인공지능 개론

인공지능은 무엇일까요?

Artifical + Inteligence

인공 지능

컴퓨터에게 일을 시킨다는 것

로켓을 발사해 달로 보낸다고 생각해 볼께요

컴퓨터에게 일을 시킨다는 것

가령, 로켓 엔진을 조작하기 위해서는 아래와 같은 수식을 풀어야 합니다.

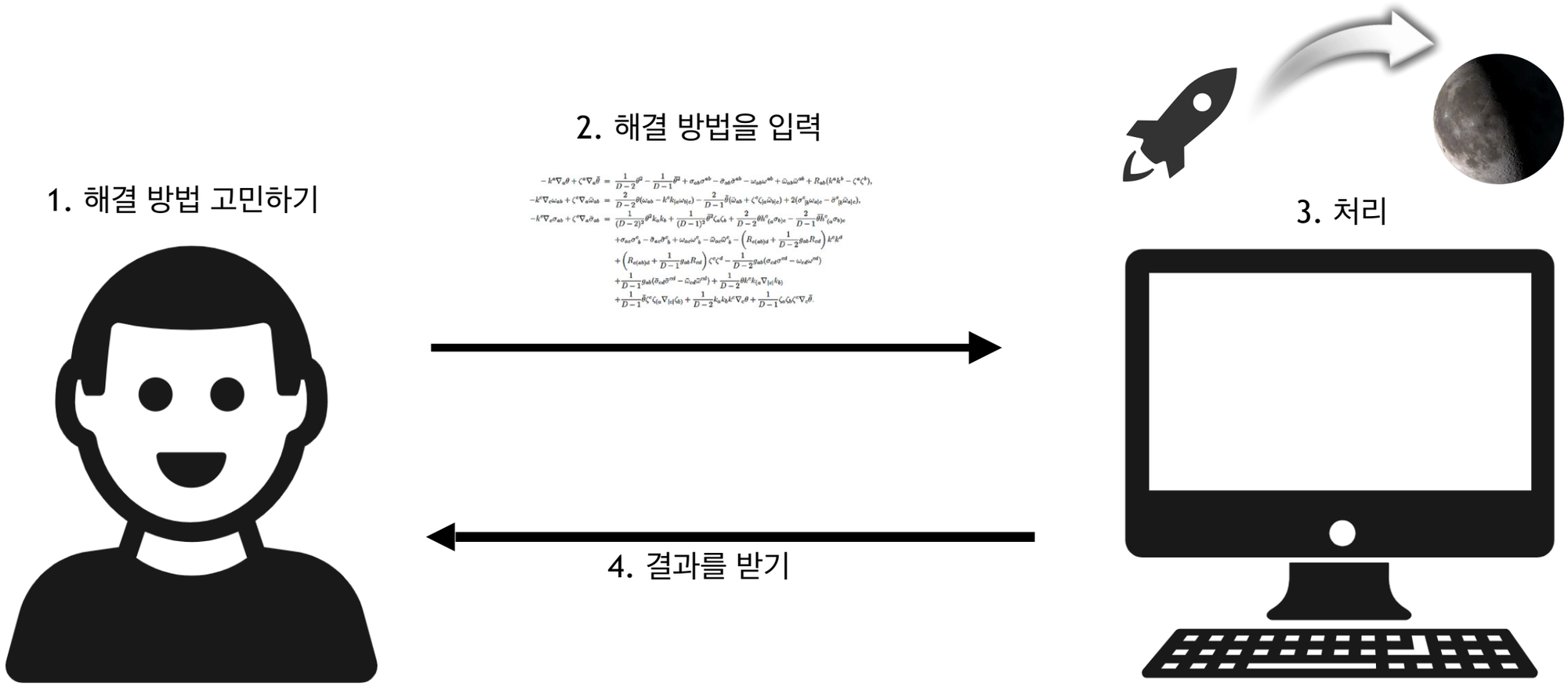
$$
\begin{aligned}
-k^a\nabla_a\theta+\zeta^a\nabla_a\bar{\theta} &= \frac{1}{D-2}\theta^2-\frac{1}{D-1}\bar{\theta}^2+\sigma_{ab}\sigma^{ab}-\bar{\sigma}_{ab}\bar{\sigma}^{ab}-\omega_{ab}\omega^{ab}+\bar{\omega}_{ab}\bar{\omega}^{ab}+R_{ab}(k^ak^b-\zeta^a\zeta^b),\\
-k^c\nabla_c\omega_{ab}+\zeta^c\nabla_a\bar{\omega}_{ab} &= \frac{2}{D-2}\theta(\omega_{ab}-k^ek_{[a}\omega_{b]c})-\frac{2}{D-1}\bar{\theta}(\bar{\omega}_{ab}+\zeta^c\zeta_{[a}\bar{\omega}_{b]c})+2(\sigma^c{}_{[b}\omega_{a]c}-\bar{\sigma}^c{}_{[b}\bar{\omega}_{a]c}),\\
-k^c\nabla_c\sigma_{ab}+\zeta^c\nabla_a\bar{\sigma}_{ab} &= \frac{1}{(D-2)^2}\theta^2k_ak_b+\frac{1}{(D-1)^2}\bar{\theta}^2\zeta_a\zeta_b+\frac{2}{D-2}\theta h^c{}_{(a}\sigma_{b)c}-\frac{2}{D-1}\bar{\theta}\bar{h}^c{}_{(a}\sigma_{b)c}\\
&\quad+\sigma_{ac}\sigma^c{}_b-\bar{\sigma}_{ac}\bar{\sigma}^c{}_b+\omega_{ac}\omega^c{}_b-\bar{\omega}_{ac}\bar{\omega}^c{}_b-\left(R_{c(ab)d}+\frac{1}{D-2}g_{ab}R_{cd}\right)k^ck^d\\
&\quad+\left(R_{c(ab)d}+\frac{1}{D-1}g_{ab}R_{cd}\right)\zeta^c\zeta^d-\frac{1}{D-2}g_{ab}(\sigma_{cd}\sigma^{cd}-\omega_{cd}\omega^{cd})\\
&\quad+\frac{1}{D-1}g_{ab}(\bar{\sigma}_{cd}\bar{\sigma}^{cd}-\bar{\omega}_{cd}\bar{\omega}^{cd})+\frac{1}{D-2}\theta k^ck_{(a}\nabla_{|c|}k_{b)}\\
&\quad+\frac{1}{D-1}\bar{\theta}\zeta^c\zeta_{(a}\nabla_{|c|}\zeta_{b)}+\frac{1}{D-2}k_ak_bk^c\nabla_c\theta+\frac{1}{D-1}\zeta_a\zeta_b\zeta^c\nabla_c\bar{\theta}.
\end{aligned}
$$

⇒ 하나하나 계산하기가 너무 어렵습니다.

컴퓨터에게 일을 시킨다는 것

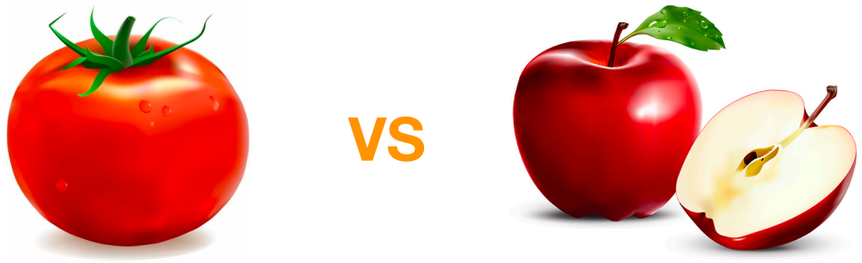
명시적 프로그래밍이란?

개발자(사람) 가 입력조건과 프로그램 상태 조건에 따라 프로그램이 동작하는 방식을 모두 구현 하는 프로그래밍 방식.

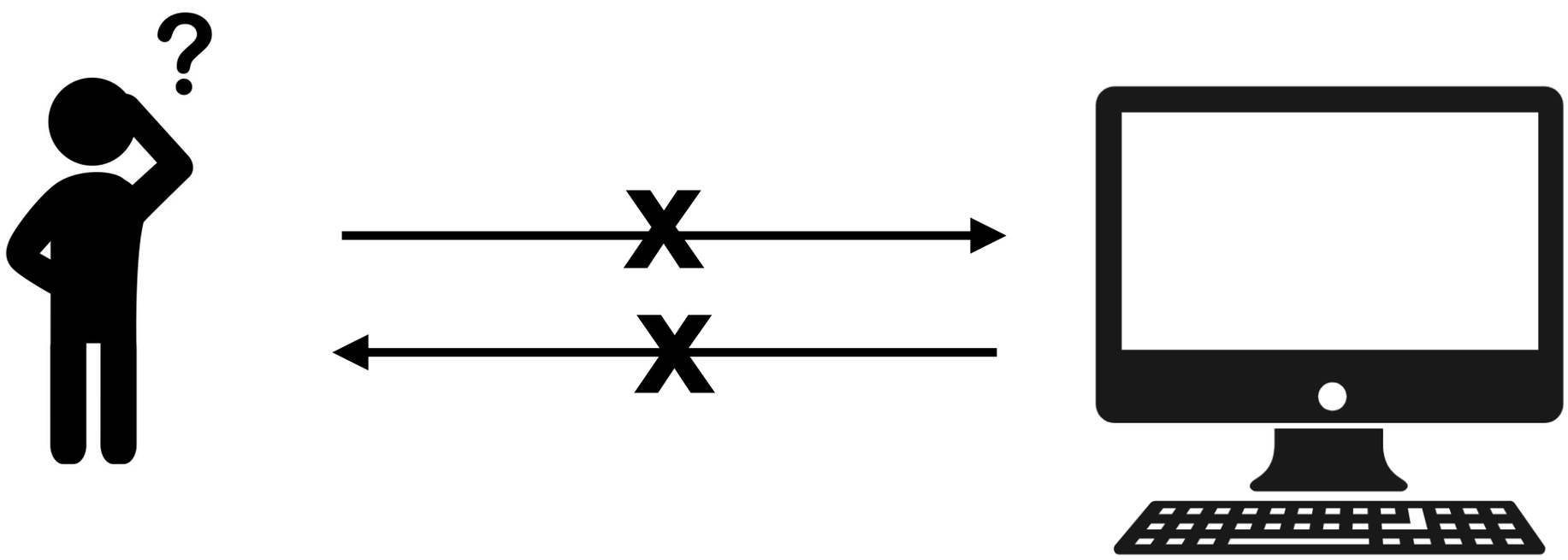


컴퓨터에게 일을 시킨다는 것

머신러닝이란?



1. 사과와 토마토를 구별하는 법 생각하기



컴퓨터에게 일을 시킨다는 것

머신러닝 프로그래밍이란?

머신러닝이란 Arthur Samuel 교수가 제안한 것으로
개발자가 프로그램의 동작방식을 일일이 구현한 것이 아니라 프로그램 스스로가 학습하여 동작방식을 결정하는 것.

1. 해결해야 할 문제 정의 및 데이터 수집

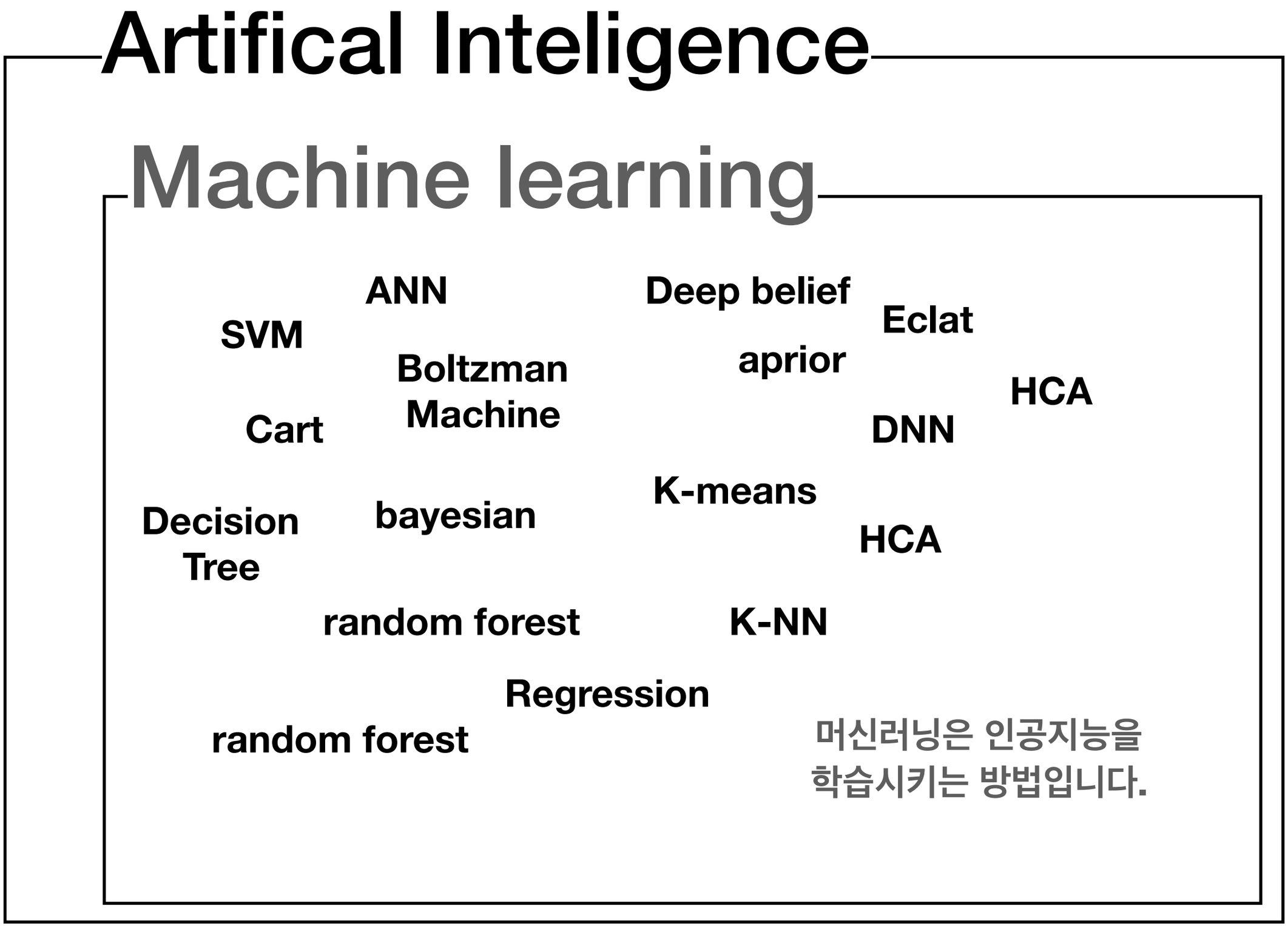
2. 데이터셋 제공

4. 결과를 받기

3. 해결 방법 생성하기

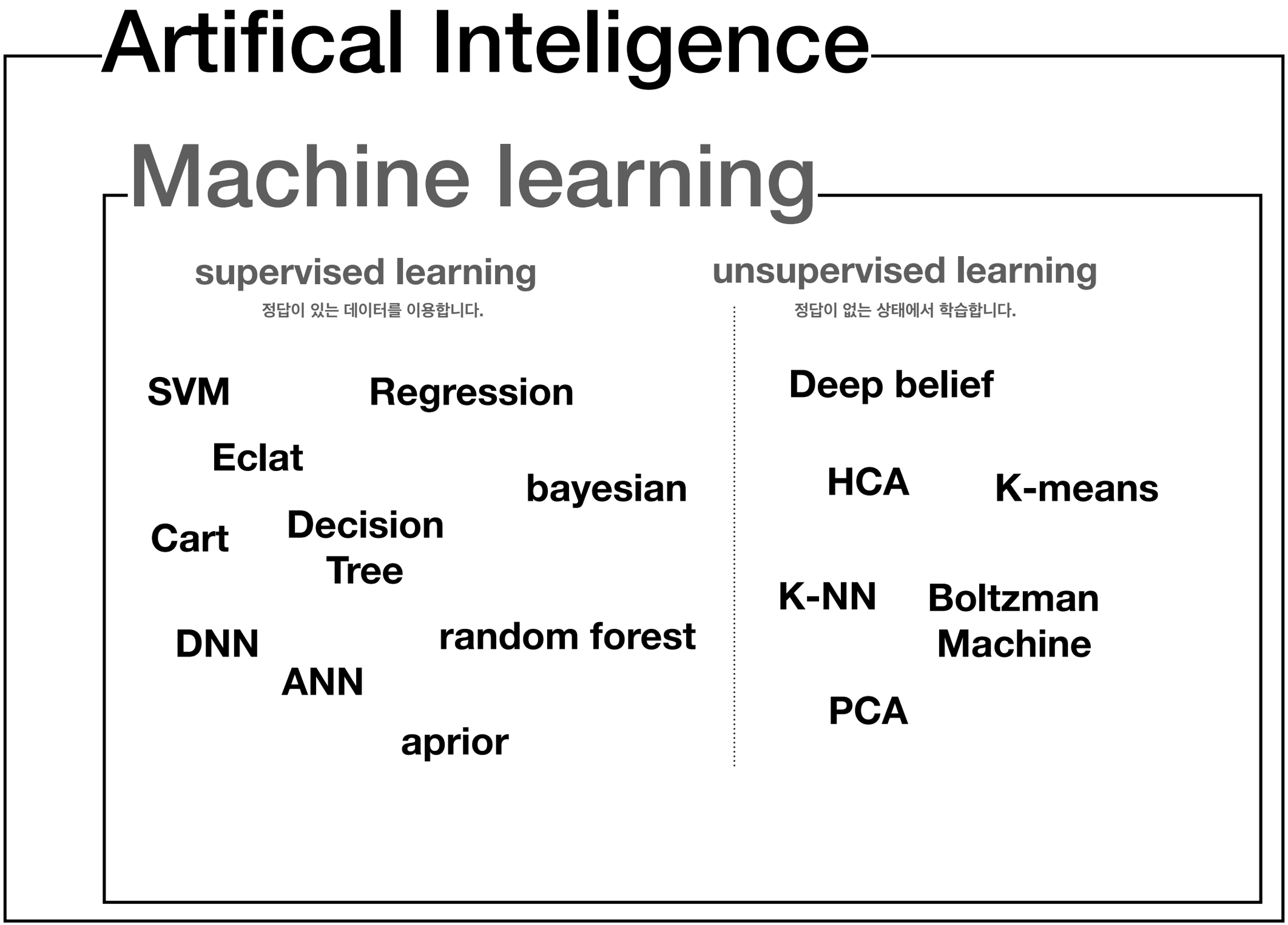
인공지능 개론

인공지능은 무엇일까요?



인공지능 개론

인공지능은 무엇일까요?



인공지능 개론

인공지능은 무엇일까요?

Artifical Inteligence

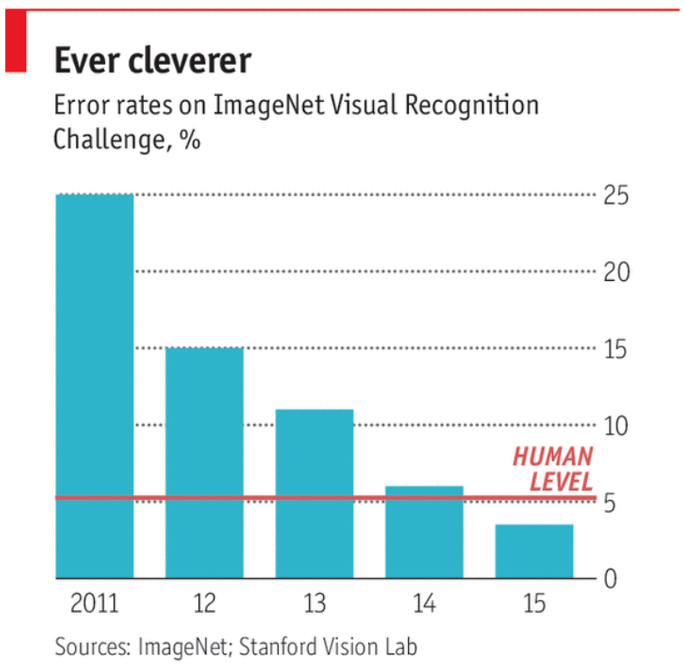
Machine learning

Deep learning

머신 러닝 기법중 하나인
깊은 신경망(Deep Neural Network)
을 이용해 인공지능을 학습

인공지능 개론

인공지능은 무엇일까요?



Deep learning

머신 러닝 기법중 하나인
깊은 신경망(Deep Neural Network)
이용해 기계가 인공적인 지능을
가지게 하는 방법

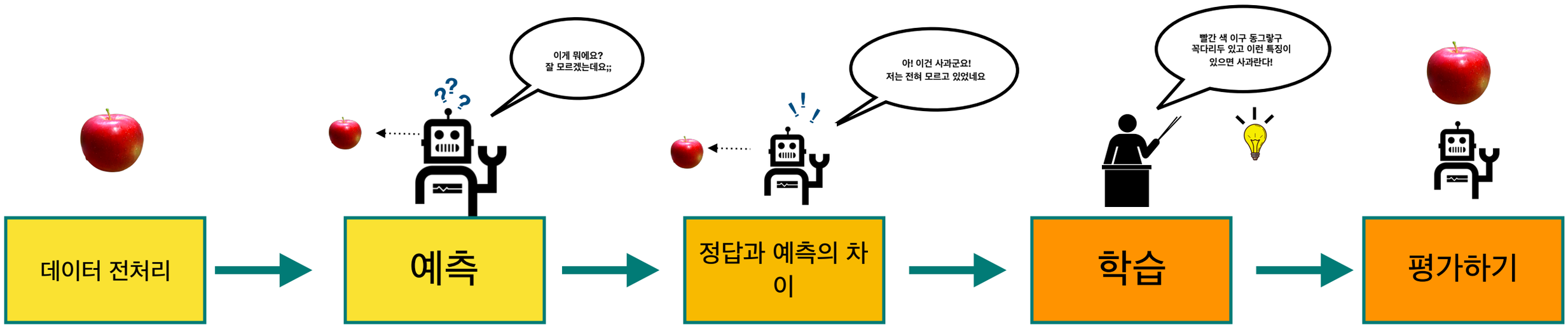
Deep learning :

DNN 을 이용해 만든 모든 알고리즘들

인공지능 개론

인공지능은 무엇일까요?

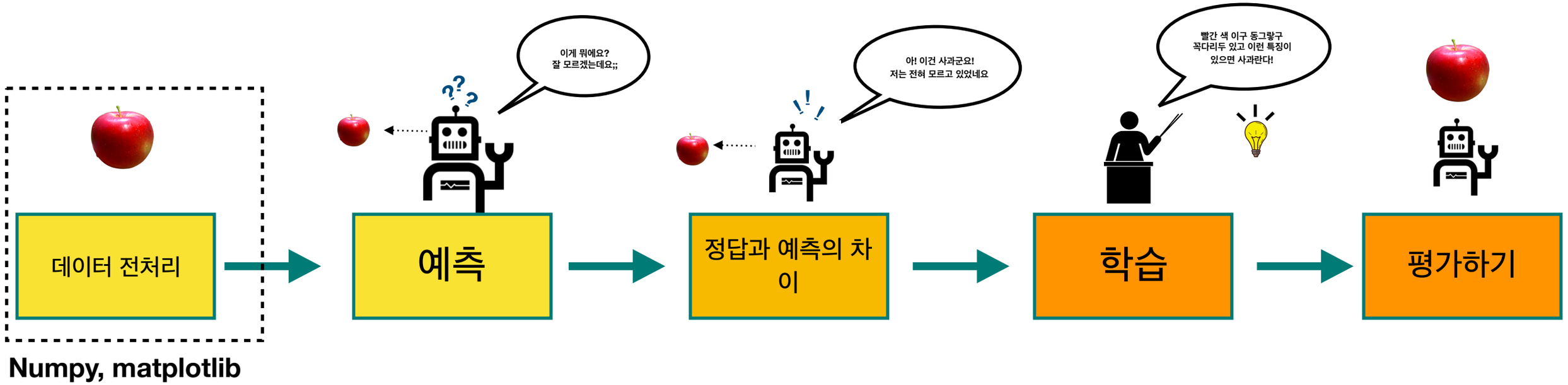
인공지능을 만드는 5가지 과정



인공지능 개론

인공지능은 무엇일까요?

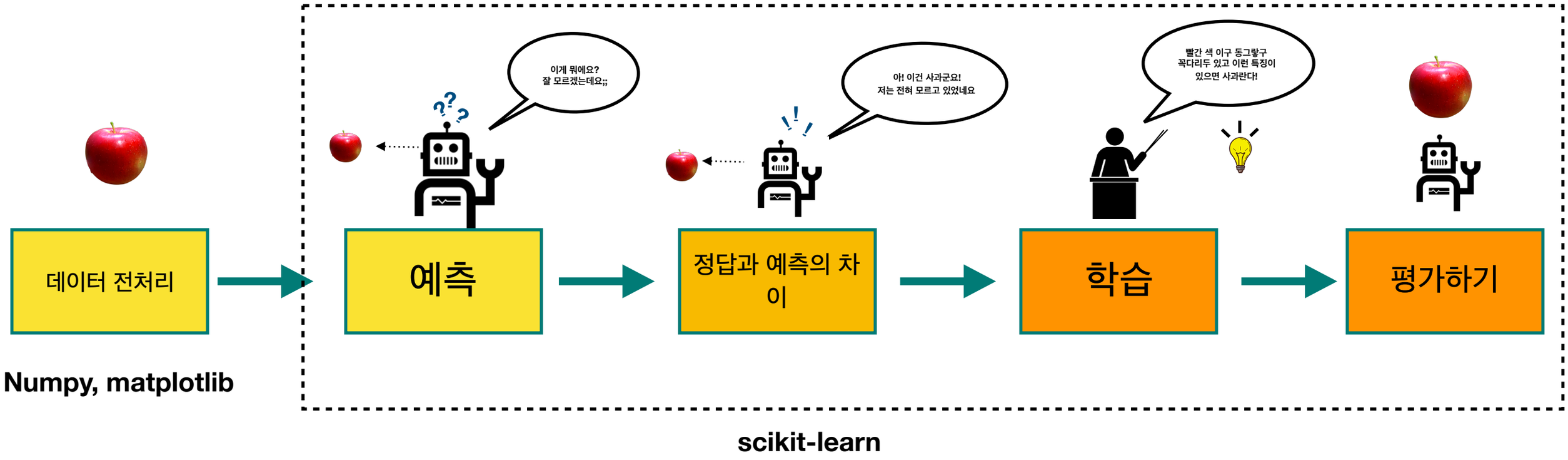
인공지능을 만드는 5가지 과정



인공지능 개론

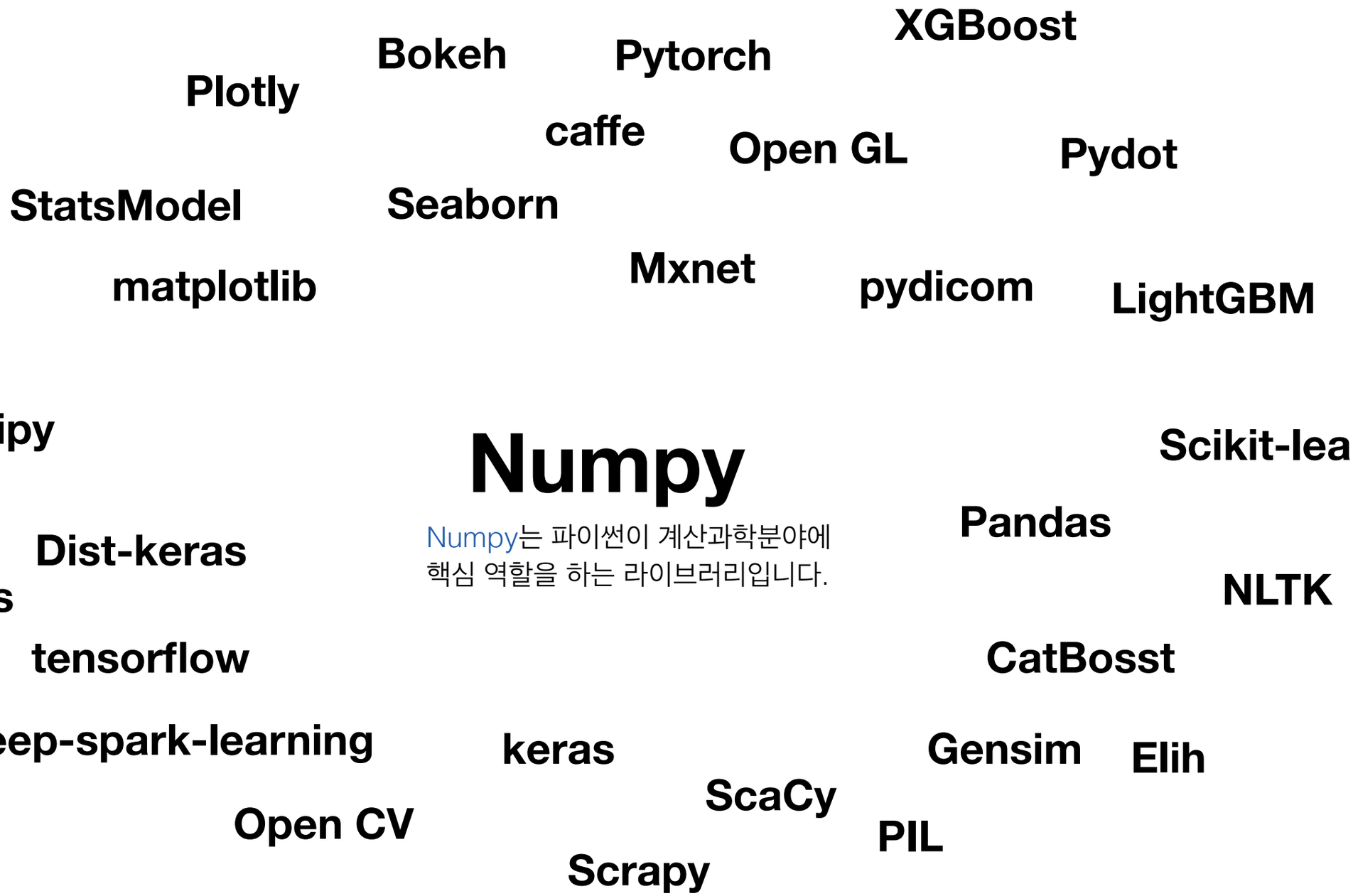
인공지능은 무엇일까요?

인공지능을 만드는 5가지 과정



Numpy 을 사용하는 Library

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기 → 2.Graph 만들기 및 변수 초기 → 3.정답과 예측의 차이 → A,B 수정하기



Numpy 을 사용하는 Library

Numpy 란?

Numpy 란 Numerical Python의 줄임말 입니다.

Numpy 는 다차원 행렬을 담는 Ndarray 클래스를 이용해 복잡하고 다양한 연산을 손쉽게 구할수 있습니다.

Class

Ndarray : N-dimension array

```
np.ndarray
# >>><class 'numpy.ndarray'>
```

Member variable

shape
ndim
itemsize
flags
…

Member function

reshape
min
max
fill
mean
…

create Ndarray

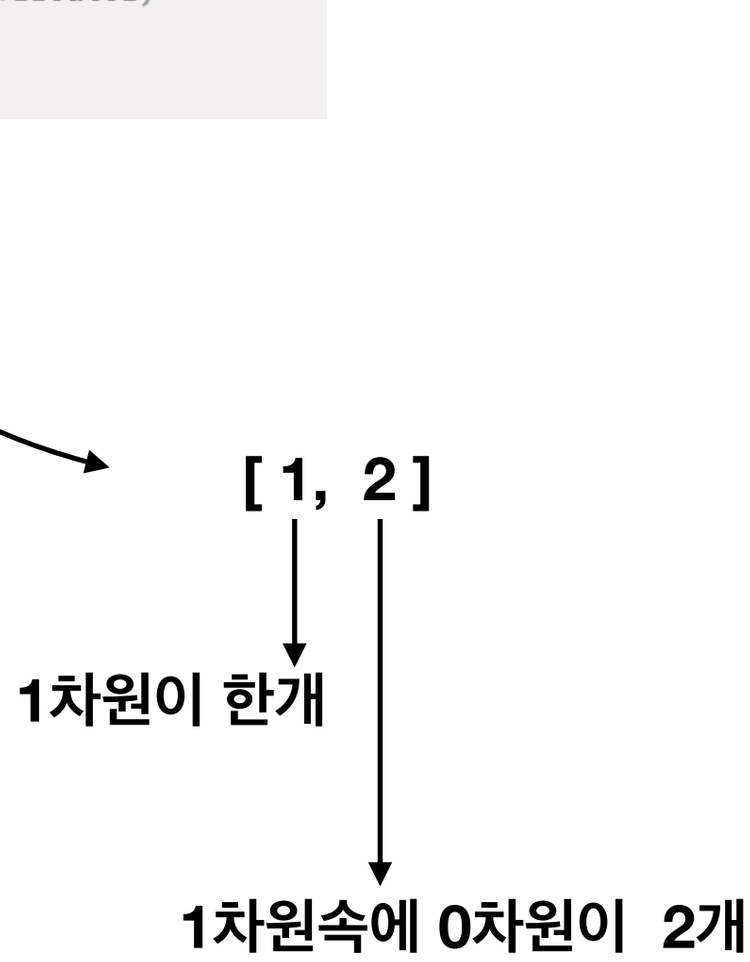
Ndarray 을 생성하는 원리

```
import numpy as np
a = np.empty(shape=[1,2], dtype=np.float32)
print(a)
# >>>[[1. 1.]]
```

create Ndarray

Ndarray 을 생성하는 원리

```
import numpy as np
a = np.empty(shape=[1,2], dtype=np.float32)
print(a)
# >>>[[1. 1.]]
```



차원 이해하기

배열과 행렬

0 차원

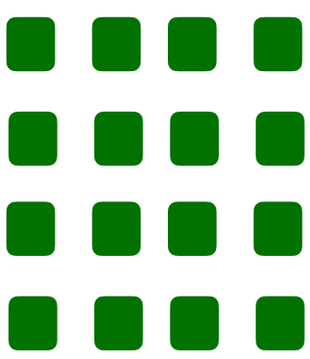
Element(요소)

1차원

Vector(백터)

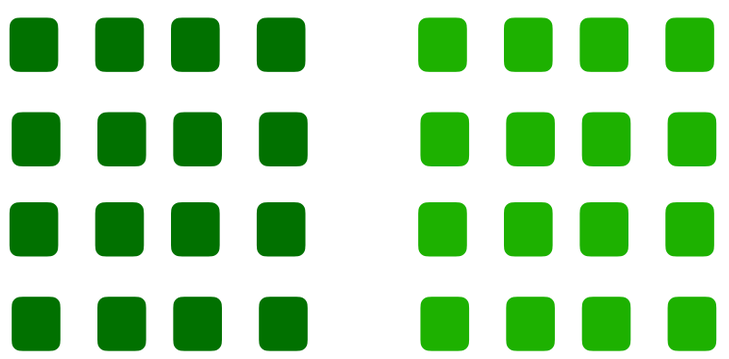
2차원

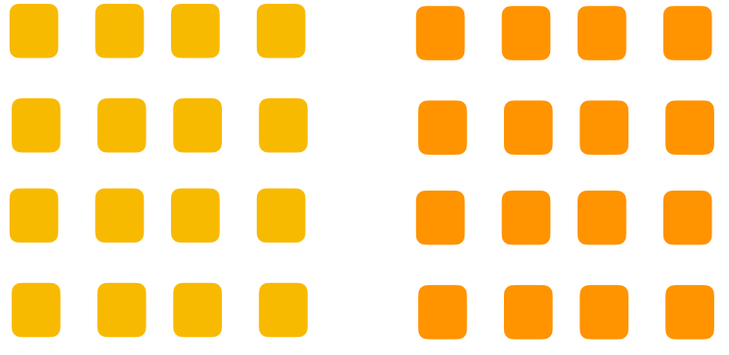
Matrix(행렬)



3차원

Multidimensional Matrix(행렬)





차원 이해하기

0 차원

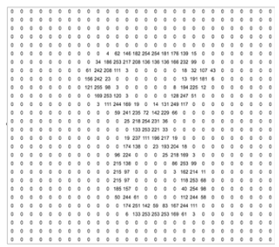
3

1차원

[3 ,3 ,1]

2차원

[[0, 0 ,1]
[1 ,1 ,1]
[0 ,0 ,1]]

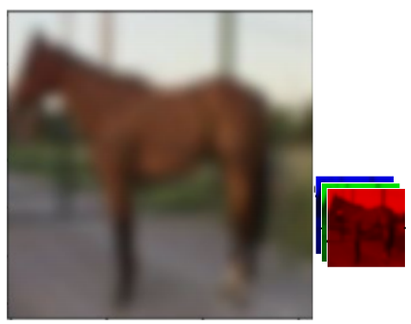


3차원

[[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]

[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]

[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]]



4차원

[[[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]

[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]

[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]]

[[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]

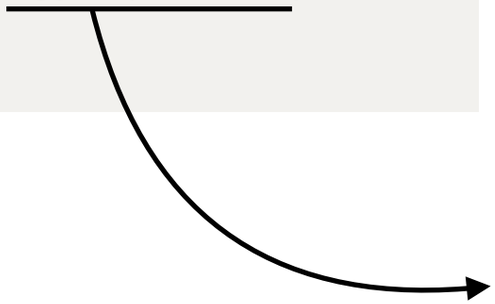
[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]

[[3 ,3 ,1]
[1 ,2 ,3]
[4 ,5 ,6]]]]

create Ndarray

Ndarray 을 생성하는 원리

```
import numpy as np
a = np.empty(shape=[1,2], dtype=np.float32)
print(a)
# >>>[[1. 1.]]
```



데이터가 어떤 형인지 파악해 봅니다.

Data type	Description
`bool_`	Boolean (True or False) stored as a byte
`int_`	Default integer type (same as C `long`; normally either `int64` or `int32`)
intc	Identical to C `int` (normally `int32` or `int64`)
intp	Integer used for indexing (same as C `ssize_t`; normally either `int32` or `int64`)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
`float_`	Shorthand for `float64`.
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
`complex_`	Shorthand for `complex128`.
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

```
f = np.array([1,2],dtype=np.bool_)
print(f)
# [ True  True]
print(f.itemsize)
# 1 1Byte을 의미함
```

Data type	Description
`bool_`	Boolean (True or False) stored as a byte
`int_`	Default integer type (same as C `long`; normally either `int64` or `int32`)
intc	Identical to C `int` (normally `int32` or `int64`)
intp	Integer used for indexing (same as C `ssize_t`; normally either `int32` or `int64`)
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
`float_`	Shorthand for `float64`.
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
`complex_`	Shorthand for `complex128`.
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

```
f = np.array([1,2],dtype=np.bool_)
print(f)
# [ True  True]
print(f.itemsize)
# 1 1Byte을 의미함
```

Create Ndarray

Ndarray 을 생성하는 원리

```
import numpy as np
a = np.empty(shape=[1,2], dtype=np.float32)
print(a)
# >>>[[1. 1.]]
```

```
b = np.zeros(shape=[1, 2], dtype=np.int32)
print(b)
# >>>
[[0, 0 ]]

b_1 = np.zeros(shape= [2, 2], dtype=np.int32)
print(b_1)
[[0 0]
 [0 0]]
```

```
c = np.ones(shape=[1, 2], dtype=np.float64)
print(c)
```

```
c = np.ones(shape=[1, 2], dtype=np.float64)
print(c)
```

```
e = np.arange(0,1,0.1)
print(e)
[0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9]

d = np.linspace(0,1,6)
print(d)
[0.  0.2 0.4 0.6 0.8 1. ]
```

Create Ndarray

Ndarray 을 생성하는 원리

```
f = np.array([1,2],dtype=np.bool_)
print(f)
# [ True  True]
print(f.itemsize)
# 1 1Byte을 의미함
```

```
g = np.zeros_like(f, dtype=np.float32)
print(g)
# [0. 0.]
```

```
f = np.array([1,2],dtype=np.bool_)
print(f)
# [ True  True]
print(f.itemsize)
# 1 1Byte을 의미함
```

```
h = np.full_like(f, fill_value=4, dtype=np.float32)
print(h)
```

Chagne to Ndarray

다른 Class 을 어떻게 ndarray 로 바꿀까요?

Python list → Ndarray

```
python_list = [1,2,3]
print(type(python_list))
#<class 'list'>
g = np.asarray([1,2,3])
print(g)
#[1 2 3]
print(type(g))
#<class 'numpy.ndarray'>
```

Image → Ndarray

```
#PIL → ndarray
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open('./sample.png')
print(img)
#<PIL.PngImagePlugin.PngImageFile ... >
plt.imshow(img)
plt.show()
np_img = np.asarray(img)
print(np_img)
```

csv→ Ndarray

```
my_data = np.genfromtxt('cancer_data.csv', delimiter=',', dtype=np.string_)
print(my_data)
```

Python

Chagne to Ndarray

다른 Class 을 어떻게 ndarray 로 바꿀까요?

Python list → Ndarray

```
python_list = [1,2,3]
print(type(python_list))
#<class 'list'>
g = np.asarray([1,2,3])
print(g)
#[1 2 3]
print(type(g))
#<class 'numpy.ndarray'>
```

문제 : sample Image 을 ndarray 로 변환하고
변환된 ndarray 을 숫자 7.5로 채우세요!

Image → Ndarray

```
#PIL → ndarray
from PIL import Image
import matplotlib.pyplot as plt
img = Image.open('./sample.png')
print(img)
#<PIL.PngImagePlugin.PngImageFile ... >
plt.imshow(img)
plt.show()
np_img = np.asarray(img)
print(np_img)
```

Image Open → np.asarray → np.full_like

구획자를 의미합니다

csv→ Ndarray

```
my_data = np.genfromtxt('cancer_data.csv', delimiter=',', dtype=np.string_)
print(my_data)
```

Python

Chagne to Ndarray

다른 Class 을 어떻게 ndarray 로 바꿀까요?

cancer_data

age	tumor_size	label
2.1708648	3.0877101	1
8.000731	5.237815	0
5.248661	5.8510585	0
3.4334962	4.2783794	1
2.3653207	3.5083961	1
6.432232	2.2827752	0
2.5806835	2.867671	1
5.9208593	6.652007	0
0.959677	3.0462556	1
4.644649	3.121122	0
3.5242965	3.7352796	1
4.6934996	7.6849127	0
5.236967	6.132978	0
3.8025222	9.168974	0
0.6811011	5.817095	0
7.3902392	1.933067	0
5.6210613	5.8455625	0
7.649406	8.496426	0

csv→ Ndarray

```
age,tumor_size,label
2.1708648,3.0877101,1.0
8.000731,5.237815,0.0
5.248661,5.8510585,0.0
3.4334962,4.2783794,1.0
2.3653207,3.5083961,1.0
6.432232,2.2827752,0.0
2.5806835,2.867671,1.0
5.9208593,6.652007,0.0
0.959677,3.0462556,1.0
4.644649,3.121122,0.0
```

Ndarray

```
[[b'age' b'tumor_size' b'label']
 [b'2.1708648' b'3.0877101' b'1.0']
 [b'8.000731' b'5.237815' b'0.0']
 [b'5.248661' b'5.8510585' b'0.0']
 [b'3.4334962' b'4.2783794' b'1.0']
 [b'2.3653207' b'3.5083961' b'1.0']
 [b'6.432232' b'2.2827752' b'0.0']
 [b'2.5806835' b'2.867671' b'1.0']
 [b'5.9208593' b'6.652007' b'0.0']
 [b'0.959677' b'3.0462556' b'1.0']
 [b'4.644649' b'3.121122' b'0.0']
 [b'3.5242965' b'3.7352796' b'1.0']
 [b'4.6934996' b'7.6849127' b'0.0']
```

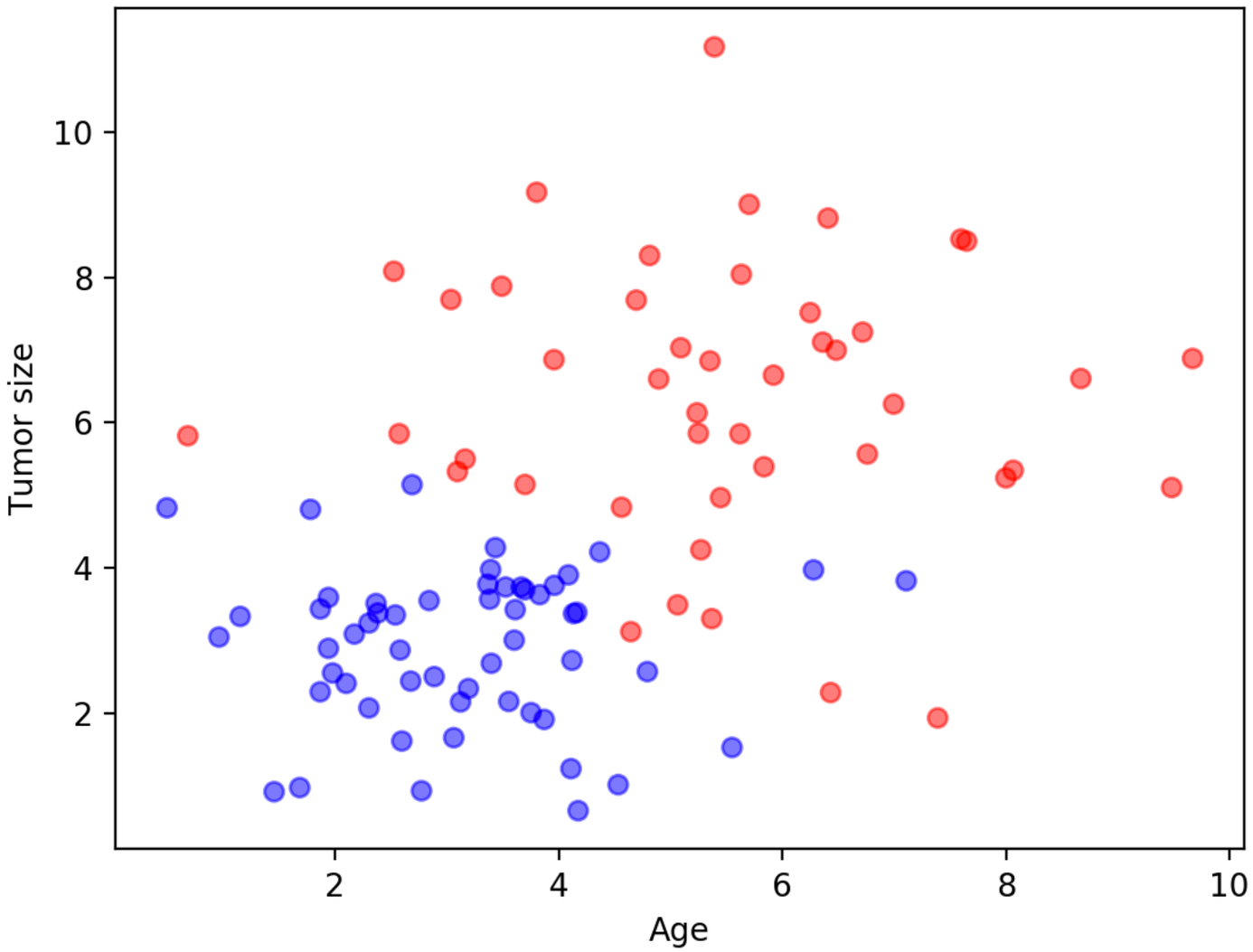
구획자를 의미합니다

```
my_data = np.genfromtxt('cancer_data.csv', delimiter=',', dtype=np.string_)
print(my_data)
```

Python

머신러닝으로 암 예측하기

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기



feature or 변수

라벨

나이	종양 크기	암 여부
2.9808695	4.175001	1.0
4.504258	6.0180507	0.0
4.2437844	5.687132	0.0
3.2565703	2.011221	1.0
2.661178	2.7638159	1.0
4.72469	3.6247754	0.0
1.5787828	2.8465047	1.0
5.461886	10.462733	0.0
0.5652324	3.1127264	1.0
6.740889	8.719268	0.0
3.5018573	2.1557863	1.0
6.0000196	7.0847054	0.0
5.3729835	7.5420237	0.0
2.2638187	9.462369	0.0
8.935356	5.328645	0.0
7.2226815	6.095941	0.0
4.3417296	6.1754203	0.0
⋮	⋮	⋮
5.236967	6.132978	1.0
3.8025222	9.168974	0.0
0.6811011	5.817095	1.0
3.6951196	0.9665335	1.0

100

0 : 암
1: 정상

Chagne to Ndarray

다른 Class 을 어떻게 ndarray 로 바꿀까요?

```
[[b'age' b'tumor_size' b'label']
 [b'2.1708648' b'3.0877101' b'1.0']
 [b'8.000731' b'5.237815' b'0.0']
 [b'5.248661' b'5.8510585' b'0.0']
 [b'3.4334962' b'4.2783794' b'1.0']
 [b'2.3653207' b'3.5083961' b'1.0']
 [b'6.432232' b'2.2827752' b'0.0']
 [b'2.5806835' b'2.867671' b'1.0']
 [b'5.9208593' b'6.652007' b'0.0']
 [b'0.959677' b'3.0462556' b'1.0']
 [b'4.644649' b'3.121122' b'0.0']
 [b'3.5242965' b'3.7352796' b'1.0']
 [b'4.6934996' b'7.0849127' b'0.0']
```

제거 해야 합니다.

Slicing을 이용해서 데이터의 필요 없는 부분을 제거 할 수 있습니다.

분리해야 합니다.

Slicing을 이용해서 하나의 데이터를 여러개로 쪼갤수 있습니다.

Slicing

Numpy 란?

1차원 데이터 slicing 해보기

```
# case 1
data = np.arange(0,9,1)
print(data)
#s = slice(2,7,2)
print(data[2:5])
print(data[2:])
print(data[2:5:2])

#[0 1 2 3 4 5 6 7 8]
#[2 3 4]
#[2 3 4 5 6 7 8]
#[2 4]
```

Slicing

Numpy 란?

1차원 데이터 slicing 해보기

```
# case 1
data = np.arange(0,9,1)
print(data)
#s = slice(2,7,2)
print(data[2:5])
print(data[2:])
print(data[2:5:2])

#[0 1 2 3 4 5 6 7 8]
#[2 3 4]
#[2 3 4 5 6 7 8]
#[2 4]
```

Slicing

Numpy 란?

1차원 데이터 slicing 해보기

```
# case 1
data = np.arange(0,9,1)
print(data)
#s = slice(2,7,2)
print(data[2:5])
print(data[2:])
print(data[2:5:2])

#[0 1 2 3 4 5 6 7 8]
#[2 3 4]
#[2 3 4 5 6 7 8]
#[2 4]
```

Slicing

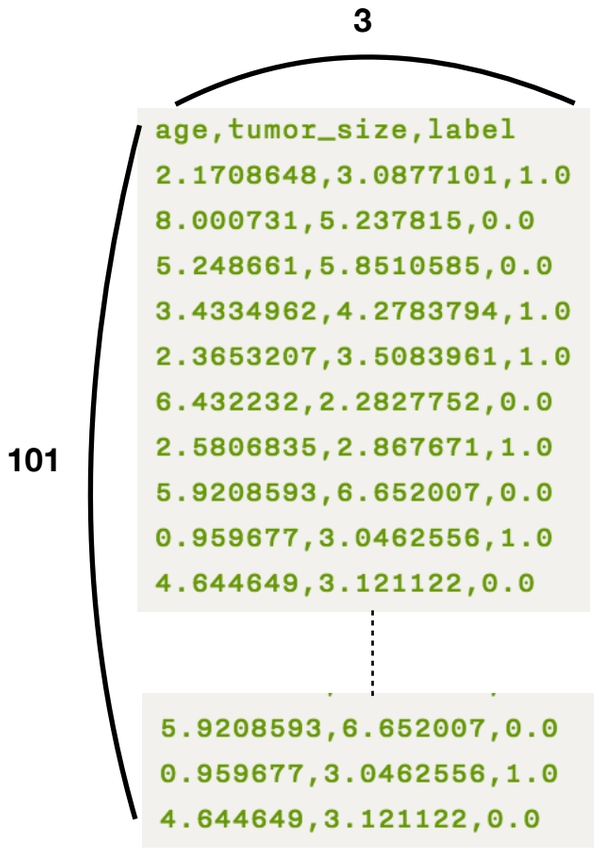
Numpy 란?

1차원 데이터 slicing 해보기

```
# case 1
data = np.arange(0,9,1)
print(data)
#s = slice(2,7,2)
print(data[2:5])
print(data[2:])
print(data[2:5:2])

#[0 1 2 3 4 5 6 7 8]
#[2 3 4]
#[2 3 4 5 6 7 8]
#[2 4]
```

2차원 데이터 slicing 해보기

```
# case 2
datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.string_)
print(datum.shape)
# (101, 3)
```

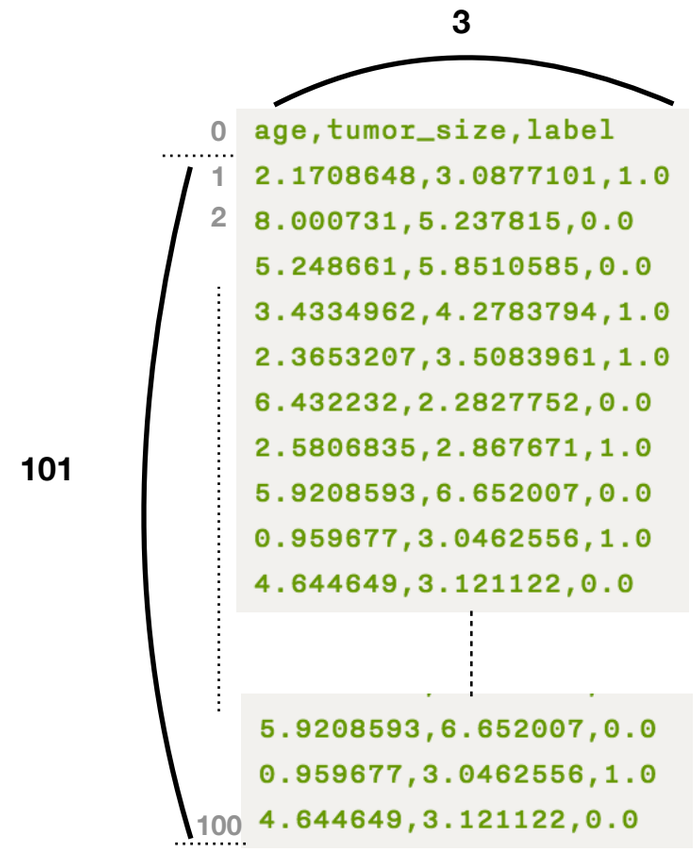


Slicing

Numpy 란?

2차원 데이터 slicing 해보기

```
# case 2
datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.string_)
print(datum.shape)
# except header
datum = datum[1:,:]
```

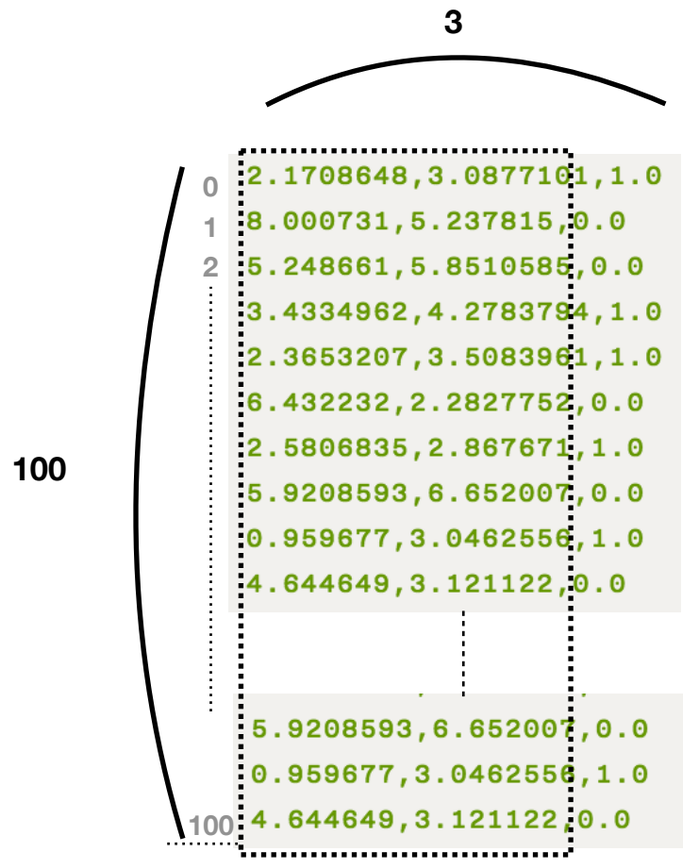


Slicing

Numpy 란?

2차원 데이터 slicing 해보기

```
# case 2
datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.string_)
print(datum.shape)
# except header
datum = datum[1:,:]
# split datum to X, Y
X = datum[:, :2]
```

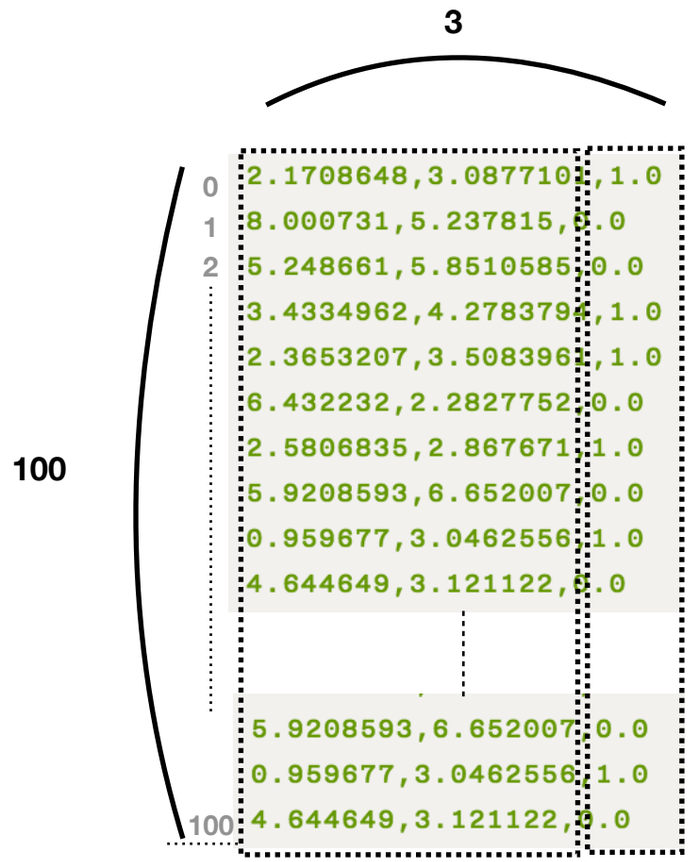


Slicing

Numpy 란?

2차원 데이터 slicing 해보기

```
# case 2
datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.string_)
print(datum.shape)
# except header
datum = datum[1:,:]
# split datum to X, Y
X = datum[:, :2]
print(X)
"""
[[b'2.1708648' b'3.0877101']
 [b'8.000731' b'5.237815']
 [b'5.248661' b'5.8510585']
 [b'3.4334962' b'4.2783794']
 [b'2.3653207' b'3.5083961']
 [b'6.432232' b'2.2827752']
 [b'2.5806835' b'2.867671']
 [b'5.9208593' b'6.652007']
 [b'0.959677' b'3.0462556']
 [b'4.644649' b'3.121122']
 [b'3.5242965' b'3.7352796']
 """
Y = datum[:, 2]
```



Train(60%)

모델을 만드는데 사용하는 데이터

Validation(20%)

모델을 검증하는데 사용하는 데이터
학습시 중간중간 학습이 잘 되고 있나
확인해야 할 때 사용함.

Test(20%)

실제로 검증해야 하는 데이터

데이터 분할 해 보기 : Slicing

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

Train(60%)

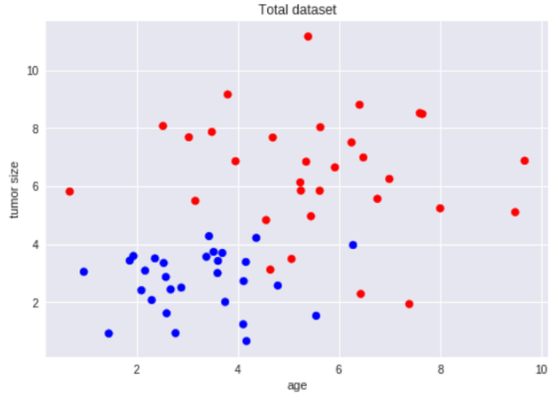
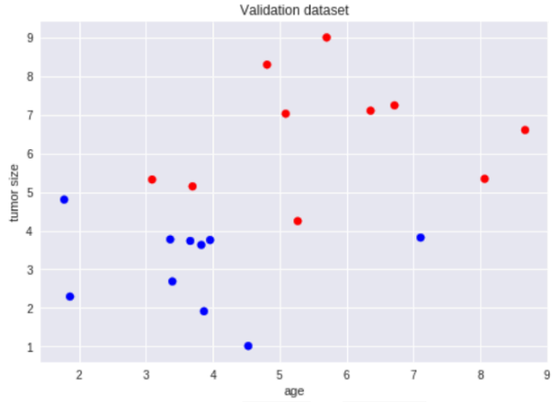
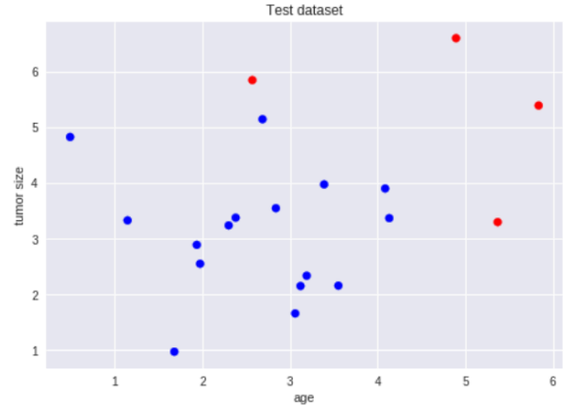
모델을 만드는데 사용하는 데이터

Validation(20%)

모델을 검증하는데 사용하는 데이터
학습시 중간중간 학습이 잘 되고 있나 확인해야 할 때 사용함.

Test(20%)

실제로 검증해야 하는 데이터

Train	Validation	Test
		

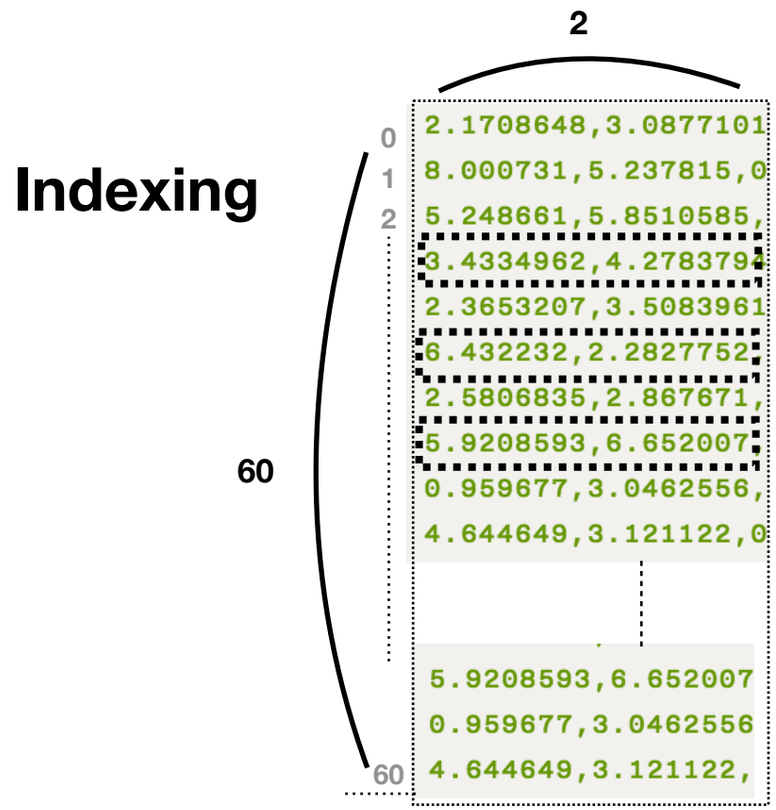
데이터 뽑아보기 : Indexing

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

train_x data중에 3행, 5행, 7행, 11행 을 확인할려면?

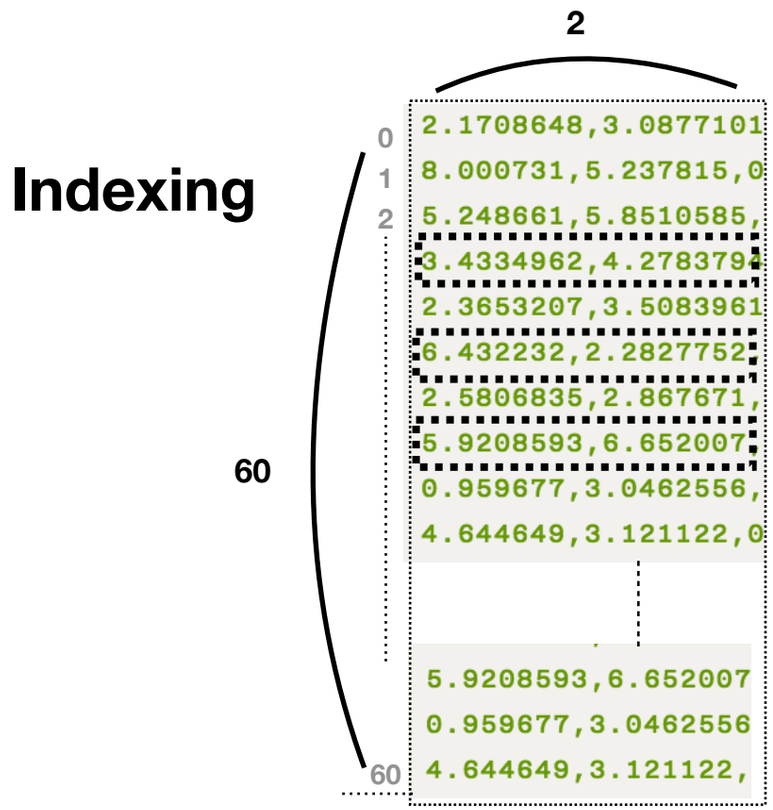
```
print(train_x[3])
print(train_x[5])
print(train_x[7])
print(train_x[11])
```



데이터 뽑아보기 : Indexing

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```



train_x data중에 3행, 5행, 7행, 11행 을 확인할려면?

```
print(train_x[3])
print(train_x[5])
print(train_x[7])
print(train_x[11])

indices = [3,5,7,11]
print(train_x[indices])
```

데이터 형 변환하기: astype

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

Data type(dtype?) |S10

10개 짜리 String이 여러개 들어가 있다.

```
dtype 접두사  설명 사용 예

b 불리언  b (참 혹은 거짓)
i 정수 i8 (64비트)
u 부호 없는 정수  u8 (64비트)
f 부동소수점 f8 (64비트)
c 복소 부동소수점 c16 (128비트)
O 객체 0 (객체에 대한 포인터)
S 바이트 문자열   S24 (24 글자)
U 유니코드 문자열 U24 (24 유니코드 글자)
```

최소값 , 최대값, 평균 , 편차 , 표준편차 구해보기

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

Data type(dtype?) |S10

10개 짜리 String이 여러개 들어가 있다.

Float32 로 바꾸고 싶습니다!

```
train_x = np.asarray(train_x, np.float32)
train_x = train_x.astype(np.float32)
```

바꾸는 방법이 2가지가 있는데 후자를 추천합니다.

최소값 , 최대값, 평균 , 편차 , 표준편차 구해보기

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

Test , Validation 을 구하고 Train 과 얼만큼 비슷한지 확인해주세요.

Data type(dtype?) S10

10개 짜리 String이 여러개 들어가 있다.

Float32 로 바꾸고 싶습니다!

```
train_x = np.asarray(train_x, np.float32)
train_x = train_x.astype(np.float32)
```

바꾸는 방법이 2가지가 있는데 후자를 추천합니다.

```
print(train_x.mean())
print(train_x.min())
print(train_x.max())
print(train_x.std())
```

최소값, 최대값, 평균, 표준편차

최소값 , 최대값, 평균 , 편차 , 표준편차 구해보기

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

Data type(dtype?) |S10

10개 짜리 String이 여러개 들어가 있다.

Float32 로 바꾸고 싶습니다!

```
train_x = np.asarray(train_x, np.float32)
train_x = train_x.astype(np.float32)
```

바꾸는 방법이 2가지가 있는데 후자를 추천합니다.

```
print(train_x.mean())
print(train_x.min())
print(train_x.max())
print(train_x.std())
```

최소값, 최대값, 평균, 표준편차

Test , Validation 을 구하고 Train 과 얼만큼 비슷한지 확인해주세요.

	Train	Validation	Teset
mean	4.544585	3.2810206	4.790821
min	0.6563968	0.49454185	1.0141542
max	11.170589	6.599054	9.002873
std	2.2260046	1.3885382	1.9838969

최소값 , 최대값, 평균 , 편차 , 표준편차 구해보기

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

Data type(dtype?) |S10

10개 짜리 String이 여러개 들어가 있다.

Float32 로 바꾸고 싶습니다!

```
train_x = np.asarray(train_x, np.float32)
train_x = train_x.astype(np.float32)
```

바꾸는 방법이 2가지가 있는데 후자를 추천합니다.

```
print(train_x.mean())
print(train_x.min())
print(train_x.max())
print(train_x.std())
```

최소값, 최대값, 평균, 표준편차

Train ≈ Test

일반화(Generalization) ↑

Test , Validation 을 구하고 Train 과 얼만큼 비슷한지 확인해주세요.

	Train	Validation	Teset
mean	4.544585	3.2810206	4.790821
min	0.6563968	0.49454185	1.0141542
max	11.170589	6.599054	9.002873
std	2.2260046	1.3885382	1.9838969

최소값 , 최대값, 평균 , 편차 , 표준편차 구해보기

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

Data type(dtype?) |S10

10개 짜리 String이 여러개 들어가 있다.

Float32 로 바꾸고 싶습니다!

```
train_x = np.asarray(train_x, np.float32)
train_x = train_x.astype(np.float32)
```

바꾸는 방법이 2가지가 있는데 후자를 추천합니다.

```
print(train_x.mean())
print(train_x.min())
print(train_x.max())
print(train_x.std())
```

최소값, 최대값, 평균, 표준편차

Train ≈ Test

일반화(Generalization) ↑

Test , Validation 을 구하고 Train 과 얼만큼 비슷한지 확인해주세요.

	Train	Validation	Teset
mean	4.544585	3.2810206	4.790821
min	0.6563968	0.49454185	1.0141542
max	11.170589	6.599054	9.002873
std	2.2260046	1.3885382	1.9838969

최소값 , 최대값, 평균 , 편차 , 표준편차 구해보기

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

Data type(dtype?) |S10

10개 짜리 String이 여러개 들어가 있다.

Float32 로 바꾸고 싶습니다!

```
train_x = np.asarray(train_x, np.float32)
train_x = train_x.astype(np.float32)
```

바꾸는 방법이 2가지가 있는데 후자를 추천합니다.

```
print(train_x.mean())
print(train_x.min())
print(train_x.max())
print(train_x.std())
```

최소값, 최대값, 평균, 표준편차

Train ≈ Test

일반화(Generalization) ↑

Test , Validation 을 구하고 Train 과 얼만큼 비슷한지 확인해주세요.

	Train	Validation	Teset
mean	4.544585	3.2810206	4.790821
min	0.6563968	0.49454185	1.0141542
max	11.170589	6.599054	9.002873
std	2.2260046	1.3885382	1.9838969

최소값 , 최대값, 평균 , 편차 , 표준편차 구해보기

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
import numpy as np
import numpy.random as npr

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.string_)

# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]
```

[1, 2, 3] ⟶ [3, 1, 2]

[[1, 2, 3]
[4, 5, 6] ⟶ [[4, 5, 6]
[7, 8, 9]] [7, 8, 9]
[1, 2, 3]]

EDA (탐색적 자료 분석)

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

Data type(dtype?) |S10

10개 짜리 String이 여러개 들어가 있다.

Float32 로 바꾸고 싶습니다!

```
train_x = np.asarray(train_x, np.float32)
train_x = train_x.astype(np.float32)
```

바꾸는 방법이 2가지가 있는데 후자를 추천합니다.

```
print(train_x.mean())
print(train_x.min())
print(train_x.max())
print(train_x.std())
```

최소값, 최대값, 평균, 표준편차

Train ≈ Test

일반화(Generalization) ↑

Test , Validation 을 구하고 Train 과 얼만큼 비슷한지 확인해주세요.

	Train	Validation	Teset
mean	4.459631	4.029236	4.297469
min	0.49454185	0.6811011	0.9173481
max	9.670666	8.496426	11.170589
std	2.1628246	1.8586345	2.133107

EDA (탐색적 자료 분석)

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

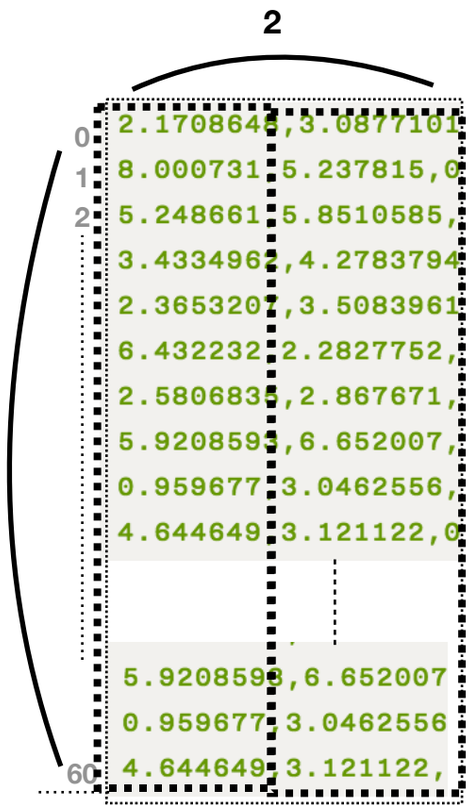
Train_x, Test_x,Validation_x
각 열별로 Mean, min , max , std 을 구해보세요.

Train ≈ Test

일반화(Generalization) ↑

Test , Validation 을 구하고 Train 과 얼만큼 비슷한지 확인해주세요.

	Train	Validation	Teset
mean	4.459631	4.029236	4.297469
min	0.49454185	0.6811011	0.9173481
max	9.670666	8.496426	11.170589
std	2.1628246	1.8586345	2.133107



EDA (탐색적 자료 분석)

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

Train ≈ Test

일반화(Generalization) ↑

	Train		Validation		Test	
	나이	종양크기	나이	종양크기	나이	종양크기
mean	4.1133013	3.9665842	4.046639	3.9357688	4.1334004	5.3768334
min	0.49454185	0.6563968	1.149596	1.0141542	0.6811011	0.9744064
max	9.670666	11.170589	7.3902392	11.170589	8.066058	9.168974
std	1.9756825	2.2250803	1.8039889	2.076409	2.0375364	2.2116494

Train_x[:,0] 에서 mean 보다 큰값을 찾아보세요

Searching

머신러닝으로 암 예측해보기 (다변수 선형회귀)
1.데이터 살펴보기

```
# divide data to train, validation , test

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]


train_y= Y[:60]
test_Y = Y[60:80]
val_Y = Y[80:]
```

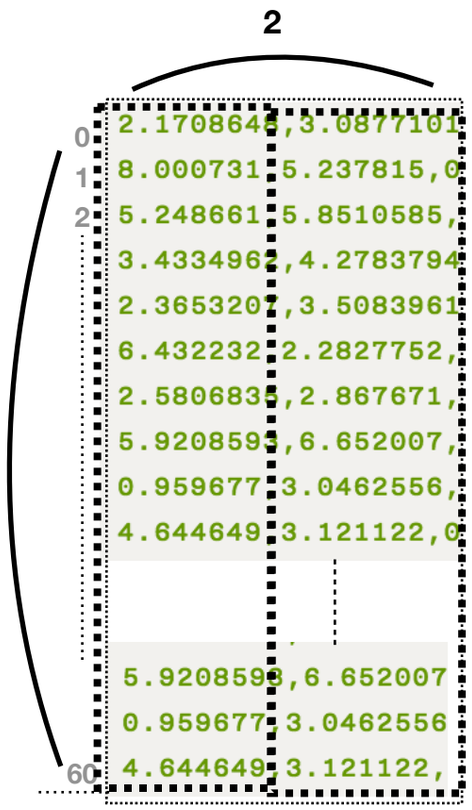
Train_x, Test_x,Validation_x
각 열별로 Mean, min , max , std 을 구해보세요.

Train ≈ Test

일반화(Generalization) ↑

Test , Validation 을 구하고 Train 과 얼만큼 비슷한지 확인해주세요.

	Train	Validation	Teset
mean	4.459631	4.029236	4.297469
min	0.49454185	0.6811011	0.9173481
max	9.670666	8.496426	11.170589
std	2.1628246	1.8586345	2.133107



Searching

```
indices = np.where(train_x[:,0]>train_x.mean())[0]
print(train_x[indices])
print(train_x.mean())
'''
(array([ 0,  3,  5,  8, 13, 18, 19, 22, 24, 33, 34, 40, 41, 43, 46, 49, 51,
       52, 53, 54, 55, 56, 58]),)
'''
```

Searching

```
indices = np.where(train_x[:,0]>train_x.mean())[0]
print(train_x[indices])
print(train_x.mean())
'''
(array([ 0,  3,  5,  8, 13, 18, 19, 22, 24, 33, 34, 40, 41, 43, 46, 49, 51,
       52, 53, 54, 55, 56, 58]),)
'''
```

Searching

```
indices = np.where(train_x[:,0]>train_x.mean())[0]
print(train_x[indices])
print(train_x.mean())
'''
(array([ 0,  3,  5,  8, 13, 18, 19, 22, 24, 33, 34, 40, 41, 43, 46, 49, 51,
       52, 53, 54, 55, 56, 58]),)
'''
```

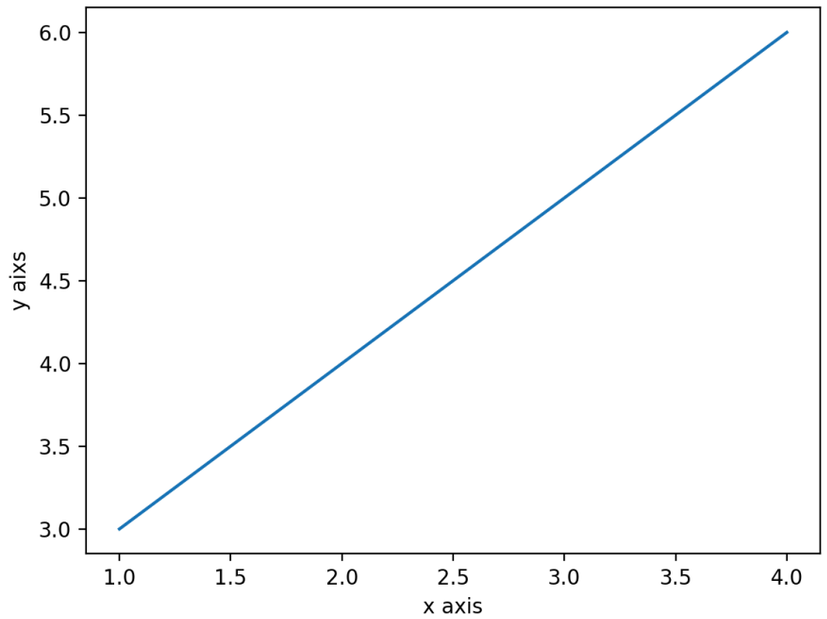
지금 배우고 있는것들은 나중에도 아주 많이 사용되는 것들입니다.

```
    return shifts


def _out_of_boundary(all_anchors, _allowed_border, im_dims):
    inds_inside = np.where(
        (all_anchors[:, 0] >= -_allowed_border) &
        (all_anchors[:, 1] >= -_allowed_border) &
        (all_anchors[:, 2] < im_dims[1] + _allowed_border) &  # <-- width
        (all_anchors[:, 3] < im_dims[0] + _allowed_border))[0]# <-- height
    return inds_inside


def _get_max_overlaps(overlaps, inds_inside):
    argmax_overlaps = overlaps.argmax(axis=1)
    max_overlaps = overlaps[np.arange(len(inds_inside)), argmax_overlaps]  # inds_inside 갯수 만큼 overlaps에서 가장 높은 overlays
    return max_overlaps, argmax_overlaps
```

Matplotlib

matplotlib 불러오기

```
import matplotlib.pyplot as plt

xs = [1,2,3,4]
ys = [3,4,5,6]
plt.plot(xs, ys)
plt.xlabel('x axis')
plt.ylabel('y aixs')
plt.show()
```

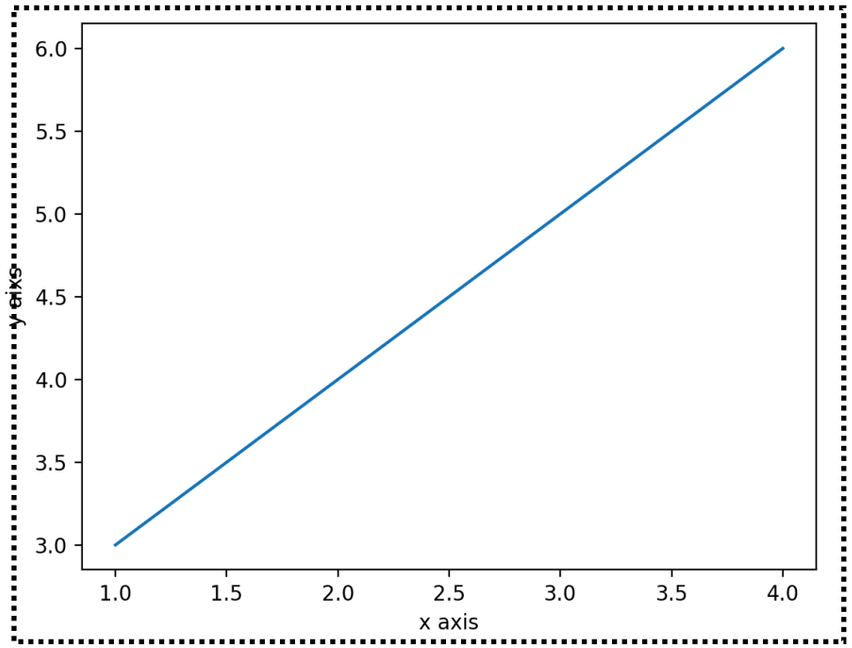


Matplotlib

데이터를 시각화하기!

Plotting

```
import matplotlib.pyplot as plt

xs = [1,2,3,4]
ys = [3,4,5,6]
plt.plot(xs, ys)
plt.xlabel('x axis')
plt.ylabel('y aixs')
plt.show()
```

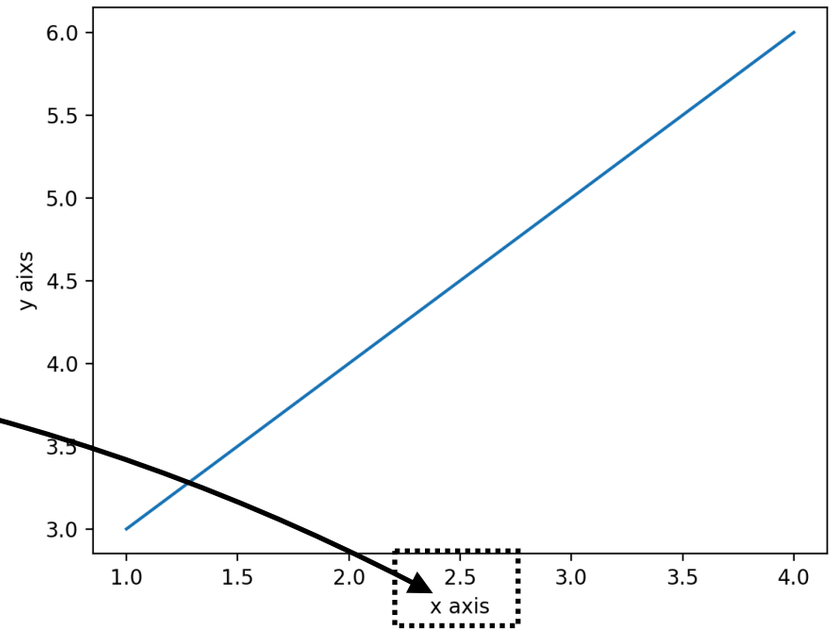


Matplotlib

데이터를 시각화하기!

Plotting

```
import matplotlib.pyplot as plt

xs = [1,2,3,4]
ys = [3,4,5,6]
plt.plot(xs, ys)
plt.xlabel('x axis')
plt.ylabel('y aixs')
plt.show()
```

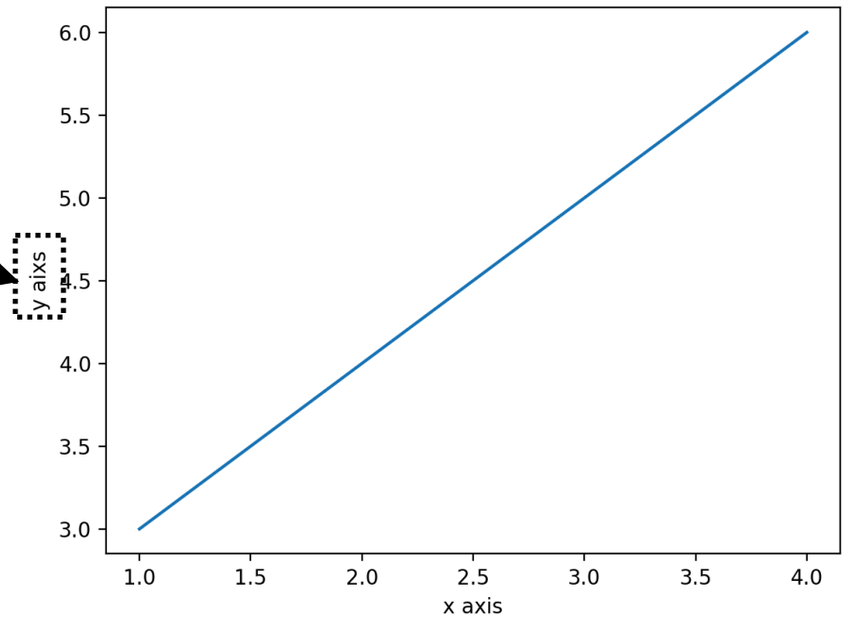


Matplotlib

데이터를 시각화하기!

Plotting

```
import matplotlib.pyplot as plt

xs = [1,2,3,4]
ys = [3,4,5,6]
plt.plot(xs, ys)
plt.xlabel('x axis')
plt.ylabel('y aixs')
plt.show()
```

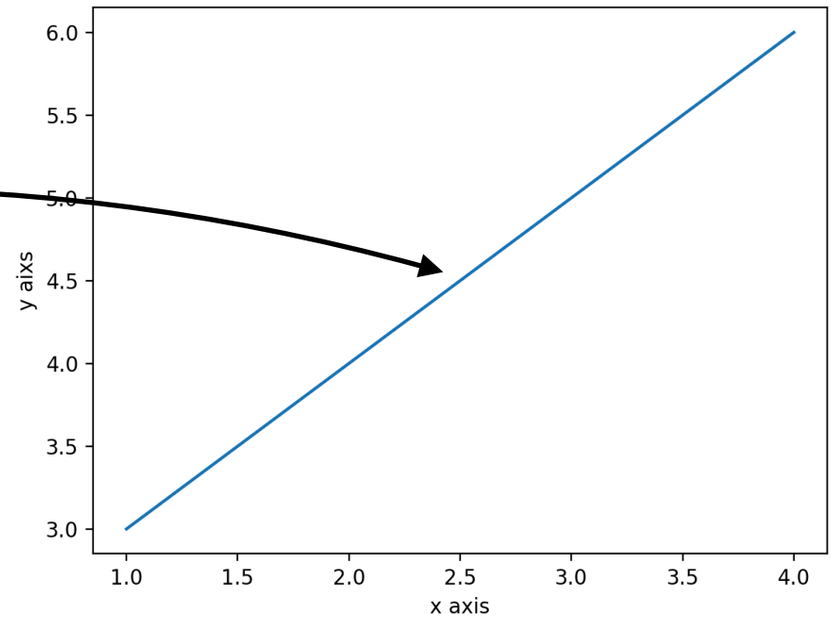


Matplotlib

데이터를 시각화하기!

Plotting 색 바꿔보기

```
import matplotlib.pyplot as plt

xs = [1,2,3,4]
ys = [3,4,5,6]
plt.plot(xs, ys, c='b')
plt.xlabel('x axis')
plt.ylabel('y aixs')
plt.show()
```



'b', 'g', 'r', 'c', 'm', 'y', 'k', 'w' 중 하나를 넣어서 선 색깔을 바꿔보세요

Matplotlib

데이터를 시각화하기!

그래프을 이쁘게 만들 요소가 2가지나 더 있습니다!

Markers

character	description
`'.'`	point marker
`','`	pixel marker
`'o'`	circle marker
`'v'`	triangle_down marker
`'^'`	triangle_up marker
`'<'`	triangle_left marker
`'>'`	triangle_right marker
`'1'`	tri_down marker
`'2'`	tri_up marker
`'3'`	tri_left marker
`'4'`	tri_right marker
`'s'`	square marker
`'p'`	pentagon marker
`'*'`	star marker
`'h'`	hexagon1 marker
`'H'`	hexagon2 marker
`'+'`	plus marker
`'x'`	x marker
`'D'`	diamond marker
`'d'`	thin_diamond marker
`'|'`	vline marker
`'_'`	hline marker

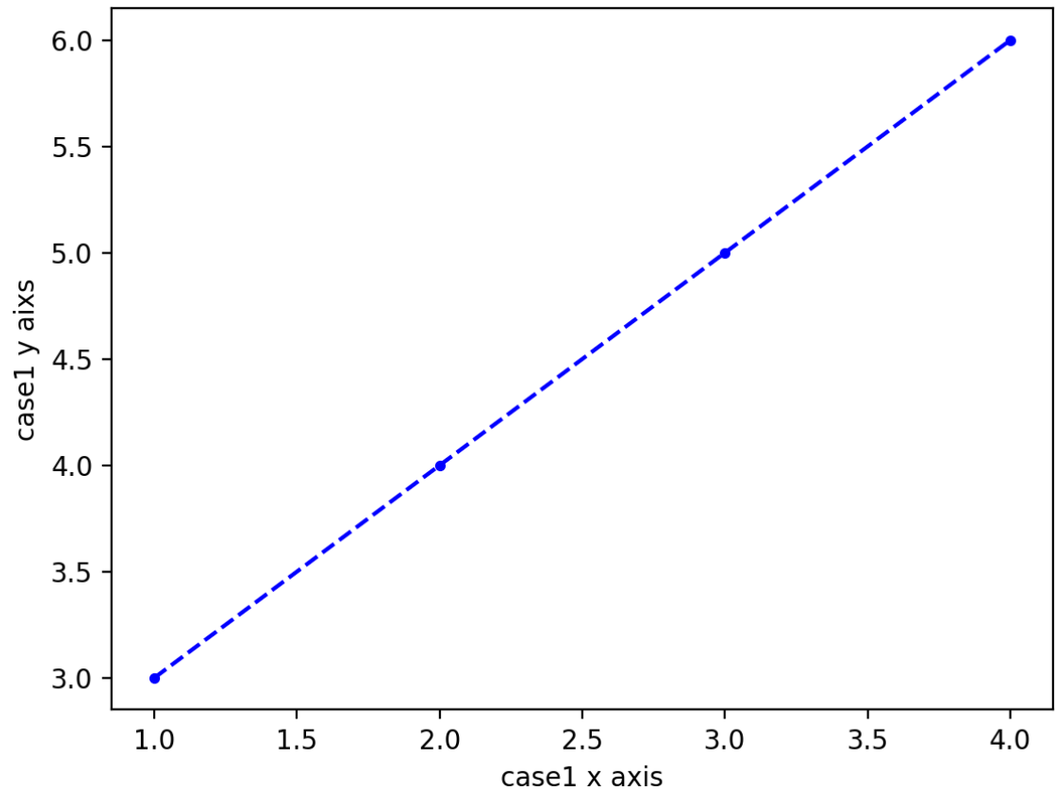
Line Styles

character	description
`'-'`	solid line style
`'--'`	dashed line style
`'-.'`	dash-dot line style
`':'`	dotted line style

Matplotlib

데이터를 시각화하기!

```
#case 1
plt.plot(xs, ys, c='b', marker='.', linestyle='--')
plt.xlabel('case1 x axis')
plt.ylabel('case1 y aixs')
plt.show()
```

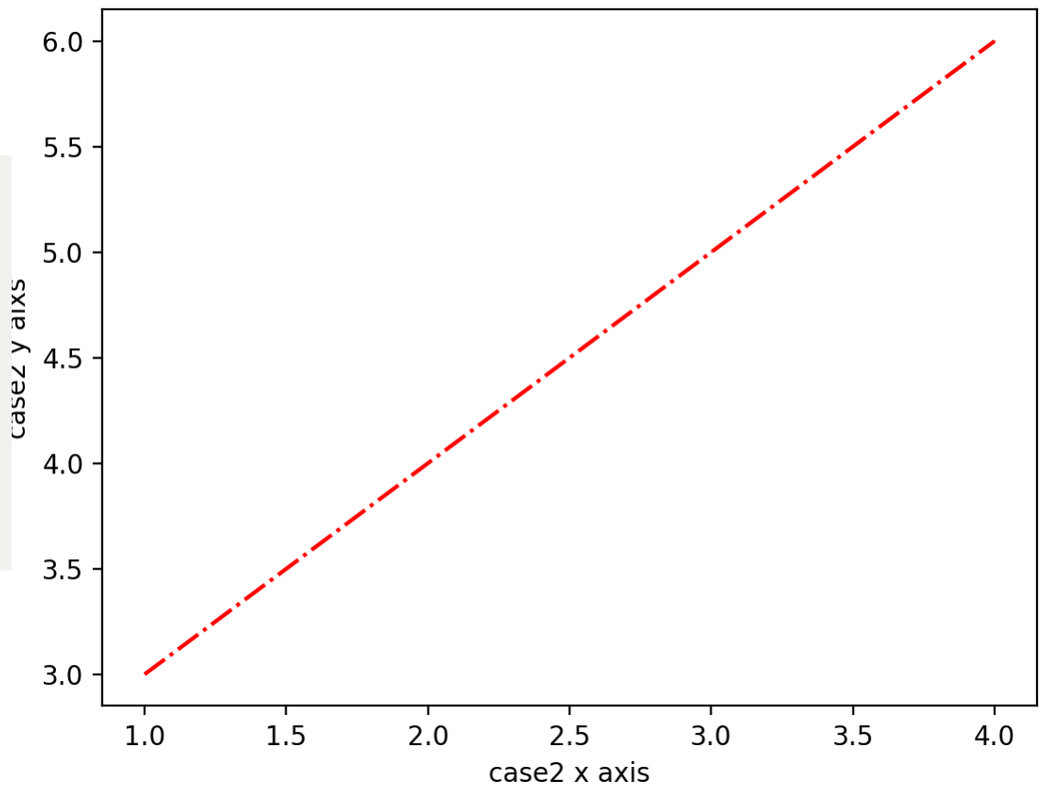


Matplotlib

데이터를 시각화하기!

축약해서 표현하기

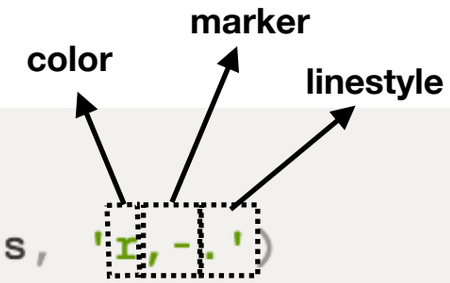
```
#case 2
plt.plot(xs, ys, 'r,-.')
plt.xlabel('case2 x axis')
plt.ylabel('case2 y aixs')
plt.show()
```



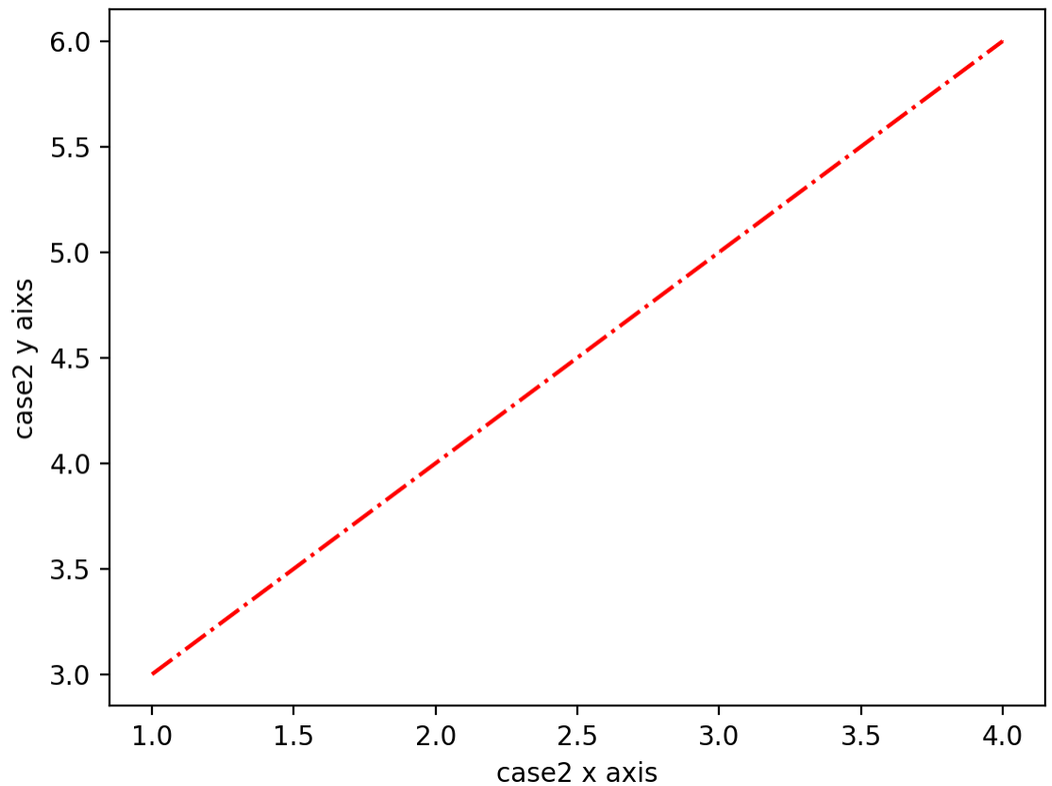
Matplotlib

데이터를 시각화하기!

축약해서 표현하기




```
#case 2
plt.plot(xs, ys, 'r,-.')
plt.xlabel('case2 x axis')
plt.ylabel('case2 y aixs')
plt.show()
```

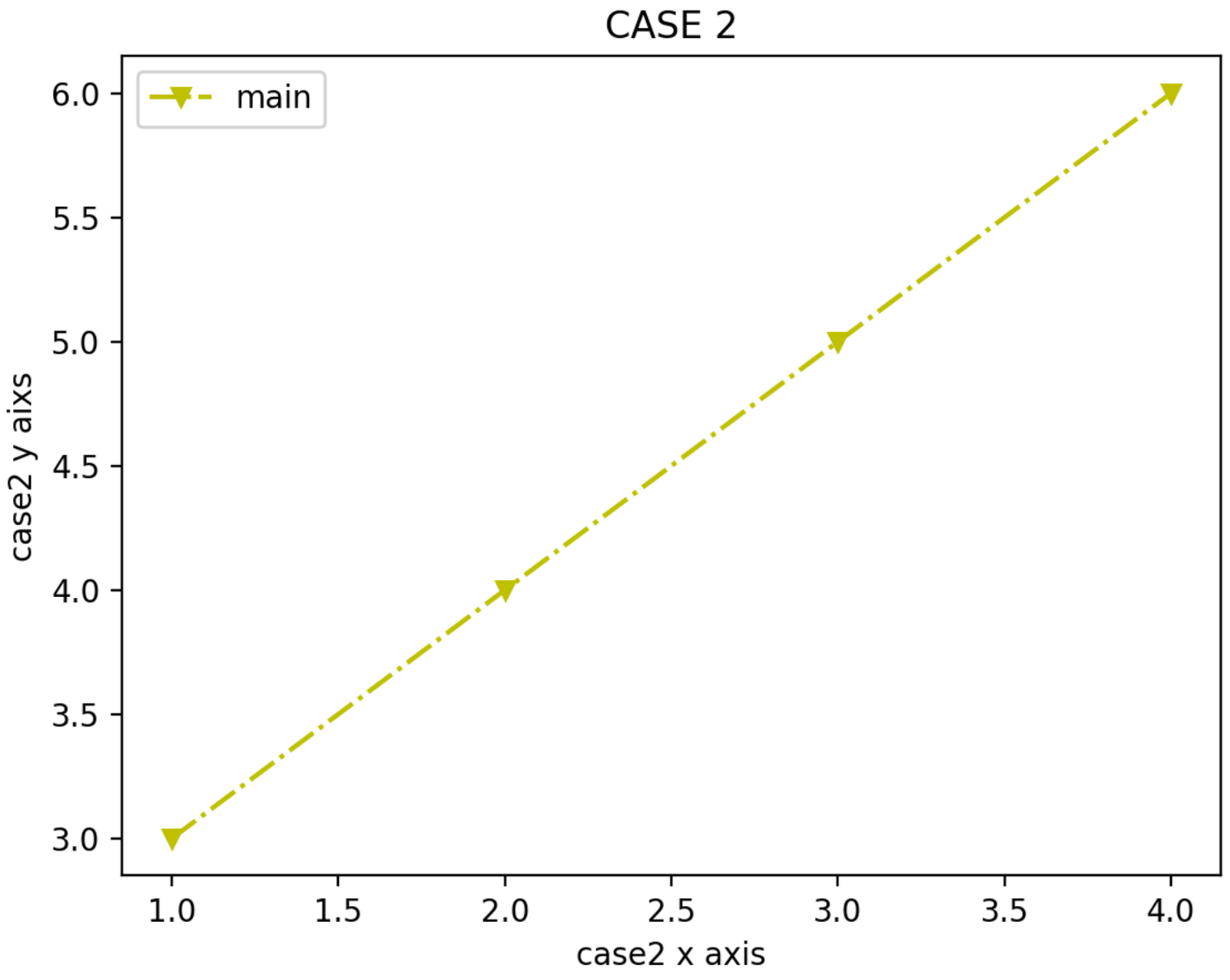


Matplotlib

데이터를 시각화하기!

더 업그레이드 해보기!

```
#case 3
plt.plot(xs, ys, 'yv-.', label='main')
plt.title('CASE 2')
plt.xlabel('case2 x axis')
plt.ylabel('case2 y aixs')
plt.legend()
plt.show()
```

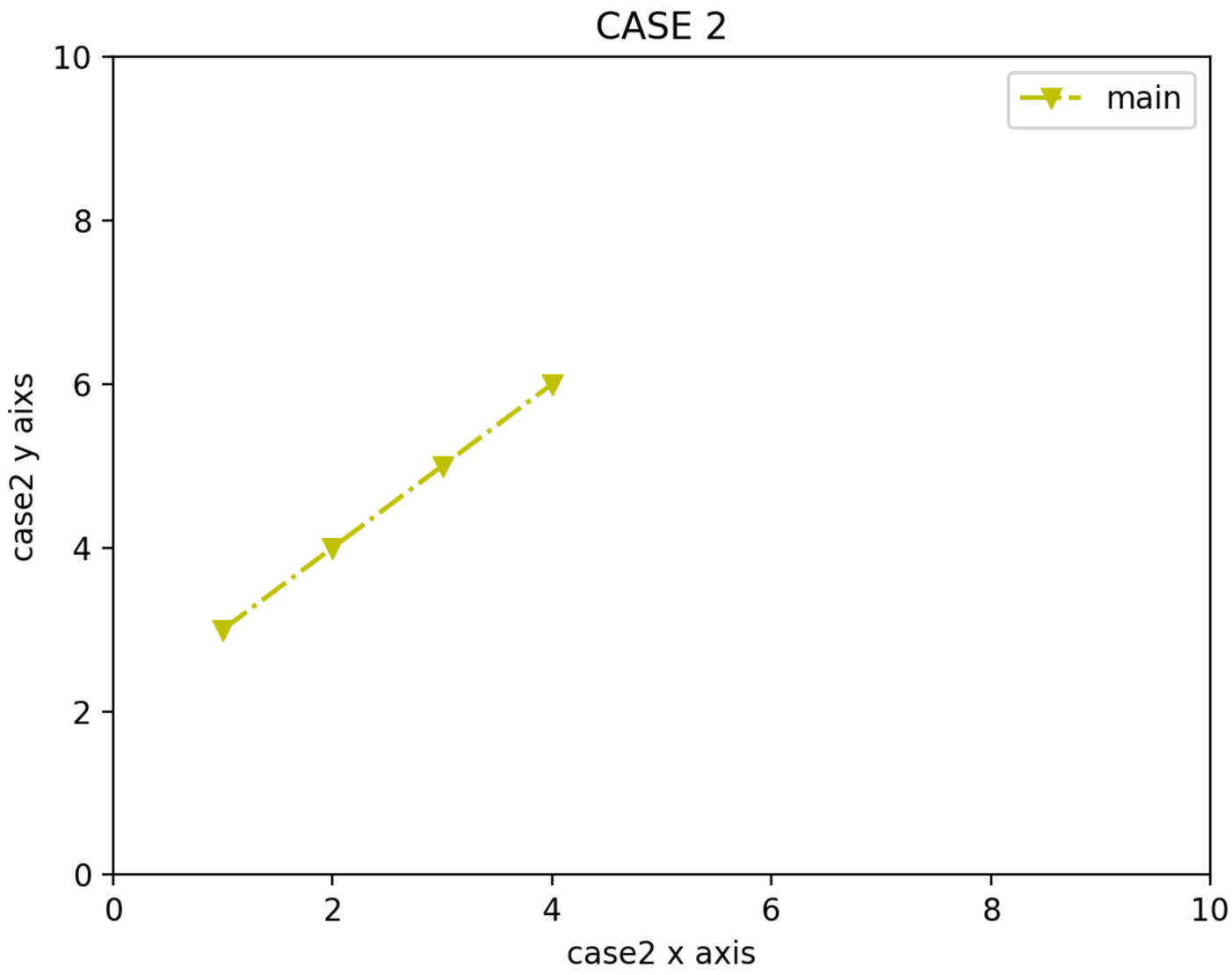


Matplotlib

데이터를 시각화하기!

더 업그레이드 해보기!

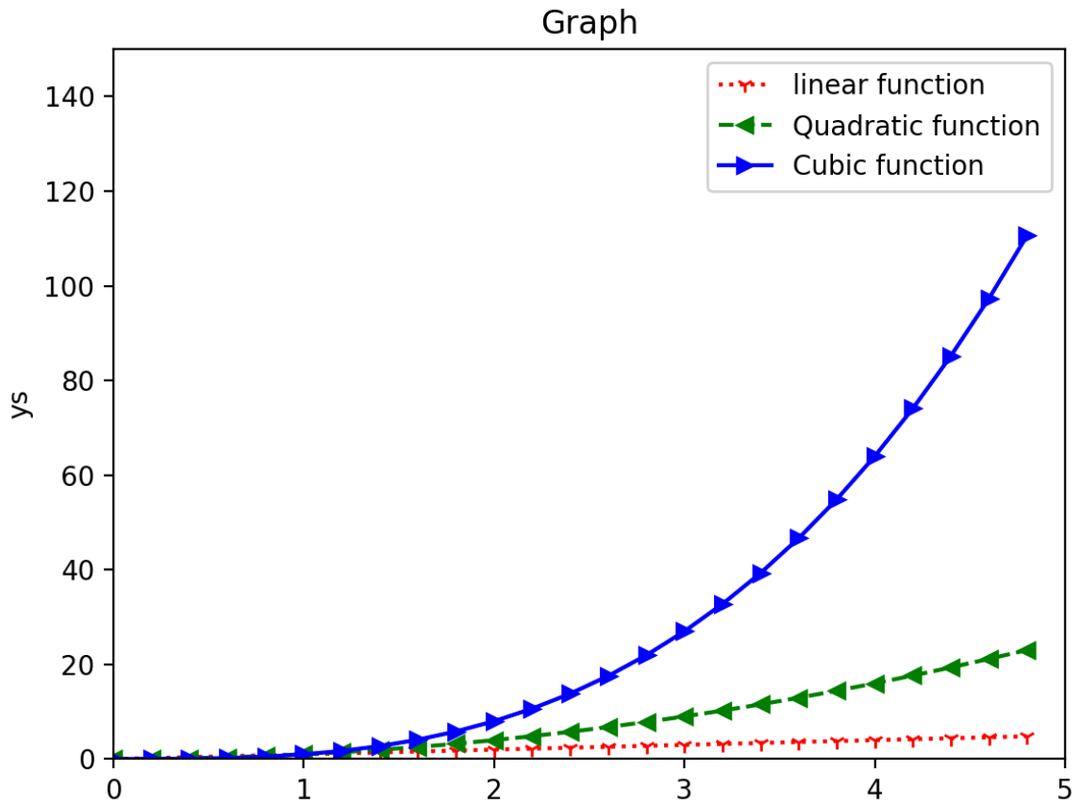
```
#case 4
plt.plot(xs, ys, 'yv-.', label='main')
plt.title('CASE 2')
plt.xlabel('case2 x axis')
plt.ylabel('case2 y aixs')
plt.xlim(0,10)
plt.ylim(0,10)
plt.legend()
plt.show()
```



Matplotlib

여러 데이터를 시각화하기!

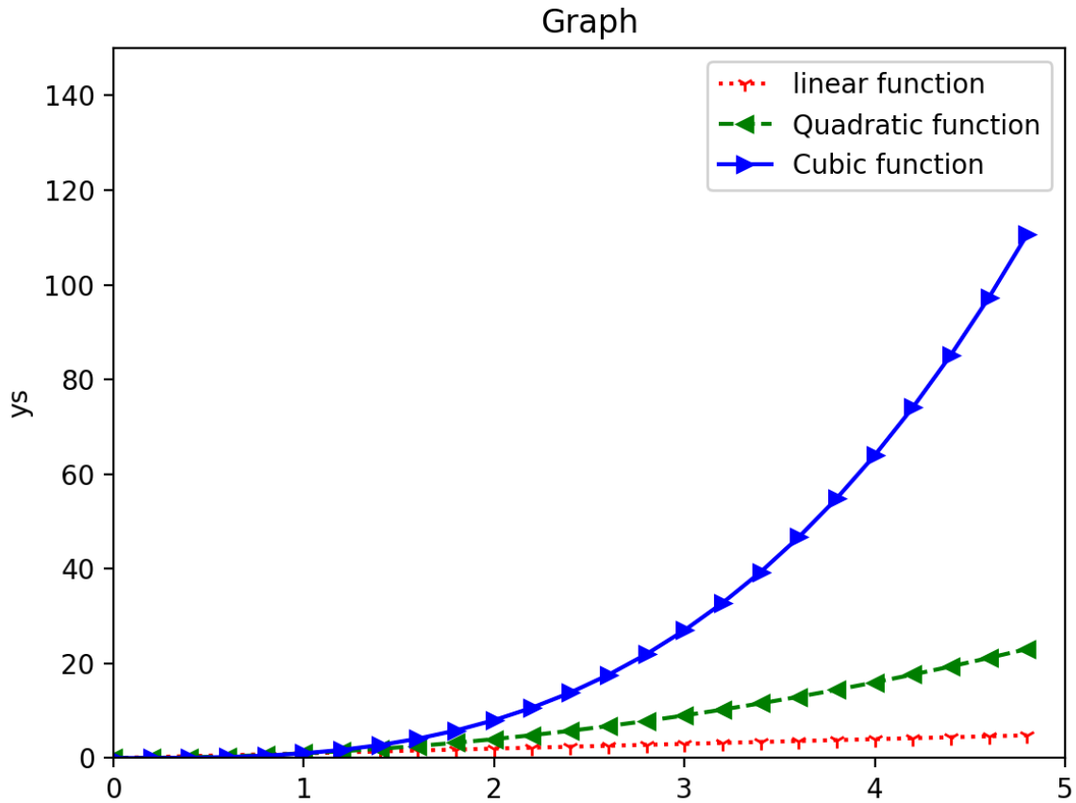
```
import matplotlib.pyplot as plt
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r1:', label='linear function')
plt.plot(t, t**2, 'g<--', label='Quadratic function')
plt.plot(t, t**3, 'b>-', label='Cubic function')

plt.xlabel('xs')
plt.ylabel('ys')
plt.xlim(0,5)
plt.ylim(0,150)
plt.title('Graph')
plt.legend()


plt.show()


# red dashes, blue squares and green triangles
#plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```



Matplotlib

여러 데이터를 시각화하기!

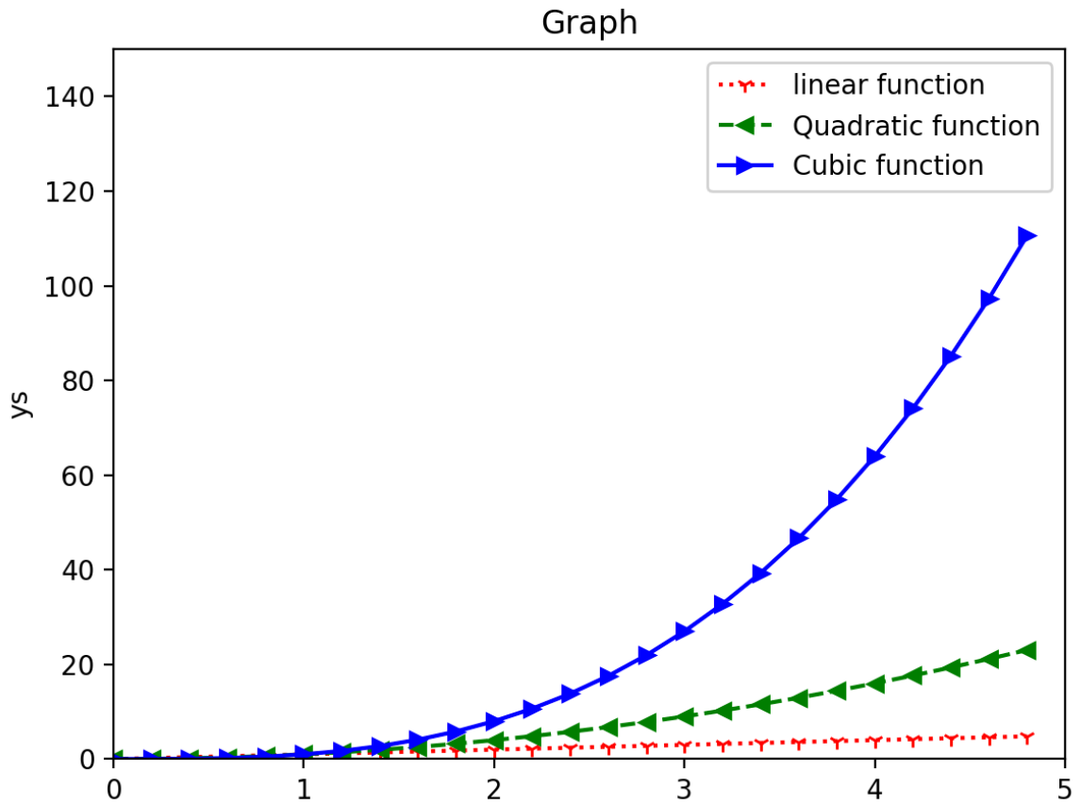
```
import matplotlib.pyplot as plt
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r1:', label='linear function')
plt.plot(t, t**2, 'g<--', label='Quadratic function')
plt.plot(t, t**3, 'b>-', label='Cubic function')

plt.xlabel('xs')
plt.ylabel('ys')
plt.xlim(0,5)
plt.ylim(0,150)
plt.title('Graph')
plt.legend()


plt.show()


# red dashes, blue squares and green triangles
#plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```



Matplotlib

여러 데이터를 시각화하기!

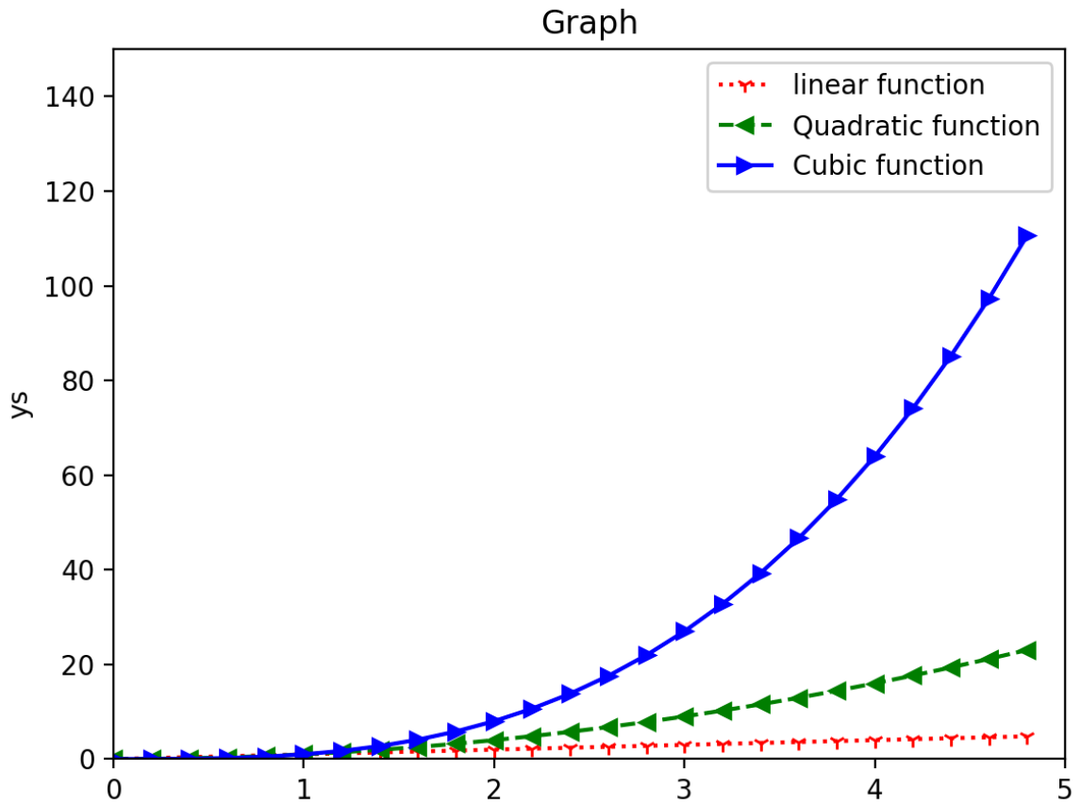
```
import matplotlib.pyplot as plt
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r:', label='linear function')
plt.plot(t, t**2, 'g<--', label='Quadratic function')
plt.plot(t, t**3, 'b>-', label='Cubic function')

plt.xlabel('xs')
plt.ylabel('ys')
plt.xlim(0,5)
plt.ylim(0,150)
plt.title('Graph')
plt.legend()


plt.show()


# red dashes, blue squares and green triangles
#plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```



Matplotlib

여러 데이터를 시각화하기!

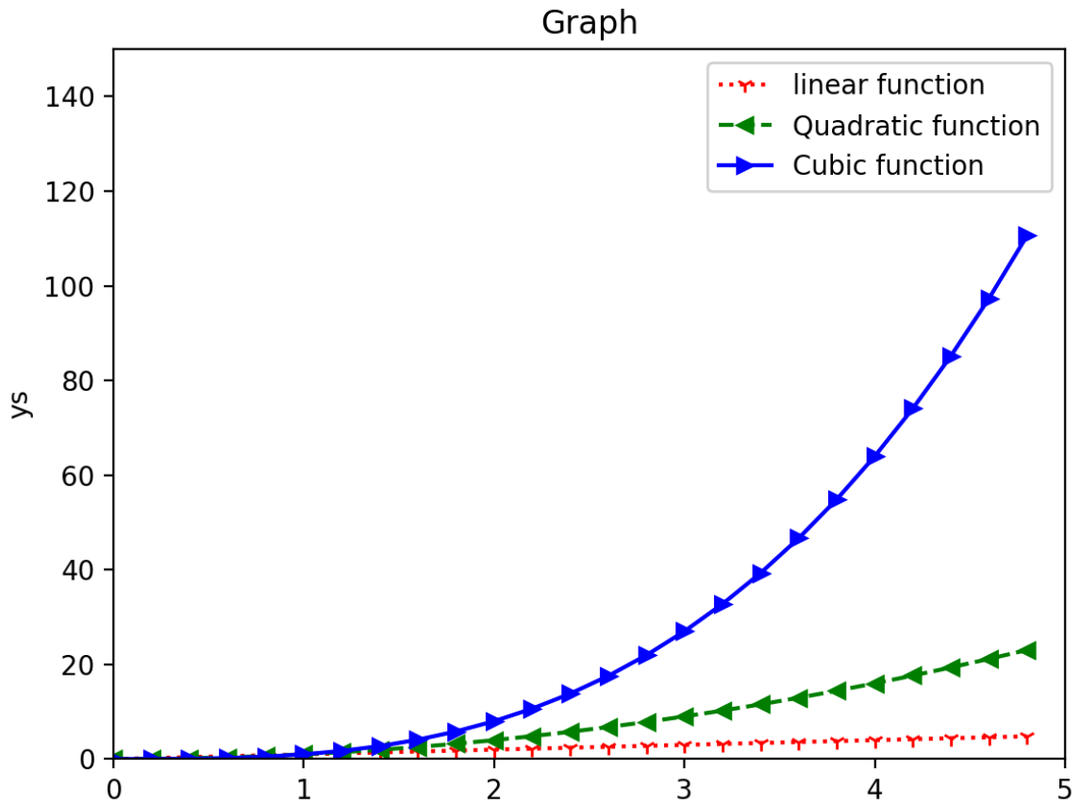
```
import matplotlib.pyplot as plt
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r1:', label='linear function')
plt.plot(t, t**2, 'g<--', label='Quadratic function')
plt.plot(t, t**3, 'b>-', label='Cubic function')

plt.xlabel('xs')
plt.ylabel('ys')
plt.xlim(0,5)
plt.ylim(0,150)
plt.title('Graph')
plt.legend()


plt.show()


# red dashes, blue squares and green triangles
#plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```



Matplotlib

여러 데이터를 시각화하기!

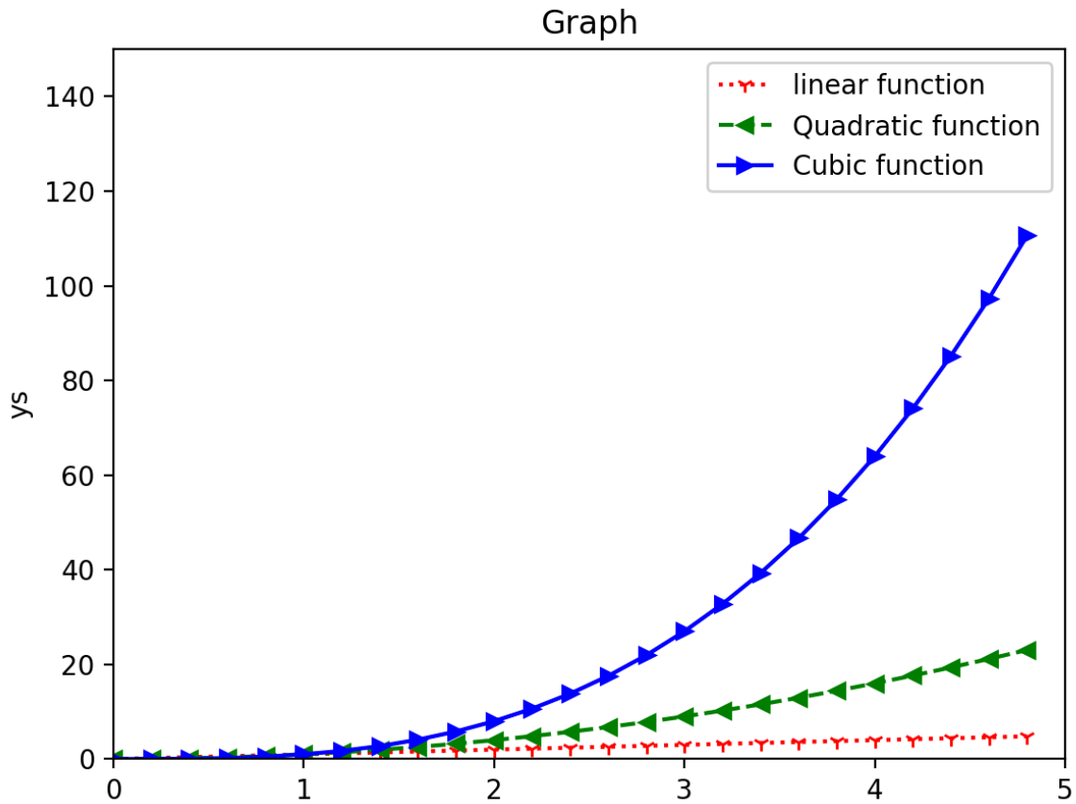
```
import matplotlib.pyplot as plt
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r1:', label='linear function')
plt.plot(t, t**2, 'g<--', label='Quadratic function')
plt.plot(t, t**3, 'b>-', label='Cubic function')

plt.xlabel('xs')
plt.ylabel('ys')
plt.xlim(0,5)
plt.ylim(0,150)
plt.title('Graph')
plt.legend()

plt.show()

# red dashes, blue squares and green triangles
#plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```



Matplotlib

여러 데이터를 시각화하기!

```
import matplotlib.pyplot as plt
import numpy as np

# evenly sampled time at 200ms intervals
t = np.arange(0., 5., 0.2)
plt.plot(t, t, 'r1:', label='linear function')
plt.plot(t, t**2, 'g<--', label='Quadratic function')
plt.plot(t, t**3, 'b>-', label='Cubic function')

plt.xlabel('xs')
plt.ylabel('ys')
plt.xlim(0,5)
plt.ylim(0,150)
plt.title('Graph')
plt.legend()

plt.show()

# red dashes, blue squares and green triangles
#plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
```

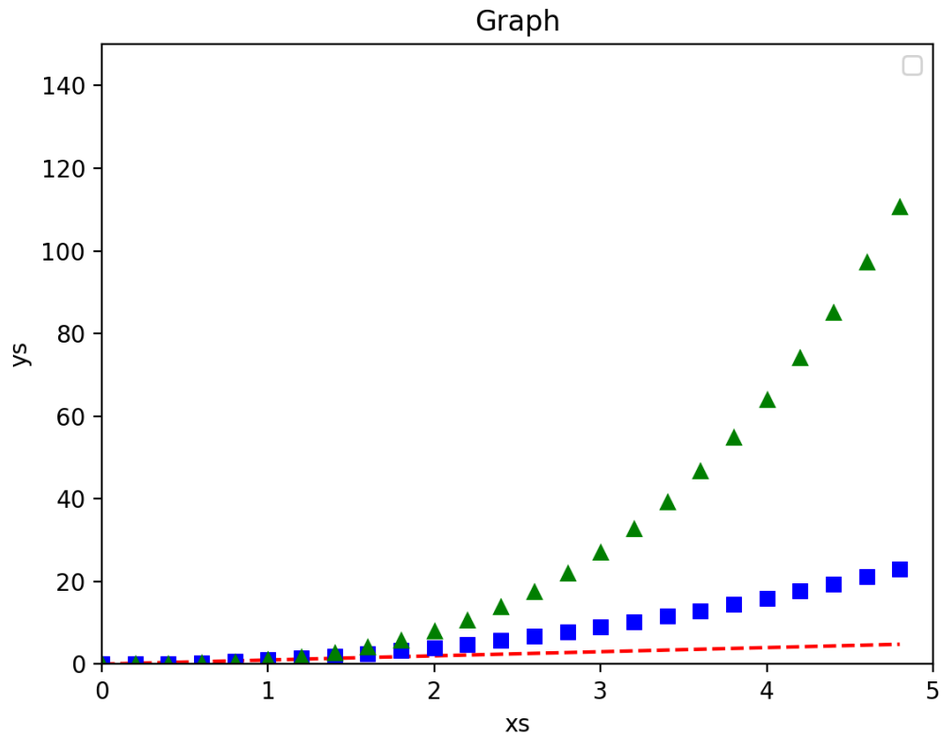


Matplotlib

여러 데이터를 시각화하기!

축약해서 표현하기

```
plt.plot(t, t, 'r--', t, t**2, 'bs', t, t**3, 'g^')
plt.xlabel('xs')
plt.ylabel('ys')
plt.xlim(0,5)
plt.ylim(0,150)
plt.title('Graph')
plt.legend()
plt.show()
```

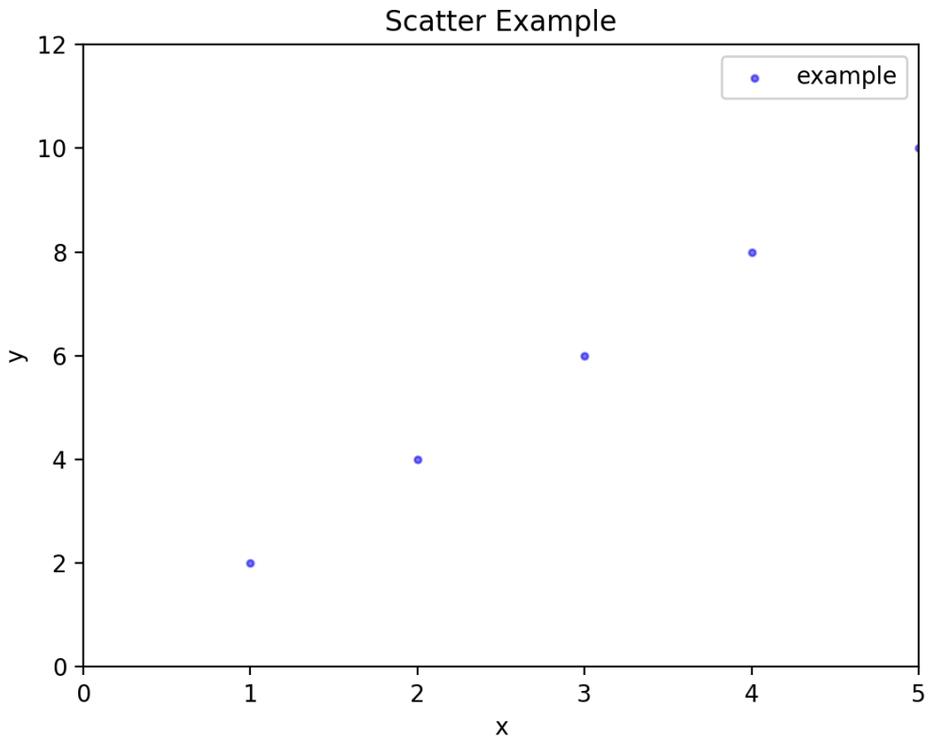


Matplotlib

scatter

sactter

```
# scatter

import matplotlib.pyplot as plt

plt.scatter(x=[1,2,3,4,5],y=[2,4,6,8,10],s=30,c='b', \
alpha=0.5,marker='.',label='example')
plt.title('Scatter Example')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim(0,5)
plt.ylim(0,12)
plt.legend()
plt.show()
```

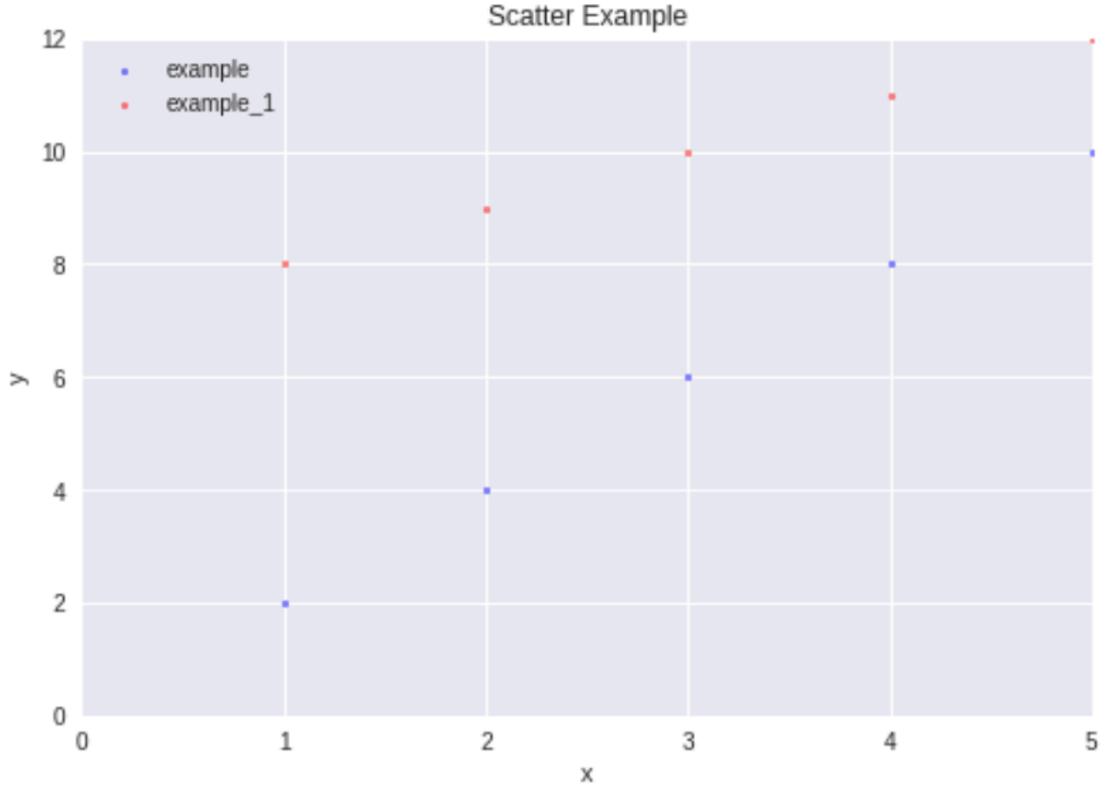


Matplotlib

scatter

sactter

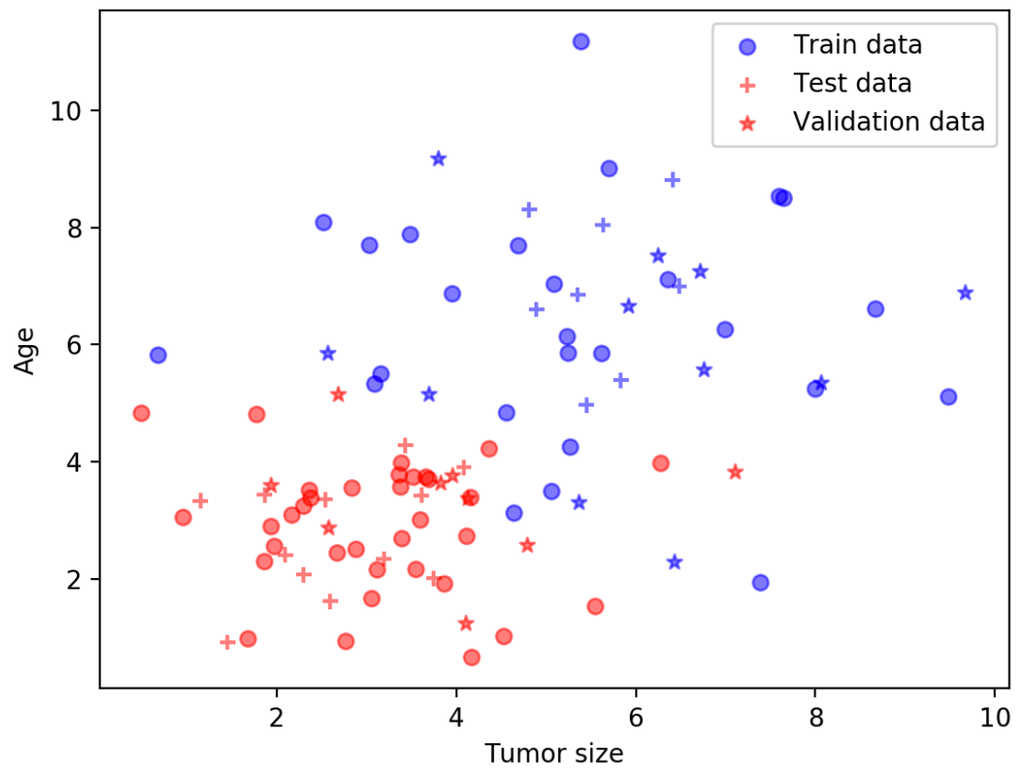
```
plt.scatter(x=[1,2,3,4,5],y=[2,4,6,8,10],s=30,c='b',alpha=0.5,marker='.',label='example')
plt.scatter(x=[1,2,3,4,5],y=[8,9,10,11,12],s=30,c='r',alpha=0.5,marker='.',label='example_1')
plt.title('Scatter Example')
plt.xlabel('x')
plt.ylabel('y')
plt.xlim(0,5)
plt.ylim(0,12)
plt.legend()
plt.show()
```



Matplotlib

데이터를 시각화하기!

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt


datum = np.genfromtxt('cancer_data.csv',dtype=np.float32, delimiter=',')
datum = datum[1:,:]
npr.shuffle(datum)
xs = datum[:,:2]
ys = datum[:,-1]

# train
train_xs = xs[:60]
train_ys = ys[:60]

# test
test_xs = xs[60:80]
test_ys = ys[60:80]

# train
val_xs = xs[80:]
val_ys = ys[80:]

print('Train shape : {} \t Test Shape : {} \t Validation shape : {}'.format(
    train_xs.shape, test_xs.shape, test_xs.shape)
)


train_colors = ['b' if label == 0 else 'r' for label in train_ys]
plt.scatter(train_xs[:,0], train_xs[:,1], label='Train data',\
 alpha=0.5, marker='o', c=train_colors)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
plt.scatter(test_xs[:,0], test_xs[:,1], label='Test data',\
 alpha=0.5, marker='+', c=test_colors)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
plt.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
plt.xlabel('Tumor size')
plt.ylabel('Age')
plt.legend()
plt.show()
```



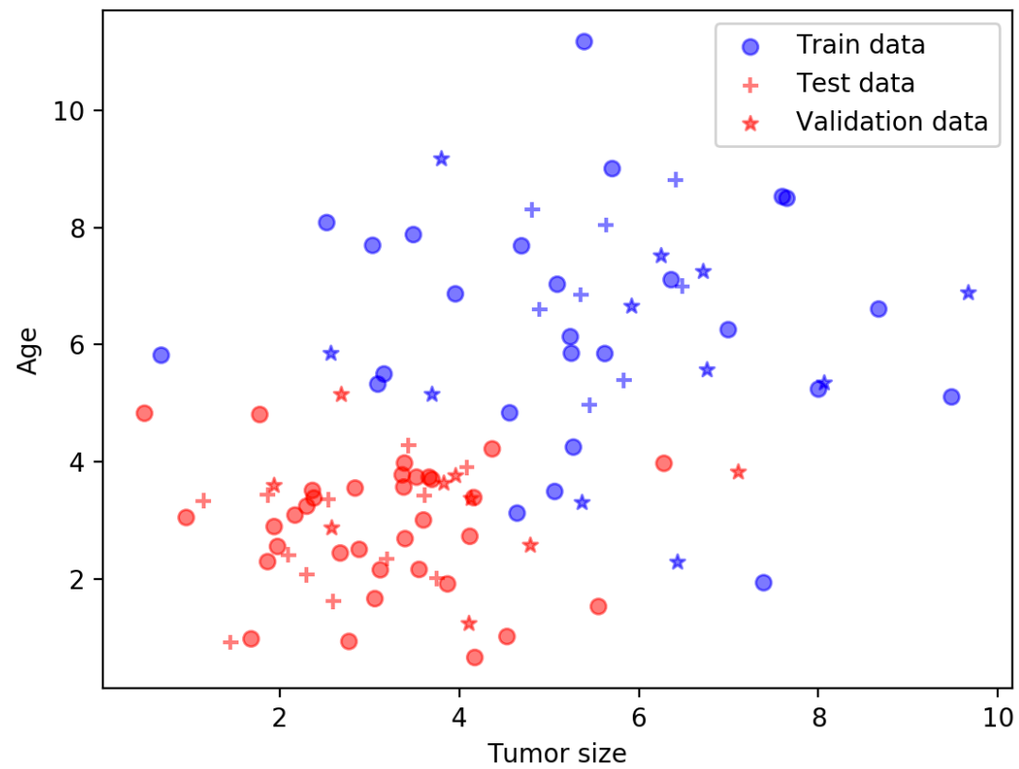
Matplotlib

데이터를 시각화하기!

```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt


datum = np.genfromtxt('cancer_data.csv',dtype=np.float32, delimiter=',')
datum = datum[1:,:]
npr.shuffle(datum)
xs = datum[:,:2]
ys = datum[:,-1]

# train
train_xs = xs[:60]
train_ys = ys[:60]

# test
test_xs = xs[60:80]
test_ys = ys[60:80]

# train
val_xs = xs[80:]
val_ys = ys[80:]

print('Train shape : {} \t Test Shape : {} \t Validation shape : {}'.format(
    train_xs.shape, test_xs.shape, test_xs.shape)
)
```

복습!

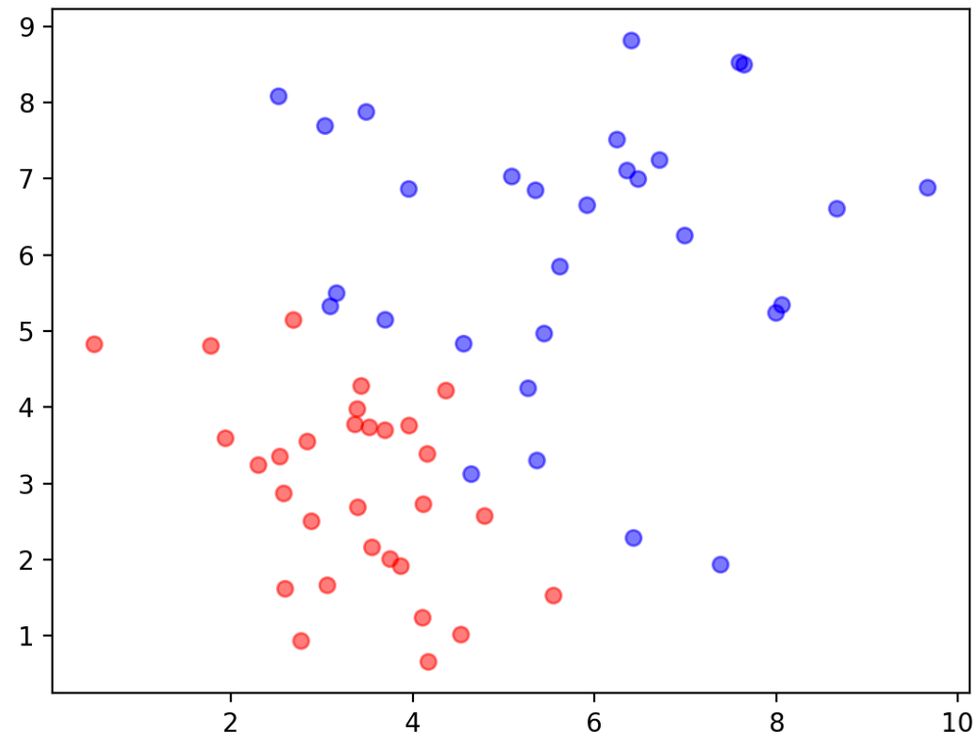
```
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
plt.scatter(train_xs[:,0], train_xs[:,1], label='Train data',\
 alpha=0.5, marker='o', c=train_colors)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
plt.scatter(test_xs[:,0], test_xs[:,1], label='Test data',\
 alpha=0.5, marker='+', c=test_colors)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
plt.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
plt.xlabel('Tumor size')
plt.ylabel('Age')
plt.legend()
plt.show()
```



Matplotlib

데이터를 시각화하기!

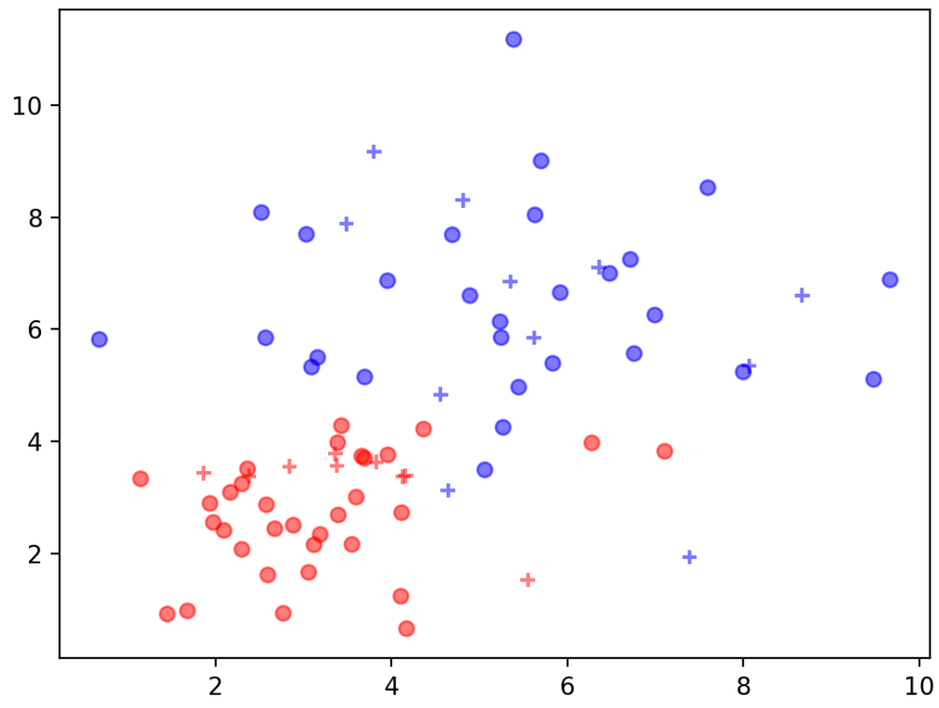
```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt


datum = np.genfromtxt('cancer_data.csv',dtype=np.float32, delimiter=',')
datum = datum[1:,:]
npr.shuffle(datum)
xs = datum[:,:2]
ys = datum[:,-1]

# train
train_xs = xs[:60]
train_ys = ys[:60]

# test
test_xs = xs[60:80]
test_ys = ys[60:80]

# train
val_xs = xs[80:]
val_ys = ys[80:]

print('Train shape : {} \t Test Shape : {} \t Validation shape : {}'.format(
    train_xs.shape, test_xs.shape, test_xs.shape)
)


train_colors = ['b' if label == 0 else 'r' for label in train_ys]
plt.scatter(train_xs[:,0], train_xs[:,1], label='Train data',\
 alpha=0.5, marker='o', c=train_colors)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
plt.scatter(test_xs[:,0], test_xs[:,1], label='Test data',\
 alpha=0.5, marker='+', c=test_colors)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
plt.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
plt.xlabel('Tumor size')
plt.ylabel('Age')
plt.legend()
plt.show()
```



Matplotlib

데이터를 시각화하기!

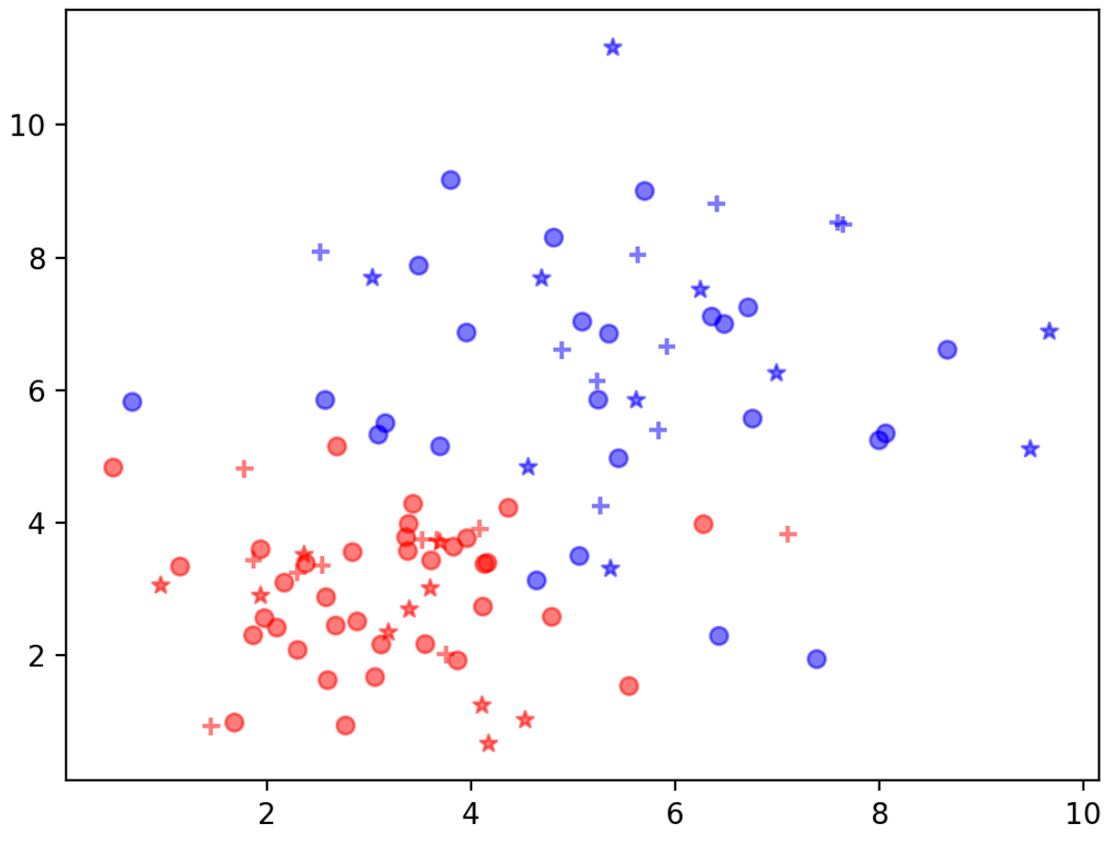
```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt


datum = np.genfromtxt('cancer_data.csv',dtype=np.float32, delimiter=',')
datum = datum[1:,:]
npr.shuffle(datum)
xs = datum[:,:2]
ys = datum[:,-1]

# train
train_xs = xs[:60]
train_ys = ys[:60]

# test
test_xs = xs[60:80]
test_ys = ys[60:80]

# train
val_xs = xs[80:]
val_ys = ys[80:]

print('Train shape : {} \t Test Shape : {} \t Validation shape : {}'.format(
    train_xs.shape, test_xs.shape, test_xs.shape)
)


train_colors = ['b' if label == 0 else 'r' for label in train_ys]
plt.scatter(train_xs[:,0], train_xs[:,1], label='Train data',\
 alpha=0.5, marker='o', c=train_colors)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
plt.scatter(test_xs[:,0], test_xs[:,1], label='Test data',\
 alpha=0.5, marker='+', c=test_colors)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
plt.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
plt.xlabel('Tumor size')
plt.ylabel('Age')
plt.legend()
plt.show()
```



Matplotlib

데이터를 시각화하기! , Visualization

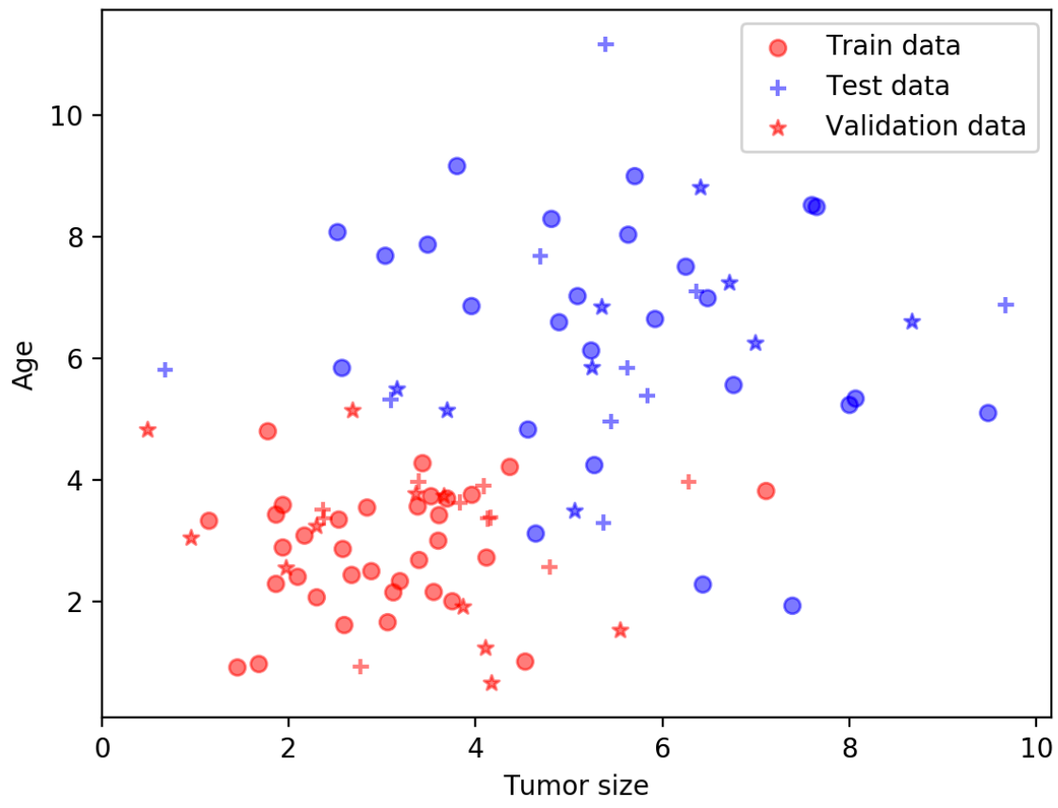
```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt


datum = np.genfromtxt('cancer_data.csv',dtype=np.float32, delimiter=',')
datum = datum[1:,:]
npr.shuffle(datum)
xs = datum[:,:2]
ys = datum[:,-1]

# train
train_xs = xs[:60]
train_ys = ys[:60]

# test
test_xs = xs[60:80]
test_ys = ys[60:80]

# train
val_xs = xs[80:]
val_ys = ys[80:]

print('Train shape : {} \t Test Shape : {} \t Validation shape : {}'.format(
    train_xs.shape, test_xs.shape, test_xs.shape)
)


train_colors = ['b' if label == 0 else 'r' for label in train_ys]
plt.scatter(train_xs[:,0], train_xs[:,1], label='Train data',\
 alpha=0.5, marker='o', c=train_colors)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
plt.scatter(test_xs[:,0], test_xs[:,1], label='Test data',\
 alpha=0.5, marker='+', c=test_colors)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
plt.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
plt.xlabel('Tumor size')
plt.ylabel('Age')
plt.legend()
plt.show()
```



Matplotlib

데이터를 시각화하기!

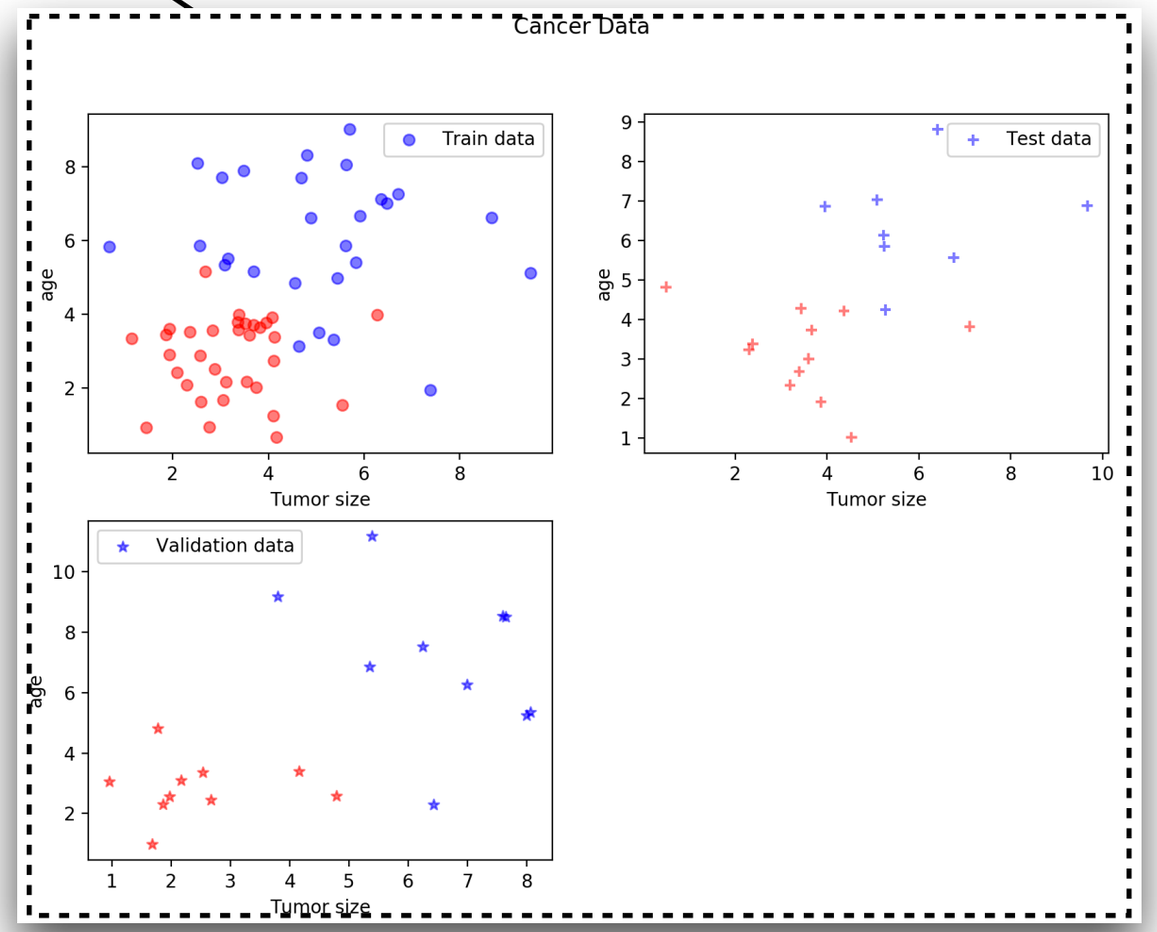
```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt


datum = np.genfromtxt('cancer_data.csv',dtype=np.float32, delimiter=',')
datum = datum[1:,:]
npr.shuffle(datum)
xs = datum[:,:2]
ys = datum[:,-1]

# train
train_xs = xs[:60]
train_ys = ys[:60]

# test
test_xs = xs[60:80]
test_ys = ys[60:80]

# train
val_xs = xs[80:]
val_ys = ys[80:]

print('Train shape : {} \t Test Shape : {} \t Validation shape : {}'.format(
    train_xs.shape, test_xs.shape, test_xs.shape)
)


train_colors = ['b' if label == 0 else 'r' for label in train_ys]
plt.scatter(train_xs[:,0], train_xs[:,1], label='Train data',\
 alpha=0.5, marker='o', c=train_colors)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
plt.scatter(test_xs[:,0], test_xs[:,1], label='Test data',\
 alpha=0.5, marker='+', c=test_colors)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
plt.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
plt.xlabel('Tumor size')
plt.ylabel('Age')
plt.legend()
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

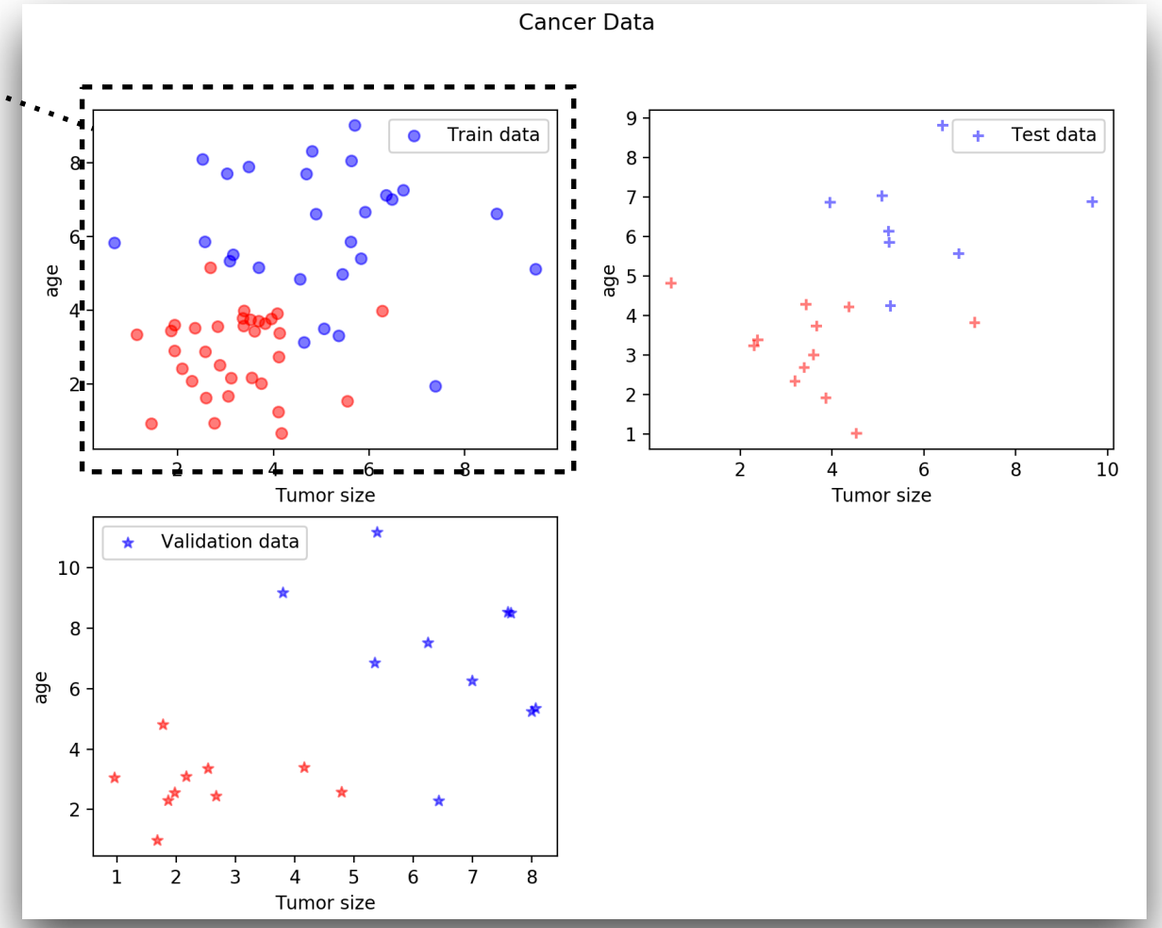
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
 marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

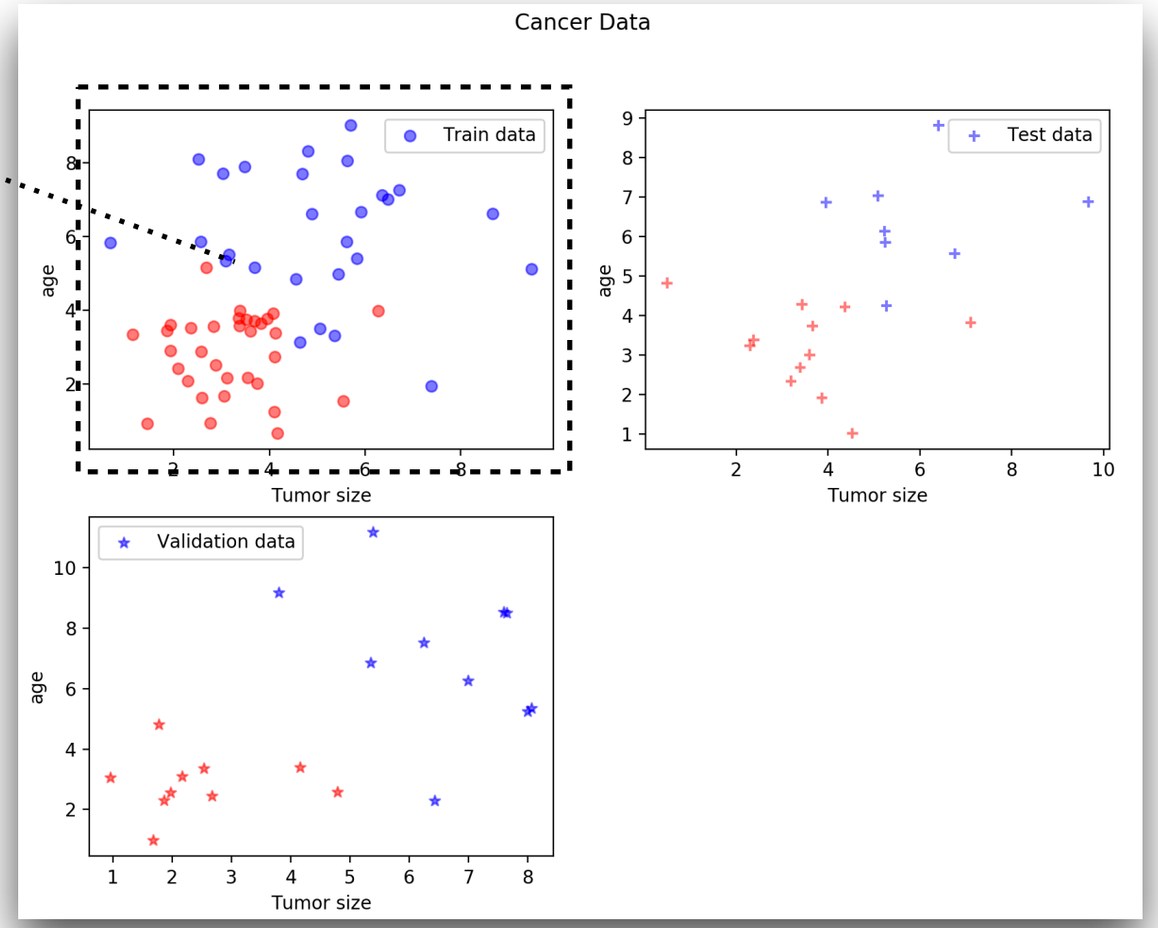
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
 marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

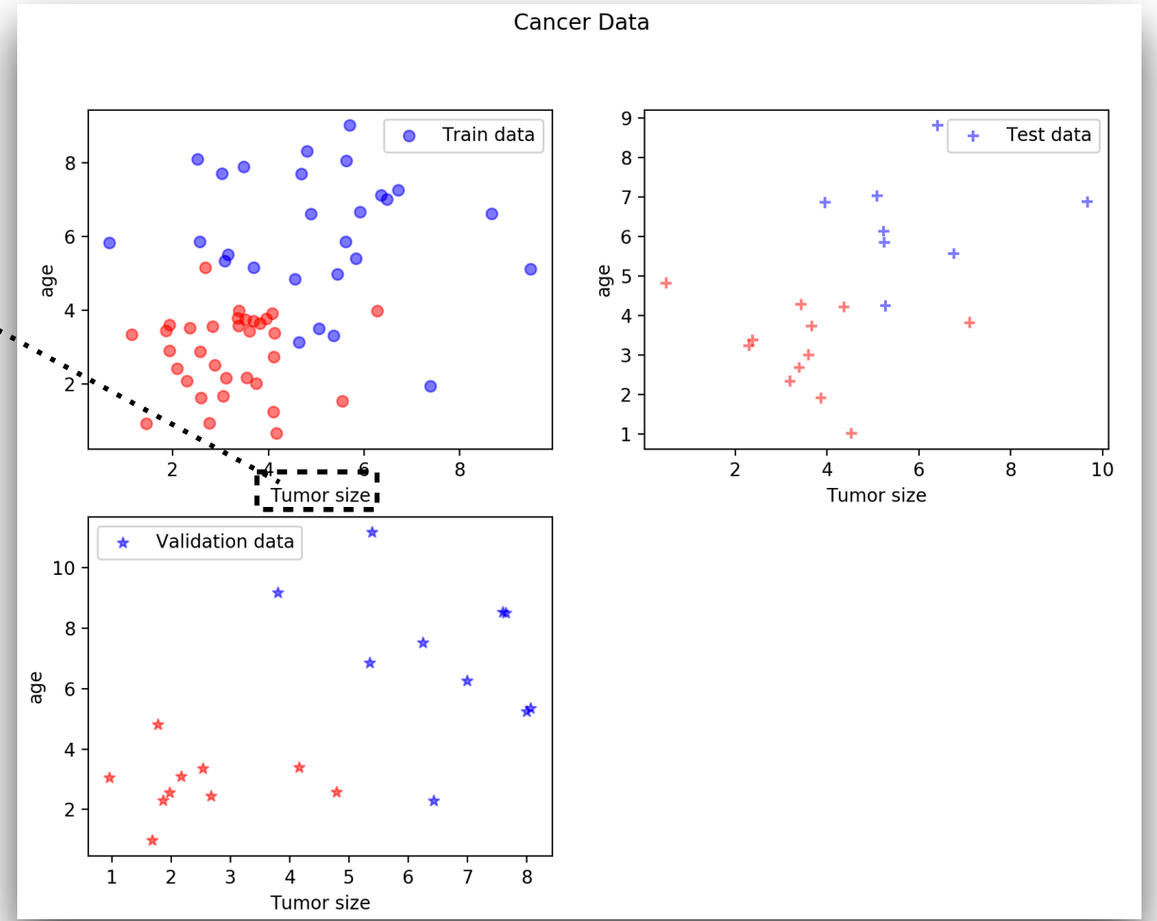
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
| marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

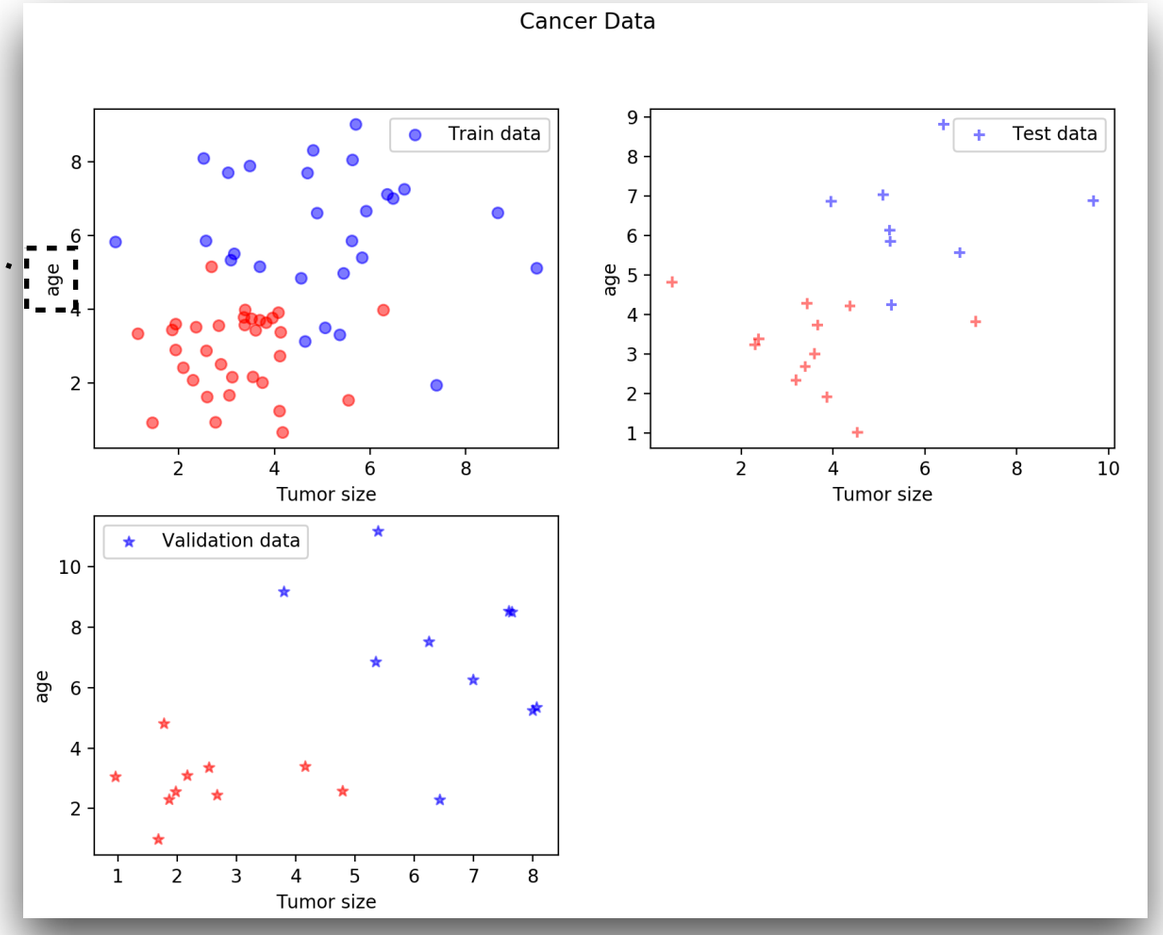
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
 marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

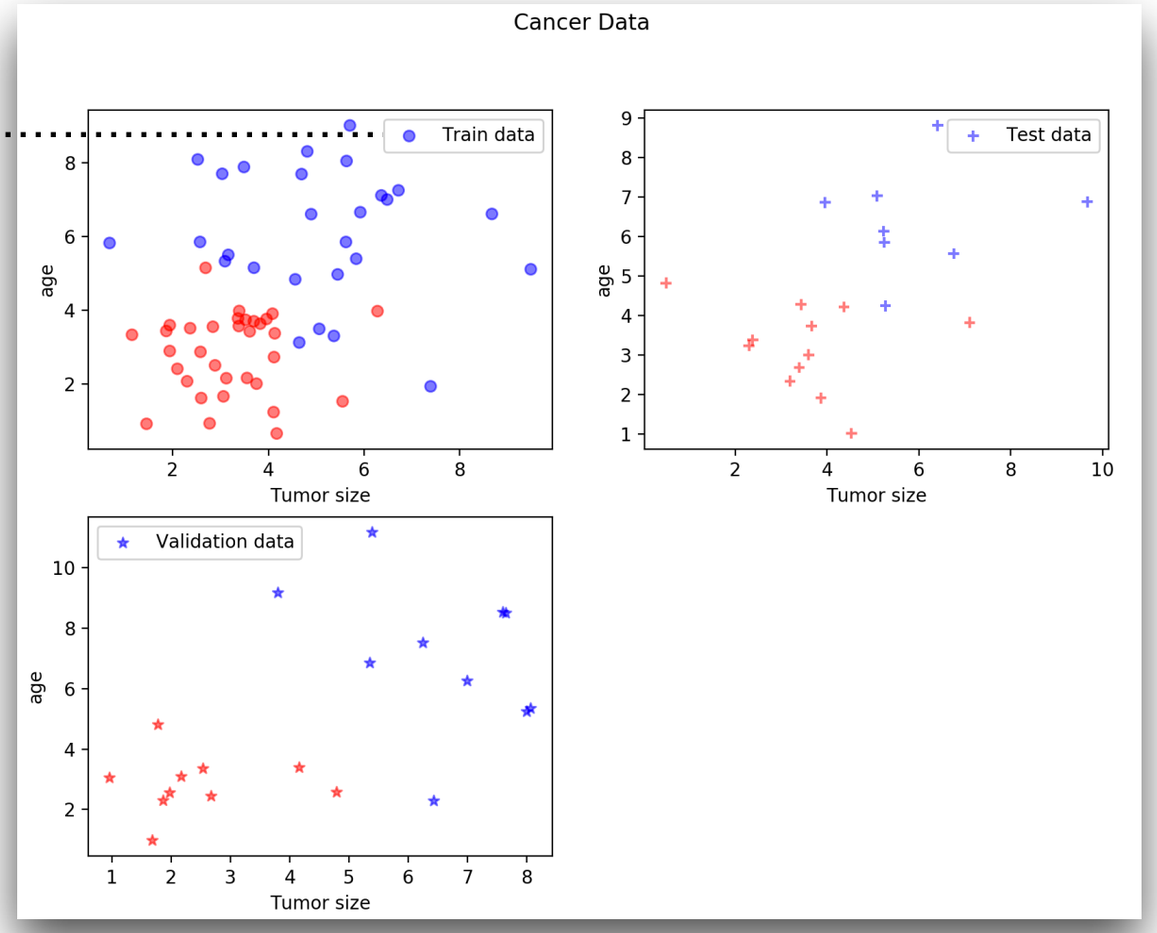
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
| marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

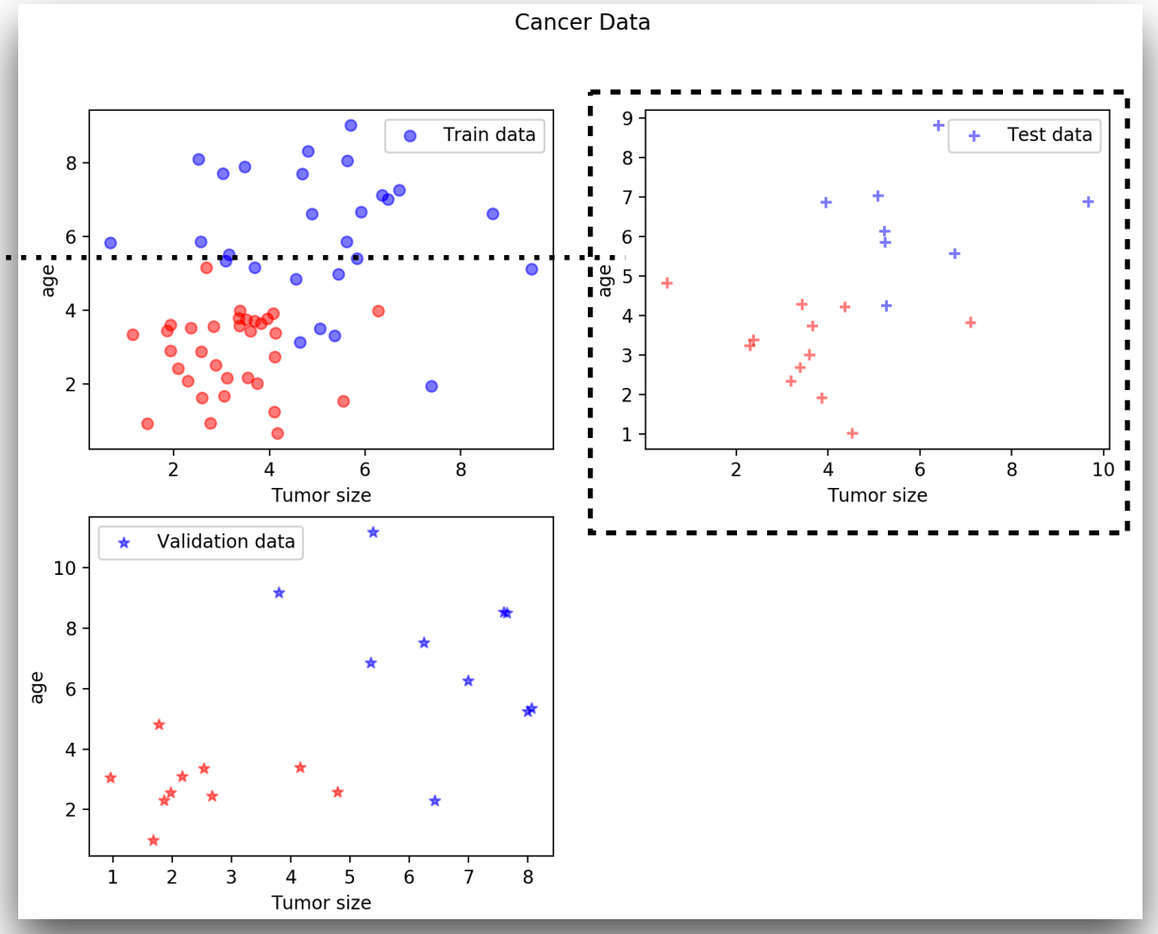
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
| marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

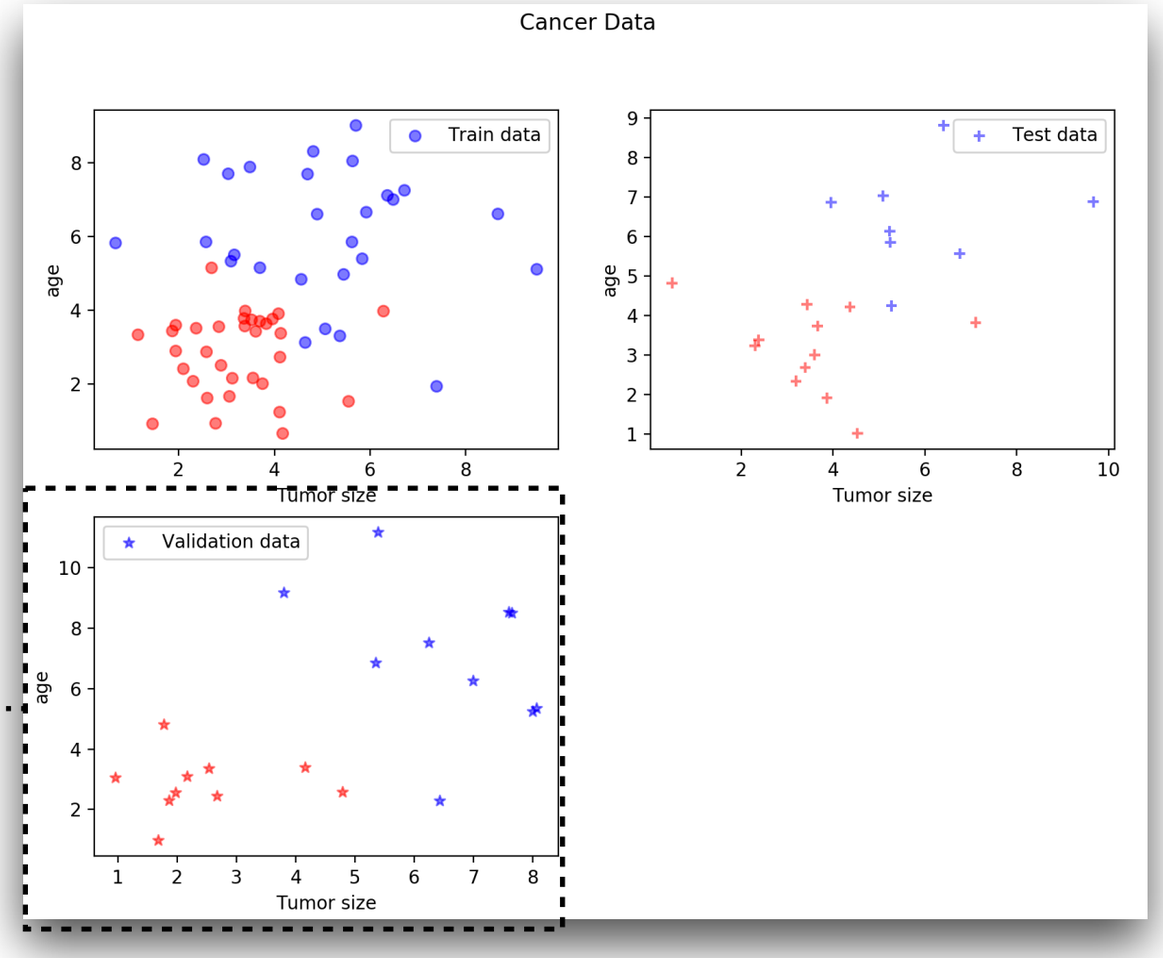
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
 marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

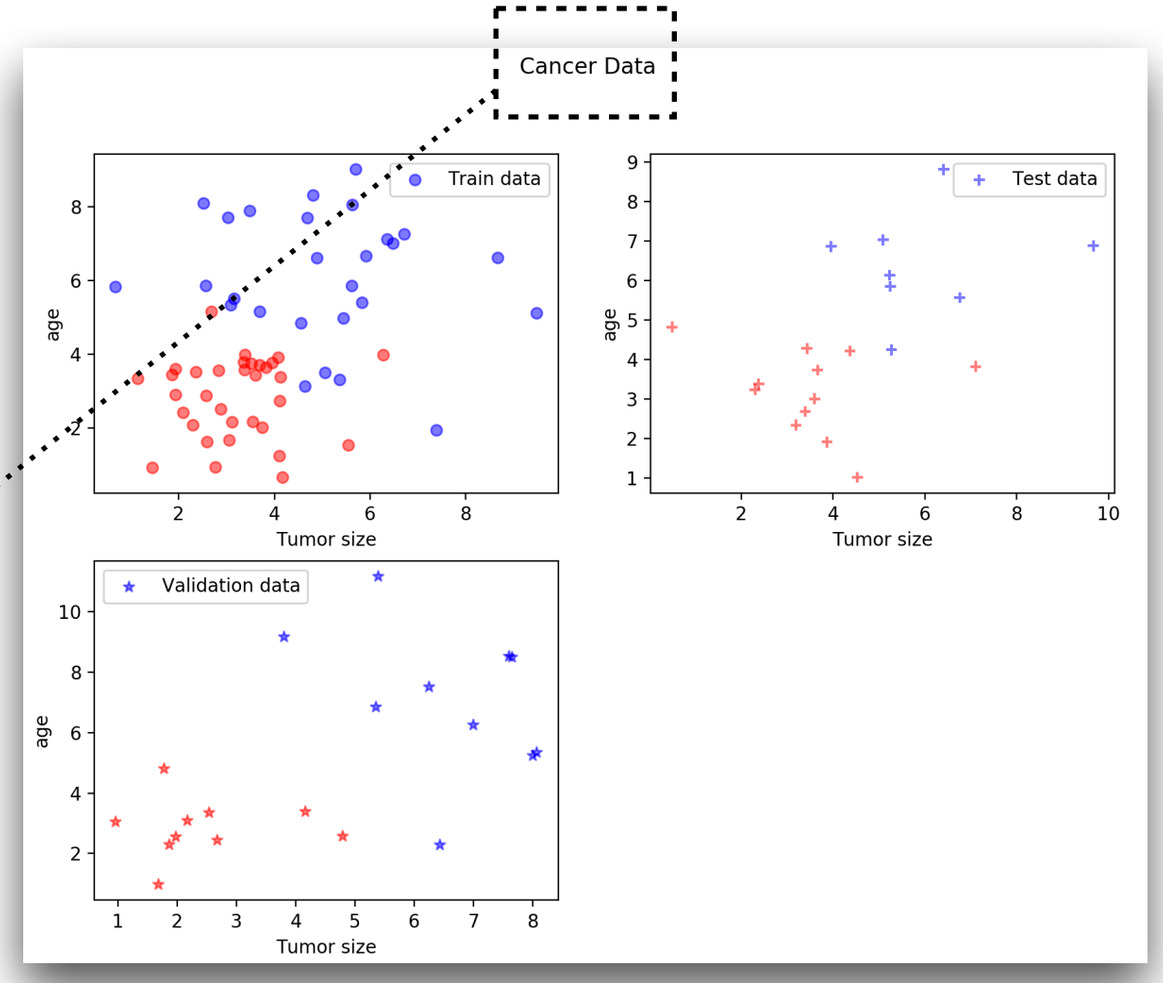
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
| marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! sub plotting

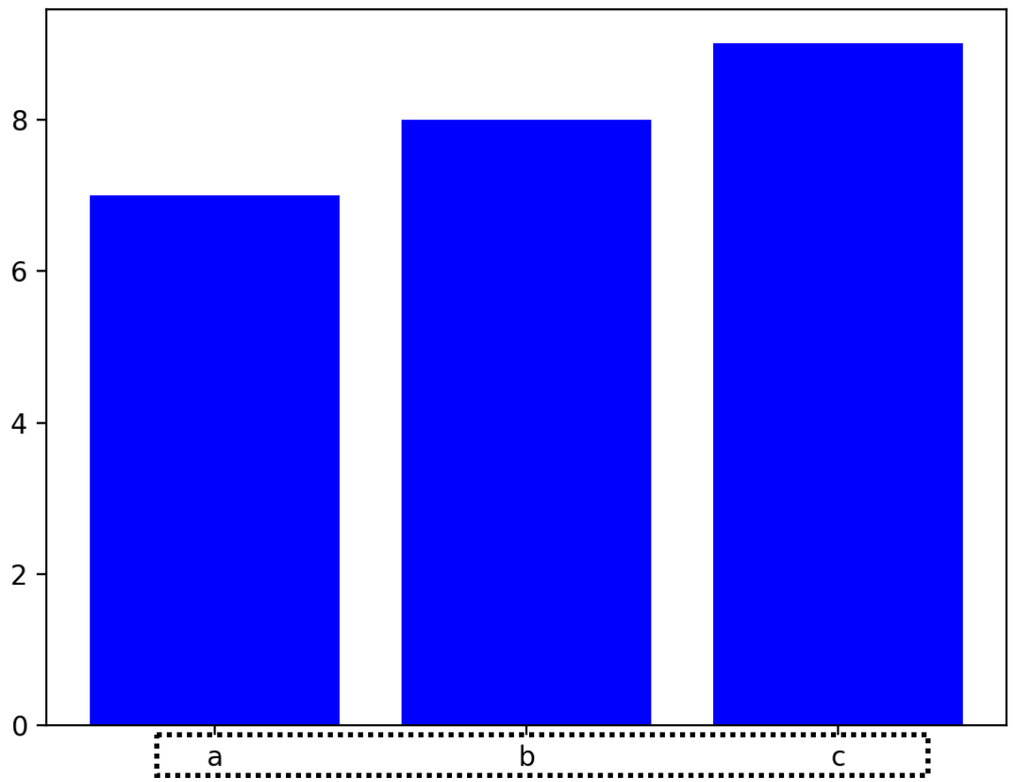
```
fig = plt.figure(figsize=(10,10))
ax = fig.add_subplot(221)
train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()


ax = fig.add_subplot(222)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5, \
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()


ax = fig.add_subplot(223)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data', alpha=0.5,\
 marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)

ax.legend()

fig.suptitle('Cancer Data')
plt.show()
```



Matplotlib

데이터를 시각화하기! bar

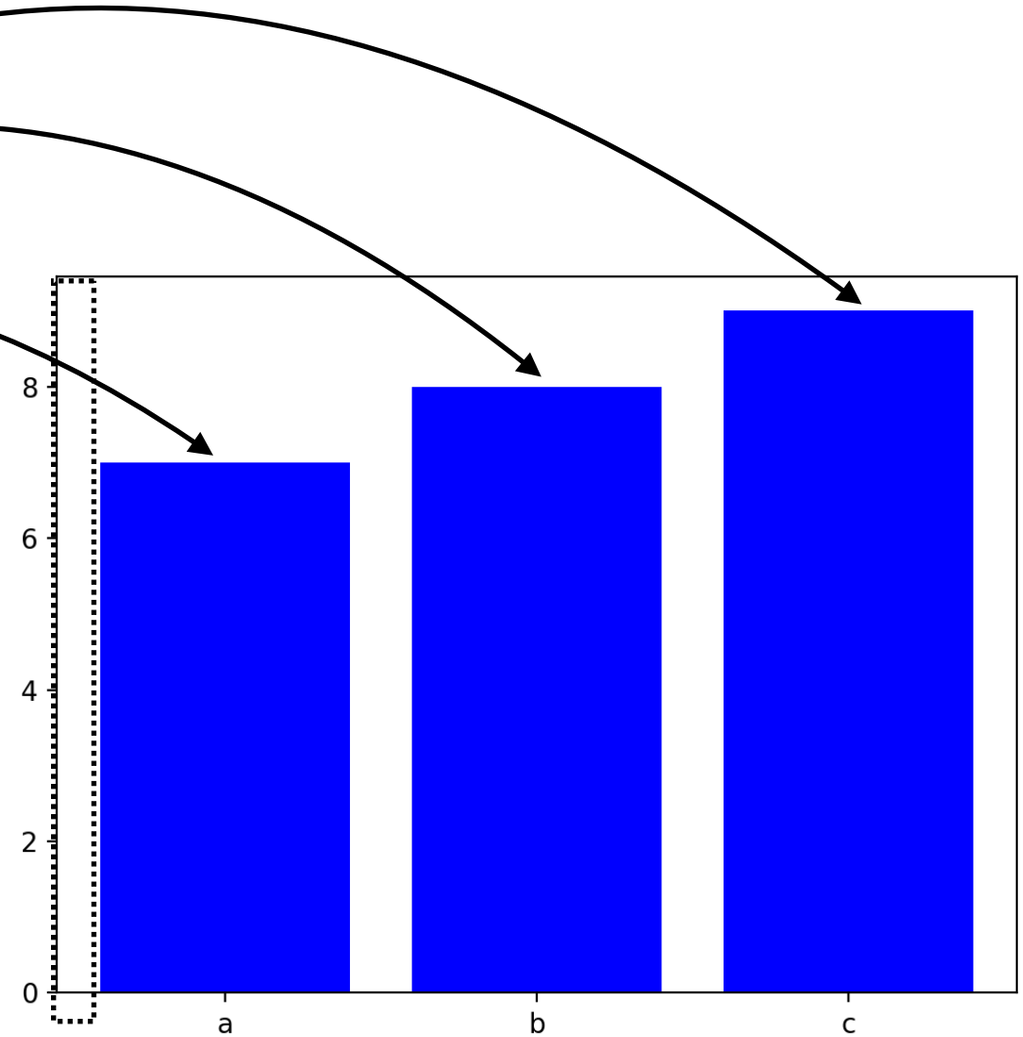
```
import matplotlib.pyplot as plt

plt.bar(['a','b','c'], [7,8,9], align='center', color='b')
plt.show()
```



Matplotlib

데이터를 시각화하기! bar

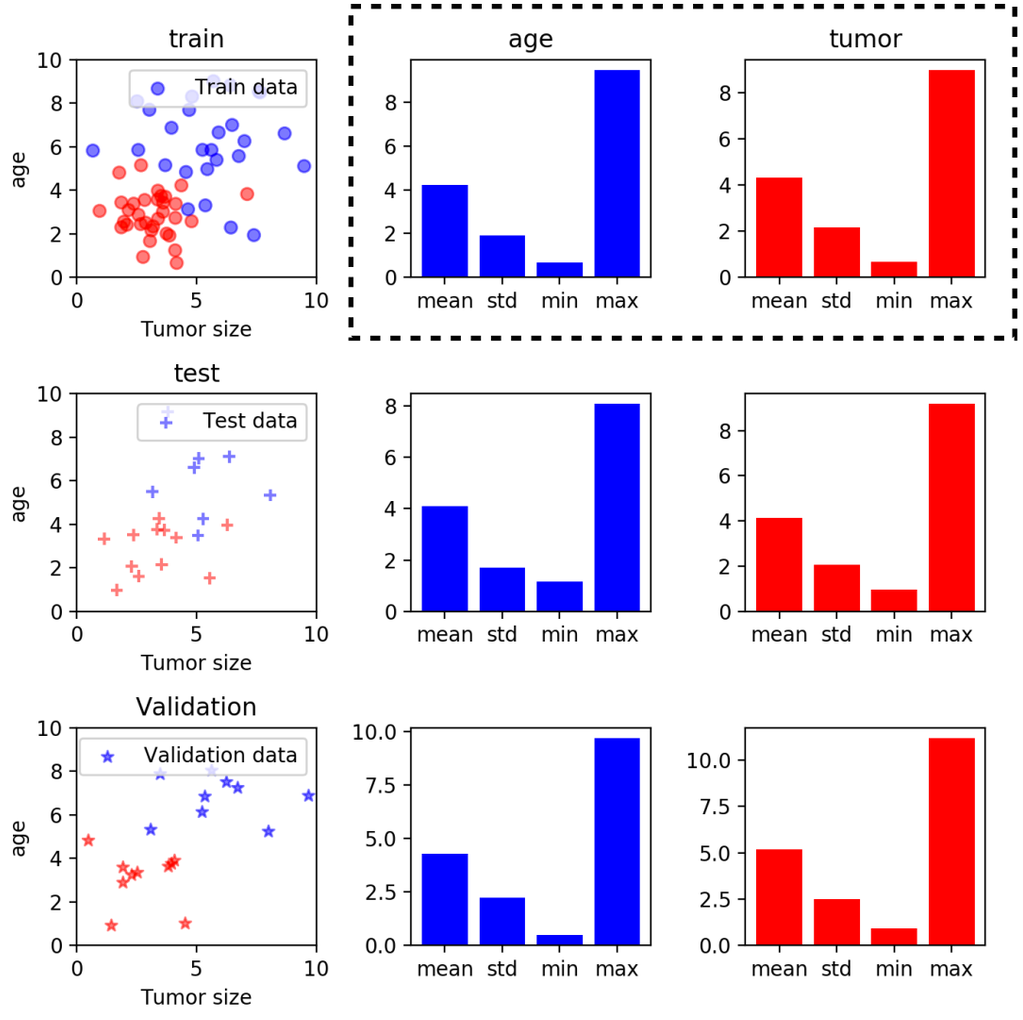
```
import matplotlib.pyplot as plt

plt.bar(['a','b','c'], [7,8,9], align='center', color='b')
plt.show()
```



Matplotlib

데이터를 시각화하기!

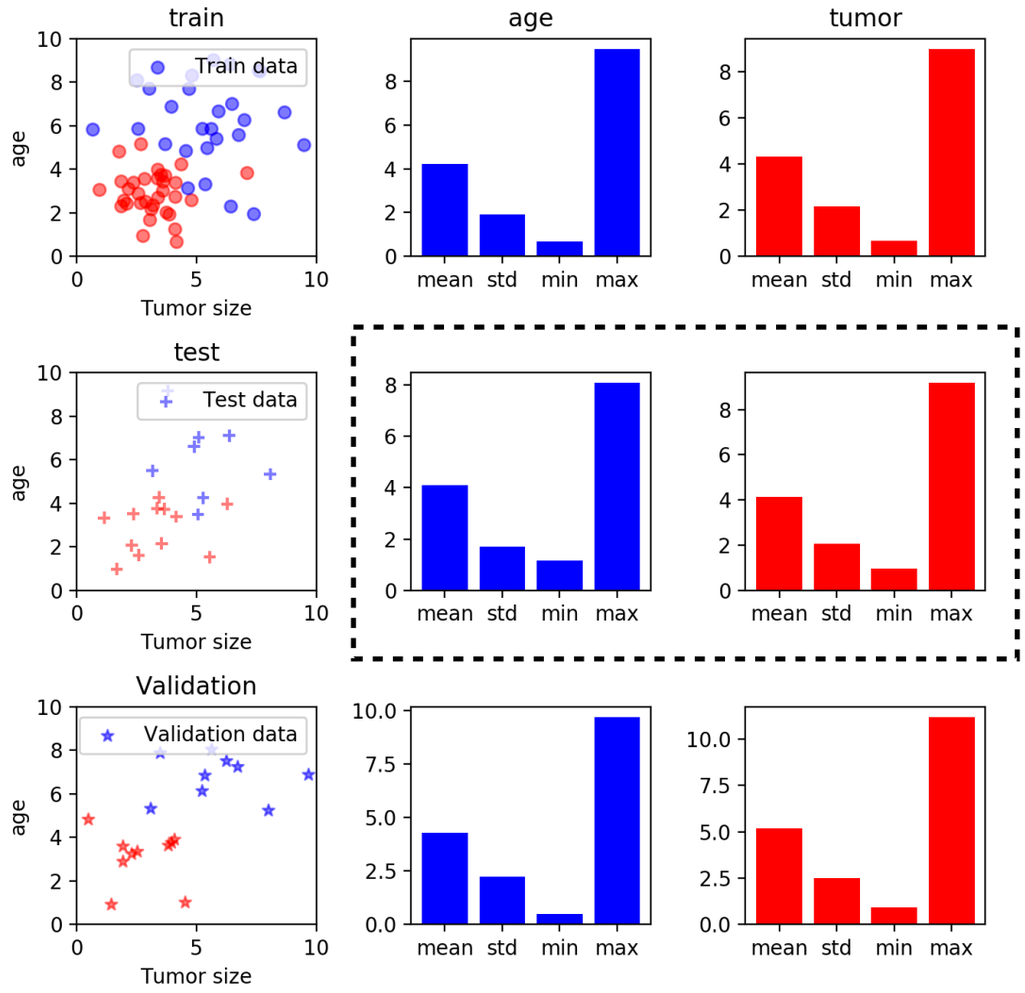
```python
def var_summary(x):
    ret_dict = {}
    if isinstance(x, np.ndarray):
        ret_dict['mean'] = x.mean()
        ret_dict['std'] = x.std()
        ret_dict['min'] = x.min()
        ret_dict['max'] = x.max()
    else:
        raise ValueError
    return ret_dict

# Train Summaries
train_xs_0_summaries = var_summary(train_xs[:,0])
train_xs_1_summaries = var_summary(train_xs[:,1])

# Test Summaries
test_xs_0_summaries = var_summary(test_xs[:,0])
test_xs_1_summaries = var_summary(test_xs[:,1])

# Validation Summaries
val_xs_0_summaries = var_summary(val_xs[:,0])
val_xs_1_summaries = var_summary(val_xs[:,1])
```



Matplotlib

데이터를 시각화하기!

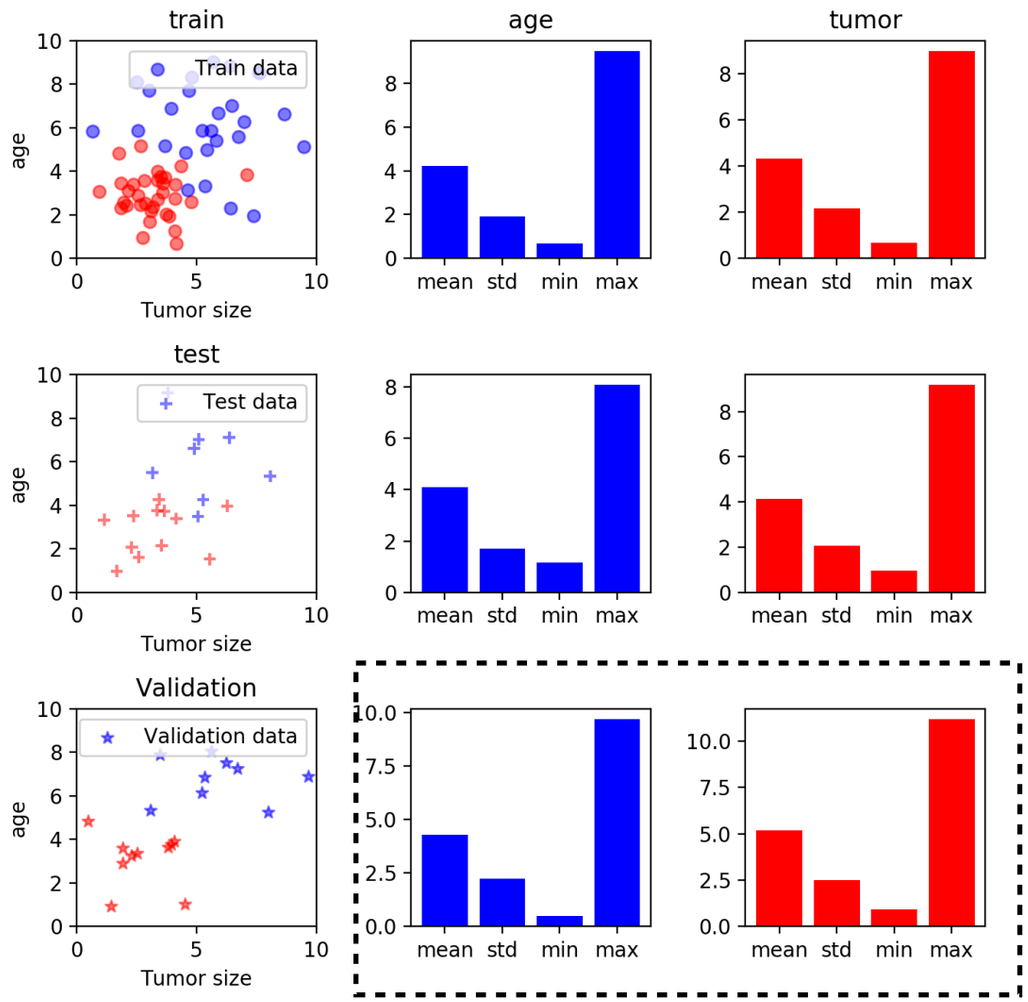
```python
def var_summary(x):
    ret_dict = {}
    if isinstance(x, np.ndarray):
        ret_dict['mean'] = x.mean()
        ret_dict['std'] = x.std()
        ret_dict['min'] = x.min()
        ret_dict['max'] = x.max()
    else:
        raise ValueError
    return ret_dict

# Train Summaries
train_xs_0_summaries = var_summary(train_xs[:,0])
train_xs_1_summaries = var_summary(train_xs[:,1])

# Test Summaries
test_xs_0_summaries = var_summary(test_xs[:,0])
test_xs_1_summaries = var_summary(test_xs[:,1])

# Validation Summaries
val_xs_0_summaries = var_summary(val_xs[:,0])
val_xs_1_summaries = var_summary(val_xs[:,1])
```



Matplotlib

데이터를 시각화하기!

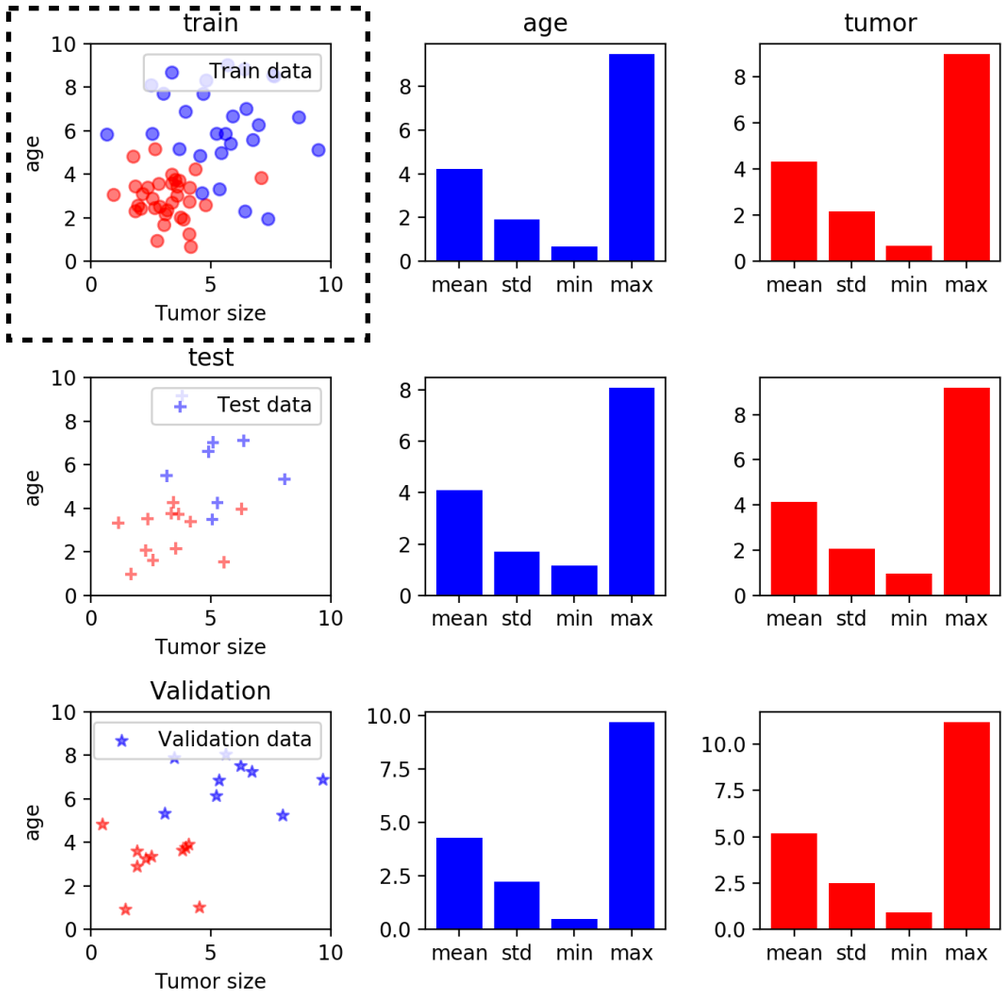
```python
def var_summary(x):
    ret_dict = {}
    if isinstance(x, np.ndarray):
        ret_dict['mean'] = x.mean()
        ret_dict['std'] = x.std()
        ret_dict['min'] = x.min()
        ret_dict['max'] = x.max()
    else:
        raise ValueError
    return ret_dict

# Train Summaries
train_xs_0_summaries = var_summary(train_xs[:,0])
train_xs_1_summaries = var_summary(train_xs[:,1])

# Test Summaries
test_xs_0_summaries = var_summary(test_xs[:,0])
test_xs_1_summaries = var_summary(test_xs[:,1])

# Validation Summaries
val_xs_0_summaries = var_summary(val_xs[:,0])
val_xs_1_summaries = var_summary(val_xs[:,1])
```



Matplotlib

데이터를 시각화하기!

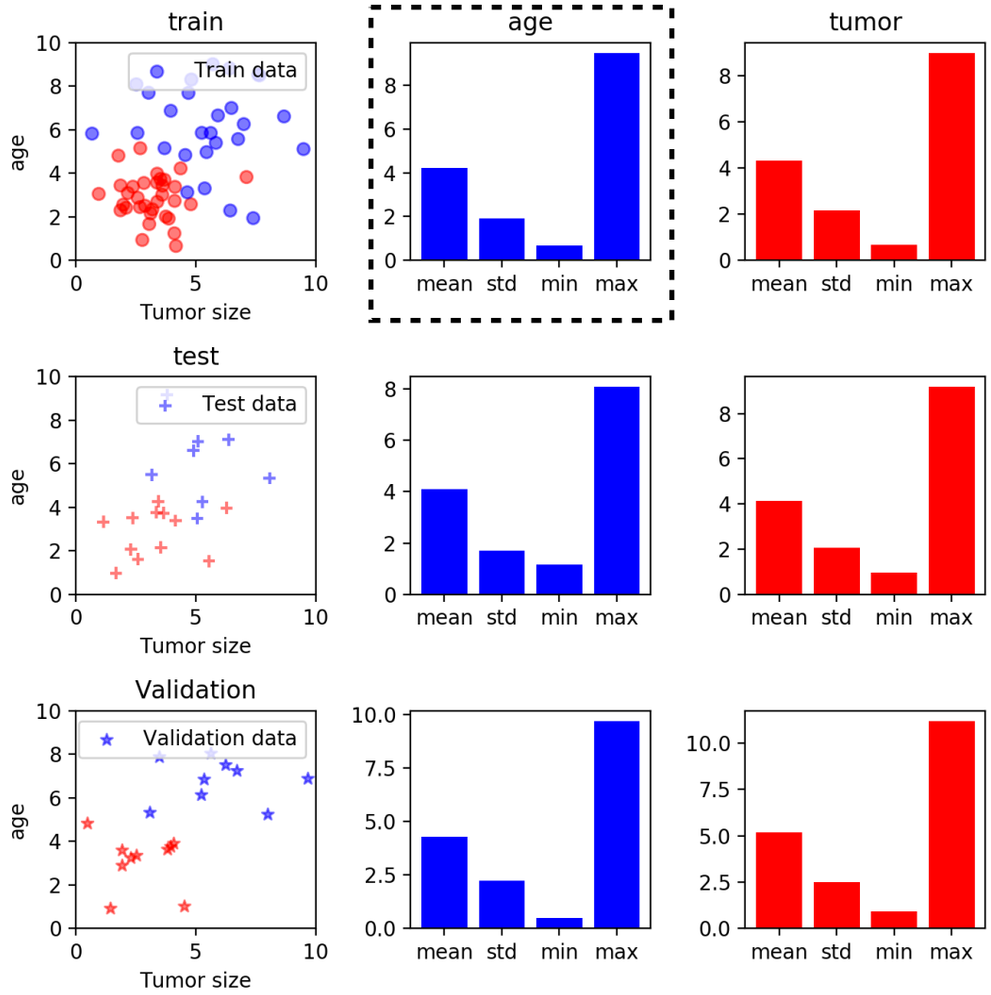
```
#
fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(331)

train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('train')
ax.legend()

ax = fig.add_subplot(332)
ax.bar(list(train_xs_0_summaries.keys()),\
 list(train_xs_0_summaries.values()), align='center', color='b')
ax.set_title('age')

ax = fig.add_subplot(333)
ax.bar(list(train_xs_1_summaries.keys()),\
 list(train_xs_1_summaries.values()), align='center', color='r')
ax.set_title('tumor')
```



Matplotlib

데이터를 시각화하기!

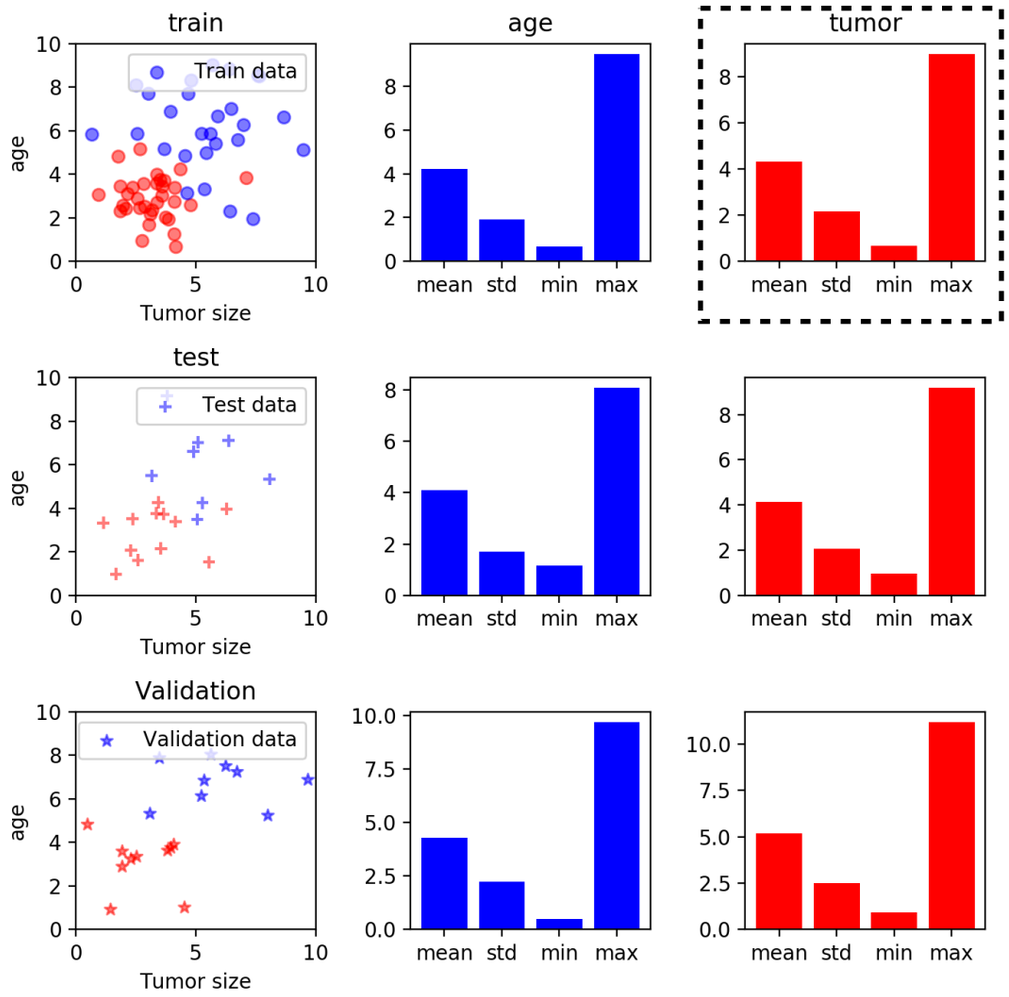
```
#
fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(331)

train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('train')
ax.legend()

ax = fig.add_subplot(332)
ax.bar(list(train_xs_0_summaries.keys()),\
 list(train_xs_0_summaries.values()), align='center', color='b')
ax.set_title('age')

ax = fig.add_subplot(333)
ax.bar(list(train_xs_1_summaries.keys()),\
 list(train_xs_1_summaries.values()), align='center', color='r')
ax.set_title('tumor')
```



Matplotlib

데이터를 시각화하기!

```
#
fig = plt.figure(figsize=(7,7))
ax = fig.add_subplot(331)

train_colors = ['b' if label == 0 else 'r' for label in train_ys]
ax.scatter(train_xs[:,0], train_xs[:,1], label='Train data', alpha=0.5, \
 marker='o', c=train_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('train')
ax.legend()

ax = fig.add_subplot(332)
ax.bar(list(train_xs_0_summaries.keys()),\
 list(train_xs_0_summaries.values()), align='center', color='b')
ax.set_title('age')

ax = fig.add_subplot(333)
ax.bar(list(train_xs_1_summaries.keys()),\
 list(train_xs_1_summaries.values()), align='center', color='r')
ax.set_title('tumor')
```

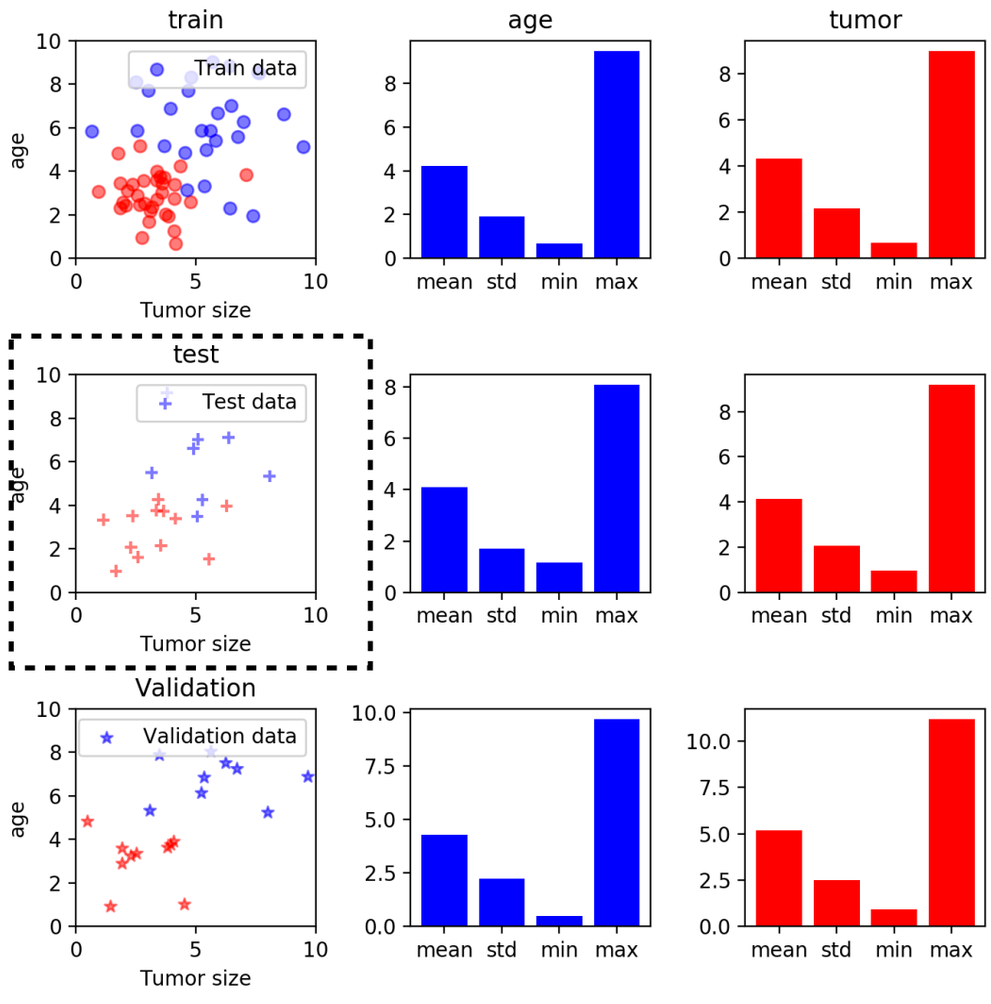


Matplotlib

데이터를 시각화하기!

```
ax = fig.add_subplot(334)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5,\
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('test')
ax.legend()
```

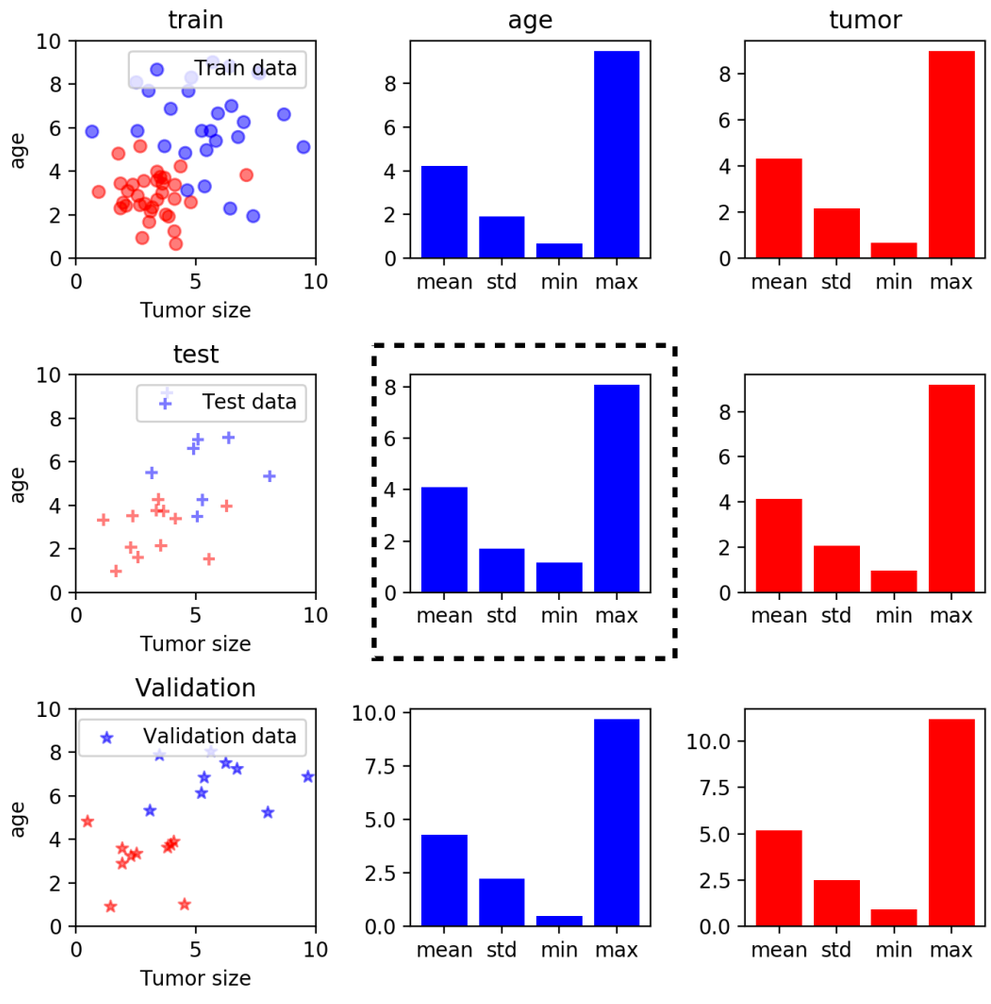
```
ax = fig.add_subplot(335)
ax.bar(list(test_xs_0_summaries.keys()),\
 list(test_xs_0_summaries.values()), align='center', color='b')

ax = fig.add_subplot(336)
ax.bar(list(test_xs_1_summaries.keys()),\
 list(test_xs_1_summaries.values()), align='center', color='r')
```



Matplotlib

데이터를 시각화하기!

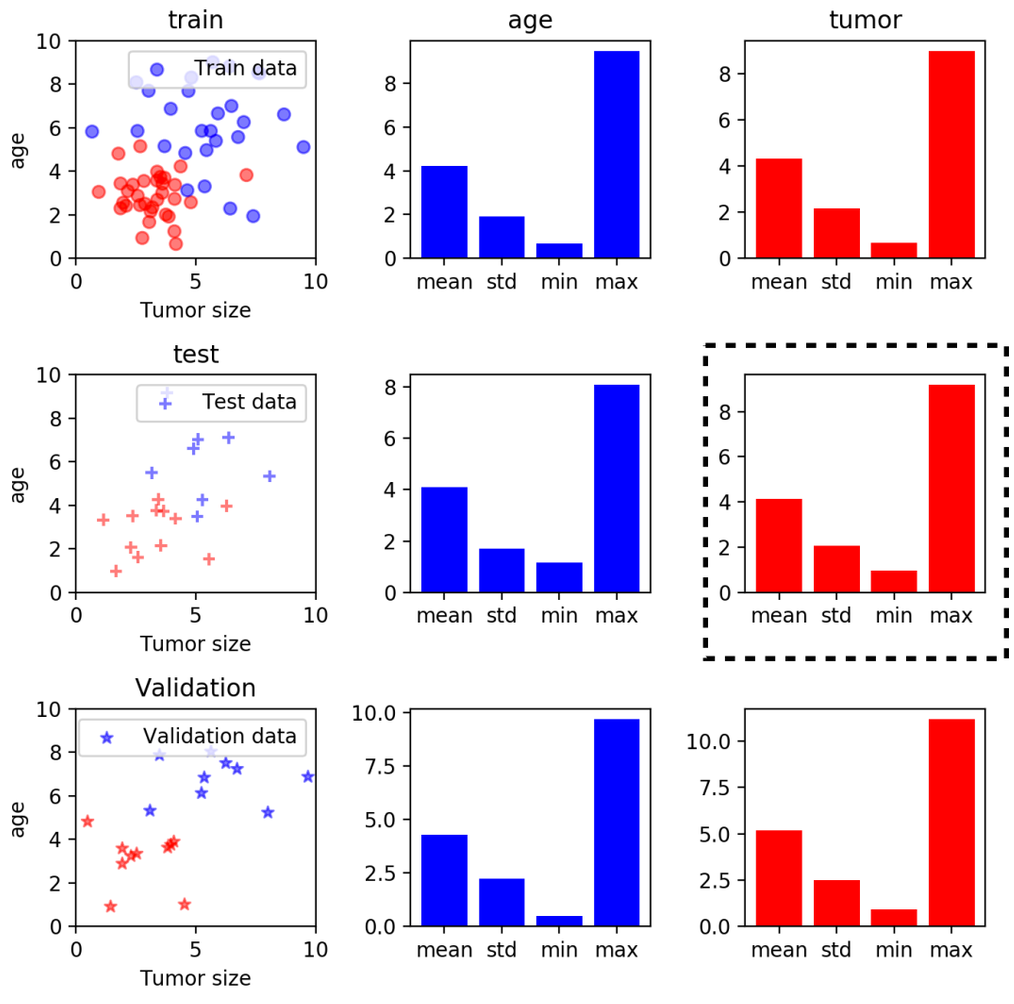
```
ax = fig.add_subplot(334)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5,\
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('test')
ax.legend()

ax = fig.add_subplot(335)
ax.bar(list(test_xs_0_summaries.keys()),\
 list(test_xs_0_summaries.values()), align='center', color='b')

ax = fig.add_subplot(336)
ax.bar(list(test_xs_1_summaries.keys()),\
 list(test_xs_1_summaries.values()), align='center', color='r')
```



Matplotlib

데이터를 시각화하기!

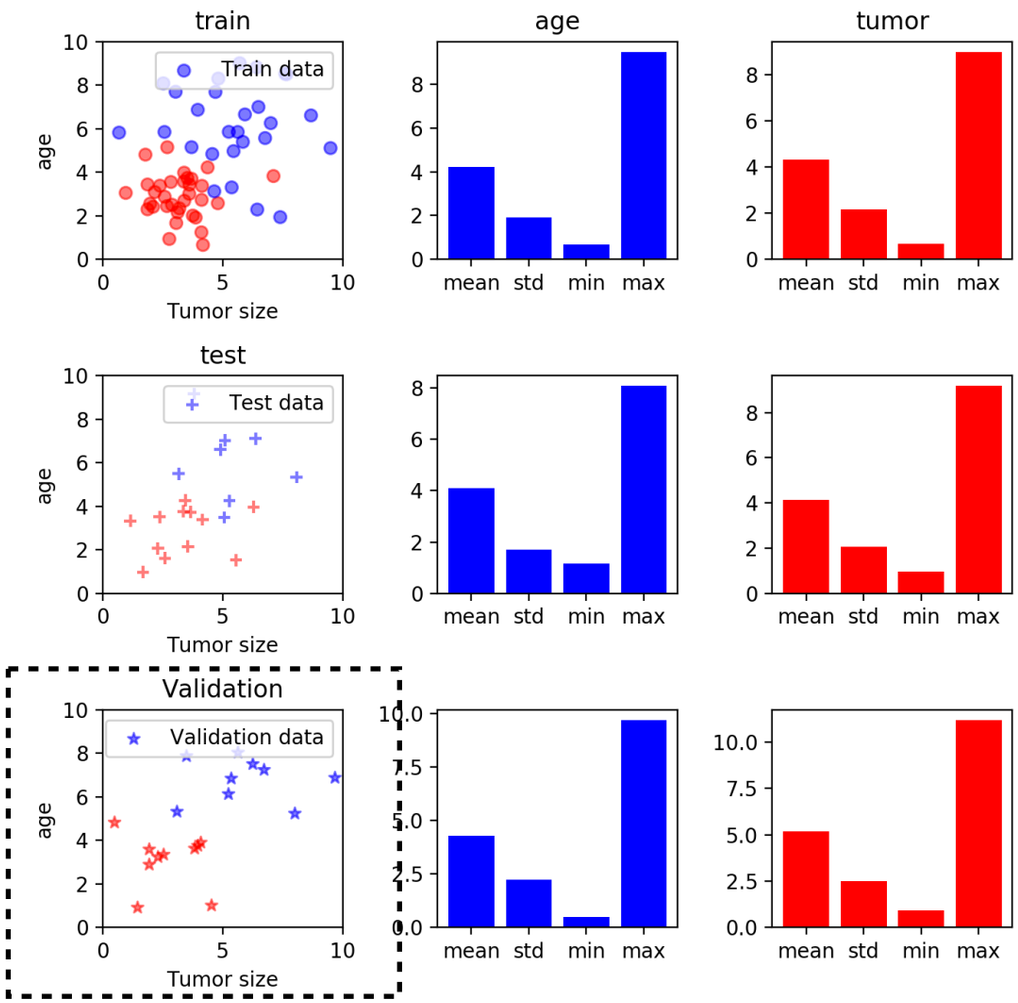
```
ax = fig.add_subplot(334)
test_colors = ['b' if label == 0 else 'r' for label in test_ys]
ax.scatter(test_xs[:,0], test_xs[:,1], label='Test data', alpha=0.5,\
 marker='+', c=test_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('test')
ax.legend()

ax = fig.add_subplot(335)
ax.bar(list(test_xs_0_summaries.keys()),\
 list(test_xs_0_summaries.values()), align='center', color='b')

ax = fig.add_subplot(336)
ax.bar(list(test_xs_1_summaries.keys()),\
 list(test_xs_1_summaries.values()), align='center', color='r')
```



Matplotlib

데이터를 시각화하기!

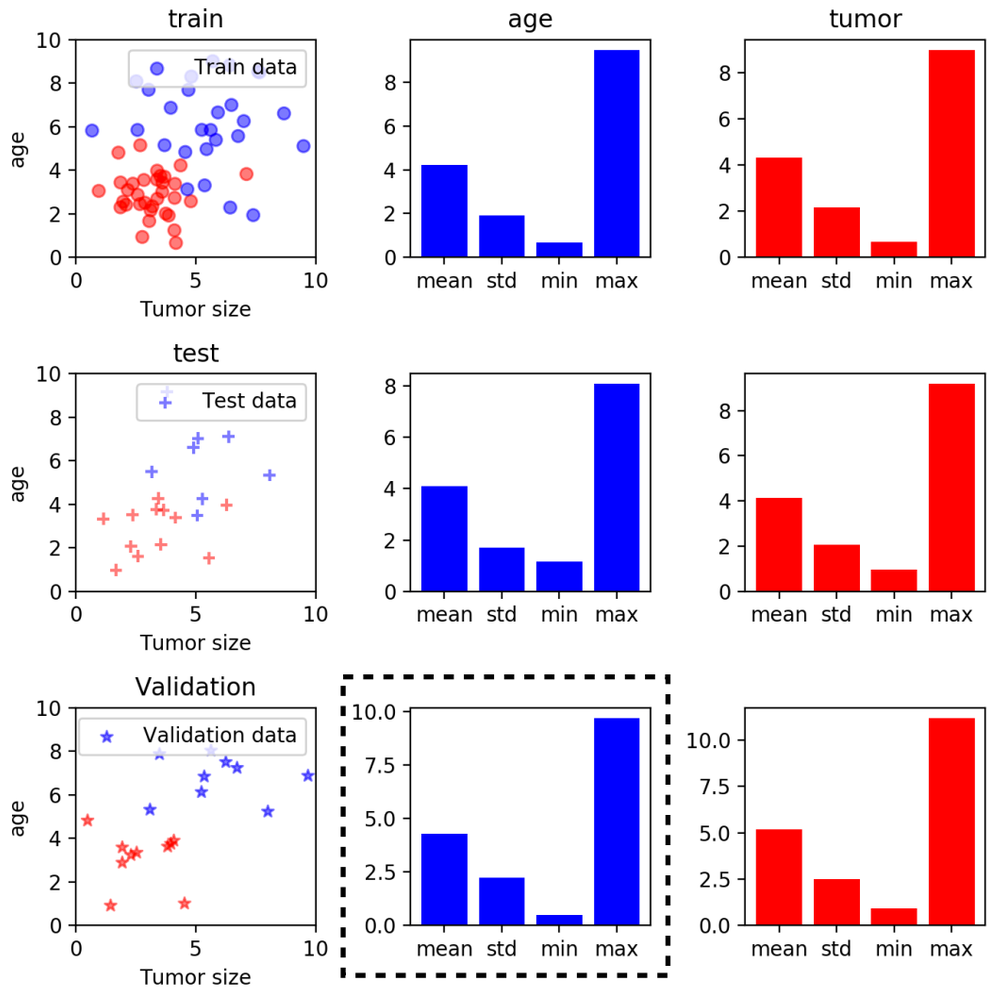
```
ax = fig.add_subplot(337)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()
ax.set_title('Validation')

ax = fig.add_subplot(338)
ax.bar(list(val_xs_0_summaries.keys()), list(val_xs_0_summaries.values()),\
 align='center', color='b')

ax = fig.add_subplot(339)
ax.bar(list(val_xs_1_summaries.keys()), list(val_xs_1_summaries.values()),\
 align='center', color='r')

plt.tight_layout()
plt.show()
```



Matplotlib

데이터를 시각화하기!

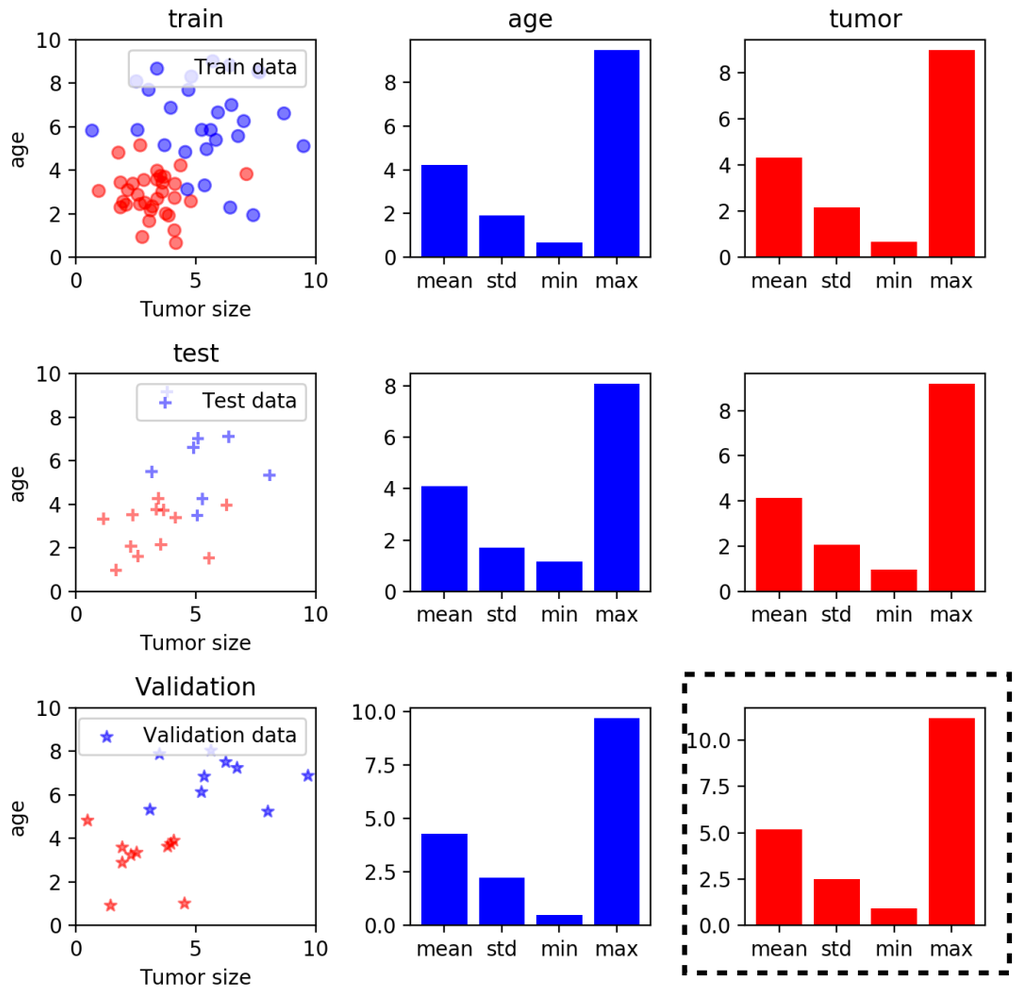
```
ax = fig.add_subplot(337)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()
ax.set_title('Validation')

ax = fig.add_subplot(338)
ax.bar(list(val_xs_0_summaries.keys()), list(val_xs_0_summaries.values()),\
 align='center', color='b')

ax = fig.add_subplot(339)
ax.bar(list(val_xs_1_summaries.keys()), list(val_xs_1_summaries.values()),\
 align='center', color='r')

plt.tight_layout()
plt.show()
```



Matplotlib

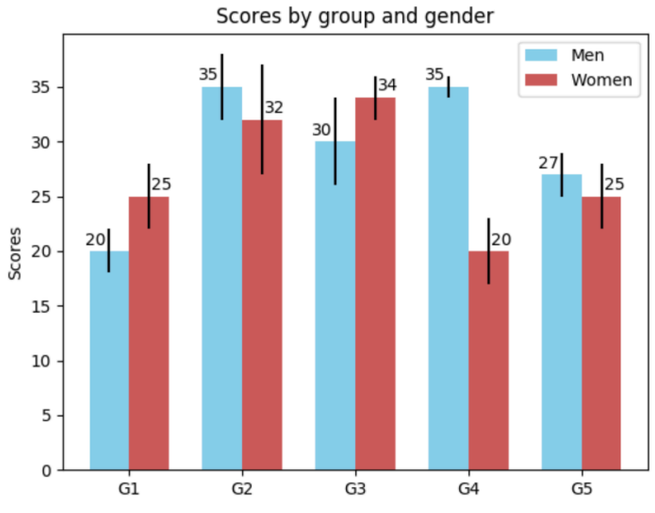
데이터를 시각화하기!

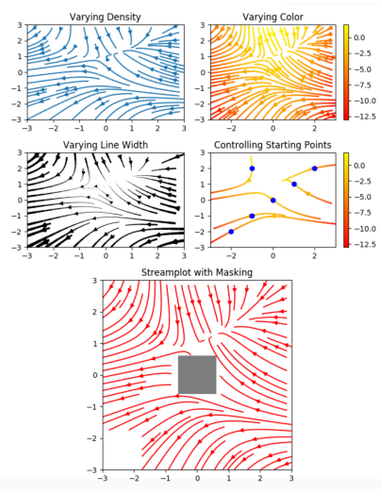
```
ax = fig.add_subplot(337)
val_colors = ['b' if label == 0 else 'r' for label in val_ys]
ax.scatter(val_xs[:,0], val_xs[:,1], label='Validation data',\
 alpha=0.5, marker='*', c=val_colors)
ax.set_xlabel('Tumor size')
ax.set_ylabel('age')
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.legend()
ax.set_title('Validation')

ax = fig.add_subplot(338)
ax.bar(list(val_xs_0_summaries.keys()), list(val_xs_0_summaries.values()),\
 align='center', color='b')

ax = fig.add_subplot(339)
ax.bar(list(val_xs_1_summaries.keys()), list(val_xs_1_summaries.values()),\
 align='center', color='r')

plt.tight_layout()
plt.show()
```

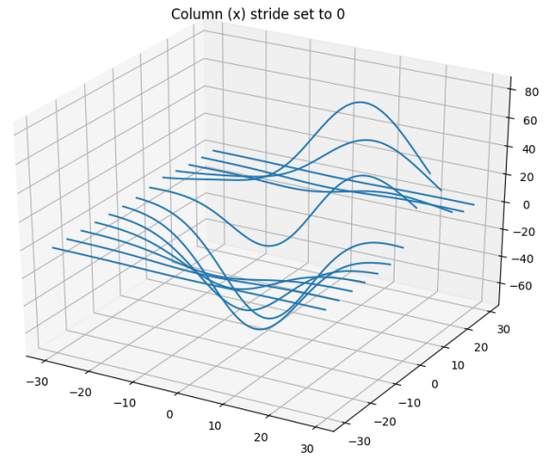


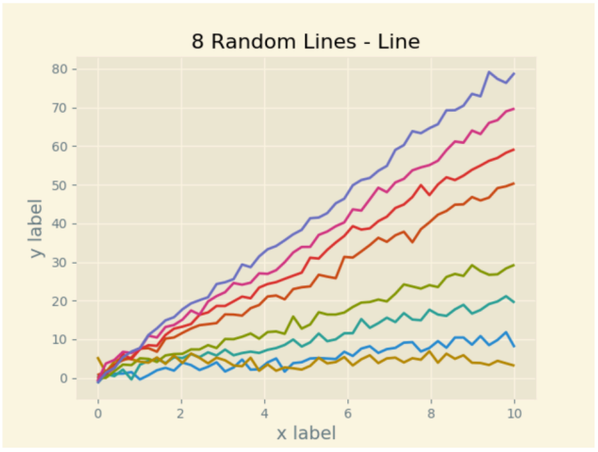
Matplotlib

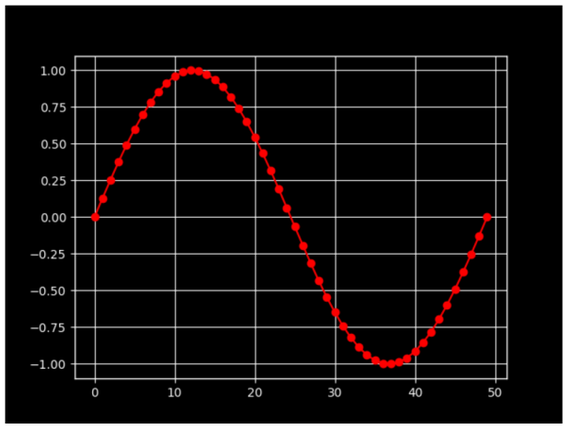
시각화의 세계는 아직 더 많이 남아있습니다.

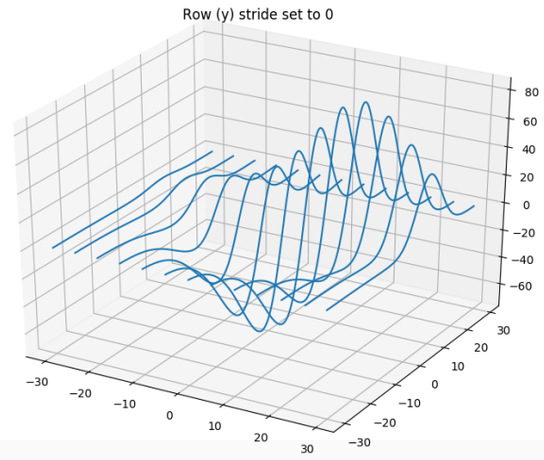








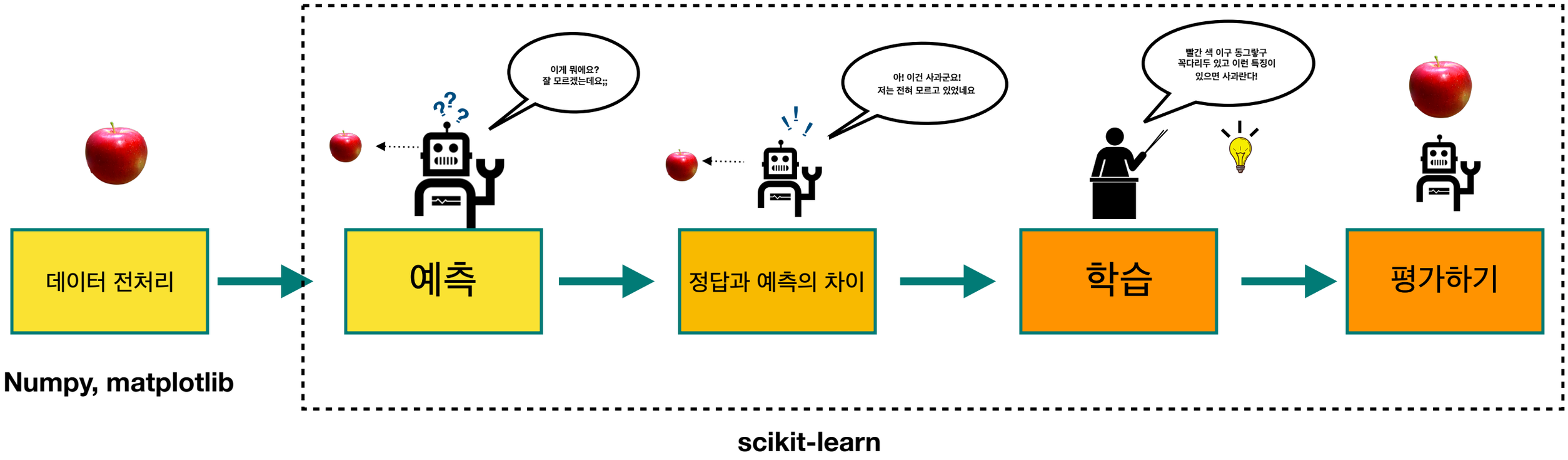




인공지능 개론

인공지능은 무엇일까요?

인공지능을 만드는 5가지 과정



인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)

ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```

Scikit-learn pakage 을 불러옵니다.
Scikit-learn 은 Python 의 Machine learning Pakage 입니다! 오늘 구현할 Linear Regression 또한 Scikit-learn 에 구현되어 있습니다.

인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)

ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```

Scikit-learn 에서 Linear Regression 모델을 불러옵니다.

인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)

ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```

scikit-learn 은 fit 함수를 통해 학습합니다.
Xs 와 정답데이터인 Ys 을 넣으면 자동으로 학습됩니다.

인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)

ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```

학습된 모델은 regr 에 저장되어 있습니다.
저장된 모델을 이용해 테스트 데이터를 평가할 수 있습니다.

인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)

ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```

Scikit-learn 에서 Linear Regression 모델을 불러옵니다.

예측값

```
[ 0.73223794 -0.01327443  1.0509077   0.54064363  1.0357858   1.0646869
  0.8996246   0.78681195  0.91288745 -0.19037986  0.5550972   0.24459589
  0.2586167   0.84552664 -0.3182311   0.95136565  0.94353557  0.6544242
  0.17461514  0.9895366 ]
```

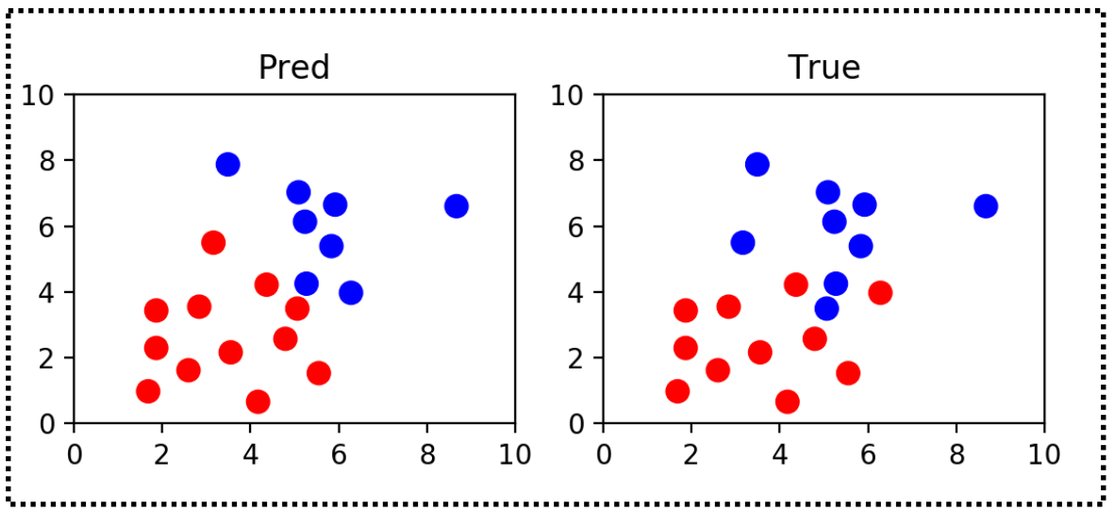
↓

[0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1]

인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)

ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```



인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)

ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```

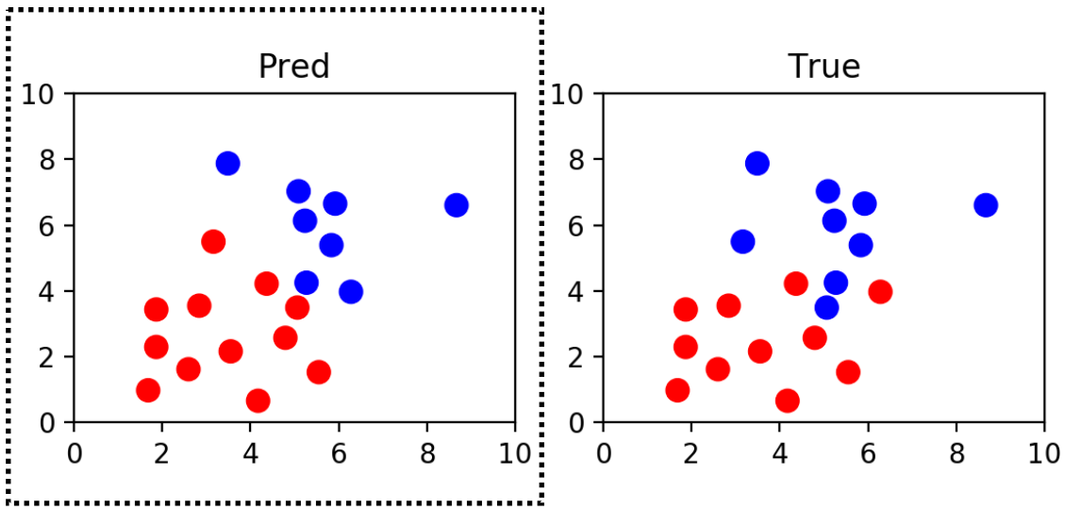
1 (정상) 빨강 , 0 (비정상) 파랑 으로 바꿉니다.

1	1	0	0	1	0	1	0	1	0	0	0	0	1	1	0	0	0	1	0]
r'	'r'	'b'	'b'	'r'	'b'	'r'	'b'	'r'	'b'	'b'	'b'	'b'	'r'	'r'	'b'	'b'	'b'	'r'	'b']

인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)

ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```

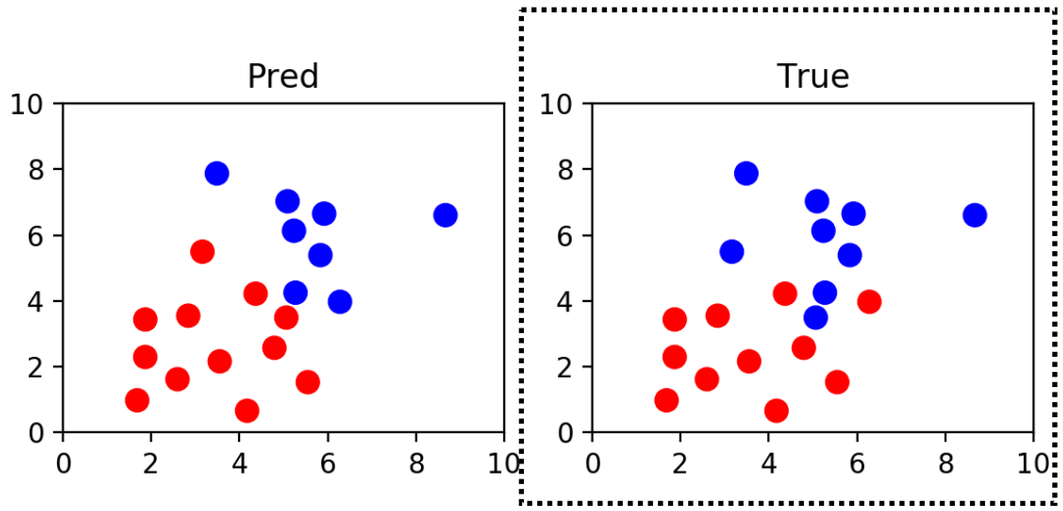


인공지능 모형

linear regression

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score

datum = np.genfromtxt('cancer_data.csv',delimiter=',',dtype=np.float32,\
 skip_header=True)
# except header
datum = datum[1:,:]

# Shuffle
npr.shuffle(datum)

X = datum[:, :2]
Y = datum[:, 2]

train_x = X[:60]
test_x = X[60:80]
val_x = X[80:]

train_y= Y[:60]
test_y = Y[60:80]
val_y = Y[80:]

regr = linear_model.LinearRegression()
regr.fit(train_x, train_y)
test_preds = regr.predict(test_x)

fig = plt.figure()
ax = fig.add_subplot(2,2,1)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('Pred')
test_preds = [0 if pred <= 0.5 else 1 for pred in test_preds]
pred_colors = ['b' if pred ==0 else 'r' for pred in test_preds]
ax.scatter(test_x[:,0], test_x[:,1], color=pred_colors, linewidth=3)
```

```
ax = fig.add_subplot(2,2,2)
ax.set_xlim(0,10)
ax.set_ylim(0,10)
ax.set_title('True')
y_colors = ['b' if y==0 else 'r' for y in test_y]
ax.scatter(test_x[:,0], test_x[:,1], color=y_colors, linewidth=3)
plt.show()
```



결과 측정 (metric)

```
acc = np.mean(np.equal(test_preds,test_y))
print(acc)
```