

End-to-end Machine Learning project [1]

Samkeun Kim <skim@hknu.ac.kr>

<http://cyber.hankyong.ac.kr>

이 장에서는 부동산 회사에 최근에 고용된 데이터 과학자인 것처럼 가장하여 예제 프로젝트를 단계별로 진행해 본다.

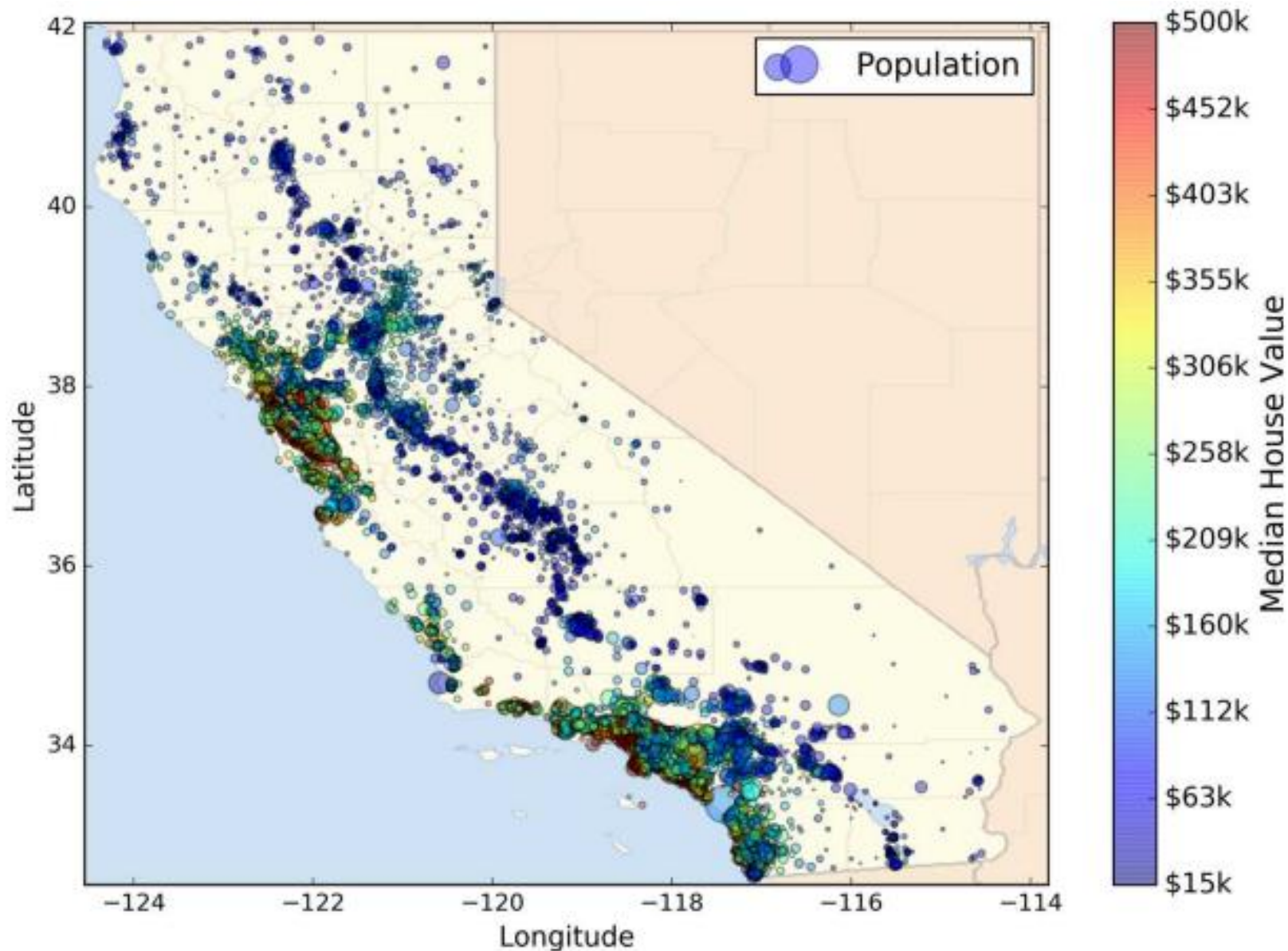
수행할 주요 단계는 아래와 같다:

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for Machine Learning algorithms.
5. Select a model and train it.
6. Fine-tune your model.
7. Present your solution.
8. Launch, monitor, and maintain your system.

Working with Real Data

Real-world data

- Popular open data repositories:
 - UC Irvine Machine Learning Repository (<http://archive.ics.uci.edu/ml>)
 - Kaggle datasets (<https://www.Kaggle.com/datasets>)
 - Amazon's AWS datasets (<http://aws.amazon.com/fr/datasets>)
- Meta portals (they list open data repositories):
 - <http://dataportals.org/>
 - <http://opendatamonitor.eu/>
 - <http://quandl.com/>
- Other pages listing many popular open data repositories:
 - Wikipedia's list of Machine Learning datasets (<https://goo.gl/SJHN2k>)
 - Quora.com question (<http://goo.gl/zDR78y>)
 - Datasets subreddit (<https://www.reddit.com/r/datasets>)



- From 1990 California census
- Added a categorical attribute
- Removed a few features for teaching purposes

California housing prices

Look at the Big Picture

Welcome to Machine Learning Housing Corporation!

첫 번째 임무: California 인구조사 데이터를 이용 => 주택 가격 모델 구축

- 데이터 구성:
 - ✓ 각 블록에 대해 population, median income, median housing price 등의 측정값을 포함
- Block 그룹:
 - ✓ 600에서 3,000명 정도의 인구로 구성, "**district**" 라고 함

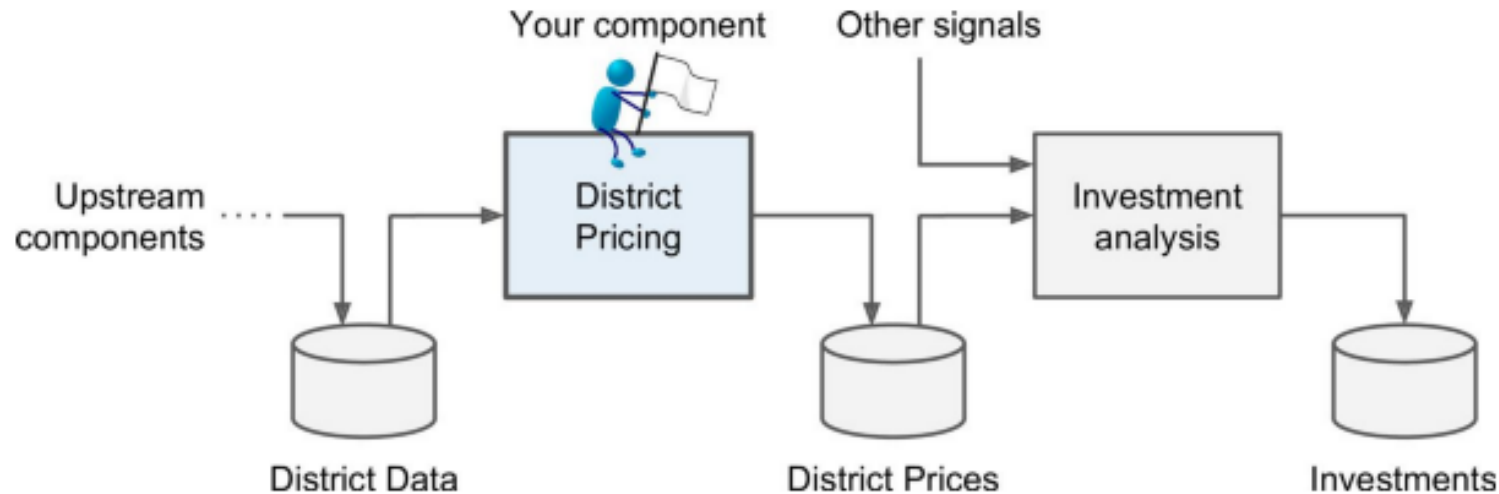
구축된 모델 => 이 데이터로부터 학습해서 어떤 "district"가 주어지면 평균 주택 가격을 예측할 수 있어야 한다.

Frame the Problem

상사에게 물어봐야 할 첫 번째 질문 => 비즈니스 목표? 모델 구축이 최종 목표가 아닐 것이다!

- 회사는 이 모델을 어떻게 사용하고 어떤 혜택을 기대할까?
- 목표를 아는 것은 문제의 구성 방식, 선택할 알고리즘, 모델 평가에 사용할 성능 측정 값 및 조정에 소요되는 노력을 결정하기 때문에 중요하다.

상사의 답 => 구축된 모델의 출력이 또 다른 ML 시스템의 입력으로 사용될 것이다!



A Machine Learning pipeline for real estate investments

Frame the Problem

두 번째 질문 => 현재의 솔루션이 존재한다면 어떤 형태인가?

- 현재의 솔루션 => 주어진 문제에 대한 통찰력 및 성능에 대한 참조를 제공

상사의 답 => 현재는 전문가들이 수작업으로 District 주택 가격을 추정:

- 전문가 팀 => 각 구역에 대한 최신의 정보를 수집
- District의 평균 주택 가격을 얻을 수 없을 때 => 복잡한 룰을 이용하여 추정
- Costly and time-consuming
- 전형적인 에러율은 20% 이상

⇒ 이런 이유로 모델을 이용하여 예측하기를 원함

Frame the Problem

문제 구조 정의 (Frame the problem):

- Is it supervised, unsupervised, or Reinforcement Learning?
 - ✓ 라벨이 있는 학습 예제가 제공되므로 Supervised 학습 태스크임이 분명하다.
- Is it a classification task, a regression task, or something else?
 - ✓ 값을 예측해야 하기 때문에 전형적인 Regression 태스크임
- Should you use batch learning or online learning techniques?
 - ✓ 시스템에 데이터가 지속적으로 유입되지 않고, 변화하는 데이터에 신속하게 조정할 필요가 없으며, 데이터가 메모리에 들어가기에 충분히 작기 때문에 평범한 배치 학습을 수행하면 됨

Select a Performance Measure

성능 측정 도구 선택:

- Root Mean Square Error (RMSE) - Regression 문제에 대한 전형적인 성능 측정 도구
- 시스템이 예측할 때 발생한 에러의 양 측정 => 에러가 클수록 큰 값을 가짐

$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2}$$

Notations

$$\text{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m} \sum_{i=1}^m (h(\mathbf{x}^{(i)}) - y^{(i)})^2} \quad : \text{hypothesis } h \text{를 이용하여 예제 집합 상에서 측정된 Cost 함수}$$

- m : RMSE를 측정하는 데이터 셋에 포함된 인스턴스 개수
- $\mathbf{x}^{(i)}$: i 번째 인스턴스의 모든 특성 값 벡터, $y^{(i)}$: label (desired output)

예:

$$\mathbf{x}^{(1)} = \begin{pmatrix} -118.29 \\ 33.91 \\ 1,416 \\ 38,372 \end{pmatrix} \quad \text{and} \quad y^{(1)} = 156,400$$

- \mathbf{X} : 전체 인스턴스의 모든 특성 값을 포함하는 행렬
- h : 시스템의 예측 함수(hypothesis)

$$\mathbf{X} = \begin{pmatrix} (\mathbf{x}^{(1)})^T \\ (\mathbf{x}^{(2)})^T \\ \vdots \\ (\mathbf{x}^{(1999)})^T \\ (\mathbf{x}^{(2000)})^T \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1,416 & 38,372 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

Select a Performance Measure

많은 Outlier 구역들이 존재한다면 => Mean Absolute Error 이용

$$\text{MAE}(\mathbf{X}, h) = \frac{1}{m} \sum_{i=1}^m |h(\mathbf{x}^{(i)}) - y^{(i)}|$$

- RMSE는 MAE 보다 outliers에 더 민감하다.
- 그러나 Outliers가 극히 적을 때는 RMSE가 선호된다.

Get the Data

Download housing data:

<https://github.com/ageron/handson-ml2>

Create the Workspace

시작 메뉴 -> Anaconda Prompt (Anaconda3)

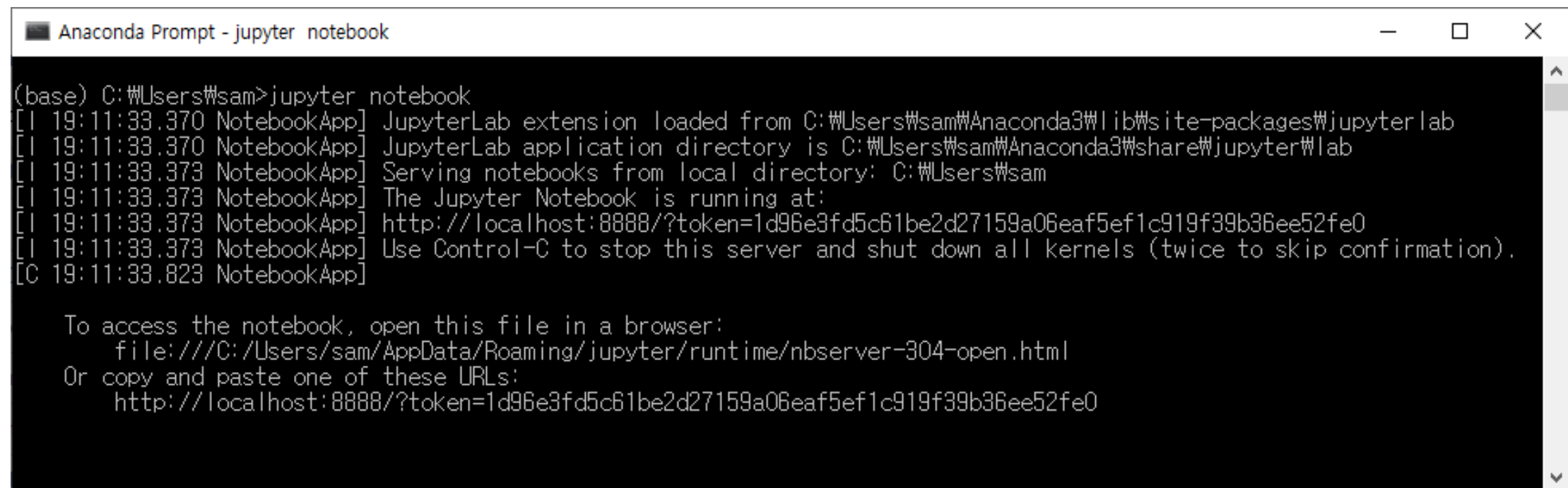
```
$ cd $ML_path (ML working directory)
```

```
$ activate aisam
```

```
$ jupyter notebook
```

Create the Workspace

Jupyter notebook 시작:



```
Anaconda Prompt - jupyter notebook

(base) C:\Users\sam>jupyter notebook
[I 19:11:33.370 NotebookApp] JupyterLab extension loaded from C:\Users\sam\Anaconda3\lib\site-packages\jupyterlab
[I 19:11:33.370 NotebookApp] JupyterLab application directory is C:\Users\sam\Anaconda3\share\jupyter\lab
[I 19:11:33.373 NotebookApp] Serving notebooks from local directory: C:\Users\sam
[I 19:11:33.373 NotebookApp] The Jupyter Notebook is running at:
[I 19:11:33.373 NotebookApp] http://localhost:8888/?token=1d96e3fd5c61be2d27159a06eaf5ef1c919f39b36ee52fe0
[I 19:11:33.373 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 19:11:33.823 NotebookApp]

To access the notebook, open this file in a browser:
    file:///C:/Users/sam/AppData/Roaming/jupyter/runtime/nbserver-304-open.html
Or copy and paste one of these URLs:
    http://localhost:8888/?token=1d96e3fd5c61be2d27159a06eaf5ef1c919f39b36ee52fe0
```

웹 브라우저에서 <http://localhost:8888/> -> Open (서버가 자동으로)



Logout

Files

Running

Clusters

Select items to perform actions on them.

Upload

New ▾



Text File

Folder

Terminal

Notebooks

Octave

Python 2

Python 3

untitled.ipynb라는 새 파일 생성

1

Jupyter Python kernel 기동!

2

Create the Workspace

Housing.ipynb로 이름 변경!

The screenshot shows the Jupyter web interface. At the top, the Jupyter logo is followed by the notebook name 'Housing', with a red arrow and the number '1' pointing to it. To the right is a 'Logout' button. Below the name is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. To the right of the menu bar is a dropdown menu showing 'Python 3'. Below the menu bar is a toolbar with various icons: a save icon, a plus icon, a scissors icon, a copy icon, a paste icon, up and down arrows, a play button (with a red arrow and the number '3' pointing to it), a square icon, and a refresh icon. Below the toolbar is a code cell with the text 'In [1]: print("Hello world!")' and the output 'Hello world!'. A red arrow and the number '2' point to the code cell. Below the code cell is an empty input field with the text 'In []:'. A handwritten note in blue text says 'Play 버튼 클릭 => Python 커널로 notebook을 보낸다!' (Click Play button => Send notebook to Python kernel!).

1

2

3

Play 버튼 클릭 => Python 커널로 notebook을 보낸다!

Download the Data

Download: **housing.tgz** (including comma-separated value (CSV) file called **housing.csv**)

<https://drive.google.com/open?id=1JXPx8Rg7SOKRXiPIGHLGFxr53VRU51WV>

Download 받은 파일을 자신의 Dothome 사이트에 업로드:

<http://ksamkeun.dothome.co.kr/datasets/housing/housing.tgz>

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

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

# Common imports
import numpy as np
import os
```

```
# To plot pretty figures
```

```
%matplotlib inline
```

← PyCharm에서 실행할 경우 Comment 처리!

```
import matplotlib as mpl
import matplotlib.pyplot as plt
```

```
mpl.rc('axes', labelsz=14)
```

```
mpl.rc('xtick', labelsz=12)
```

```
mpl.rc('ytick', labelsz=12)
```

```
# 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)
```

```
os.makedirs(IMAGES_PATH, exist_ok=True)
```

```
def save_fig(fig_id, tight_layout=True, fig_extension="png", resolution=300):
```

```
    path = os.path.join(IMAGES_PATH, fig_id + "." + fig_extension)
```

```
    print("Saving figure", fig_id)
```

```
    if tight_layout:
```

```
        plt.tight_layout()
```

```
    plt.savefig(path, format=fig_extension, dpi=resolution)
```

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


```
import warnings
```

```
warnings.filterwarnings(action="ignore", message="^internal gelsd")
```

Function to fetch the data:

```
import os
import tarfile
import urllib
```

자신의 dothome url로 대체!
(예: <http://ksamkeun.dothome.co.kr/>)



```
DOWNLOAD_ROOT = "https://raw.githubusercontent.com/ageron/handson-ml2/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):
    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()
```

Function to load the data using pandas:

```
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)
```

- This function returns a Pandas DataFrame object containing all the data

Take a Quick Look at the Data Structure

```
In [4]: housing = load_housing_data()  
housing.head()
```

Out [4]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
0	-122.23	37.88	41.0	880.0	129.0	322.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0

- 각 행은 1 구역을 의미
- 10개의 특성을 가짐

Take a Quick Look at the Data Structure

```
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
```

← 데이터셋: 20,640 인스턴스

← 20,433 non-null values, 207 districts missing values

- Categorical attribute
- value_counts() 메소드를 이용하여 각 category별 구역 수 파악 가능

Category별 구역 수 파악 가능: `value_counts()`

```
>>> housing["ocean_proximity"].value_counts()
<1H OCEAN      9136
INLAND          6551
NEAR OCEAN      2658
NEAR BAY        2290
ISLAND           5
Name: ocean_proximity, dtype: int64
```

Take a Quick Look at the Data Structure

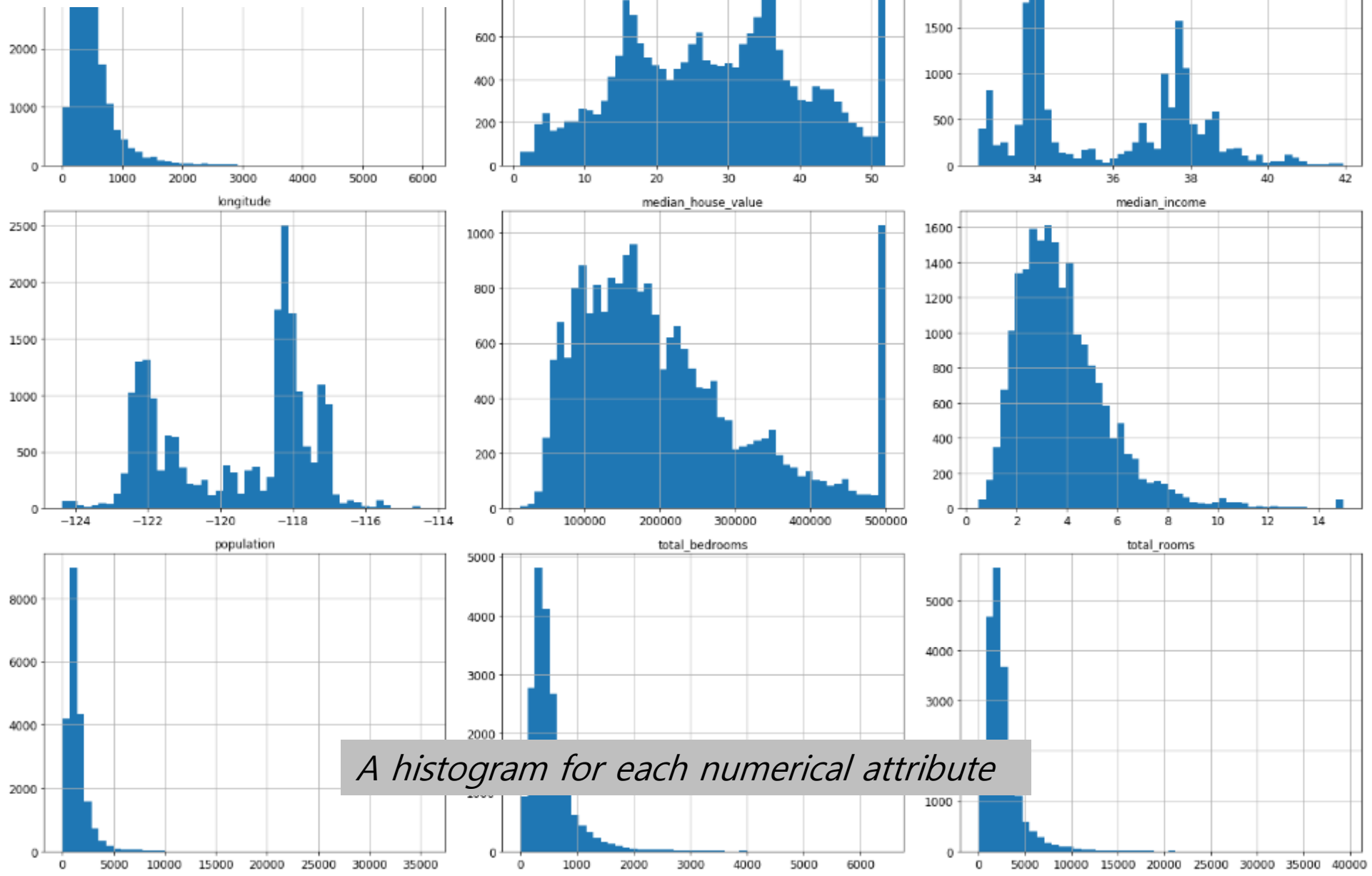
describe(): shows a summary of the numerical attributes

```
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


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



A histogram for each numerical attribute

Create a Test Set

테스트 셋 생성은 이론적으로 매우 간단하다. 데이터 세트의 20%를 무작위로 선택하여 따로 보관해 둔다:

```
In [10]: # to make this notebook's output identical at every run  
np.random.seed(42)
```

```
In [11]: 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 [12]: train_set, test_set = split_train_test(housing, 0.2)  
print(len(train_set), "train +", len(test_set), "test")
```

16512 train + 4128 test

Scikit-Learn: 데이터셋을 나누는 다양한 방법 제공

```
In [13]: from sklearn.model_selection import train_test_split  
         train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
```

```
In [14]: test_set.head()
```

Out [14]:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population
20046	-119.01	36.06	25.0	1505.0	NaN	1392.0
3024	-119.46	35.14	30.0	2943.0	NaN	1565.0
15663	-122.44	37.80	52.0	3830.0	NaN	1310.0
20484	-118.72	34.28	17.0	3051.0	NaN	1705.0
9814	-121.93	36.62	34.0	2351.0	NaN	1063.0

Discover and Visualize the Data to Gain Insights

데이터를 좀 더 깊게 파헤쳐 보자.

먼저 test set은 제쳐 두자. 다음은 training set에 영향 주지 않도록 카피한다:

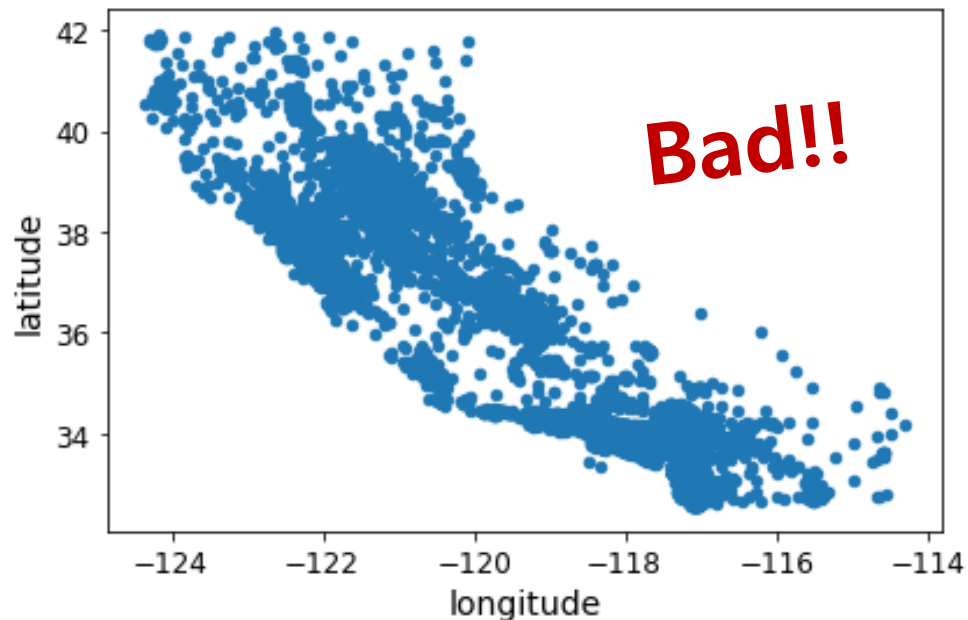
```
housing = train_set.copy()
```

Visualizing Geographical Data

지리 정보(경위도)가 있으므로 모든 구역의 scatterplot을 그려볼 수 있다:

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

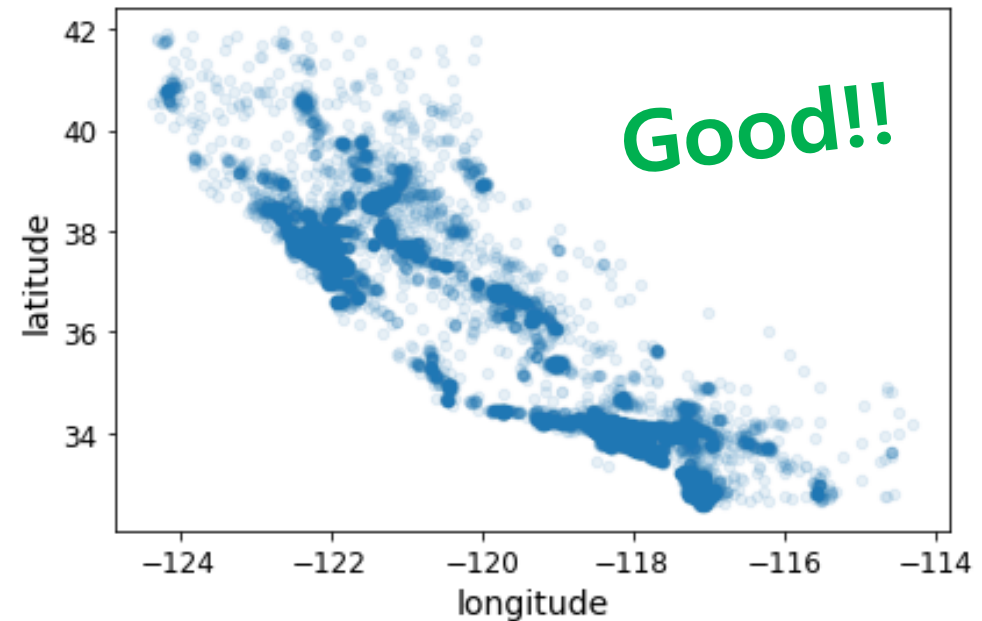
Saving figure bad_visualization_plot



Can clearly see the high-density areas

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

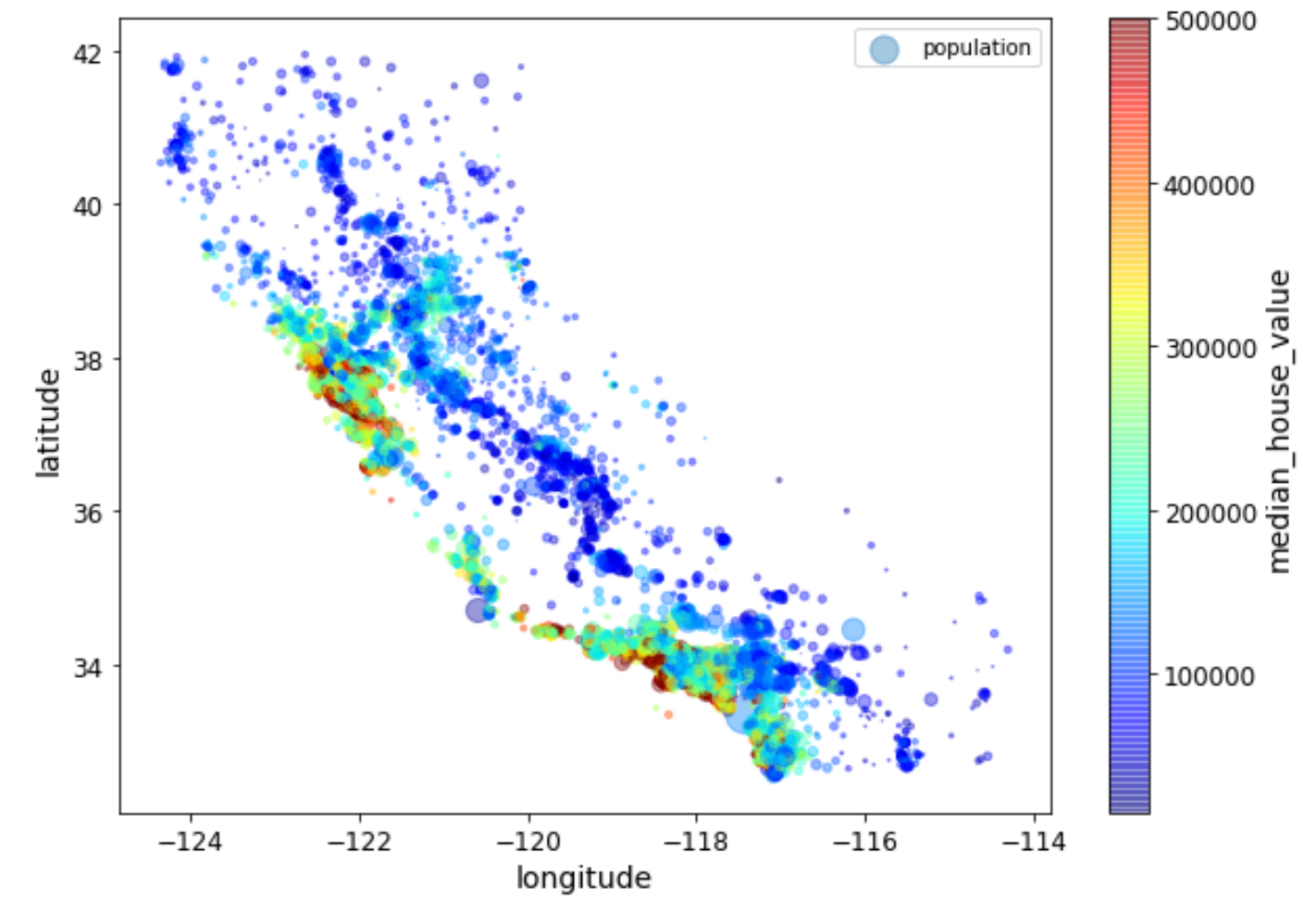
Saving figure better_visualization_plot



Visualizing Geographical Data

```
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()
```

<matplotlib.legend.Legend at 0x274dbb67c88>



주택 가격을 살펴보자.

- 각 원의 반경 – district의 인구 수 (옵션 s)
- 컬러 – 주택 가격 (옵션 c)
- 미리 정의된 컬러 맵 jet 사용:
blue(저가 주택) ~ red(고가 주택)

주택 가격:

- 위치(예: 해안과 가까운 정도) 및 인구 밀집도와 큰 상관관계 있음

Looking for Correlations

데이터셋이 그리 크지 않기 때문에 모든 특성 사이의 표준 상관 계수(Pearson's r)를 쉽게 계산 가능:

- `corr()` 메소드 이용:

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

- 각 특성이 median house value와 얼마나 많이 연관되었나?

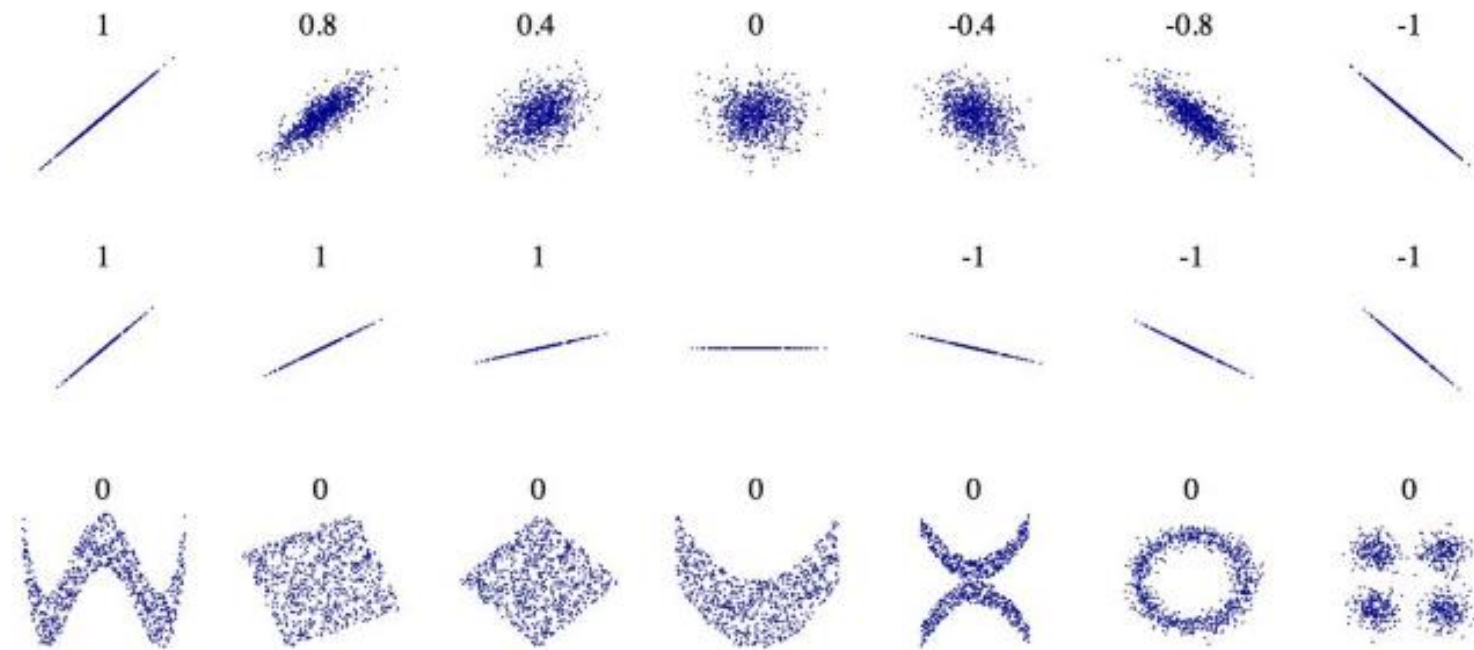
```
corr_matrix["median_house_value"].sort_values(ascending=False)
```

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

Looking for Correlations

상관계수 범위: $-1 \sim 1$

- 1에 가까우면 강한 양의 상관관계
- -1에 가까우면 강한 음의 상관관계



Standard correlation coefficient of various datasets (source: Wikipedia; public domain image)

특성 간의 상관관계를 체크하기 위한 또 다른 방법:

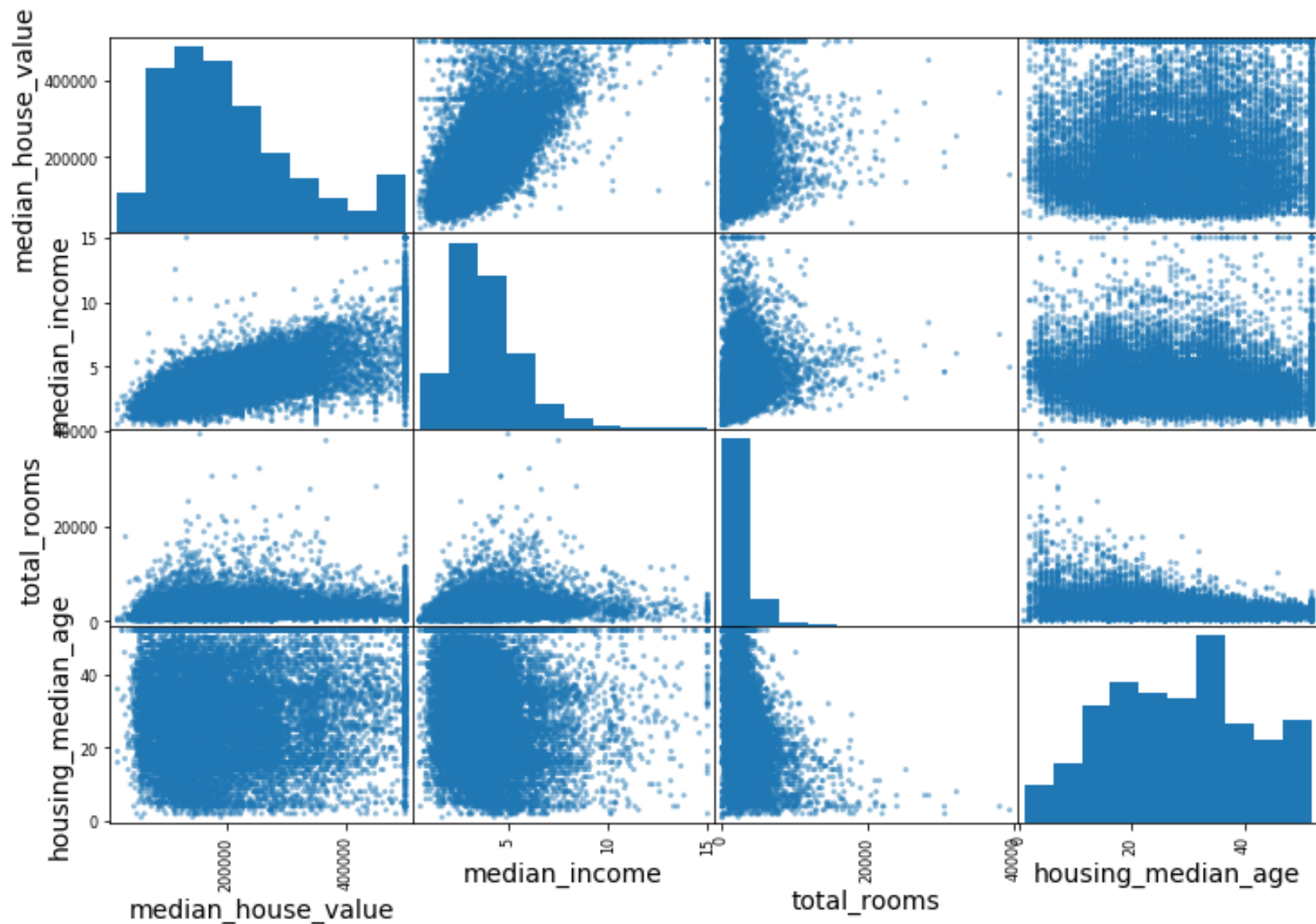
- Pandas' `scatter_matrix` function – 모든 numerical 특성 간의 상관관계를 plot
- 11개의 numerical 특성: $11 \times 11 = 121$

예: median housing value와 가장 상관관계가 높은 특성 몇 개만 Plot

```
from pandas.plotting import scatter_matrix

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

Looking for Correlations



Pandas가 각 변수를 그 자체에 플로팅하면 주 대각선(왼쪽에서 오른쪽으로)이 직선으로 가득 차므로 유용하지 않다.

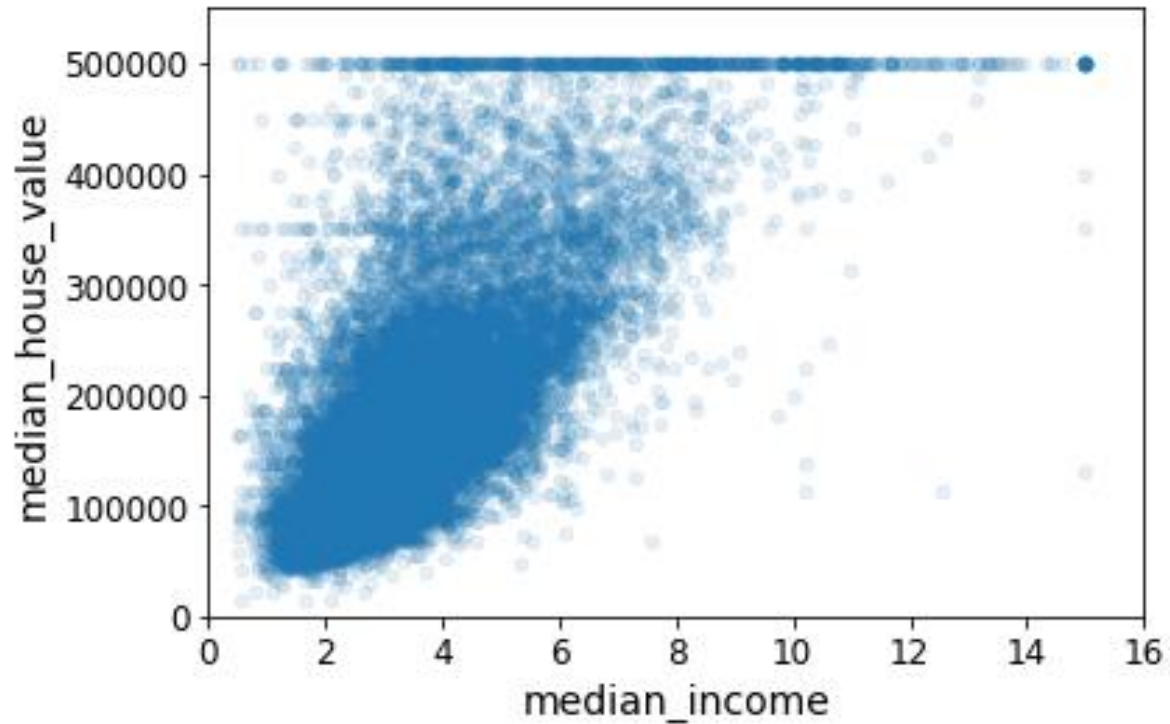
따라서 Pandas는 각 속성의 히스토그램을 표시한다.

Looking for Correlations

Median house value를 예측하기 위한 가장 가능성 높은 특성: **median income**

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

[0, 16, 0, 550000]



Experimenting with Attribute Combinations

특성 조합을 통한 새 특성 도출:

```
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"]
```

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

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

Name: median_house_value, dtype: float64

한 구역의 total room 수는 유용하지 않다.
=> 가구당 room 수 도출

새로 도출된 특성들:
Hey, not bad!

실습과제 2-1

본문에 나오는 전체 내용 PyCharm에서 실행하기

(참조: 제 02강 실습과제 #2 End-to-End Machine Learning Project [1].pdf)

