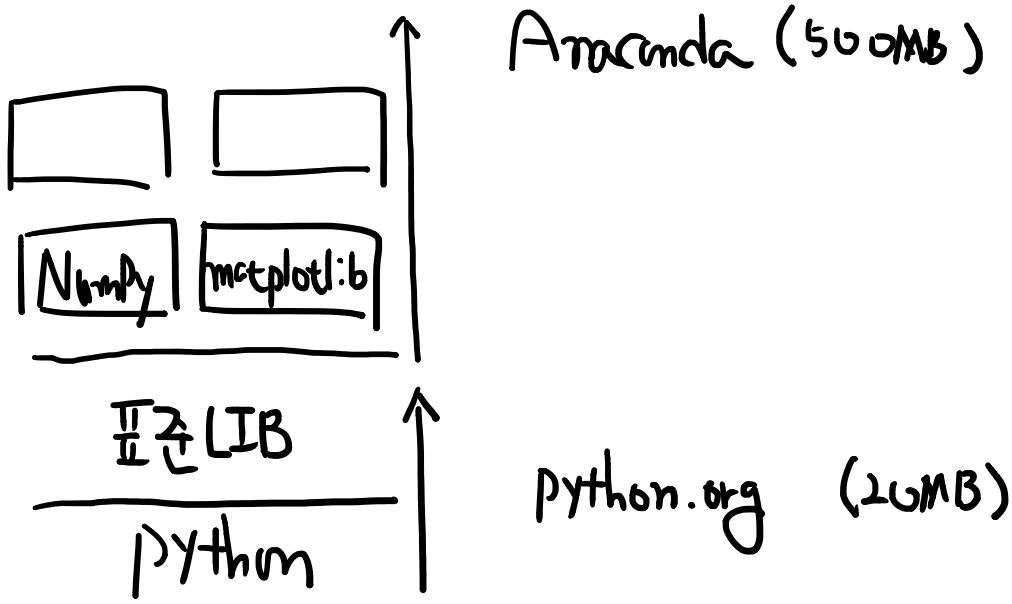
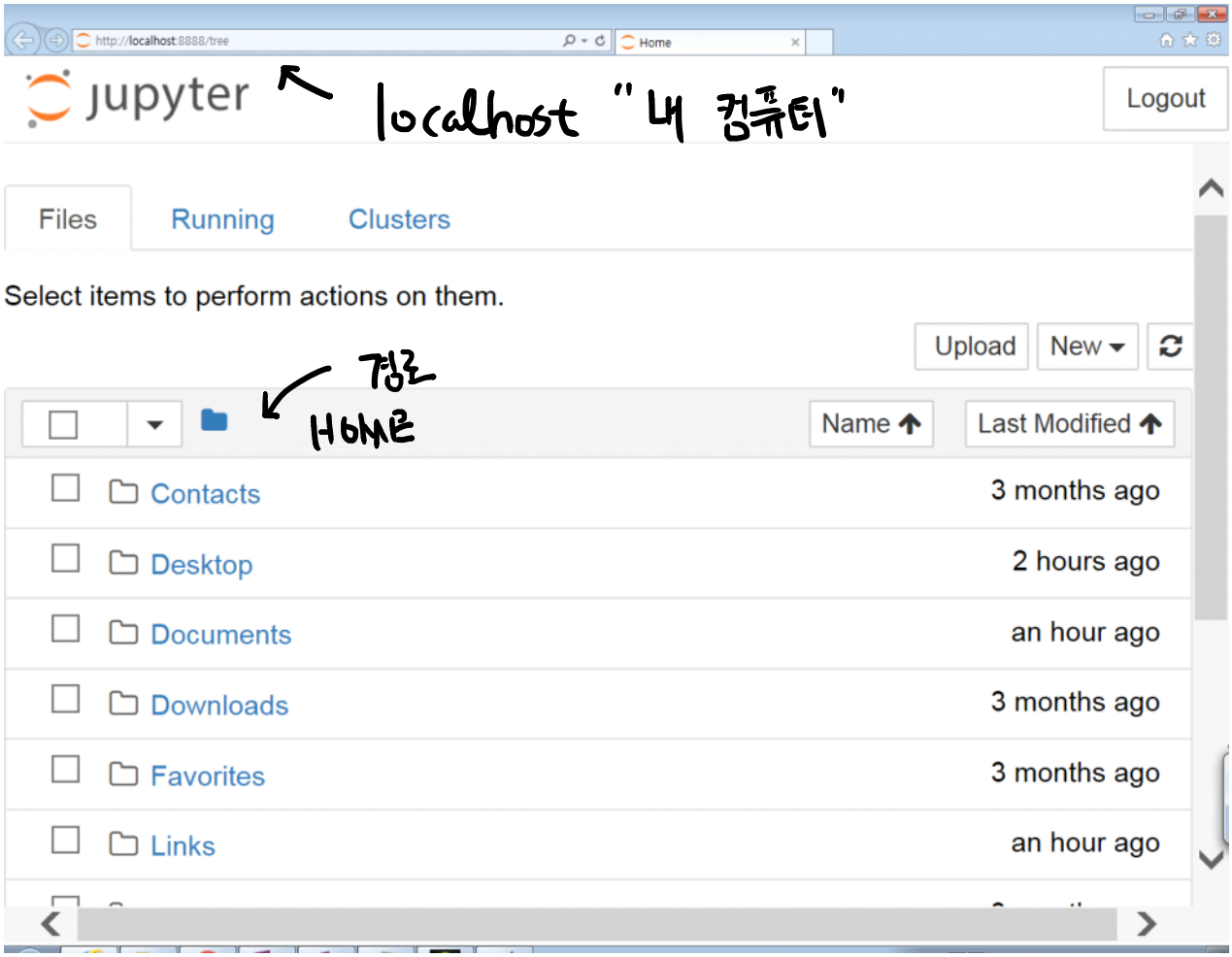
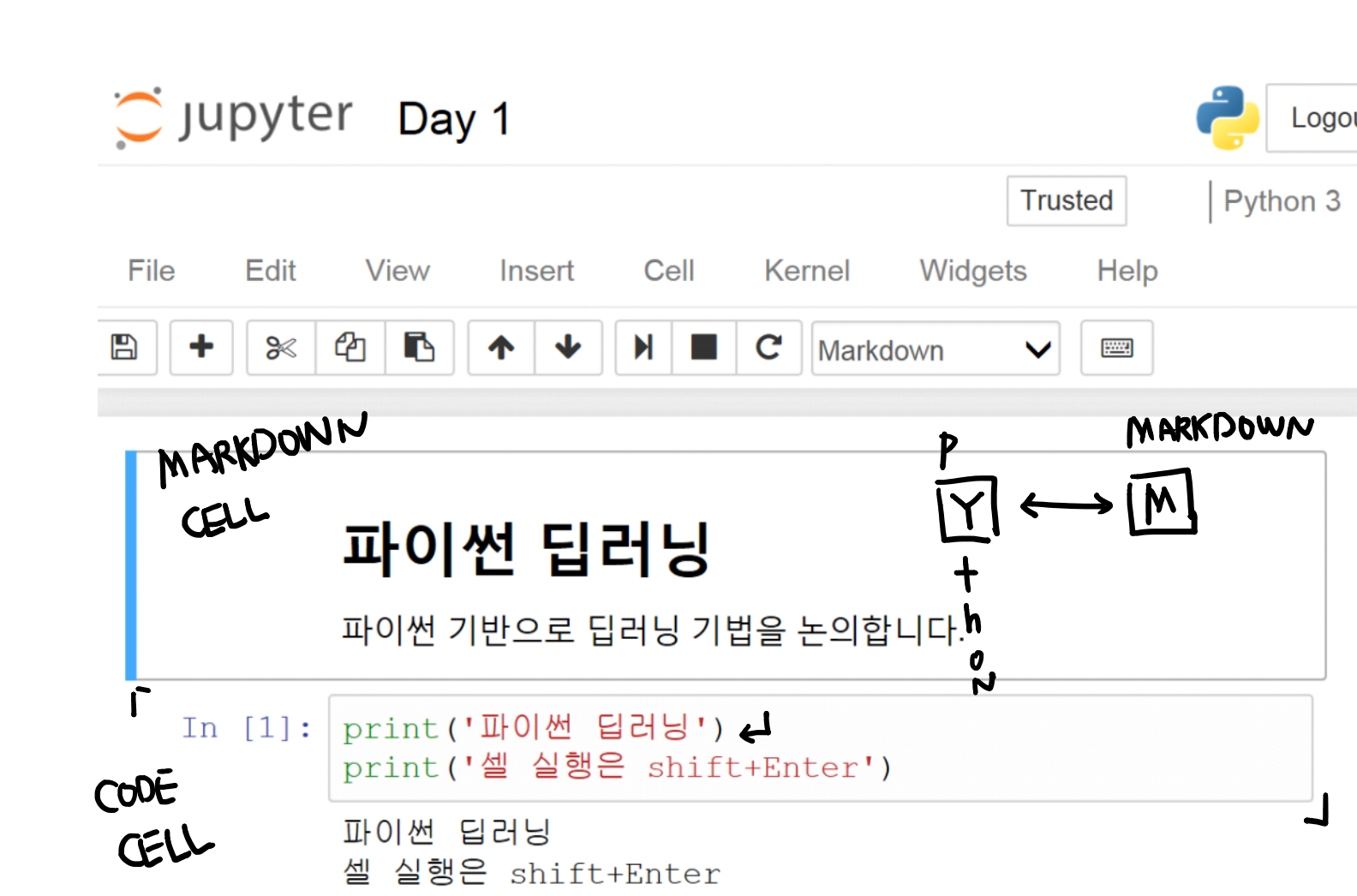
환경설정

2018년 6월 18일 월요일 오전 10:07



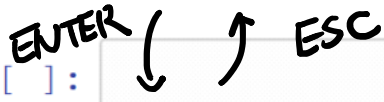




파이썬 기반으로 딥러닝 기법을 논의합니다.

In [2]:
```
print('파이썬 딥러닝')
print('셀 실행은 shift+Enter')
```

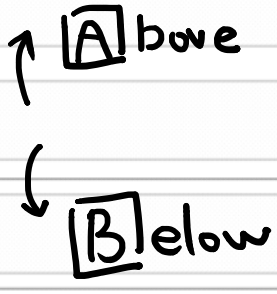
```
파이썬 딥러닝
셀 실행은 shift+Enter
```



In [ ]:

In [19]:
```
x = 1
```

In [ ]:

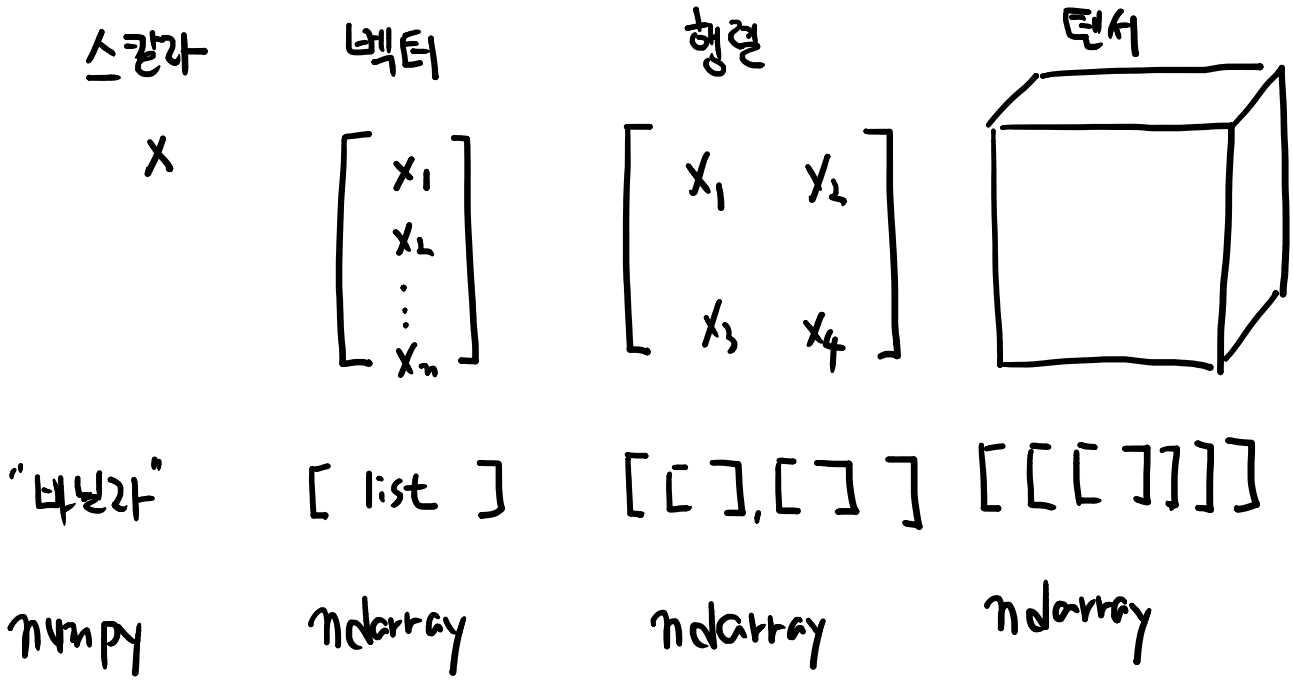


NumPy

2018년 6월 18일 월요일 오전 10:57

```
In [28]: import numpy as np

In [29]: arr = np.array(data)

In [30]: arr
Out[30]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

In [31]: type(arr)
Out[31]: numpy.ndarray
```

n-dimension



```
In [29]: arr = np.array(data)

In [30]: arr

Out[30]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9]])

In [31]: type(arr)

Out[31]: numpy.ndarray

In [32]: arr.dtype

Out[32]: dtype('int32')
```

바닐라

[1, '2', 3.14]

py	numpy
int	(U)int8,16,32,64
float	float16,32,64

np.array (list (range(1e7)))

np.arange (1e7)

```
In [39]: arr = np.arange(1e7)

In [40]: nums = arr.tolist()

In [41]: len(arr)

Out[41]: 10,000,000
```

"매직"

```
In [42]: %timeit arr * 1.1

         28.6 ms ± 763 µs per loop (mean ± std. dev. of
         7 runs, 10 loops each)
```

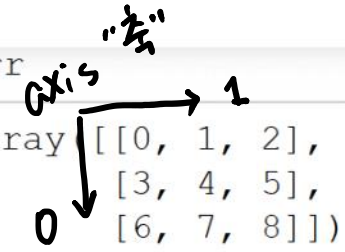
X30

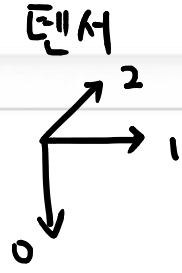
```
In [42]: %timeit arr * 1.1

28.6 ms ± 763 µs per loop (mean ± std. dev. of
7 runs, 10 loops each)
```

x30

```
In [43]: %timeit [n*1.1 for n in nums]

852 ms ± 137 ms per loop (mean ± std. dev. of 7
runs, 1 loop each)
```

```
In [52]: arr
```






```
Out[52]: array([[0, 1, 2],
                [3, 4, 5],
                [6, 7, 8]])
```

```
In [53]: arr.shape

Out[53]: (3, 3)
```

arr [0축 , 1축, …]

```
In [54]: arr[0]

Out[54]: array([0, 1, 2])
```

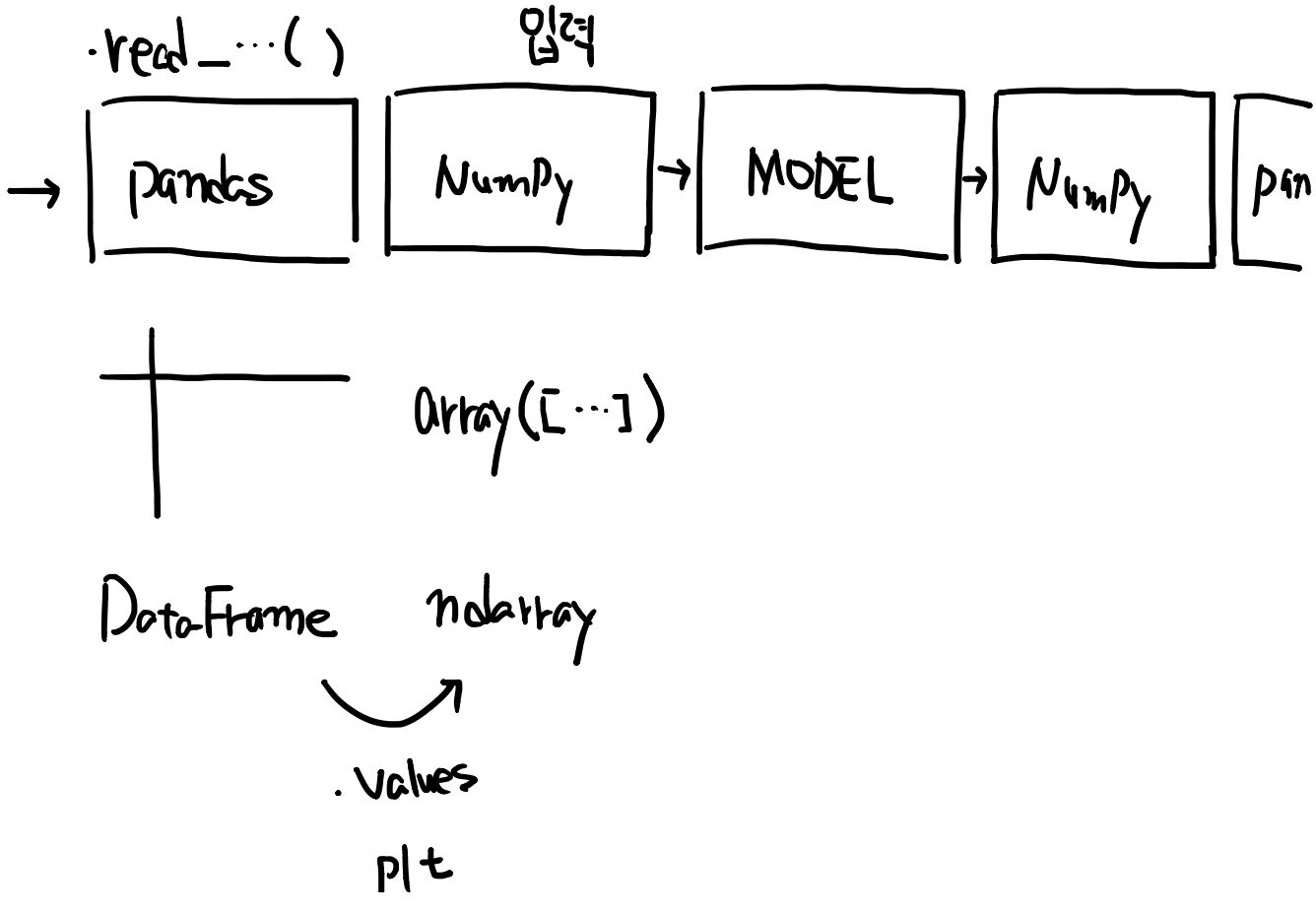
```
In [55]: arr[:, 0]

Out[55]: array([0, 3, 6])
```

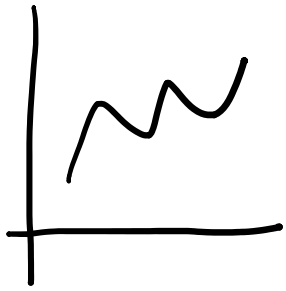
[: , 0]

```
In [101]: iris = pd.read_csv('data/iris.data', header=None)

In [103]: type(iris)

Out[103]: pandas.core.frame.DataFrame

In [104]: iris.values

Out[104]: array([[5.1, 3.5, 1.4, 0.2, 'Iris-setosa'],
                 [4.9, 3.0, 1.4, 0.2, 'Iris-setosa'],
                 [4.7, 3.2, 1.3, 0.2, 'Iris-setosa'],
                 [4.6, 3.1, 1.5, 0.2, 'Iris-setosa'],
                 [5.0, 3.6, 1.4, 0.2, 'Iris-setosa'],
                 [5.4, 3.9, 1.7, 0.4, 'Iris-setosa'],
                 [4.6, 3.4, 1.4, 0.3, 'Iris-setosa'],
                 [5.0, 3.4, 1.5, 0.2, 'Iris-setosa'],
                 [4.4, 2.9, 1.4, 0.2, 'Iris-setosa'],
```

v

기계학습 파이프라인





```
In [112]: arr = np.arange(4).reshape(2, 2)
          arr

Out[112]: array([[0, 1],
                 [2, 3]])
```

[ ] → array([···])

↓

DataFrame

```
In [113]: frame = pd.DataFrame(arr)
          frame
```

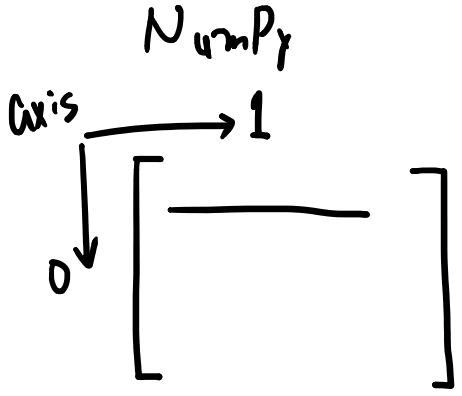
Out[113]:

	0	1
0	0	1
1	2	3

```
In [116]: iris[0][:5]

Out[116]: 0    5.1
          1    4.9
          2    4.7
          3    4.6
          4    5.0
          Name: 0, dtype: float64
```

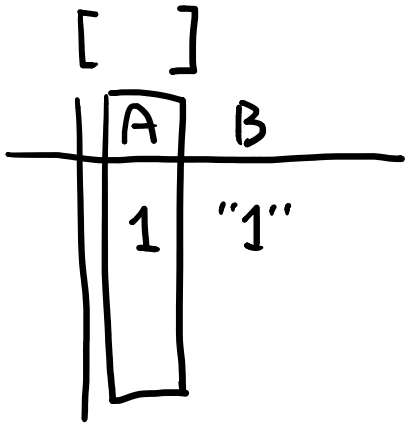
```
In [117]: iris[4][:5]

Out[117]: 0    Iris-setosa
          1    Iris-setosa
          2    Iris-setosa
          3    Iris-setosa
          4    Iris-setosa
          Name: 4, dtype: object
```

Name: 4, dtype: object



"행 우선"

[0]



"열 우선"

['A']

기계학습

2018년 6월 18일 월요일 오후 1:57

기계학습 리뷰

```
In [128]: iris = pd.read_csv('data/iris.data', header=None)
          iris[:5]
```

Out[128]:

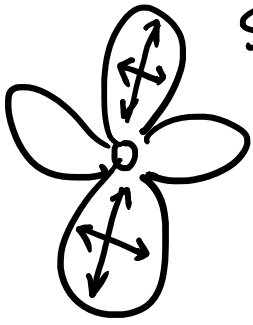
특징(features)

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

샘플

label

붓꽃



Setosa

```
In [129]: iris.shape
```

Out[129]: (150, 5)

```
In [131]: iris[4].value_counts()
```

Out[131]:
```
Iris-versicolor    50
Iris-setosa        50
Iris-virginica     50
Name: 4, dtype: int64
```

클래스('분류')

지도학습 ← 보조 ← 비지도학습

Supervised Learning

X, y 정답

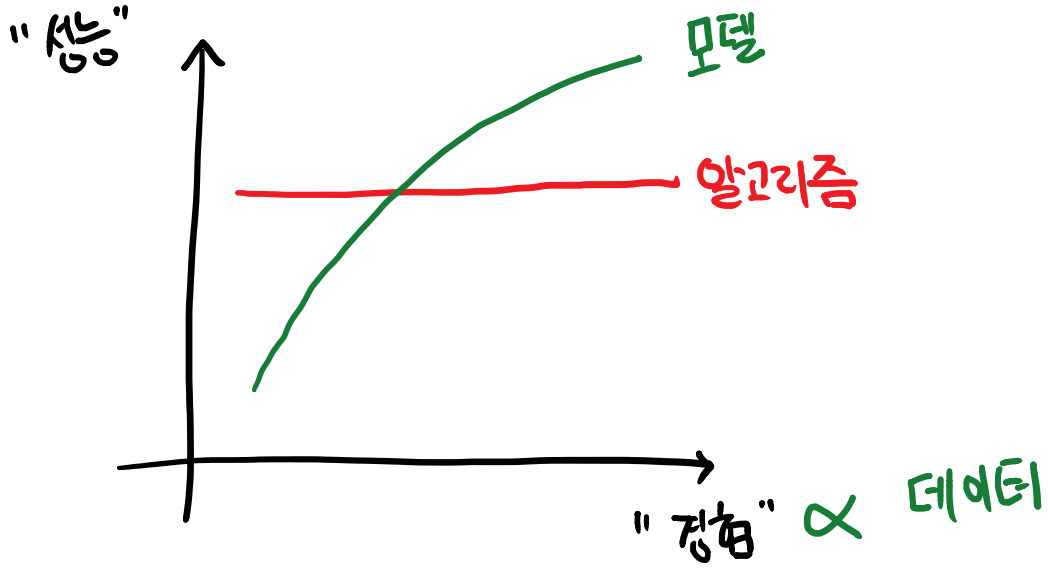
Unsupervised

X, ?

0.1 2.3 0.4 1.0 → 31

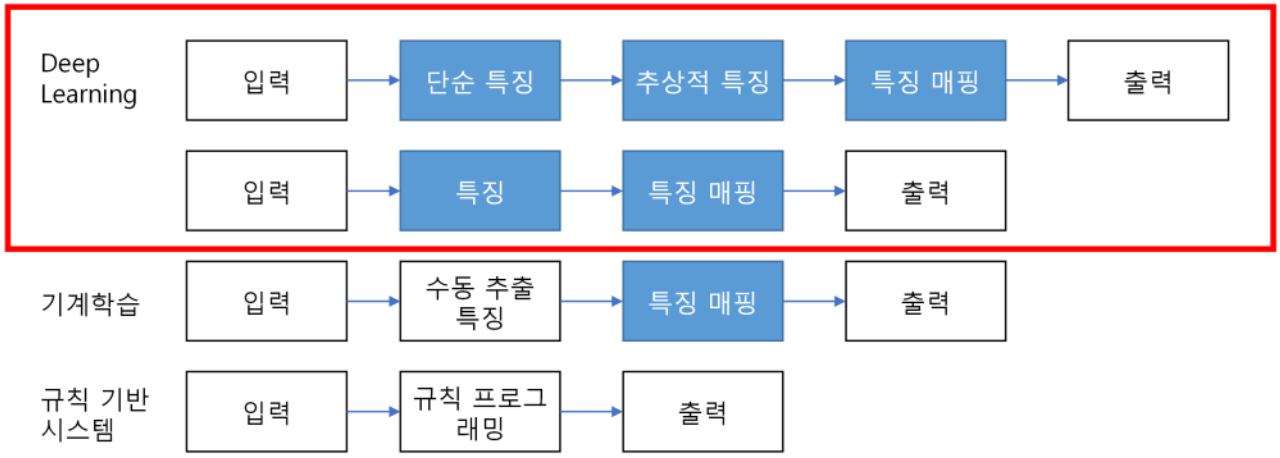
4.9 3.4 0.5 1.1 → ?

학습 모델과 알고리즘

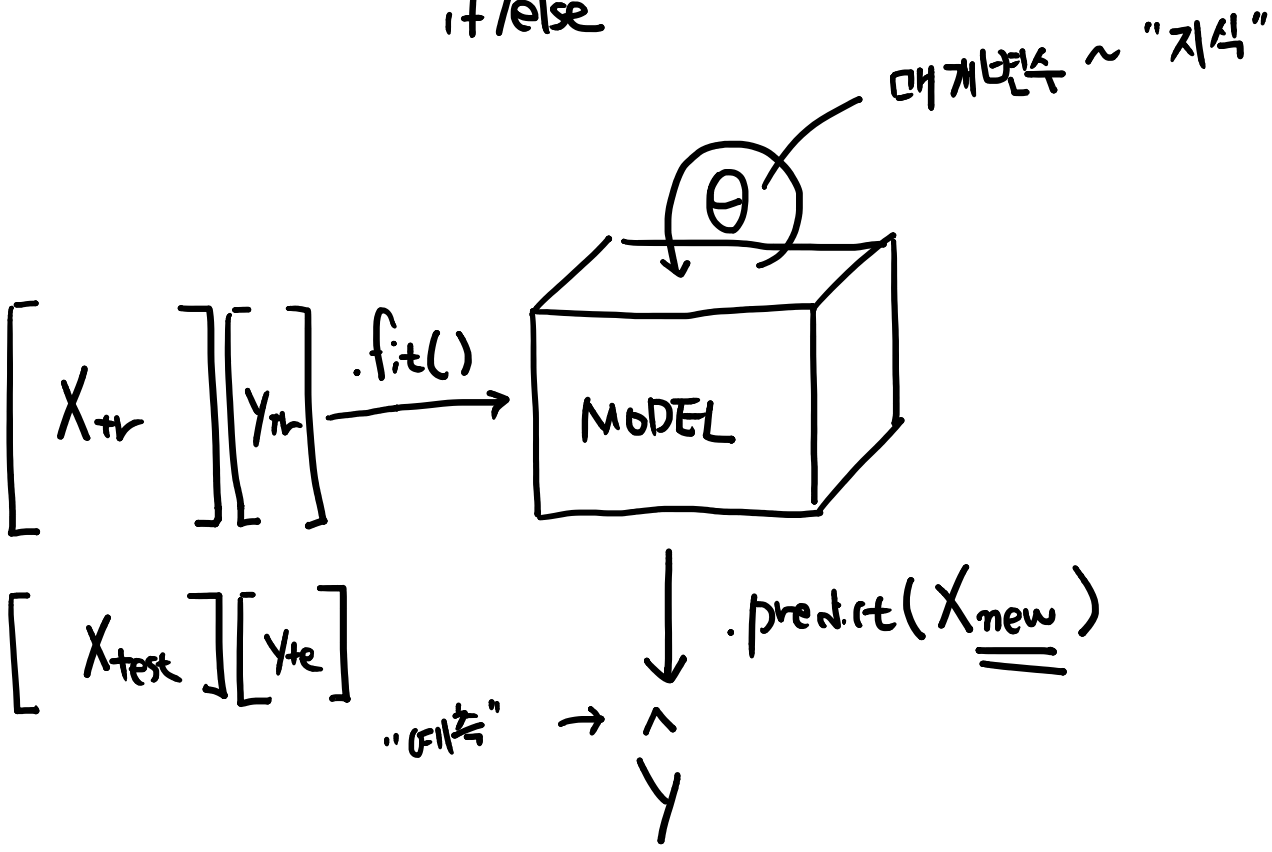


비교

표현 기반 학습



if/else



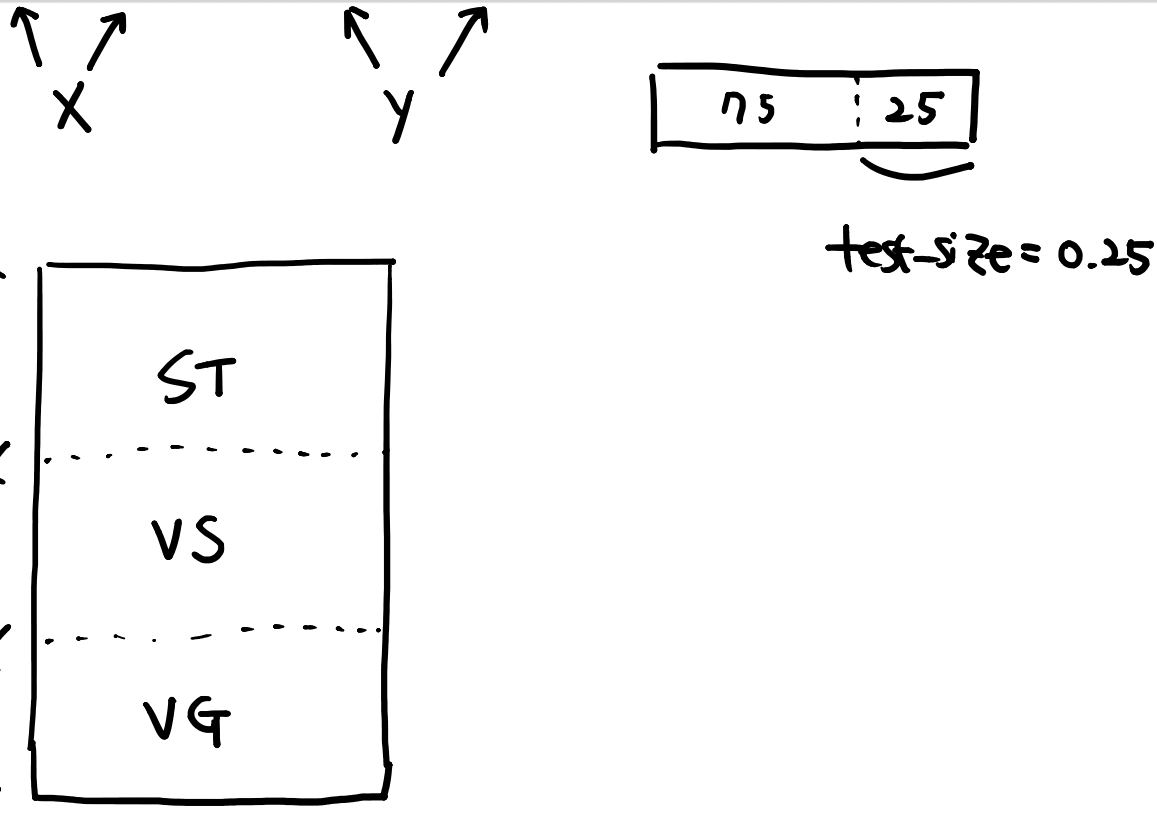
훈련/테스트 분리

In [157]:

```
from sklearn.model_selection import train_test_split
```

In [158]:

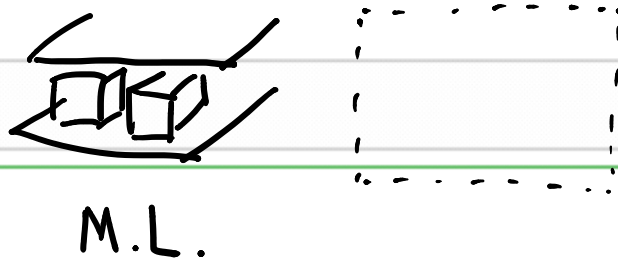
```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```



```
X_train, X_test, y_train, y_test = train_test_split(X, y)
```

In [173]:

```
model = LogisticRegression()
```



In [174]:

```
model.fit(X_train, y_train)
```

...

In [175]:

```
model.score(X_test, y_test)
```

Out[175]:

0.944055944055944406

분류

$$y = \begin{cases} 0 \\ 1 \\ 2 \\ \vdots \\ n \end{cases}$$

.value_counts()

회귀

$y = 0.123\cdots$

In [184]:

```
model.score(X_train, y_train)
```

Out[184]:

0.70704449413358561

회귀 평가

$$R^2 = 1 - \frac{\Sigma(y-\hat{y})^2}{\Sigma(y-\bar{y})^2}$$

예측

평균

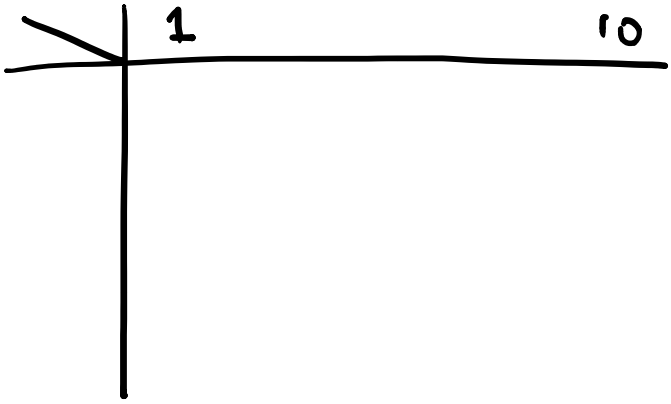
In [185]:

```
model.score(X_test, y_test)
```

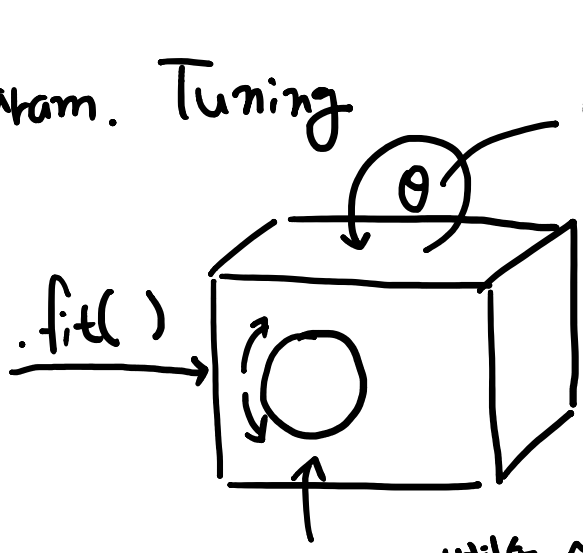
Out[185]:

0.58071127318418336

교차확인 Cross-Validation



Hyper Param. Tuning



매개변수 ~ "지식"

상위 매개변수 ~ "환경"

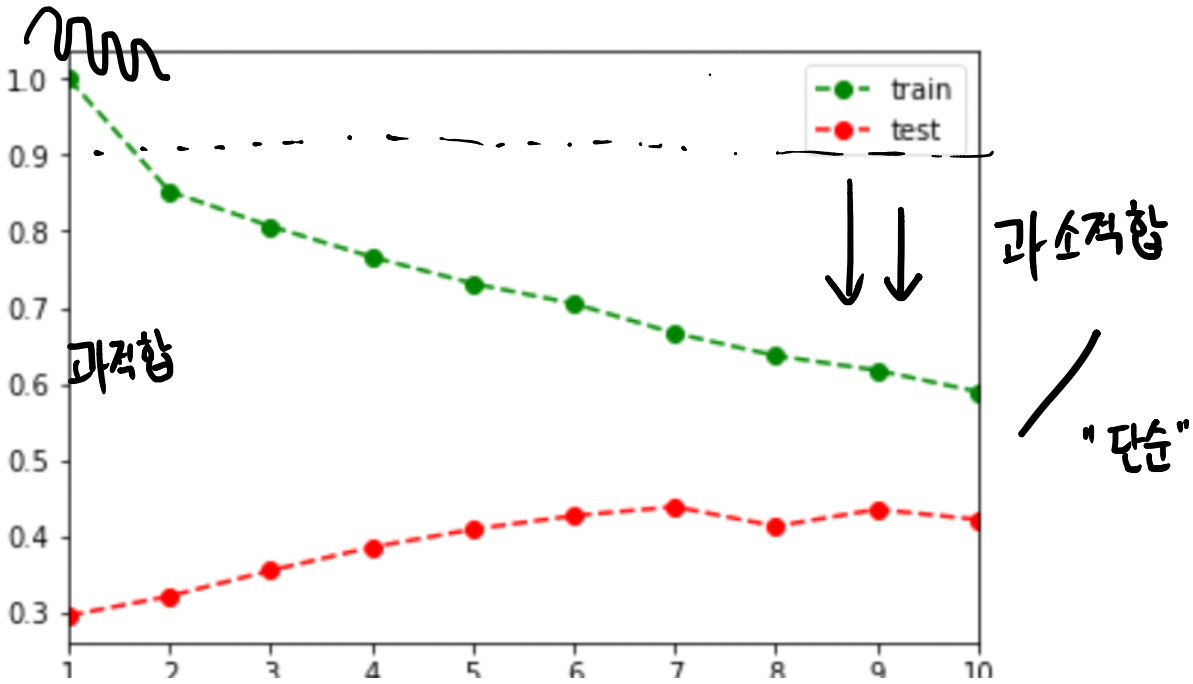
Param.

상위 매개변수
Hyper Param.

```
훈련결과.plot(style=['go--', 'ro--'])
```

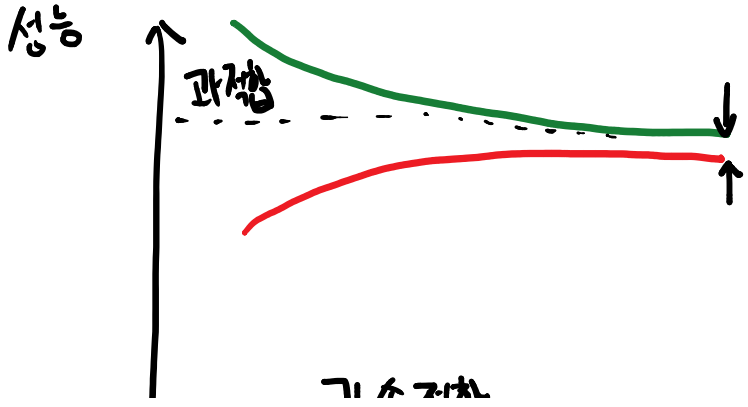
Out[217]:

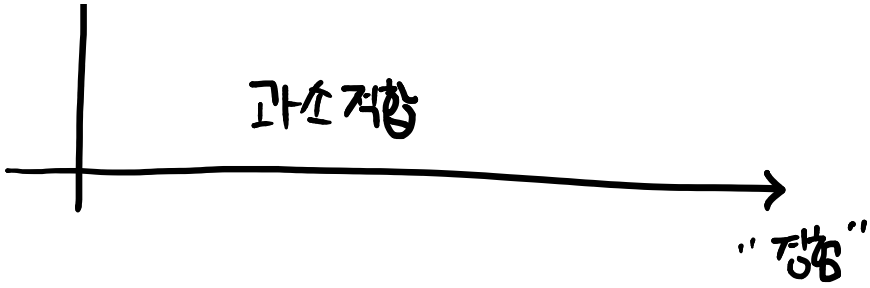
```
<matplotlib.axes._subplots.AxesSubplot at 0xa19ffd0>
```



K=1 이웃수 k=10

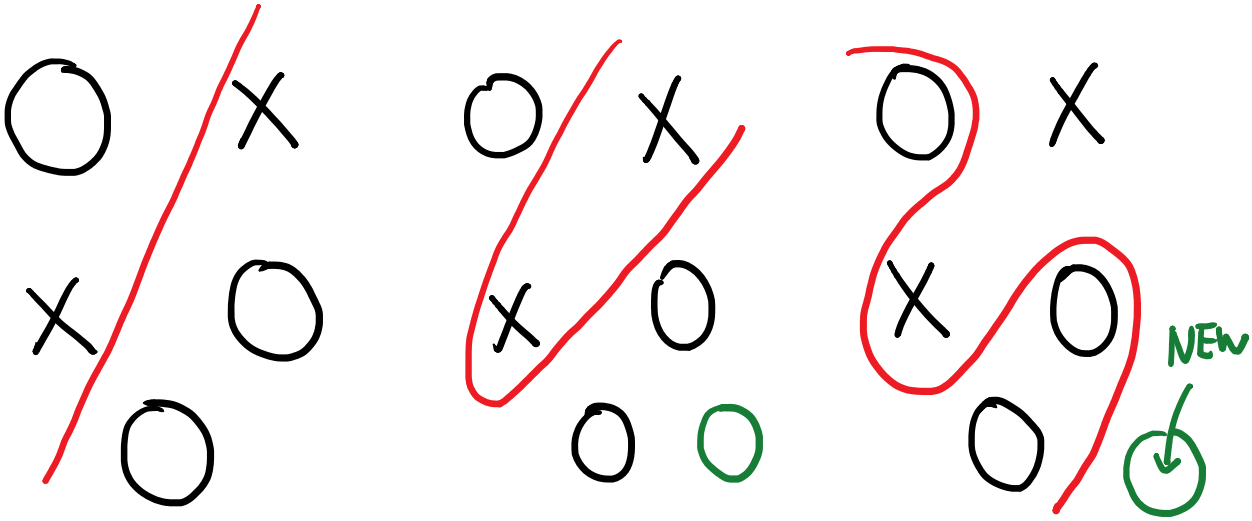
이상적인 훈련?





모델 표현력 = 복잡도 = ...

Capacity



"단순" 과소적합 적합 과적합

UNDERFIT FIT OVERFIT

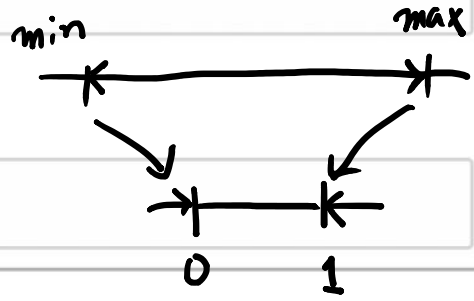
데이터 전처리

```
from sklearn.preprocessing import MinMaxScaler
```

In [232]:

```
scaler = MinMaxScaler()
```

$$x' \leftarrow \frac{X - min}{Max - Min}$$



In [234]:

```
Xmm_train = scaler.fit_transform(X_train)
```

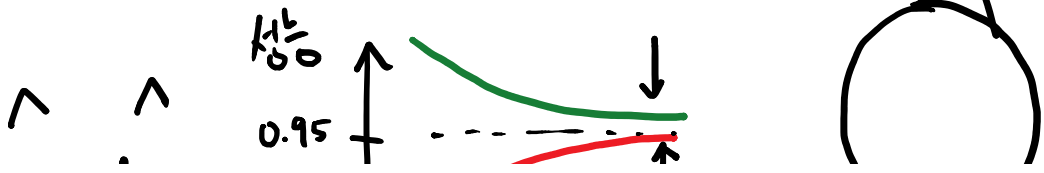
In [235]:

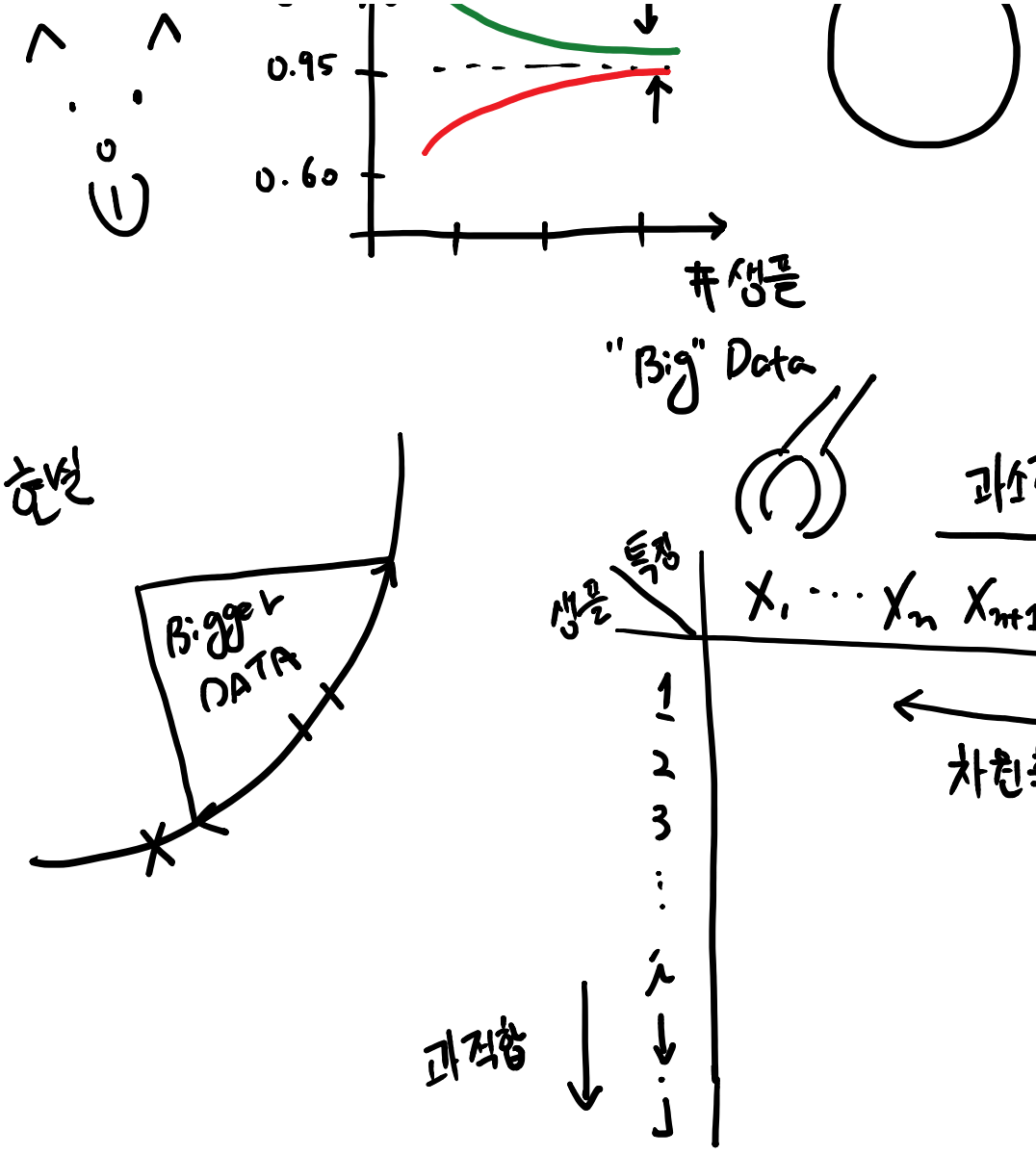
```
pd.DataFrame(Xmm_train)[:5]
```

Out[235]:

	0	1	2	3	4	
0	0.000534	0.000000	0.098889	0.0	0.207469	0.66731

데이터와 모델 표현력




0.95
0.60
샘플
"Big" Data
손실
Bigger DATA
과소적합
특징
샘플
$X_1 \cdots X_n \; X_{n+1} \cdots$
1
2
3
⋮
λ
J
과적합
차원축소

차원축소

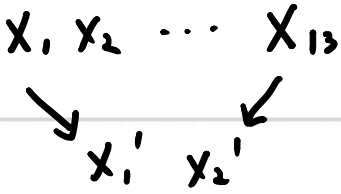
```
from sklearn.decomposition import PCA
```

In [241]:

```
pca = PCA(n_components=2)
```

In [242]:

```
Xpca_train = pca.fit_transform(X_train)
```



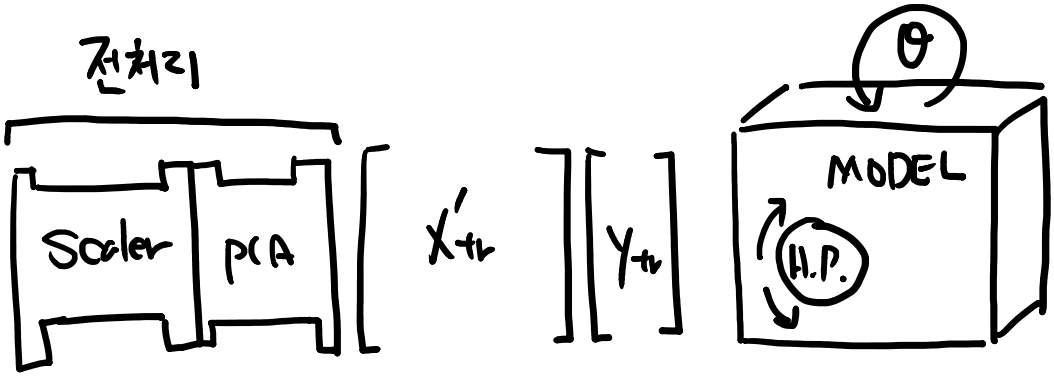
In [243]:

```
X_train.shape
```

Out[243]:

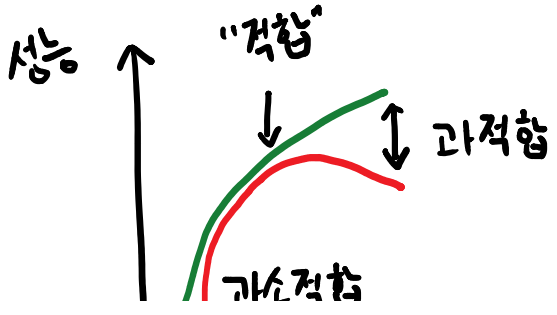
(379, 13) → (379,2)

정리



↓ .predict(Xnew)

$\hat{Y}$



훈련DATA

1. 최적화 ~ 과소적합

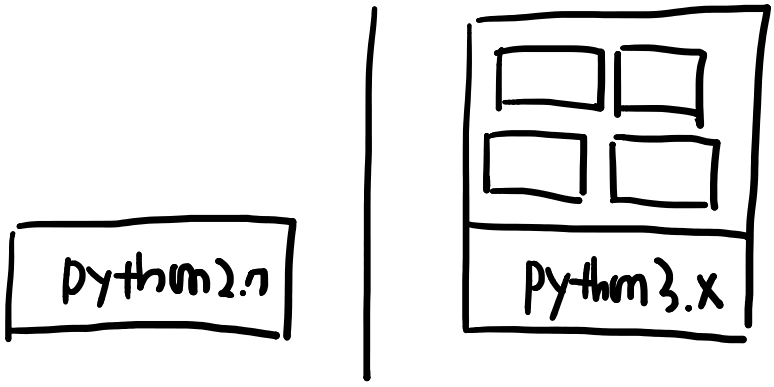
실해하면?

1. 최적화 ~ 과소적합

2. 일반화 → 과적합

/usr/bin/python → python2.7

/usr/me/anaconda3/python



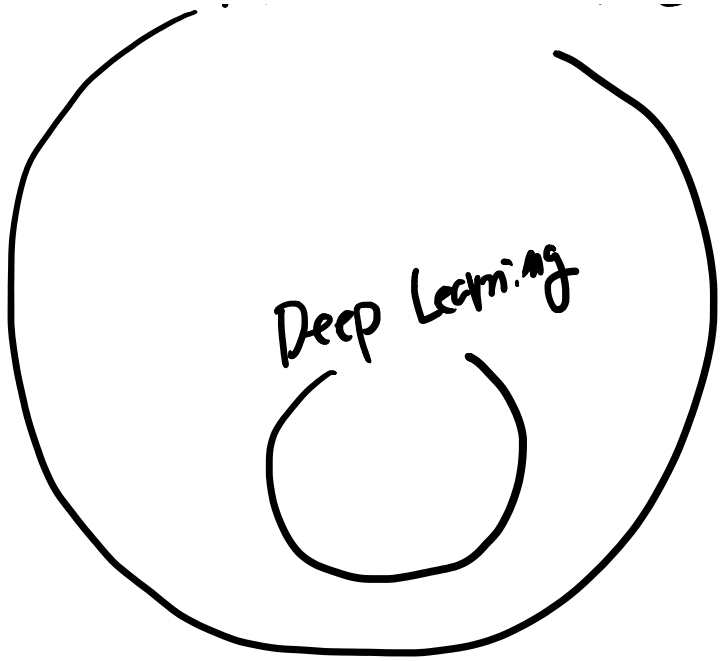
Q. matplotlib 한글?

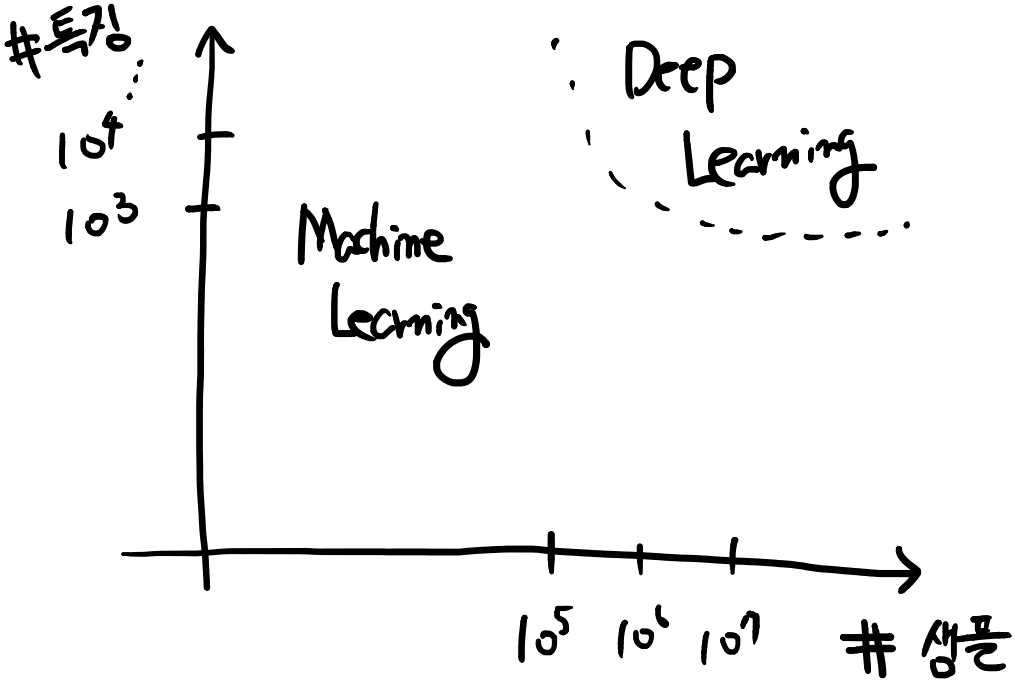
matplotlib.rcParams['font-family'] = 한글글씨체

(rc) 환경설정

Q. A.I. Vs Machine Learning Vs. Deep Learning

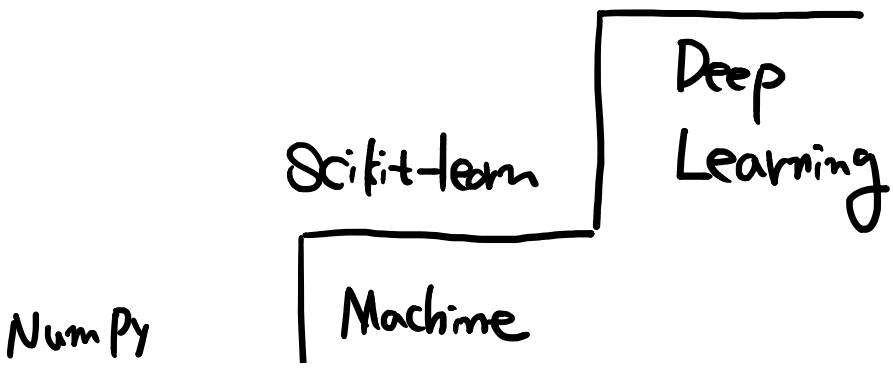
A.I. == Machine Learning

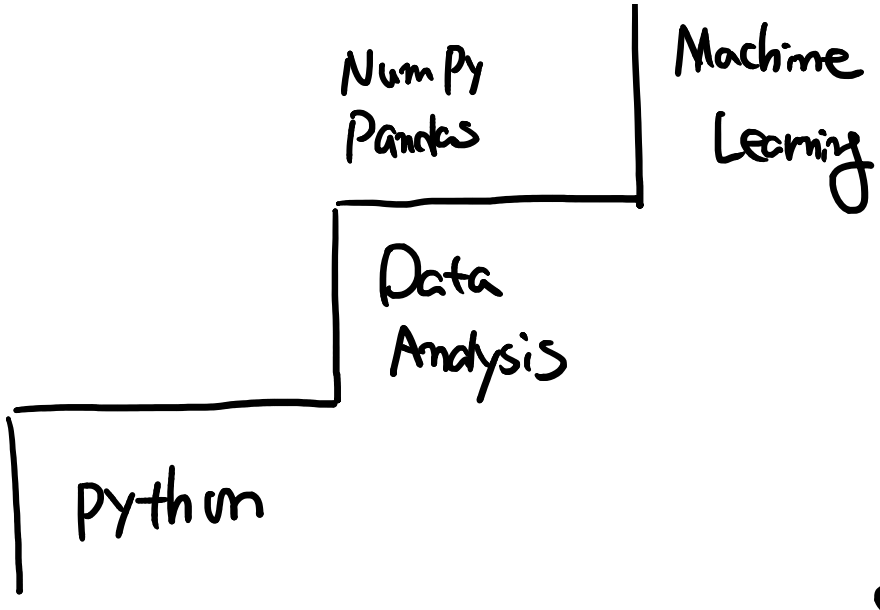




"텐크트리"

TensorFlow+keras ...





Scikit-Learning