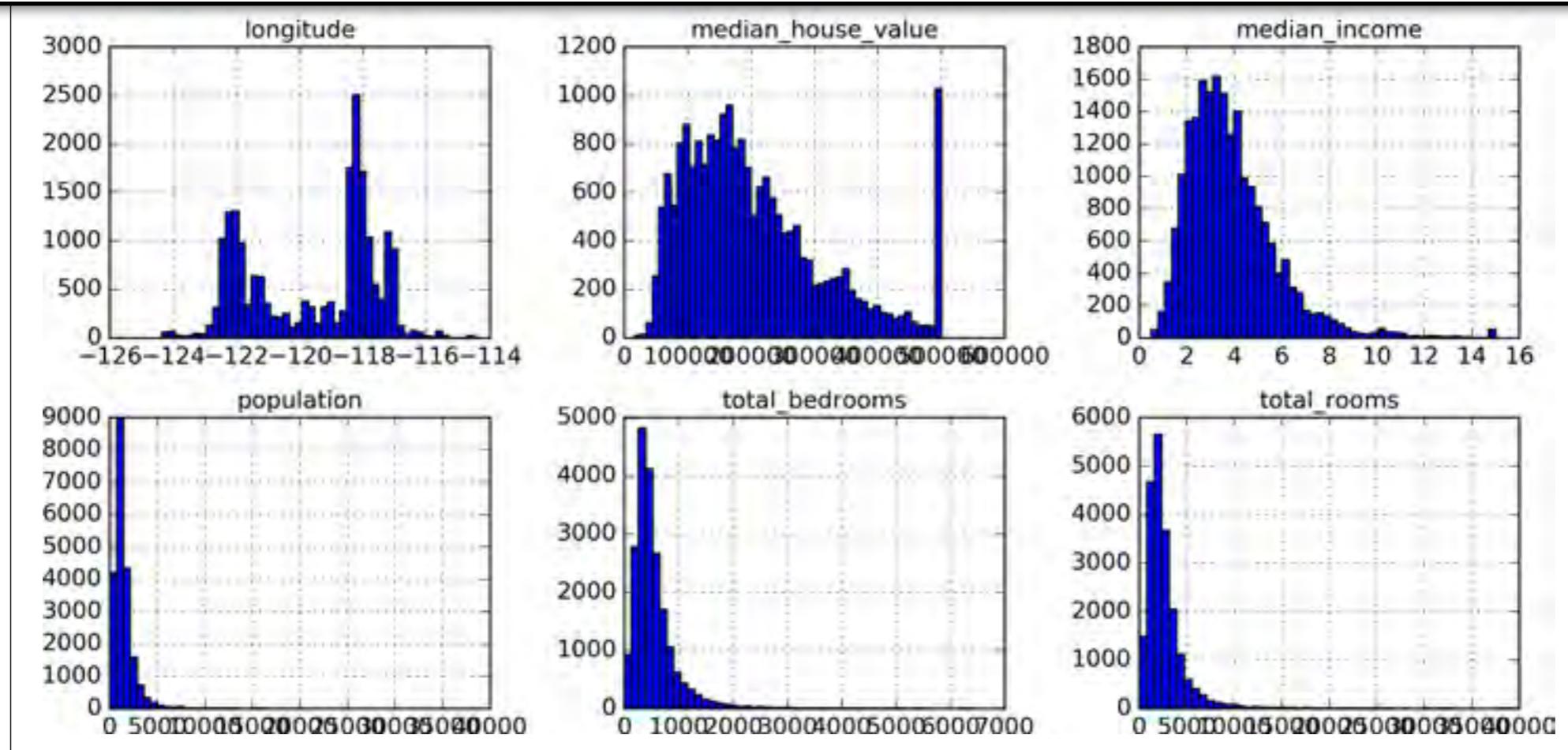
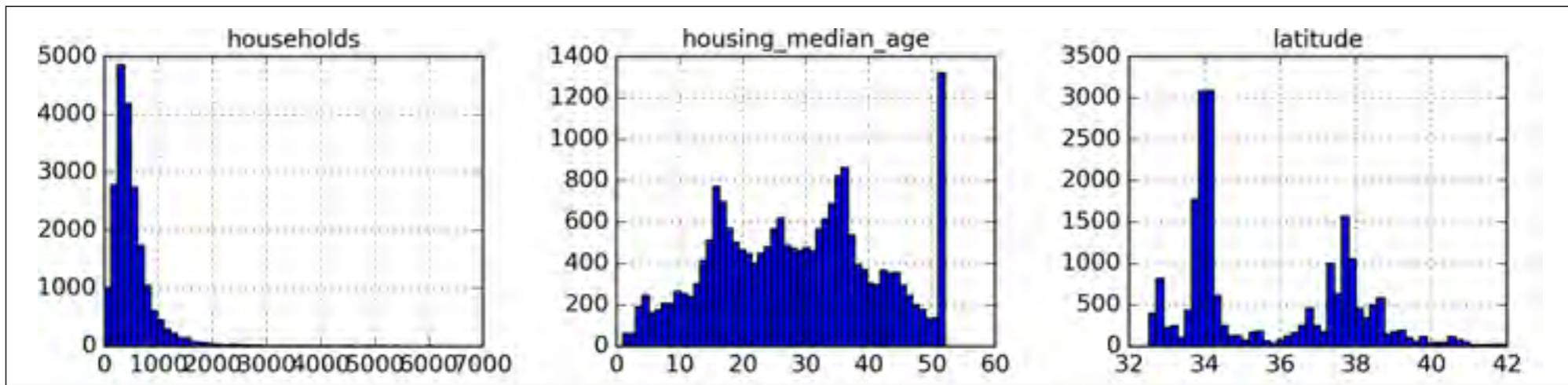


Figure 2-8. A histogram for each numerical attribute

```
%matplotlib inline # only in a Jupyter notebook
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
plt.show()
```



The housing median age and the median house value were also capped. The latter may be a serious problem since it is your target attribute (your labels). Your Machine Learning algorithms may learn that prices never go beyond that limit. You need to check with your client team (the team that will use your system's output) to see if this is a problem or not. If they tell you that they need precise predictions even beyond \$500,000, then you have mainly two options:

- Collect proper labels for the districts whose labels were capped.
- Remove those districts from the training set (and also from the test set, since your system should not be evaluated poorly if it predicts values beyond \$500,000).

Finally, many histograms are *tail heavy*: they extend much farther to the right of the median than to the left. This may make it a bit harder for some Machine Learning algorithms to detect patterns. We will try transforming these attributes later on to have more bell-shaped distributions.

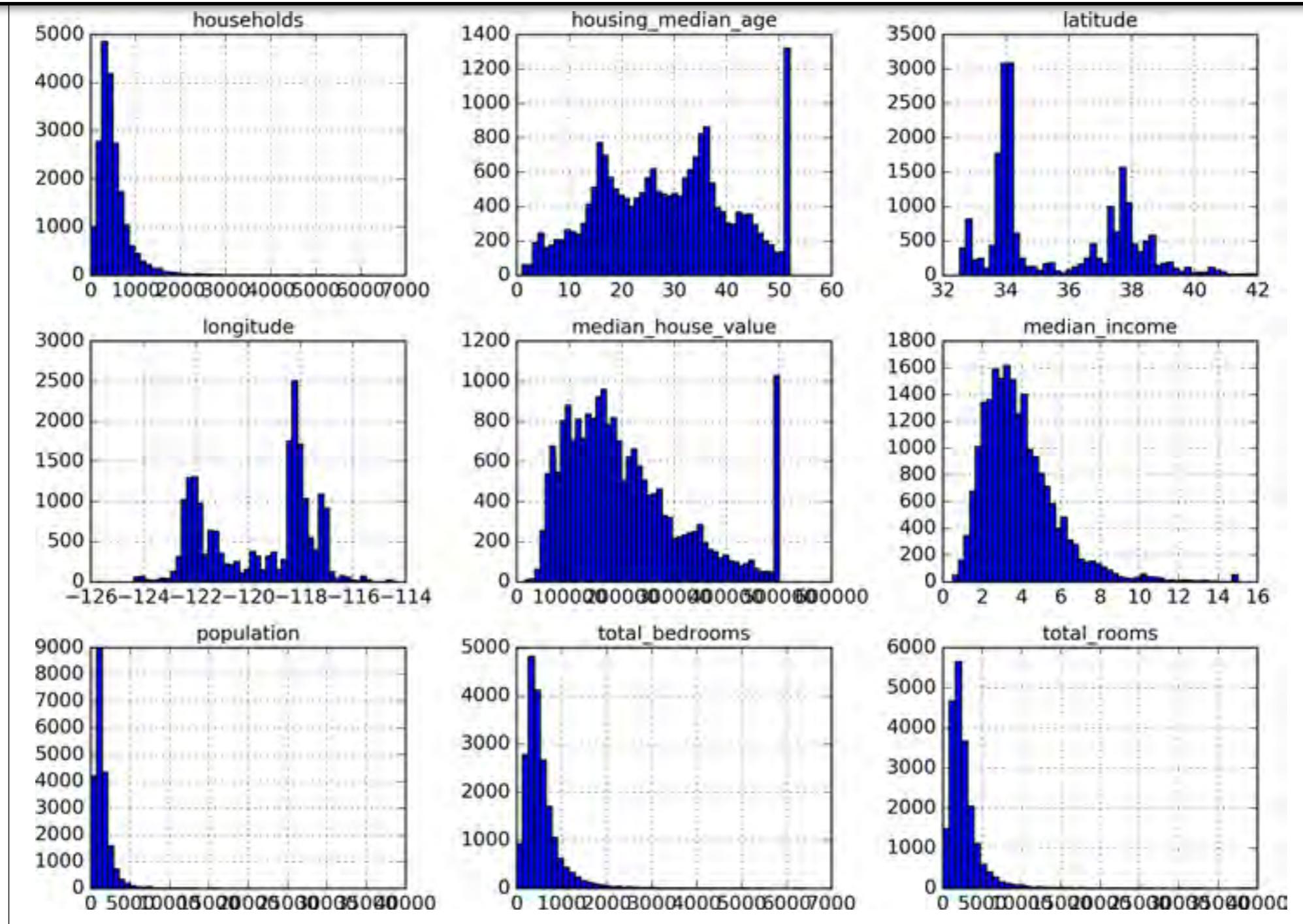


Figure 2-8. A histogram for each numerical attribute

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

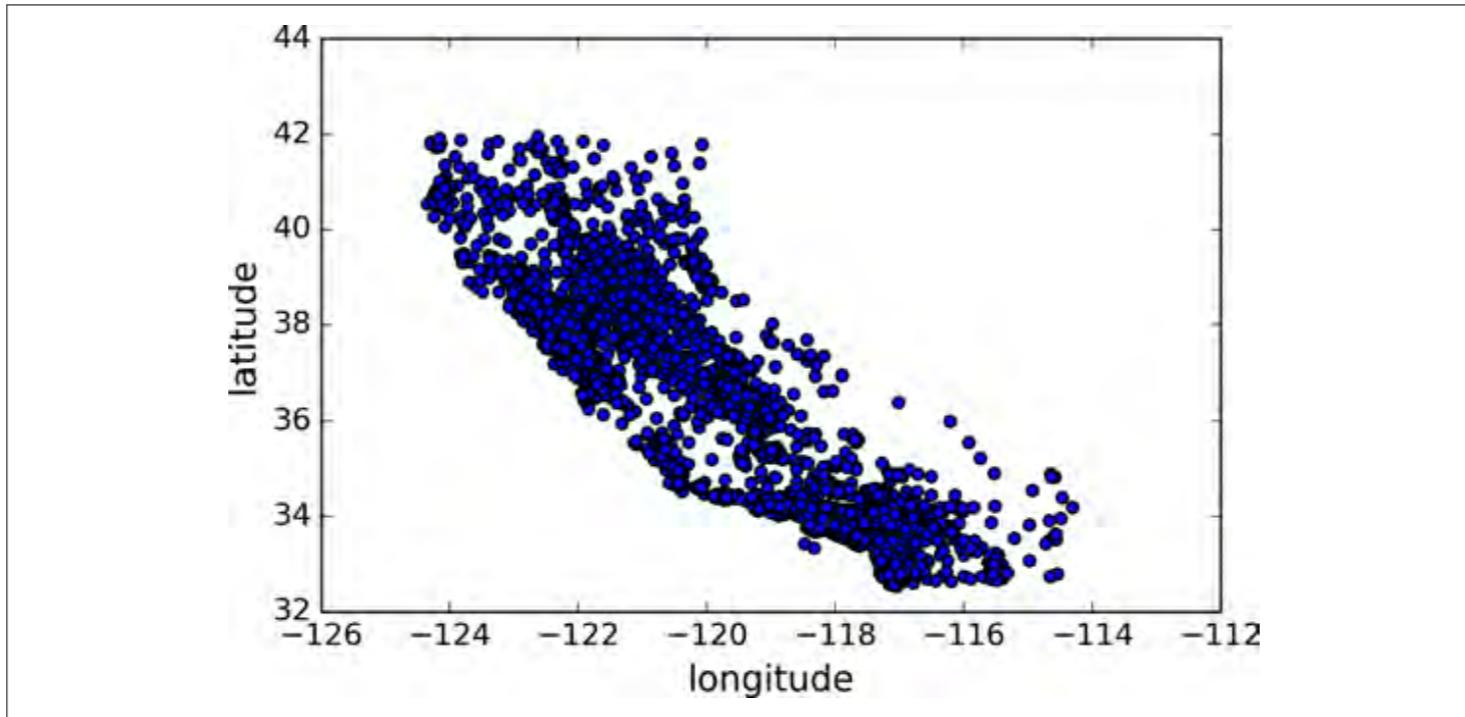
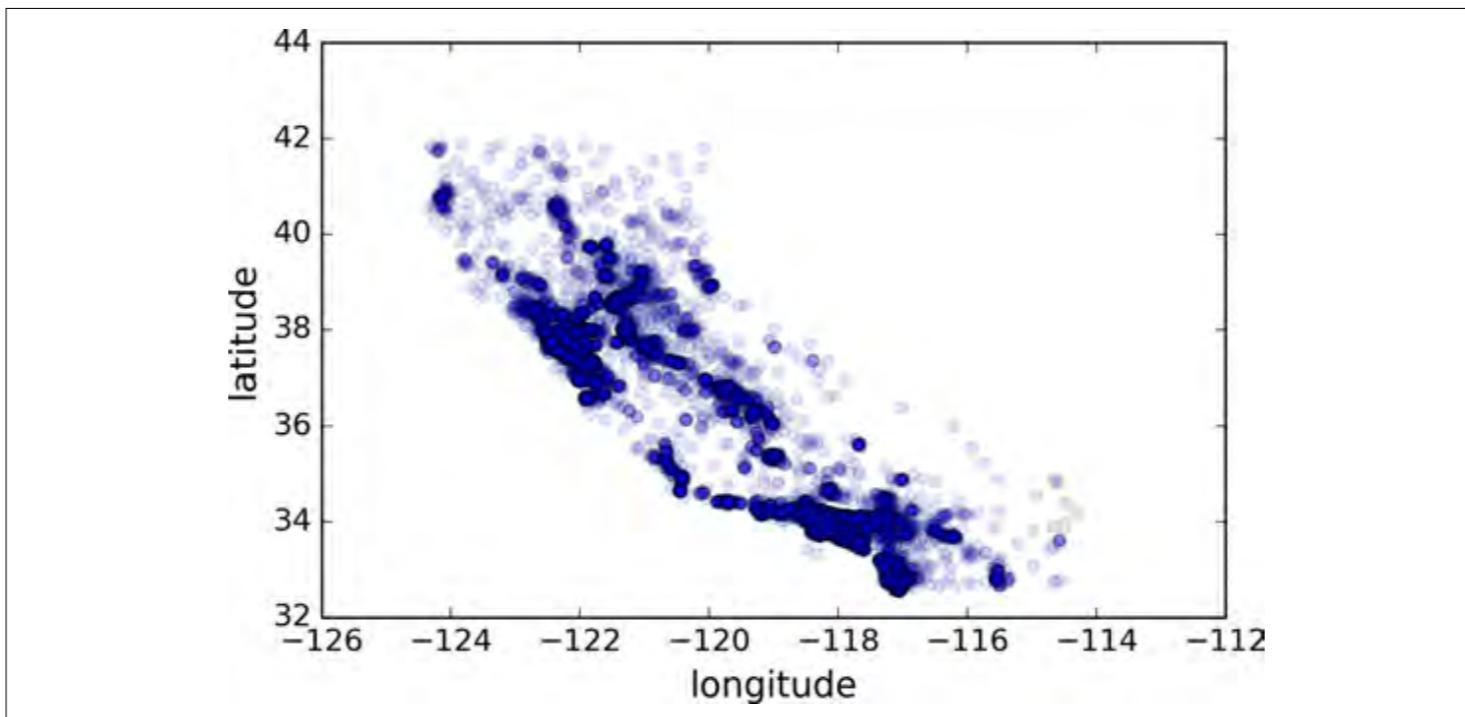


Figure 2-11. A geographical scatterplot of the data

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



Setting alpha to 0.1 emphasizes high density areas

Figure 2-12. A better visualization highlighting high-density areas

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

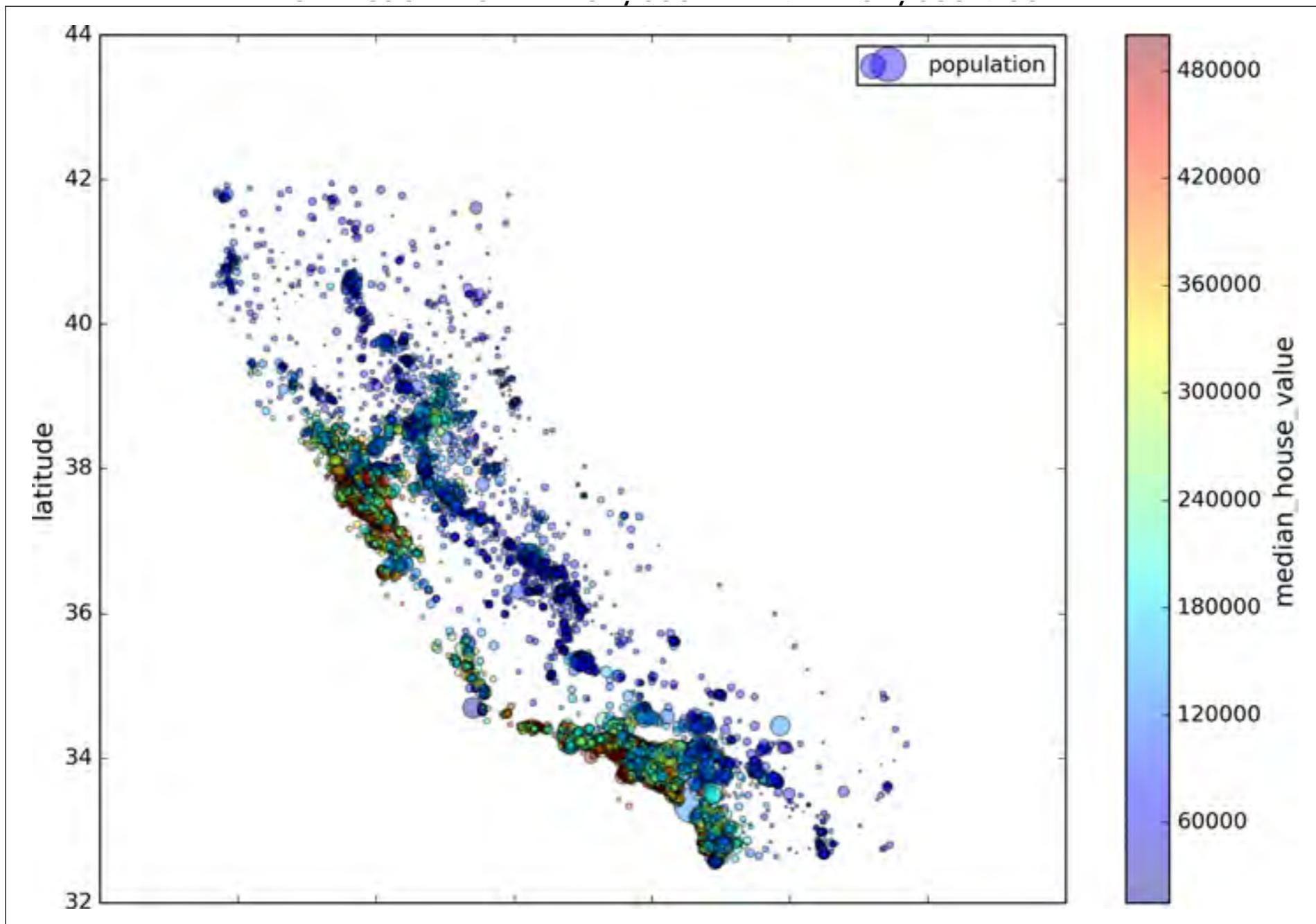


Figure 2-13. California housing prices

Look for Correlations

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

```
>>> corr_matrix["median_house_value"].sort_values(ascending=False)
median_house_value      1.000000
median_income           0.687170
total_rooms             0.135231
housing_median_age     0.114220
households              0.064702
total_bedrooms          0.047865
population              -0.026699
longitude                -0.047279
latitude                 -0.142826
Name: median_house_value, dtype: float64
```

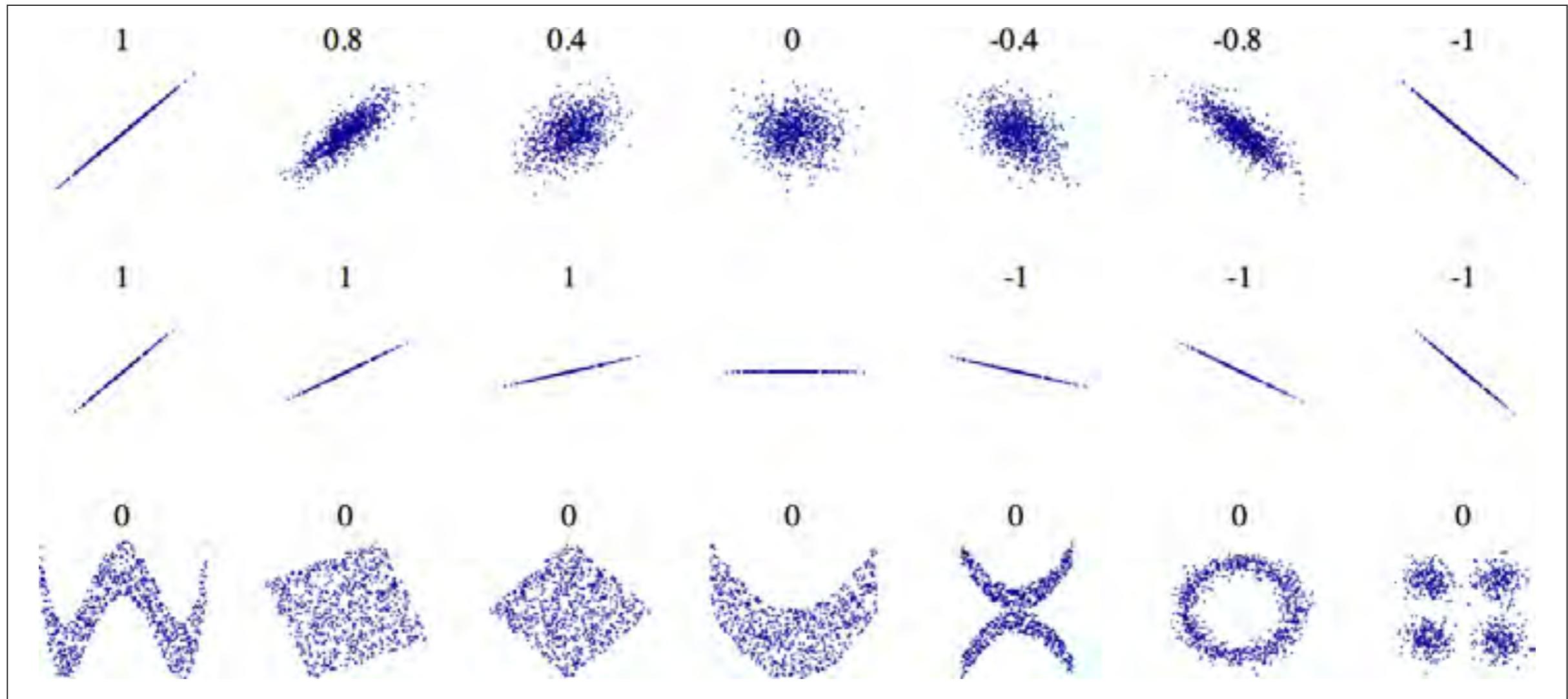


Figure 2-14. Standard correlation coefficient of various datasets (source: Wikipedia; public domain image)

```
from pandas.plotting import scatter_matrix

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

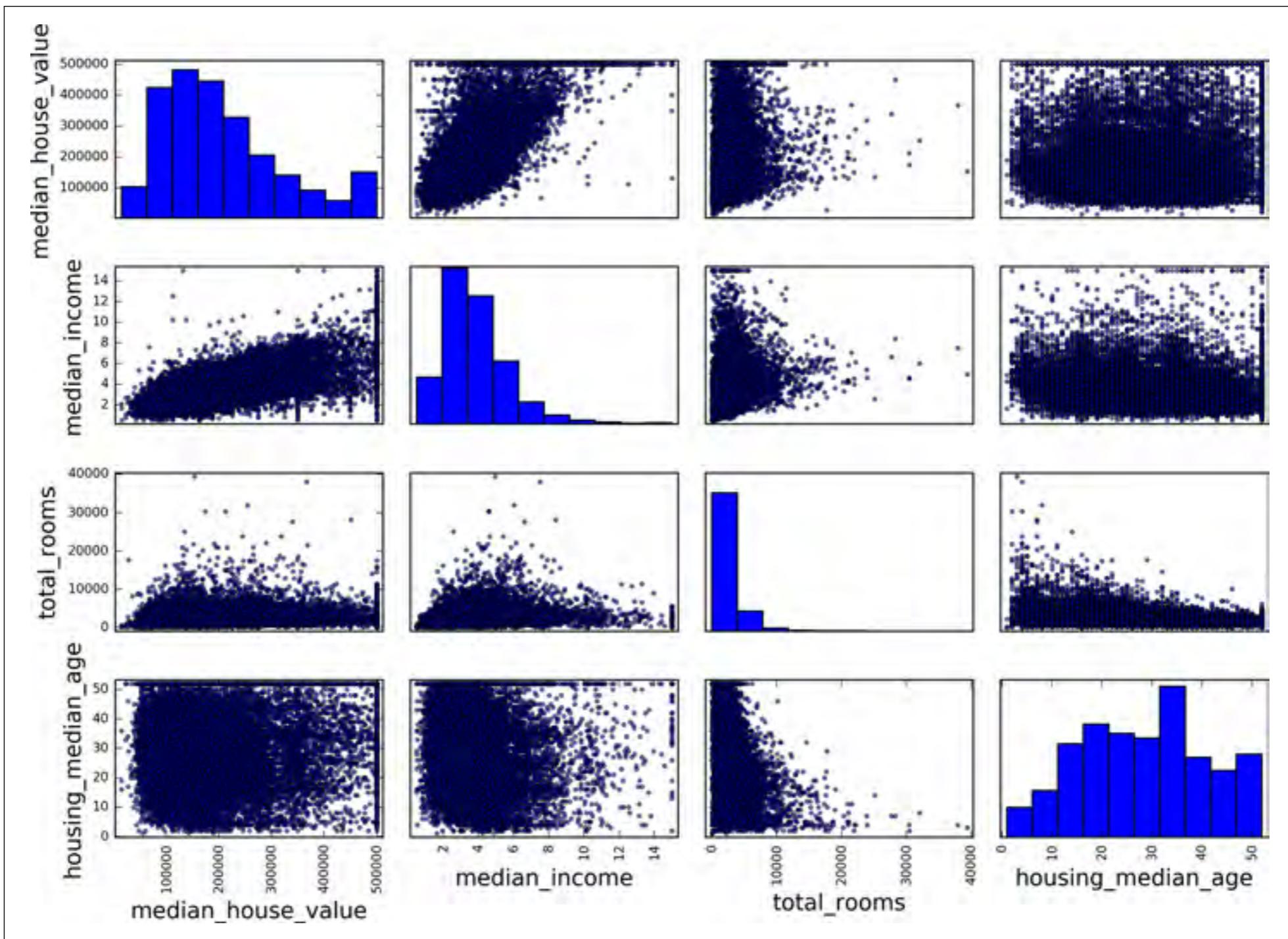


Figure 2-15. Scatter matrix

```
from pandas.plotting import scatter_matrix

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

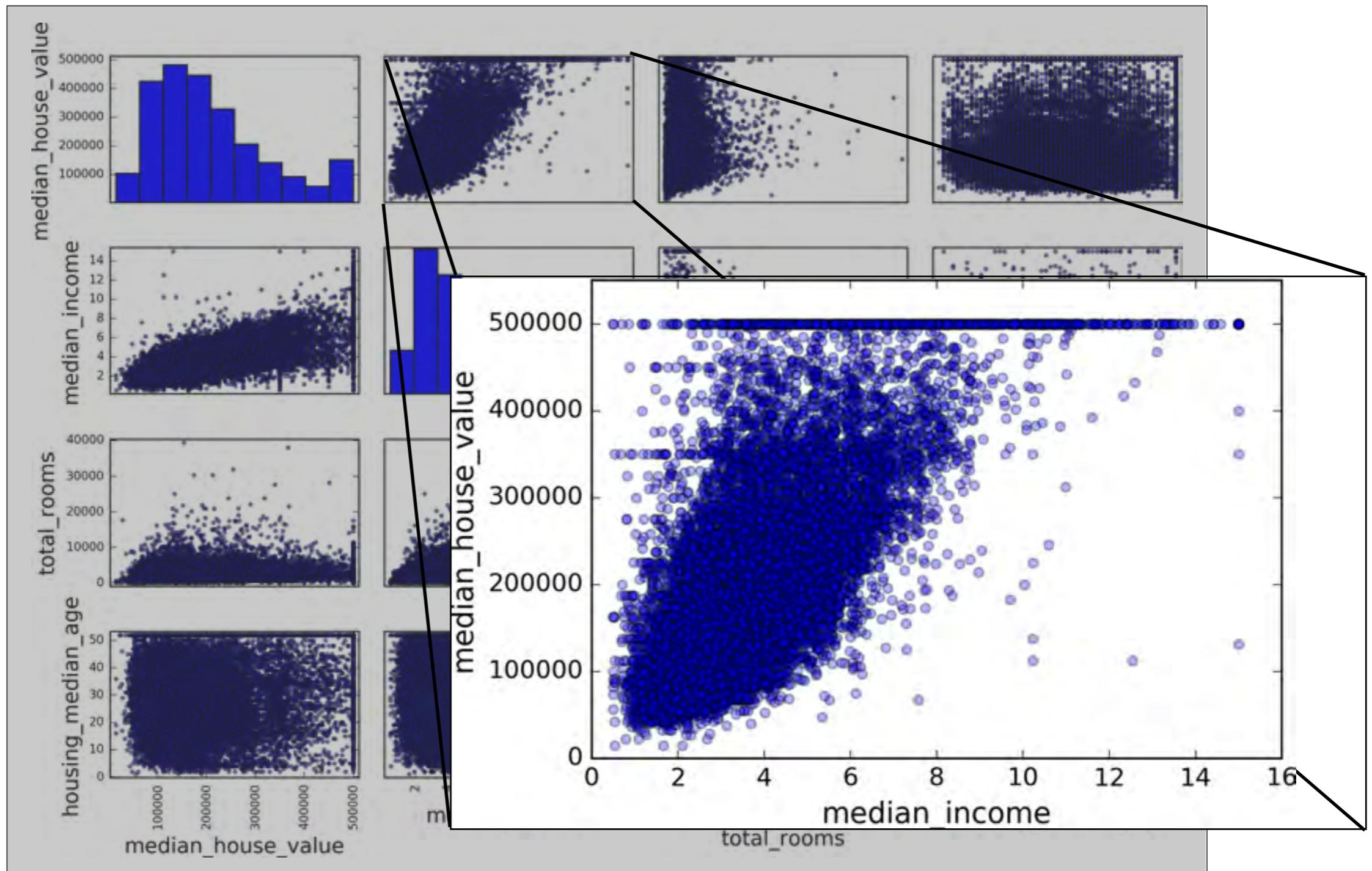


Figure 2-15. Scatter matrix

In-Class Activity

- ▶ **Replicate some of the plots from the California Housing Dataset with the Boston Housing Dataset included with scikit learn**
 - ```
import pandas as pd
from sklearn.datasets import load_boston
boston = load_boston()
boston_pd = pd.DataFrame(boston.data)
```
- ▶ **Be sure to add median value to the main dataframe**
  - ```
boston_pd.columns = boston.feature_names
```
 - ```
boston_pd['MEDV'] = boston.target
```
- ▶ **Generate a histogram for all of the data columns**
- ▶ **Generate a scatter matrix for the attributes "MEDV", "LSTAT", "RM", "AGE"**

Also check out: [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/visualization.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/visualization.html)

# Project 1

- ▶ **Due June 14**
- ▶ **Keep exploring potential datasets**
  - kaggle.com
  - archive.ics.uci.edu/ml/datasets.php
  - libguides.nypl.org/eresources
  - opendata.cityofnewyork.us/data/
- ▶ **The data set will need to be labeled as you are going to use it for both supervised and unsupervised learning tasks**

# Assignments for tomorrow

## ► **Reading**

- Ch 4: "Representing Data/Engineering Features" in  
Guido, Sarah and Andreas C. Muller. (2016). Introduction  
to Machine Learning with Python, O'Reilly Media, Inc.  
213–55.