

# 송내고 특강

---

**Prof. Jibum Kim**

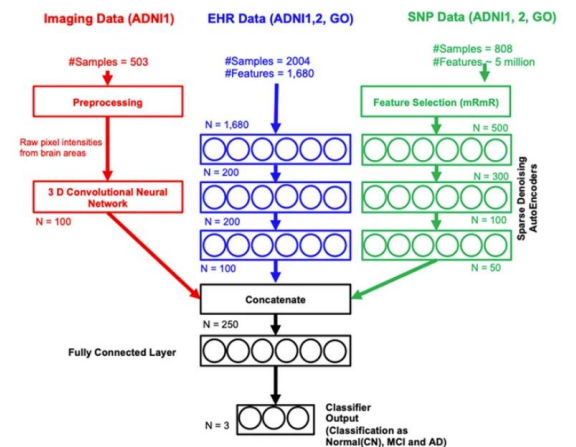
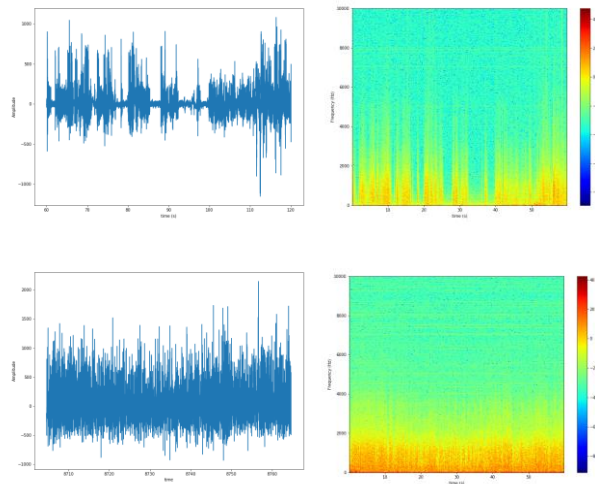
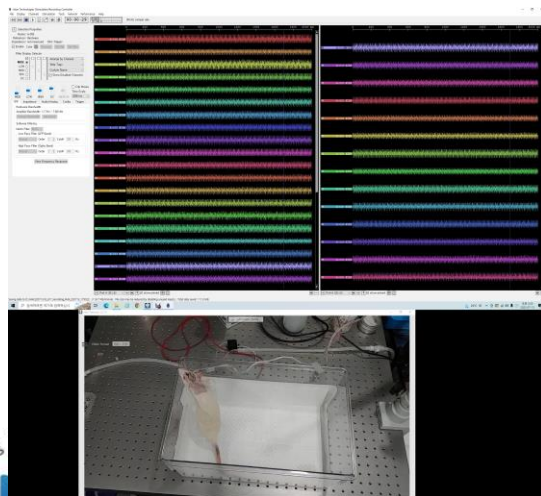
**Department of Computer Science & Engineering**

**Incheon National University**

- 
- **김지범 (Ph.D):** 출신: 인천광역시 미추홀구, 서인천고등학교 졸업
  - Education: 연세대학교 전기전자공학부 학사, 석사
  - (미) 펜실베니아 주립대학교 컴퓨터공학과 박사 (2012)
  - (미) 로스알라모스 국립연구소 Post Doc (2013)
  - **현재: 인천대학교 컴퓨터공학부 조교수, 부교수 (2013 – 현재)**

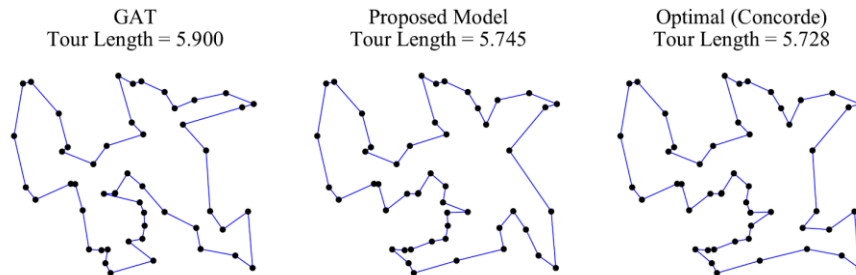
- **연구 분야: 수치 해석, 기하 딥러닝, 인공지능 응용**
- **수상: 2019, 2020 인천대 학술연구상 수상**
- **과제: 현재 연구재단 중견연구, 기초연구실 과제 수행중.  
이전 연구재단 신진연구, 중기부 과제, 산업체과제등 수행  
경험**
- **특허, 기술이전 실적**
- **최근 5년간 기술이전 2건, 총 3000만원 기술이전 실적**
- **최근 5년간 특허 등록: 10건 이상**
- **현재 인천과학예술평재학교 학생들 STEAM 과제 지도중**

- **현재 진행중인 연구**
- **1. 뇌파, 행동데이터, 뇌 image등을 사용한 뇌졸중 사전진단 multi-modal 딥러닝 모델 설계 연구 (기초연구실)**
- **동물 실험 기반으로 뇌파 분석, 딥러닝 모델 설계, 기전 파악**

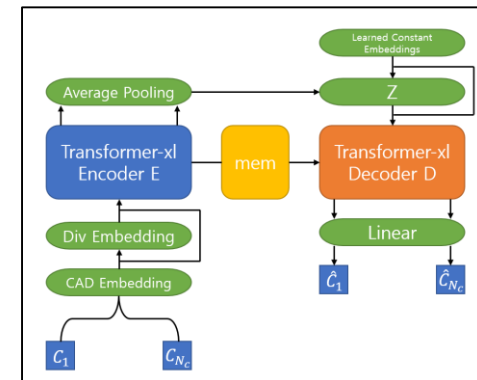
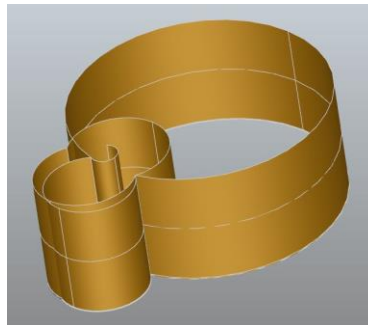
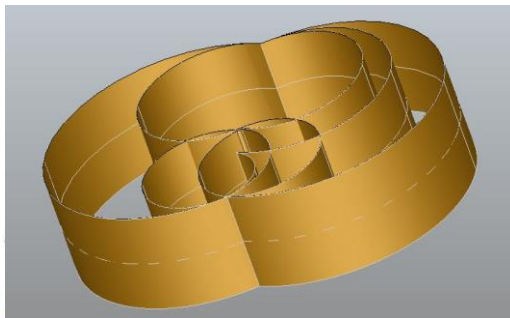


## ■ 2. 데이터 기반의 기하 딥러닝 기술 연구 (중견연구)

### ■ 1) 딥러닝 기반의 최적화 문제의 근사해 찾기



### ■ 2) 딥러닝 기반의 3D CAD 생성 모델 연구



- 
- 이재승 (GAI Lab 석사 3학기)
  - E-mail: [hunni10@inu.ac.kr](mailto:hunni10@inu.ac.kr)
  - 경력: ACM-ICPC (대학생 프로그래밍 경시대회) Korea 2020 본선 26위
  - 연구주제: Graph Convolutional Network 기반의 메쉬 refinement 방법 연구

- **연구실 인공지능 장비**
- **Multi-GPGPU Data Processing System**
- **1. Nvidia Tesla GPU-V100 6기 (학과 공동 운용)**
- **NVIDIA V100 | NVIDIA**
- **2. Nvidia Titan XP 8기|x2 (16기)**
- **총 1억원 이상의 장비. 7호관 서버실에 위치함**



- 
- 오늘 특강 내용
  - 1. 학부 소개
  - 2. (이론) Machine learning 이란?
  - 3. (실습) Python의 Pandas를 이용하여 실제 데이터를 갖고 간단한 data 시각화 및 상관 관계 분석
  - 4. (실습) Scikit-learn과 MNIST dataset을 이용한 간단한 이진 분류기



---

## ■ 1. 인천대 컴퓨터공학부 소개





- 
- 인천대 컴퓨터공학부 역사
  - 1984년: 10월 인천대 전자계산학과 신설 인가로 시작 (정원 50명)
  - 1985년: 제 1회 입학
  - 2015년: 설립 30주년
  - 2017년: 컴퓨터공학부 (주간 78명, 야간 30명)
  - 2022년: 컴퓨터공학부 (주간 108명)

- 인천대학교 컴퓨터 공학부
- 7호관 4층, 5층



- **전임교수: 총 17명**
- **직원 (조교): 4명**
- **재학생 (주/야) – 2022.4.6 기준**
- **총합: 436/102명**
- **4개의 컴퓨터 실습실**
- **자체 서버실 보유**

• 실습실 환경

	컴퓨터실습실1
	위치: 정보기술대 408호 규모: 54석
	컴퓨터실습실2
	위치: 정보기술대 415호 규모: 48석
	컴퓨터실습실3
	위치: 정보기술대 501호 규모: 47석
	컴퓨터실습실4
	위치: 정보기술대 511호 규모: 48석

## ■ 학부생 대표 기업 취업 현황

- 2022년: 우아한형제들, 현대HT, 한전KDN, 스마일게이트
- 2021년: 삼성 리서치(**Samsung Research**), 네이버, 카카오엔터프라이즈, 쿠팡
- 2020년: 삼성전자, 우아한 형제들, 쿠팡, AJ 네트워크, IBK시스템
- 2019년: KT, 삼성전자, 카카오, GS 리테일, 농협은행, 라인플러스, 롯데정보통신, 삼성 SDS, LG CNS, 예금보험공사, 네오위즈 게임즈, 한전 KDN, 한화 시스템
- 2018년: 삼성전자, 카카오, 카카오게임즈, SK인포섹, 롯데정보통신, GS ITM, 관세청, 동아제약

- 대학원 대표 기업 취업 현황
- 삼성전자, 안랩, NHN, SK 하이닉스, 네이버  
파이낸셜등



- 2022학년도 학생부교과성적우수자 전형
- 경쟁률: 13.46대 1
- 경쟁률, 성적 모두 인천대 내에서 최상위권

단과 대학	모집단위	지원현황			성적				교과 환산점			총원 합격
					지원 자	최초 합격자	최종 등록자	최종 등록자	최종등록자			예비 합격 순위
		모집 인원	지원 인원	경쟁률	평균			70% cut	평균	70% cut	가산점 평균	
정보 기술 대학	컴 퓨 터 공 학 부	24	323	13.46	3.40	2.21	2.52	2.75	343.38	342	4.95	55
	정 보 통 신 공 학 과	19	213	11.21	3.55	2.68	2.91	2.99	340.06	339.5	4.96	26
	임베디드시스템공학과	6	58	9.67	3.89	2.35	3.11	3.27	336.93	334.7	4.92	12

- 2024년도 입학생 부터 인천대 컴퓨터공학부 대폭개편
- 교육부 특성화 사업 선정 (사업 기간 2022-2024)
- ‘컴퓨터인공지능’ 학부로 개편하여 AI 부분 강화 계획
- 1. 인공지능 전공
- 2. 컴퓨터전공으로 전공을 2개로 운영 계획 (커리큘럼 2개)





## ■ 인공지능 전공 커리큘럼 예시

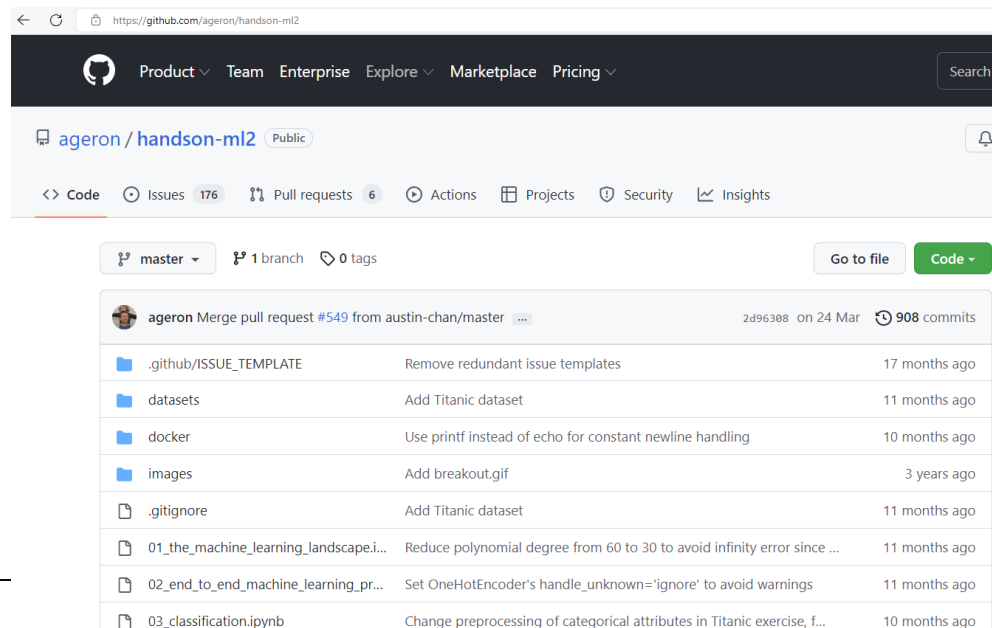
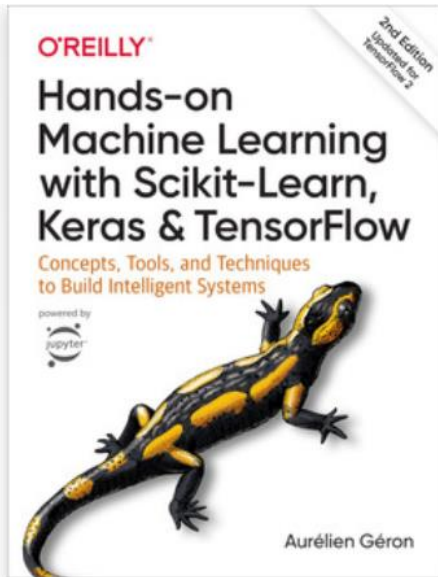
## ■ 자연어처리, 음성인식, 생물정보학등

: 전공기초  
  : 전공핵심(필수)  
  : 전공심화(선택)

1-1	1-2	2-1	2-2	3-1	3-2	4-1	4-2
프로그래밍 입문 (3)	데이터 사이언스 입문 (3)	자료구조 (3)	데이터 마이닝 (3)	데이터 베이스 (3)	AI캡스톤 디자인기초 (2)	AI캡스톤 디자인심화 (2)	빅데이터 입문 (3)
이산수학 (3)	C언어 (3)	데이터 프로그래밍 (3)	인공지능 개론 (3)	기계학습 (3)	알고리즘 (3)	컴퓨터비전 (3)	HCI (3)
컴퓨터공학 개론 (2)	선형대수 (3)	객체지향 프로그래밍 (3)	컴퓨터구조 (3)	운영체제 (3)	컴퓨터 네트워크 (3)	음성인식 (3)	생물정보학 (3)
공학수학I (3)	확률및통계 (3)	시뮬레이션 기초및실습 (3)	수치해석 (3)	AI소프트웨 어공학 (3)	심층학습 (3)	AI실전문제 해결 (2 or 3)	인공지능과 강화학습 (3)
물리 (3)	공학수학II (3)				자연어처리 (3)		지능형IoT (3)

- 
- 2. (이론) Machine learning 이란?
  - 핸즈온 책 chapter 1

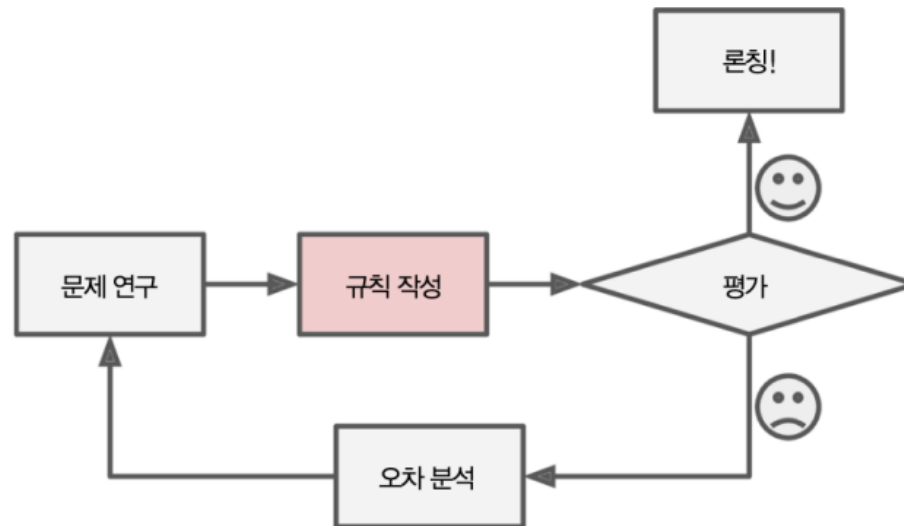
- 데이터 사이언스나 인공지능 입문으로 좋은 책인
- “Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition”의 챕터 1, 2와 3의 일부를 오늘 실습에 소개
- 구글에서 위의 이름으로 검색하면 깃허브도 있음



- 
- **Machine learning (ML)? 한글로 기계 학습**
  - **1. Machine learning is the science of programming computers so they can learn from data**
  - **2. Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed (명시적으로 규칙을 정하기보다는 학습을 통해서 배움)**

- 스팸 필터 예
- 1. 기존의 프로그래밍 기술
- 1) 스팸 메일이 어떻게 보이는지 관찰. 특정 단어나 구절 (예: “credit card”, “amazing”)이 많이 보인다
- 2) 각각의 패턴에 대해서 규칙을 작성해서 1)에서 관찰한 패턴이 여러 개 보이면 스팸 메일이라고 판단
- 3) 작성한 프로그램을 평가 (test)하고 성능이 좋을때까지 1)과 2) 반복

## 전통적인 프로그래밍

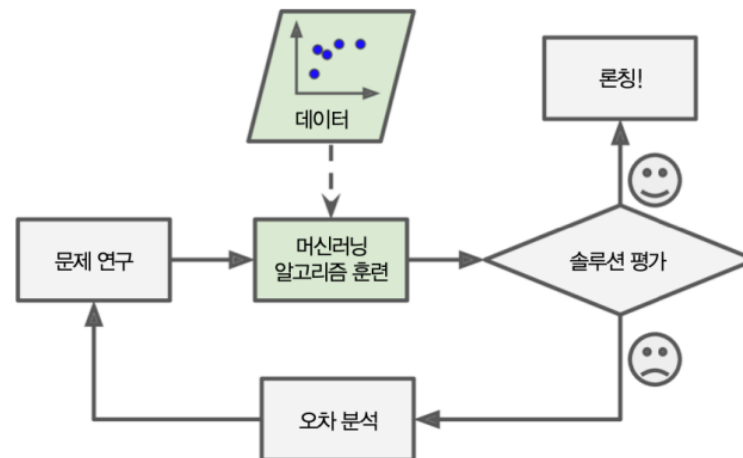


- 전통적인 프로그래밍 접근 방법은 다음과 같다.
  - **문제 연구**: 누군가가 문제를 해결하기 위해 해결책을 찾음
  - **규칙 작성**: 결정된 규칙을 개발자가 프로그램을 작성
  - **평가**: 만들어진 프로그램을 테스트
  - 문제가 없다면 **론칭**, 문제가 있다면 **오차**를 분석한 후 처음 과정부터 다시 실시

- 이러한 전통적인 방법의 문제점?
- 스팸 분류 프로그램을 작성되고 론칭된 후에 **새로운 스팸단어가 생기면** 프로그램이 이 단어를 자동으로 분류할 수 없음
- 예를 들어 최초에 스팸메일들에서 “4U”라는 단어가 포함된 스팸메일을 보냈고 이것이 차단되고 다시 “for you”가 포함된 스팸메일을 보낸 경우
- **개발자가 새로운 규칙을 매번 새롭게 업데이트 해주어야 함**
- 새로운 규칙이 생겼을 때 사용자가 매번 업데이트를 해주어야하므로 **유지 보수가 어렵다**

- Machine learning 방법
- 데이터로부터 스스로 스팸메일의 특징을 찾음
- 차이: 문제 연구후에 규칙이 아닌 머신 러닝 알고리즘 훈련 (학습)을 통해 주어진 데이터를 바탕으로 훈련함

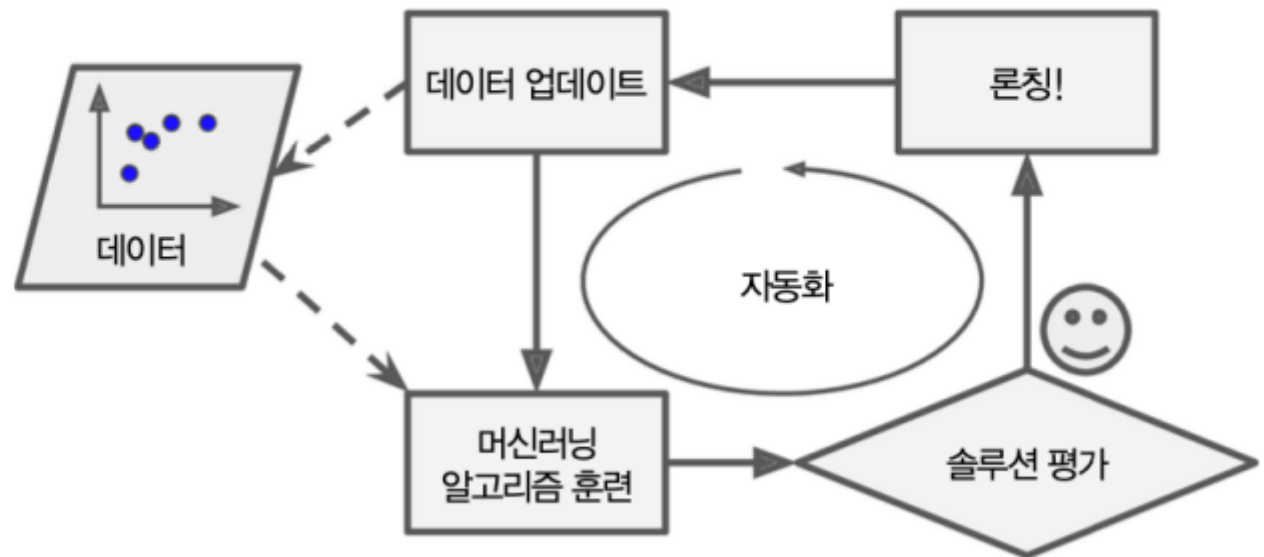
머신러닝





- 예를 들어 최초에 스팸 메일들에서 “4U”라는 단어가 포함된 스팸메일을 보냈고 이것이 차단되고 다시 “for you”가 포함된 스팸메일을 보낸 경우
- Machine learning의 경우 사용자가 스팸으로 지정한 메일에 'For U'가 자주 등장하는 경우 **이 단어가 포함된 이메일을 스팸 으로 분류하도록 스스로 학습**
- **장점: 규칙을 매번 새롭게 정할 필요가 없고 자동화 가능**

## 머신러닝 학습 자동화



- 
- 실제 데이터를 가지고 ML 모델을 학습시키고 테스트 시킬때는 전체 데이터를 나누어서 사용
  - **Training set:** 머신 러닝 프로그램이 훈련 (학습)하는데 사용하는 데이터 집합
  - **Test set:** 실제 머신 러닝 프로그램이 얼마나 잘 동작하는지 테스트 (평가)하는 데이터 집합

- 
- **Supervised learning (지도 학습)**, unsupervised learning (비지도 학습), semi-supervised learning (준지도 학습), Reinforcement learning (강화 학습)
  - In supervised learning, the training data you feed to the algorithm includes the **desired solutions, called labels**
  - Label의 예: 스팸 메일이면 1, 정상 메일이면 0

- 
- A typical supervised learning task is **classification (분류)**
  - The spam filter is a good example of this: it is trained with many example emails along with **their class (spam or not)**, and it must learn how to classify new emails

## 분류



- 특성을 사용한 데이터 분류
- 예제: 스팸 필터

- 
- 3. (실습) Python의 Pandas를 이용하여 실제 데이터를 갖고 간단한 data 시각화 및 상관 관계 분석 (핸즈온 책 chapter 2)

- 
- **유명한 실제 공개 데이터 저장소**
  - UC 얼바인 머신러닝 저장소
  - UCI Machine Learning Repository
  - Kaggle dataset
  - Find Open Datasets and Machine Learning Projects | Kaggle

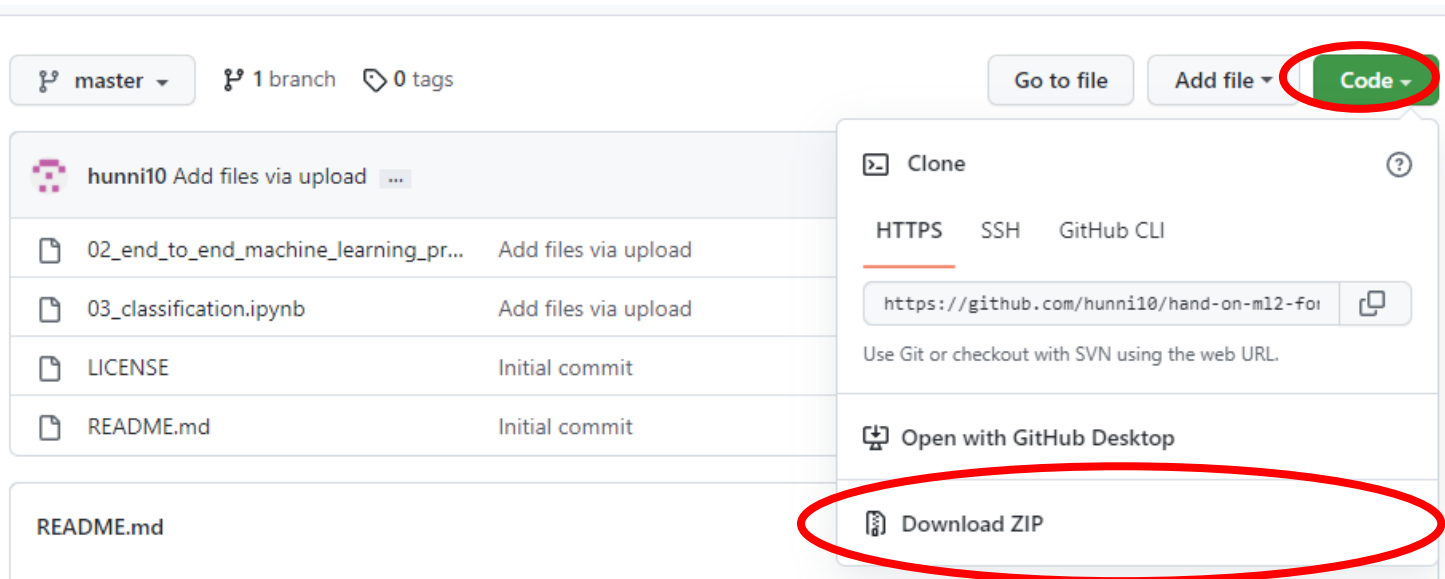


- 오늘 최초 실습에서 사용할 실제 데이터
- 주어진 데이터: 미국 캘리포니아 인구조사 데이터
- 특징: 구역(block)별 인구, 중간 소득, 경도, 위도 등
- Label: 중간 주택 가격
- 학습 내용: 데이터 시각화, 특징간 상관관계
- 사용할 라이브러리: Python Pandas (데이터 처리 분석), NumPy (행렬 수치 연산) Matplotlib (시각화), Scikit-learn (머신러닝 알고리즘 구현) 등

---

## ■ 실습 시작

## ■ <https://github.com/hunni10/hand-on-ml2-for-share>



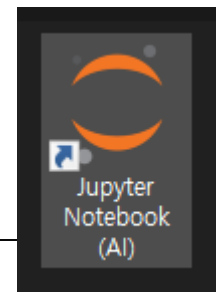
D 드라이브에 압축 해제

- 오늘 실습에서는 Python 코드를 사용 예정
- Jupyter Notebook을 사용 예정

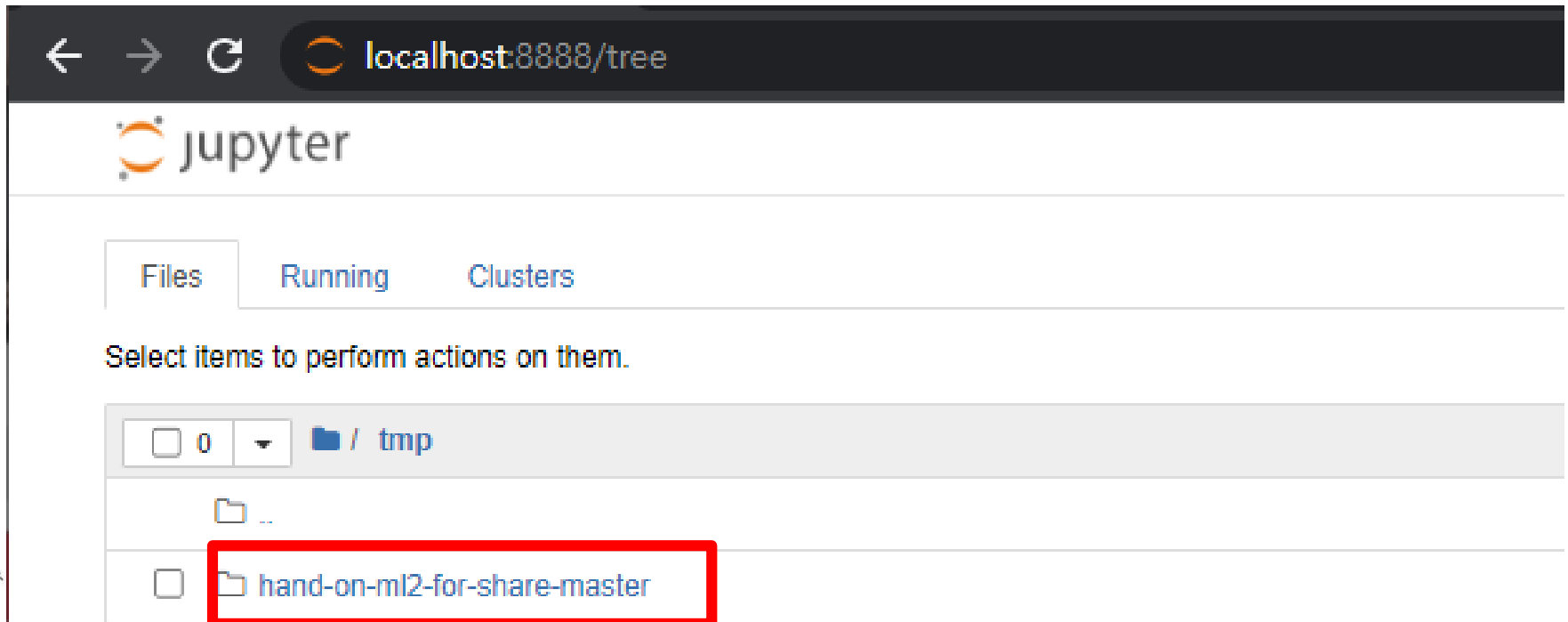
### The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

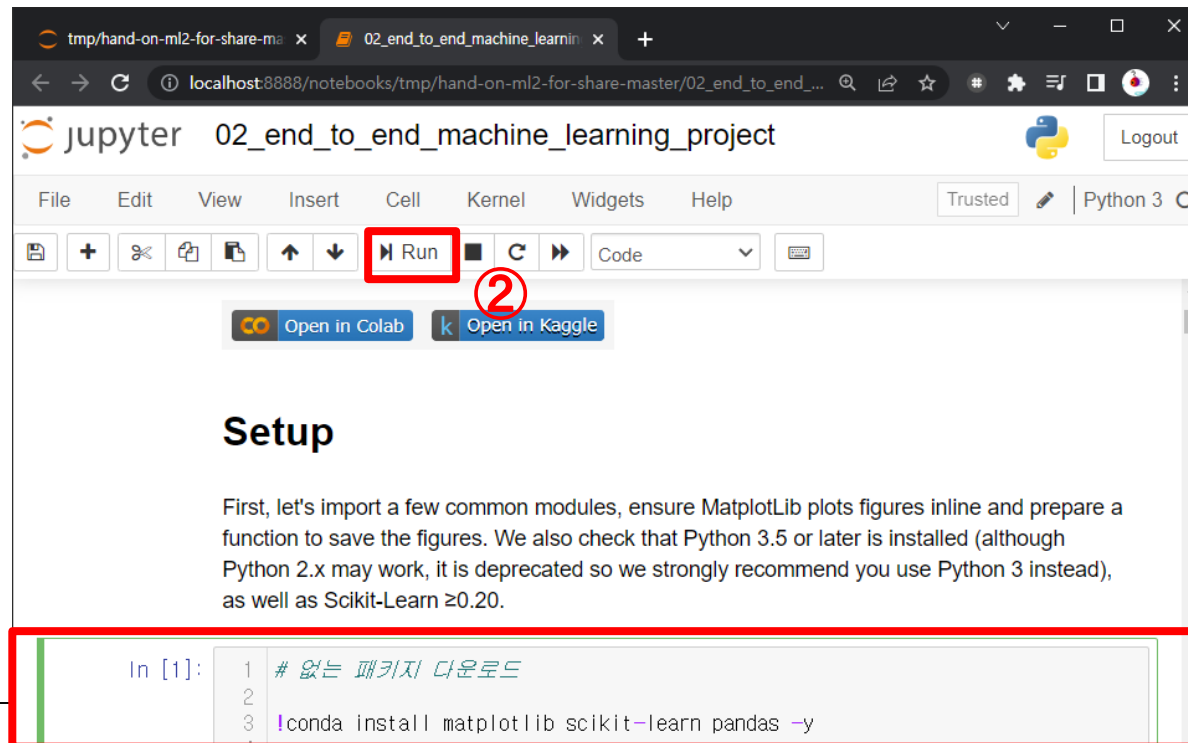
- 바탕화면의 “Jupyter Notebook (AI)” 실행



- 크롬 브라우저에 표시되는 Jupyter notebook
- 실수로 크롬탭을 꺾을 경우 크롬 주소창에 “localhost:8888” 입력
- 압축 푼 “hand-on-ml2-for-share-master” 폴더 클릭



- “02\_end\_to\_end\_machine\_learning\_project.ipynb” 클릭
- 실행하고픈 코드블럭(Cell) 클릭-> 상단 바의 “Run” 클릭



① 클릭

## ■ 실행중 (별표시)

```
In [*]: 1 # 없는  
2  
3 !conda  
4  
5 # Pytho  
6 import  
7 assert  
8  
9 # Sciki  
10 import  
11 assert  
12  
13 # Commo  
14 import  
15 import  
16
```

## ■ 실행완료 (숫자표시)

```
In [1]: 1 # 없는  
2  
3 !conda  
4  
5 # Pytho  
6 import  
7 assert  
8  
9 # Sciki  
10 import  
11 assert  
12  
13 # Commo  
14 import  
15 import  
16
```

- 코드블럭(Cell)의 출력 결과는 블록 하단에 나타남
- (코드에 따라서 실행이 끝나고 출력이 없을 수도 있음)

```
33 print("saving figure", fig_id)
34 if tight_layout:
35     plt.tight_layout()
36 plt.savefig(path, format=fig_extension, dpi=resolution)
```

```
==> WARNING: A newer version of conda exists. <==
current version: 4.9.1
latest version: 4.14.0
```

Please update conda by running

```
$ conda update -n base -c defaults conda
```

```
Collecting package metadata (repodata.json): ...working... done
Solving environment: ...working... done
```

```
# All requested packages already installed.
```



- 
- [https://github.com/hunni10/hand-on-ml2-for-share/blob/master/02\\_end\\_to\\_end\\_machine\\_learning\\_project.ipynb](https://github.com/hunni10/hand-on-ml2-for-share/blob/master/02_end_to_end_machine_learning_project.ipynb)
  - 교재 깃허브의 chapter 2
  - 설치되어있는 Jupyter Notebook로 실행

- 
- 제일 먼저 ‘Setup’ 부분 실행 확인
  - 에러가 뜨지는 않는지?
  - Python 3과 Scikit-Learn 0.20이상 버전 설치되어있어야함
  - ‘Get the Data’ 확인
  - ‘Take a Quick Look at the Data Structure’ 확인

- 
- **Scikit Learn**
  - Machine learning 분야에서 가장 폭넓게 사용되는 tool 중의 하나
  - Classification, Regression, Clustering 등
  - scikit-learn: machine learning in Python  
— scikit-learn 1.1.2 documentation

## Take a Quick Look at the Data Structure

```
In [5]: housing = load_housing_data()  
housing.head()
```

```
Out[5]:
```

	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

- 
- **Python의 pandas의 dataframe 활용**
  - **head(): 데이터의 상위 N행 봄. 기본 N=5**
  - **info(): 데이터에 대한 전반적인 정보**
  - **describe(): 열별 요약 통계량 (수치형만)**
  - **hist(): histogram 보기**

- **특이점: 다른 열 (수치형)과 다르게 ocean\_proximity 열은 자료형이 다름 (object형)**

In [6]:

```
housing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 20640 entries, 0 to 20639
```

```
Data columns (total 10 columns):
```

#	Column	Non-Null Count	Dtype
0	longitude	20640 non-null	float64
1	latitude	20640 non-null	float64
2	housing_median_age	20640 non-null	float64
3	total_rooms	20640 non-null	float64
4	total_bedrooms	20433 non-null	float64
5	population	20640 non-null	float64
6	households	20640 non-null	float64
7	median_income	20640 non-null	float64
8	median house value	20640 non-null	float64
9	ocean_proximity	20640 non-null	object

```
dtypes: float64(9), object(1)
```

```
memory usage: 1.6+ MB
```

## ■ “ocean\_proximity”: 해안 근접도

```
In [7]: housing["ocean_proximity"].value_counts()
```

```
out[7]: <1H OCEAN      9136  
        INLAND    6551  
        NEAR OCEAN 2658  
        NEAR BAY   2290  
        ISLAND      5  
        Name: ocean_proximity, dtype: int64
```

## ■ 각 열별 통계정보 보여줌

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

Out[8]:

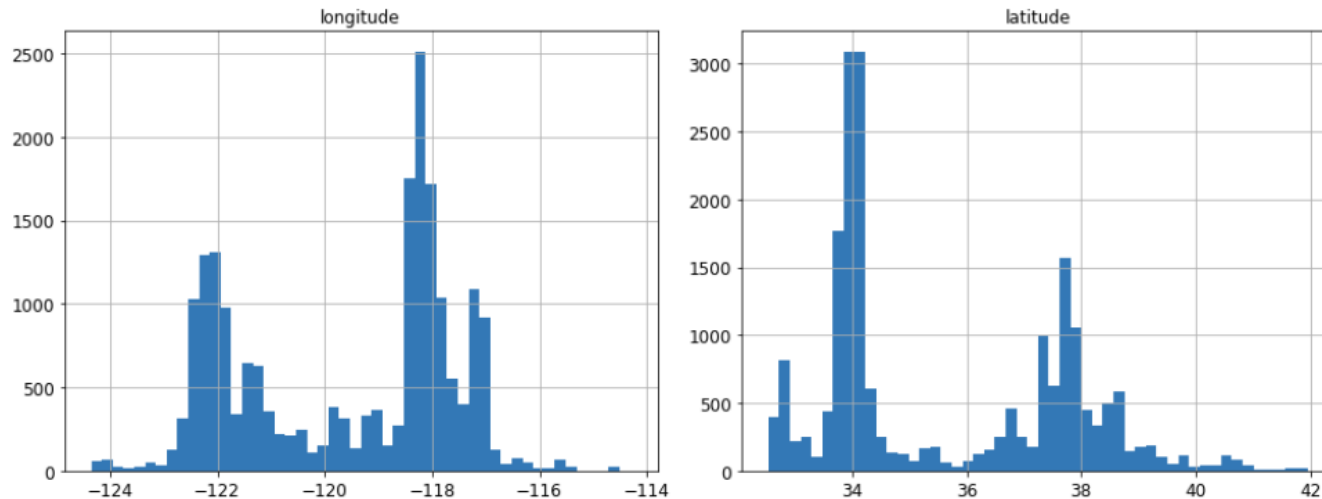
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
<b>count</b>	20640.000000	20640.000000	20640.000000	20640.000000	20433.000000	20640.000000	20640.000000	20640.000000	20640.000000
<b>mean</b>	-119.569704	35.631861	28.639486	2635.763081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



## ■ 히스토그램 시각화 (각 열별)

```
In [9]: %matplotlib inline
import matplotlib.pyplot as plt
housing.hist(bins=50, figsize=(20,15))
save_fig("attribute_histogram_plots")
plt.show()
```

Saving figure attribute\_histogram\_plots

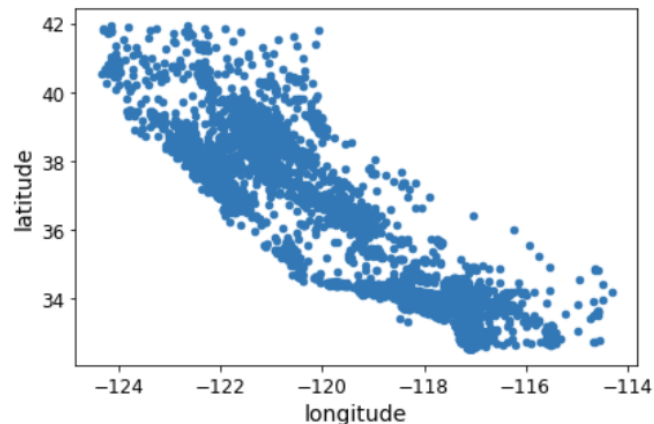


- **Training data 시각화:** 지리적 데이터 시각화 (scatter plot)
- 구역이 집결된 지역과 그렇지 않은 지역 구분 가능
- 샌프란시스코의 베이 에어리어, LA, 샌디에고 등 밀집된 지역 확인 가능
- 아래: bad visualization 예

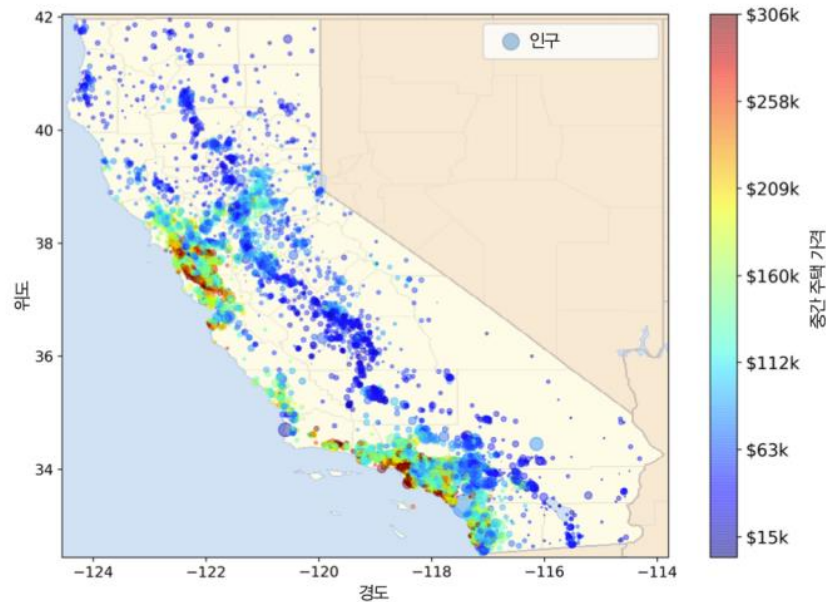
In [33]:

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

Saving figure bad\_visualization\_plot



- 주택 가격이 해안 근접도, 인구 밀도와 관련이 큼
- 해안 근접도: 위치에 따라 다르게 작용 **대도시 근처: 해안 근처 주택 가격이 상대적 높음**
- 북부 캘리포니아 지역: 높지 않음

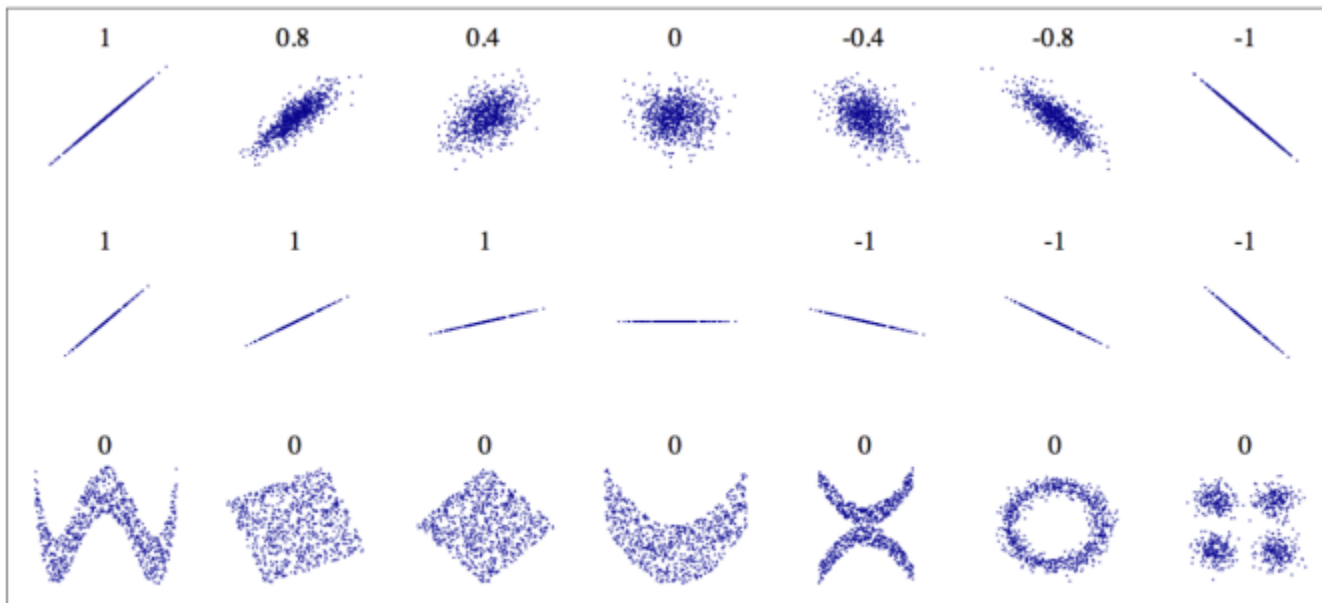


- 상관계수 조사
- 중간 주택 가격 특성과 다른 특성 사이의 상관계수 활용

```
In [39]: corr_matrix["median_house_value"].sort_values(ascending=False)
```

```
Out[39]: 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
```

- 상관계수 (standard correlation coefficient)의 특징
- 상관계수:  $[-1, 1]$ 구간의 값
- 1에 가까울 수록: 강한 양의 선형 상관관계
- -1에 가까울 수록: 강한 음의 선형 상관관계
- 0에 가까울 수록: 매우 약한 선형 상관관계



- 상관계수를 통해 확인할 수 있는 정보 중간 주택 가격과 중간 소득의 상관계수가 0.68로 가장 높음  
중간 소득이 올라가면 중간 주택 가격도 상승하는 경향이 있음

```
In [39]: corr_matrix["median_house_value"].sort_values(ascending=False)
```

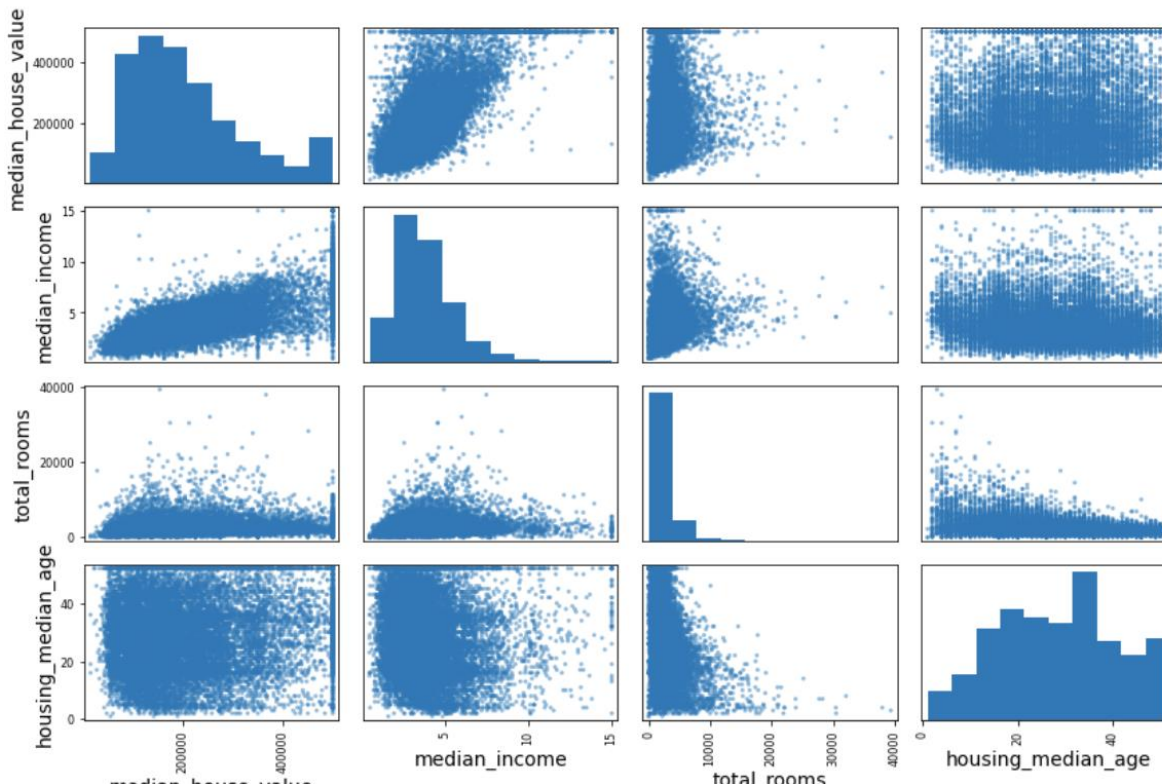
```
Out[39]: median_house_value    1.000000  
         median income        0.687160  
         total_rooms          0.135097  
         housing_median_age    0.114110  
         households            0.064506  
         total_bedrooms        0.047689  
         population            -0.026920  
         longitude             -0.047432  
         latitude              -0.142724  
         Name: median_house_value, dtype: float64
```

## ■ 각 특징 (열)끼리의 상관관계 시각화

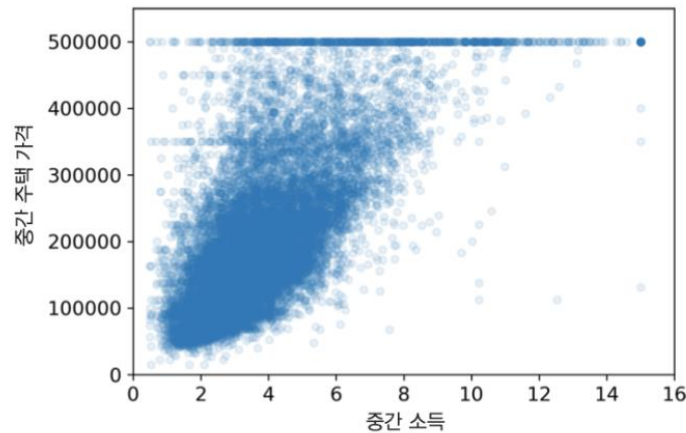
```
In [40]: # from pandas.tools.plotting import scatter_matrix # For older versions of Pandas
from pandas.plotting import scatter_matrix

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



- 중간 주택 가격과 중간 소득의 관계: 산점도 활용
- 점들이 너무 넓게 퍼져 있음. 완벽한 선형관계와 거리 멀.
- 50만 달러 수평선: 가격 제한
- 35만, 28만, 그 아래 정도에서도 수평선 존재





---

- 4. (실습) Scikit-learn과 MNIST dataset을 이용한 간단한 이진 분류기 (핸즈온 챕터 3)

- 
- Most common supervised learning tasks are **classification (predicting class)** and regression (predicting values)
  - MNIST dataset을 이용한 classification (분류기) 실습

- 
- **MNIST dataset**
  - 미국 고등학생과 인구조사국 직원들이 손으로 쓴 70,000개의 숫자 이미지로 구성된 데이터셋
  - 사용된 0부터 9까지의 숫자는 각각  $28 \times 28 = 784$ 크기의 픽셀로 구성된 이미지 데이터
  - 2차원 어레이가 아닌 길이가 784인 1차원 어레이로 제공
  - Label: 총 70,000개의 사진 샘플이 표현하는 값

5	0	4	1	9	2	1	3	1	4
3	5	3	6	1	7	2	8	6	9
4	0	9	1	1	2	4	3	2	7
3	8	6	9	0	5	6	0	7	6
1	8	1	9	3	9	8	5	9	3
3	0	7	4	9	8	0	9	4	1
4	4	6	0	4	5	6	1	0	0
1	7	1	6	3	0	2	1	1	7
9	0	2	6	7	8	3	9	0	4
6	7	4	6	8	0	7	8	3	1

- 
- 문제 정의
  - 지도학습: 각 이미지가 담고 있는 숫자가 레이블 (label)로 지정됨
  - 분류: 이미지 데이터를 분석하여 0부터 9까지의 숫자로 분류
  - 이미지 그림을 총 10개의 클래스로 (class)로 분류하는 multiclass classification

- 
- Training data와 test data 나누기
  - MNIST dataset은 이미 6:1로 분류되어 있음
  - 훈련세트: 앞쪽 60,000개 이미지
  - 테스트세트: 나머지 10,000개 이미지

- 
- [https://github.com/hunni10/hand-on-ml2-for-share/blob/master/03\\_classification.ipynb](https://github.com/hunni10/hand-on-ml2-for-share/blob/master/03_classification.ipynb)
  - 교재 깃허브의 chapter 3

- 
- **Setup**
  - **Matplotlib 설치: Python에서의 시각화 라이브러리**
  - **Python: 3.5이상 설치**
  - **Scikit-Learn: 0.20이상 버전**



- 
- handson-ml2/03\_classification.ipynb at master · ageron/handson-ml2 · GitHub
  - 최초 setup 실행후 오류가 뜨는지 확인

## ■ 1. Loading the dataset

```
In [2]: from sklearn.datasets import fetch_openml
mnist = fetch_openml('mnist_784', version=1, as_frame=False)
mnist.keys()
```

```
Out[2]: dict_keys(['data', 'target', 'frame', 'categories', 'feature_names', 'target_names', 'DESCR', 'details', 'url'])
```

- Scikit-learn을 사용해 MNIST dataset 가져옴
- ‘DESCR’ key: dataset 설명
- ‘target’ key: label이 들어있음

## ■ 2. Exploring data

```
In [3]: X, y = mnist["data"], mnist["target"]  
        X.shape
```

```
Out[3]: (70000, 784)
```

```
In [4]: y.shape
```

```
Out[4]: (70000,)
```

- 행렬의 차원을 shape으로 표현

- 차원수 return: 70,000행, 784열

즉, 70,000개의 data, 784 특징 (28x28 pixel)

- $y$ 는 label을 의미
- $x[0]$ 는 5처럼 보임 실제  $x[0]$ 의 label 확인



`y[0]`

'5'

- 
- Image의 픽셀값은 0-255사이의 정수로 표현. 이러한 자료형 (uint8)으로 변환

```
In [8]: y = y.astype(np.uint8)
```

## ■ 최초 100개의 image 시각화

In [11]:

```
plt.figure(figsize=(9,9))  
example_images = X[:100]  
plot_digits(example_images, images_per_row=10)  
save_fig("more_digits_plot")  
plt.show()
```



5	0	4	1	9	2	1	3	1	4
3	5	3	6	1	7	2	8	6	9
4	0	9	1	1	2	4	3	2	7
3	8	6	9	0	5	6	0	7	6
1	8	7	9	3	9	8	5	9	3
3	0	7	4	9	8	0	9	4	1
4	4	6	0	4	5	6	1	0	0
1	7	1	6	3	0	2	1	1	7
9	0	2	6	7	8	3	9	0	4
6	7	4	6	8	0	7	8	3	1

- 
- **3. Splitting dataset into training and test dataset**
  - **MNIST dataset은 이미 training set (60,000개)과 test data (10,000개)로 구분되어 있음**

```
In [13]: X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]
```

## ■ 4. Binary classification

### ■ 이진 분류기: “5” 인가? “5”가 아닌가?

```
In [14]: y_train_5 = (y_train == 5)
          y_test_5 = (y_test == 5)
```

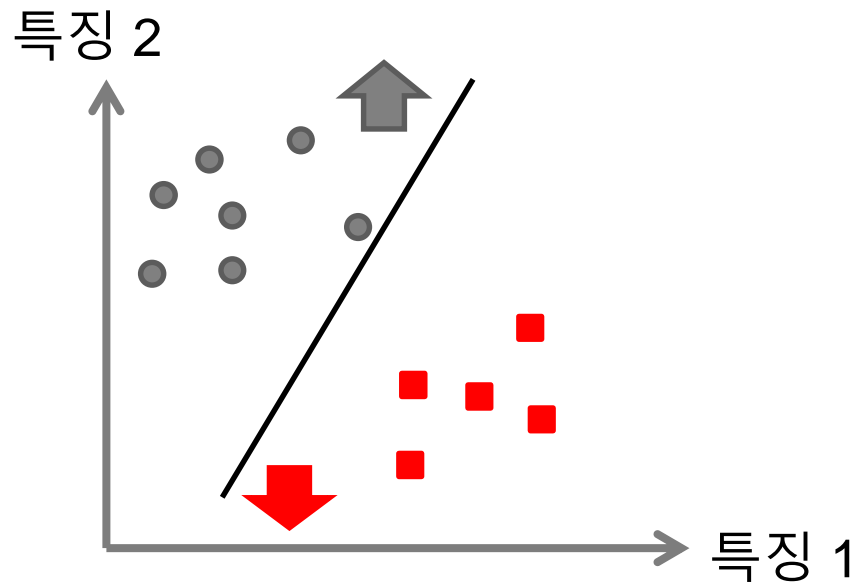
### ■ 분류기: stochastic gradient descent (SGD) classifier 사용 (Scikit-learn 제공)

```
In [15]: from sklearn.linear_model import SGDClassifier

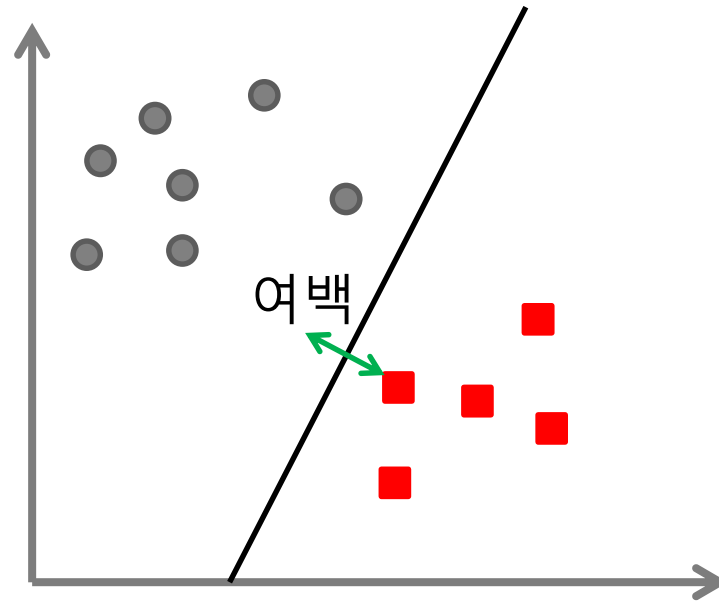
          sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)
          sgd_clf.fit(X_train, y_train_5)
```



- 여기서 사용되는 SGD classifier는 선형 Support Vector Machine (SVM)이라는 기계학습 모델이다.



선형 분류기는 간단히 말하면 데이터를 직선으로 분류한다는 의미이다  
(2D의 경우)

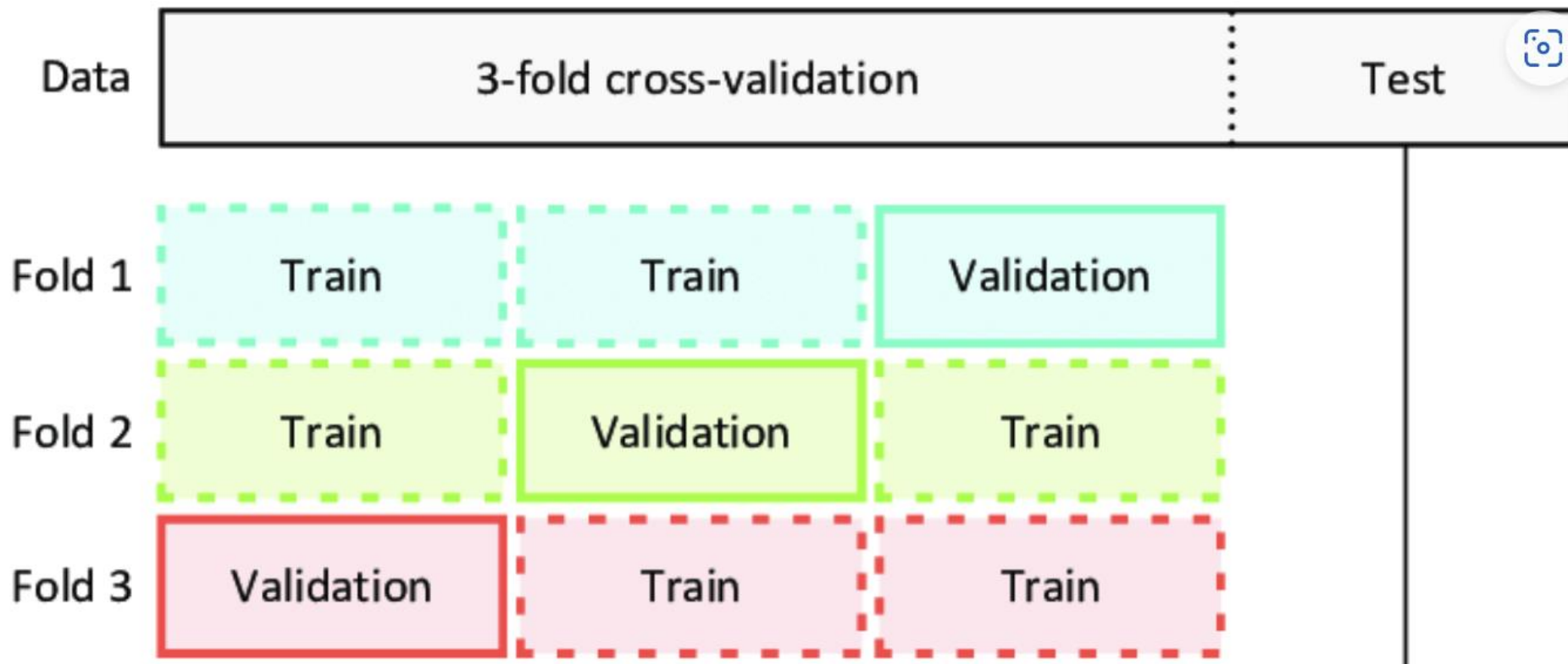


- SVM에서 여백 (margin)이란 선형 분류기 (직선)으로부터 가장 가까운 샘플데이터까지의 거리의 2배 (위의 그림에서 초록색)라고 정의 된다

**선형 SVM (SVC)의 목적은 여백을 가장 크게 하는 직선을 찾는 것이다!**

- 분류기 성능 평가: **K-fold Cross validation (CV)**
- 왜 필요한가? 학습한 dataset에 평가까지 하면 왜곡된 결과가 나올 수 있음 (학습하지 않은 데이터에 test를 해보야함). **Overfitting 문제**
- **한쪽 데이터에 치우친 결과를 방지하기 위하여**
- **Splitting the training set into K-folds (in this case 3), then making predictions and evaluating them on each fold using a model trained on the remaining folds**

- 모든 training data를 활용하여 검증
- 일부만 사용하여 왜곡된 결과를 방지하기 위함



---

- **Above 95% accuracy on all cross-validation folds**

```
In [17]: from sklearn.model_selection import cross_val_score  
cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

```
Out[17]: array([0.95035, 0.96035, 0.9604 ])
```

- **Good? Let's look at a very dumb classifier that just classifies every single image in the “not-5” class**

- It has over 90% accuracy? Why?
- Because only about 10% of the images are 5s, so if you always guess that an image is not a 5, you will be right about 90% of the time

In [19]:

```
from sklearn.base import BaseEstimator
class Never5Classifier(BaseEstimator):
    def fit(self, X, y=None):
        pass
    def predict(self, X):
        return np.zeros((len(X), 1), dtype=bool)
```

In [20]:

```
never_5_clf = Never5Classifier()
cross_val_score(never_5_clf, X_train, y_train_5, cv=3, scoring="accuracy")
```

Out[20]:

```
array([0.91125, 0.90855, 0.90915])
```

- 
- This demonstrates why accuracy is generally not the preferred performance measure for classifiers
  - 오차 행렬 (confusion matrix)
  - 클래스별 예측 결과를 정리한 행렬
  - 오차행렬의 행은 실제 class를 열린 예측된 class를 가리킴

- The first row of this matrix considers non-5 images: 53,892 of them were correctly classified as non-5s, while 687 were wrongly classified as 5s
- 1,891 were wrongly classified as non-5s, 3,530 were correctly classified as 5s

```
In [21]: from sklearn.model_selection import cross_val_predict  
  
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
```

```
In [22]: from sklearn.metrics import confusion_matrix  
  
confusion_matrix(y_train_5, y_train_pred)
```

```
Out[22]: array([[53892,   687],  
               [ 1891, 3530]])
```



- 완벽한 분류기
- 오차 행렬의 대각부분 (diagonal)에만 값이 있음

```
In [23]: y_train_perfect_predictions = y_train_5 # pretend we reached perfection  
confusion_matrix(y_train_5, y_train_perfect_predictions)
```

```
Out[23]: array([[54579,    0],  
               [    0, 5421]])
```