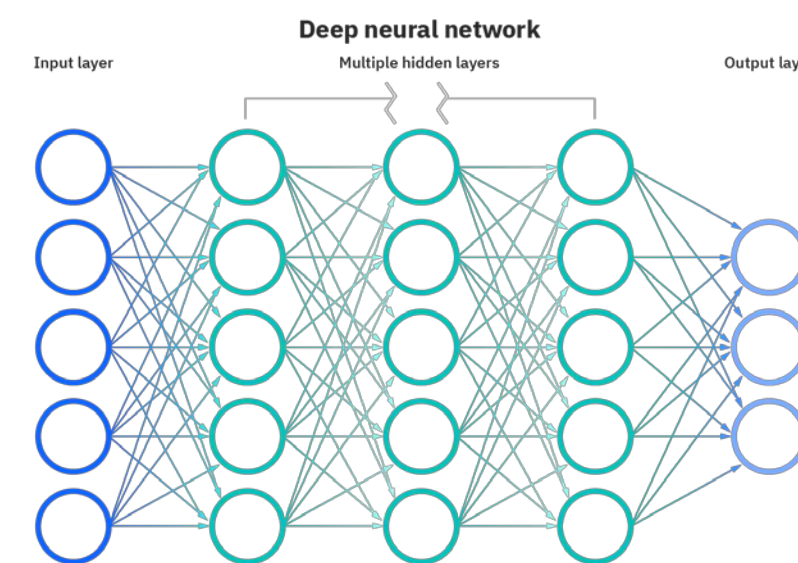


# 딥러닝 입문

## Deep Learning Primer



**서지혜** 교수

서울과학기술대학교•경영대학

[jihae@seoultech.ac.kr](mailto:jihae@seoultech.ac.kr)



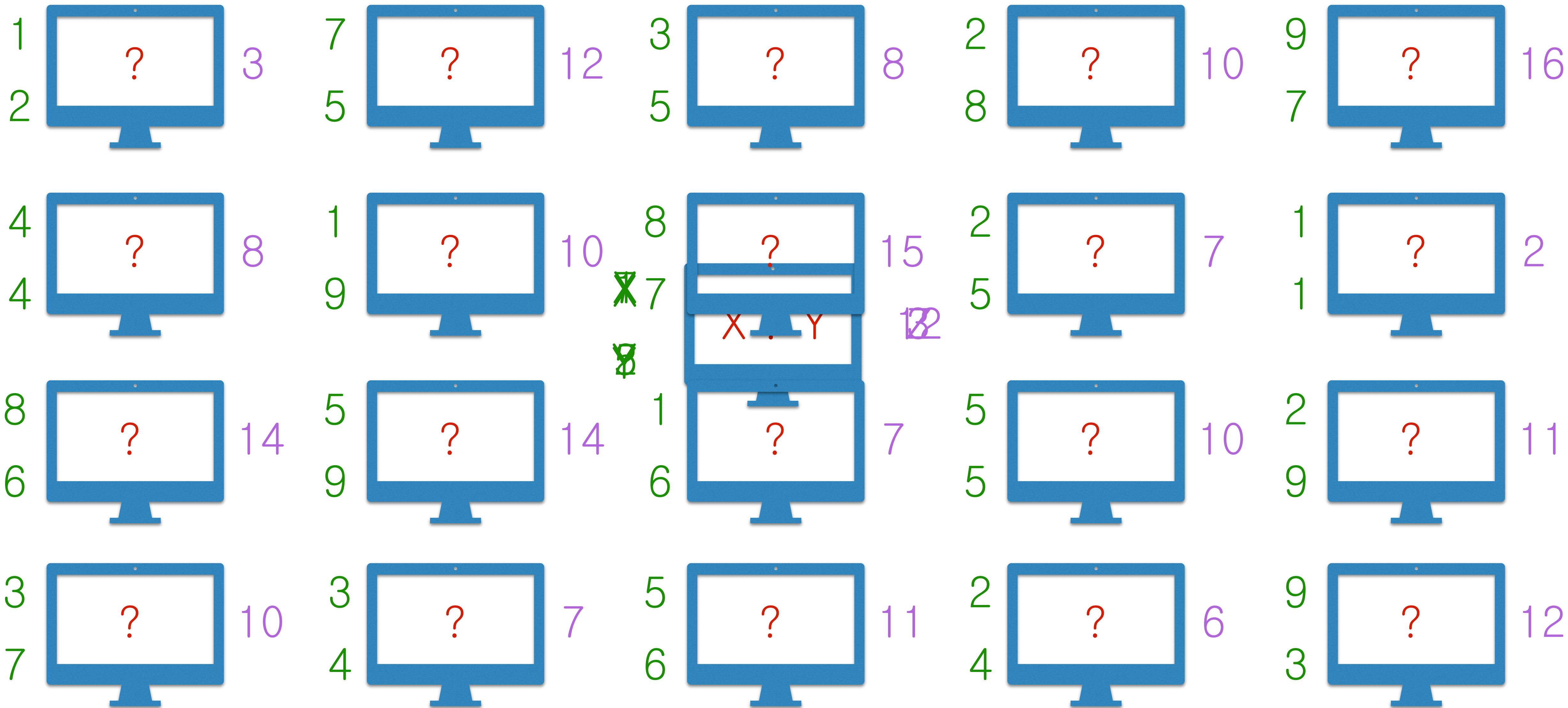
# 개요

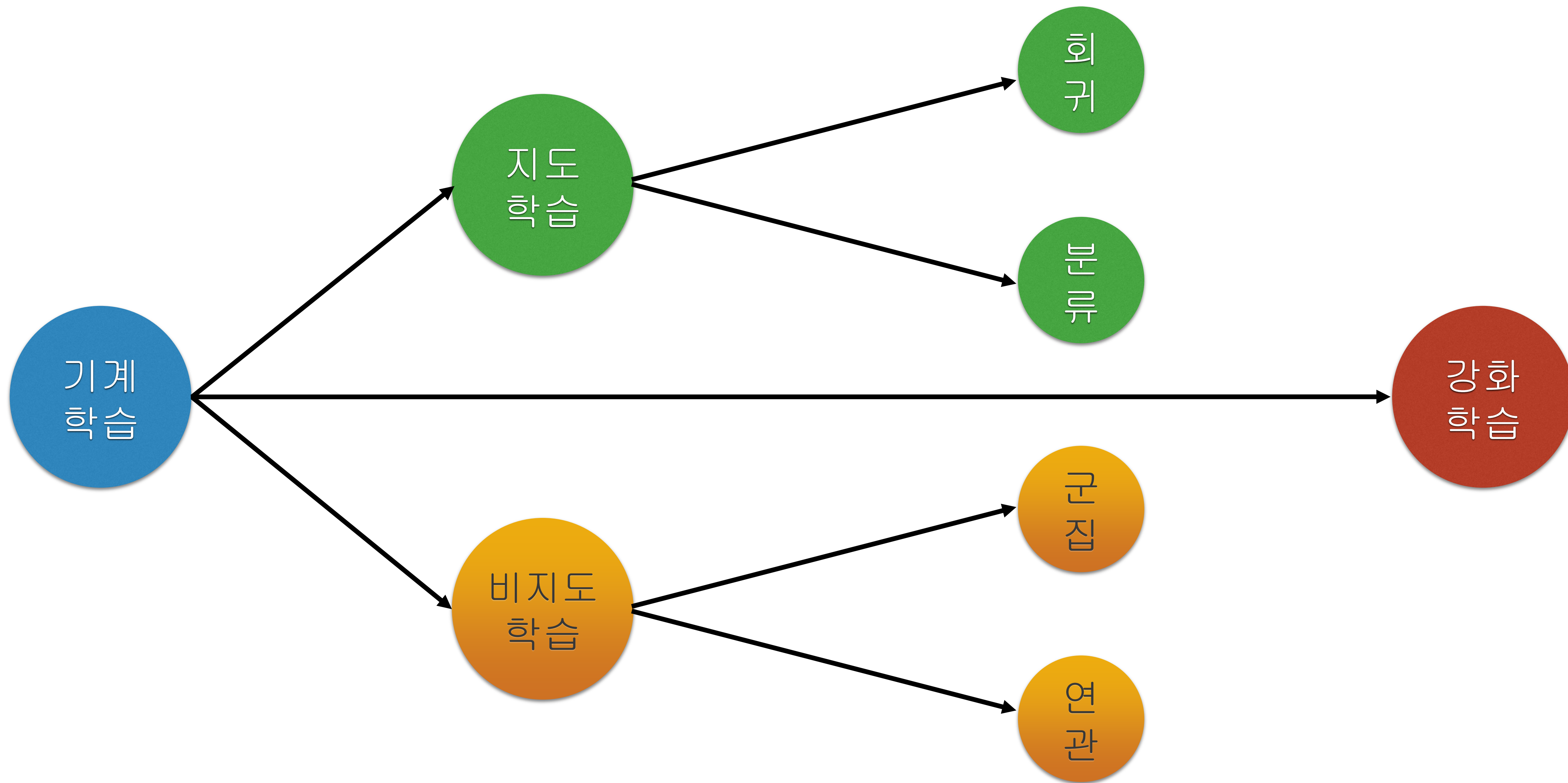
---

## Overview



# 기계학습이란?





선형  
회귀

의사  
결정  
나무

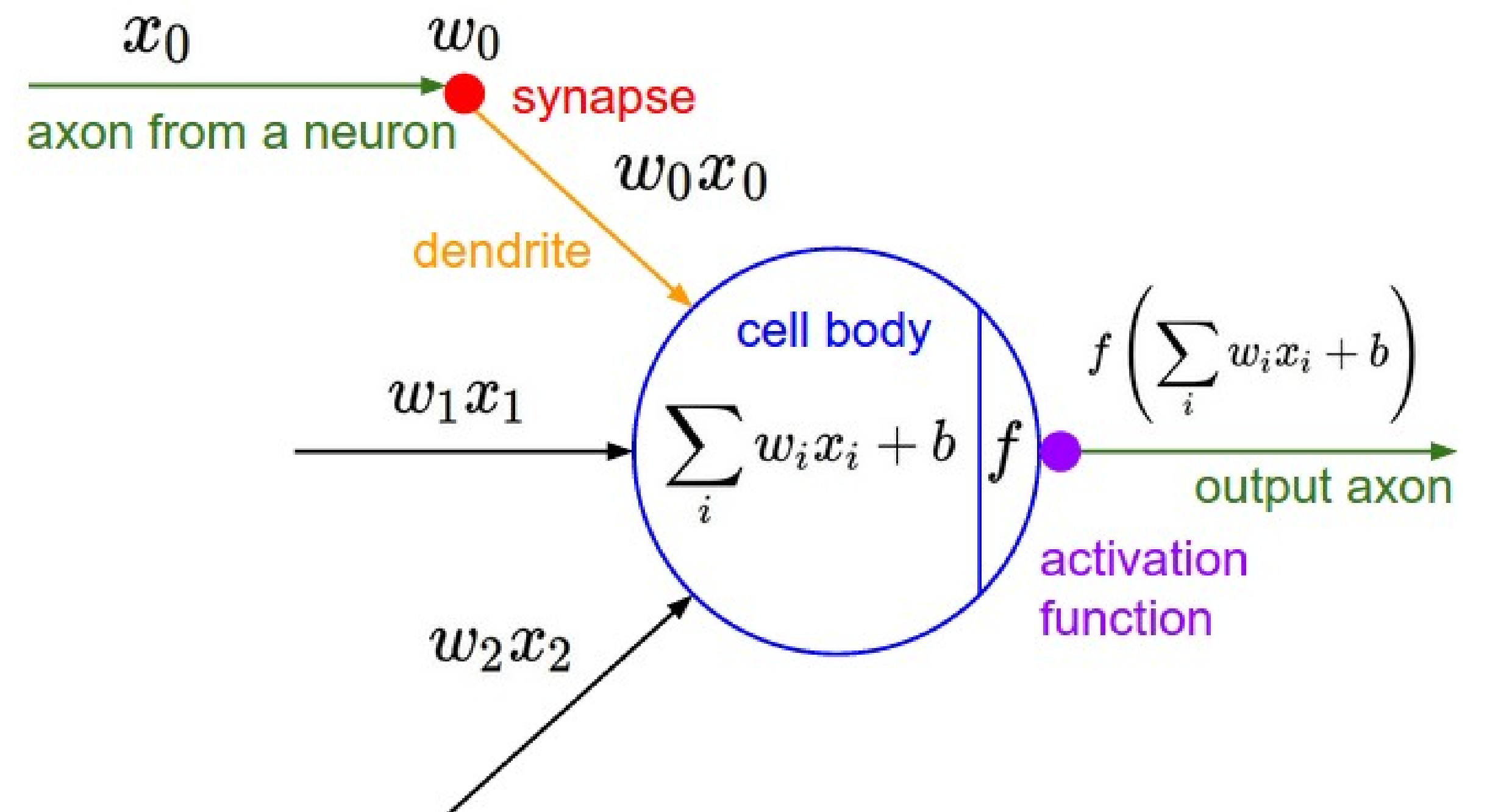
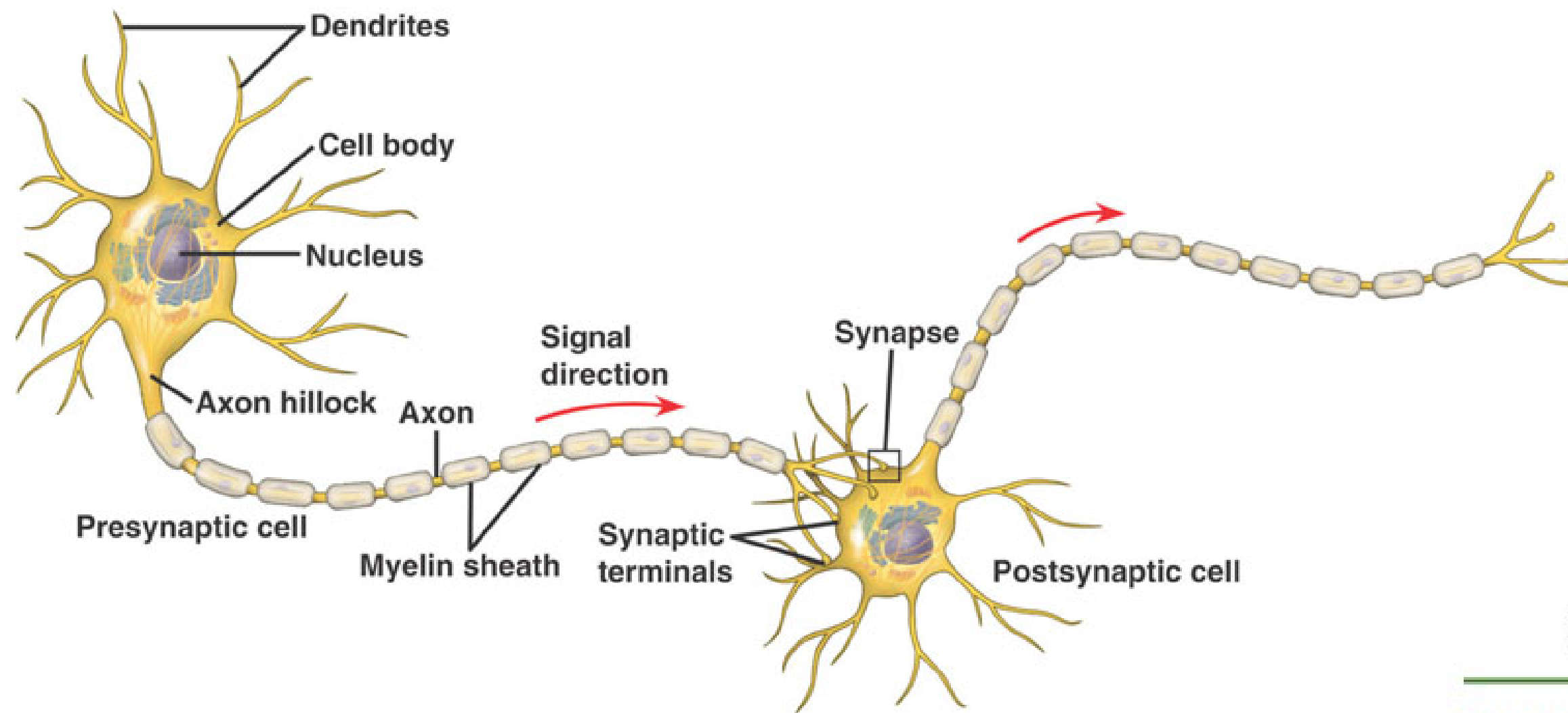
랜덤  
포레스트

최근접  
이웃

서포트  
벡터  
머신

인공  
신경망





# 데이터에 대한 이해

---

Data, Data, Data



**열(column)**

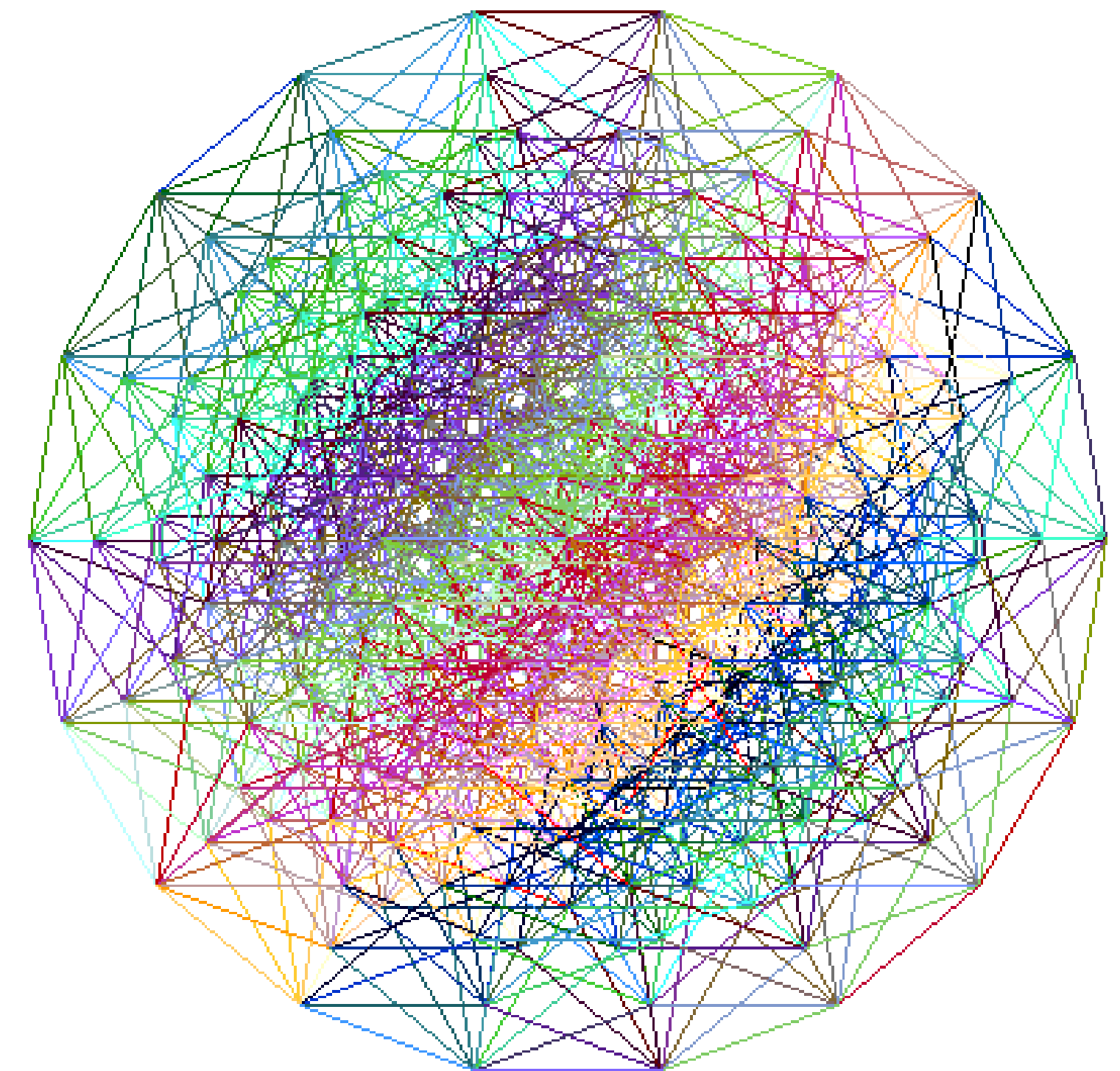
특징(feature)  
변수(variable)  
속성(attribute)  
필드(field)

평균 공부 시간 (1주일)	평균 수면 시간 (1일)	학점 (4.5 만점)	...	$N$
13	7.5	3.1	...	...
4	6	2.45	...	...
19	6	4.12	...	...
7	8	1.9	...	...
15	7.5	2.29	...	...
4	7	1.7	...	...
11	7	4.32	...	...
10	4	3.53	...	...

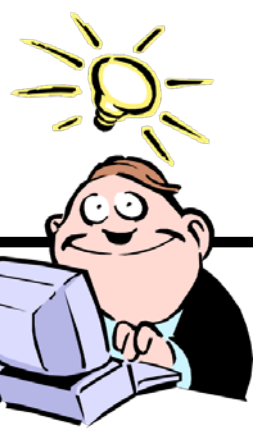
**행(row)**

관측치(observation)  
인스턴스(instance)  
레코드(record)

$N$  개 열(특징, 변수) =  $N$  차원  
관측치 =  $N$  차원 공간 상의 한 점







독립변수

종속변수

평균 공부 시간 (1주일)	평균 수면 시간 (1일)	학점 (4.5 만점)
13	7.5	3.1
4	6	2.45
19	6	4.12
7	8	1.9
15	7.5	2.29
4	7	1.7
11	7	4.32
10	4	3.53

# 독립변수와 종속변수를 분리한다.

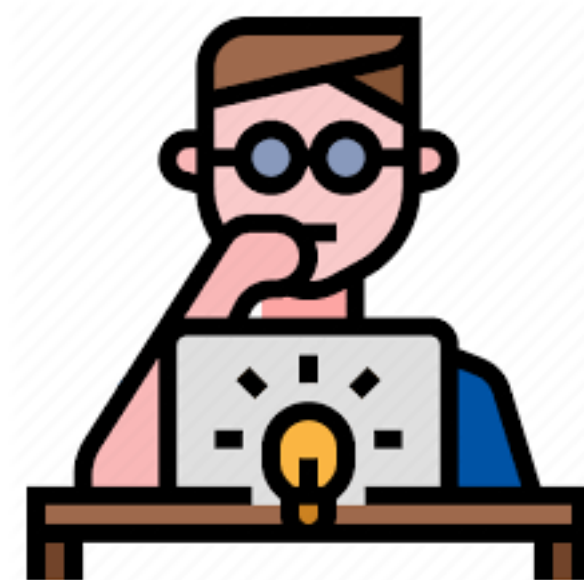
X = df[['평균공부시간(1주일)', '평균수면시간(1일)']]

y = df[['학점(4.5만점)']]

# 실습을 통한 기계학습/ 딥러닝 이해

---

(Deep) Learning by Doing for Dummies



독립 변수 종속 변수

평균 공부 시간 (1주일)	평균 수면 시간 (1일)	학점 (4.5 만점)
13	7.5	3.1
4	6	2.45
19	6	4.12
7	8	1.9
15	7.5	2.29
4	7	1.7
11	7	4.32
10	4	3.53

1. 데이터 준비

3. 모델 학습(fit)

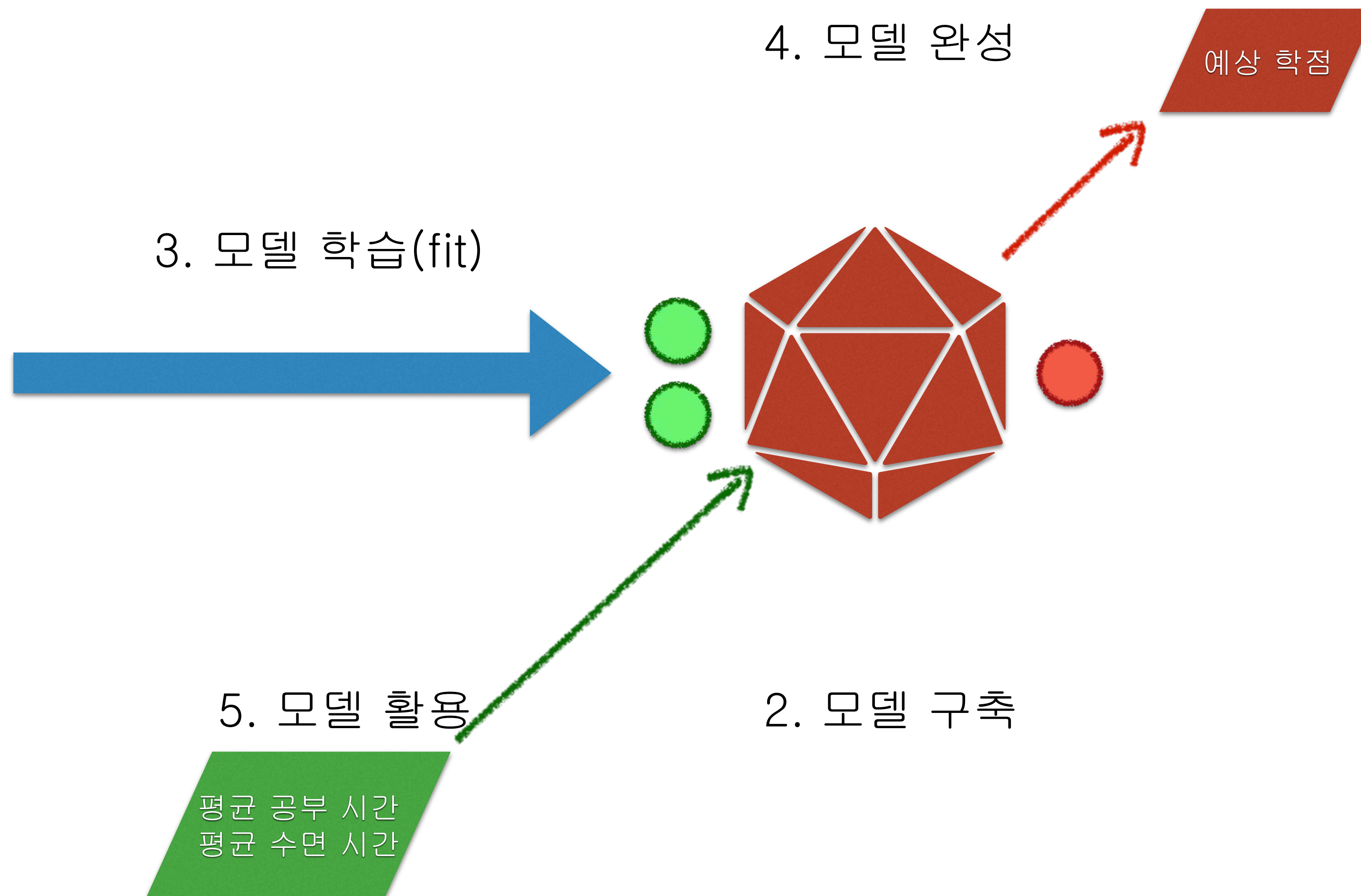
4. 모델 완성

예상 학점

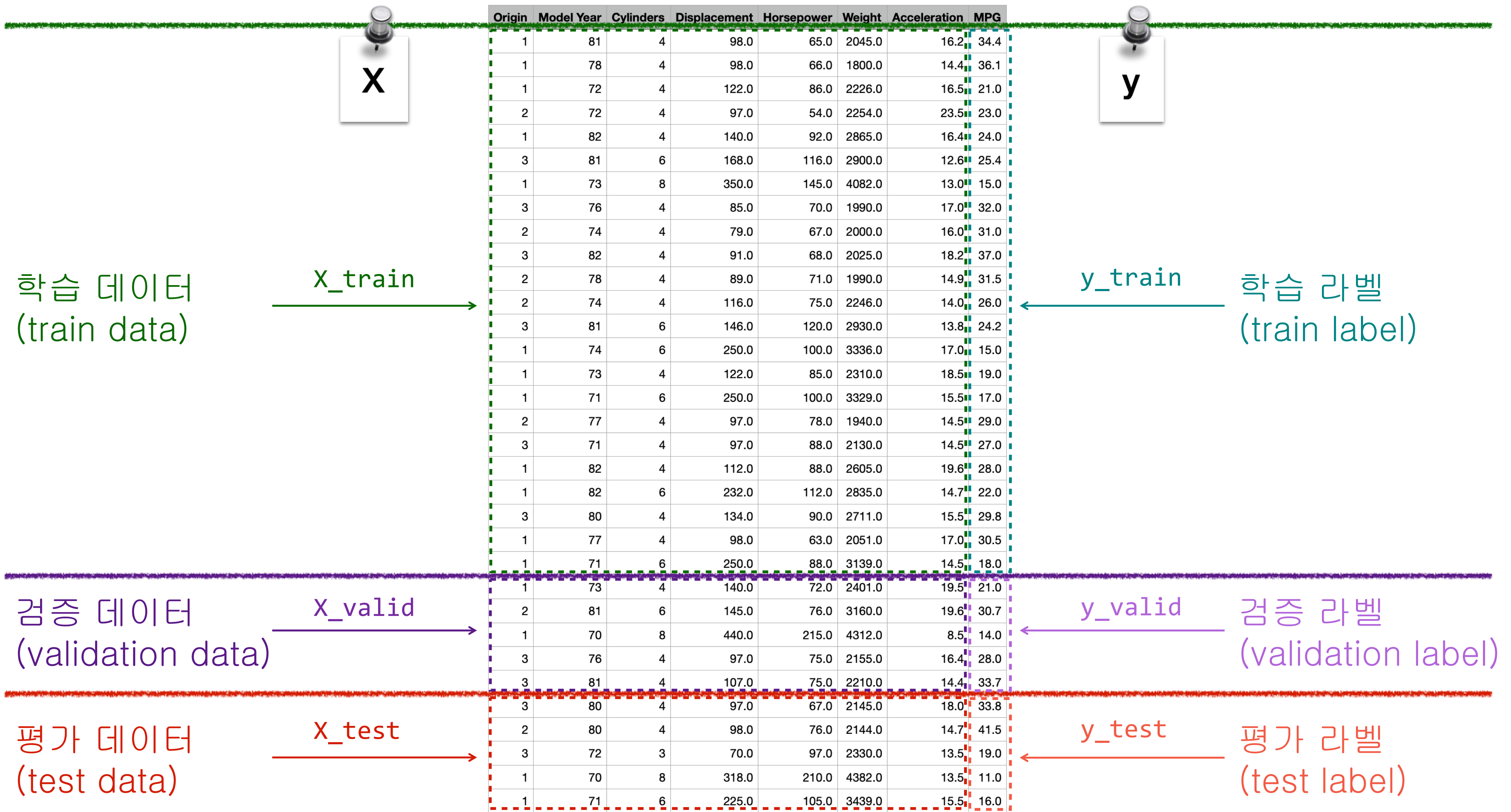
5. 모델 활용

평균 공부 시간  
평균 수면 시간

2. 모델 구축



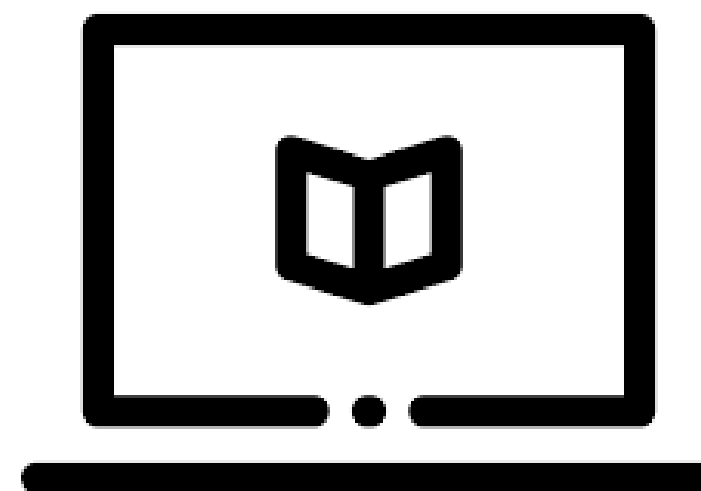
# 지도학습에 사용하는 데이터 분리하기





# 회귀 문제

Lab Exercises





순차적 모델(Sequential API)

기능적 모델(Functional API)

사용자 정의 모델(Custom Model)



## 1. 데이터 준비

```
# --- 데이터를 가져온다.
import pandas
df = pandas.read_csv(URL, encoding='utf-8')
```

```
# --- 학습 데이터, 검증 데이터, 평가 데이터로 나눈다.
from sklearn.model_selection import train_test_split
train_df, test_df = train_test_split(df, test_size=0.1)
train_df, valid_df = train_test_split(train_df, test_size=0.2)
```

```
# --- 독립변수와 종속변수를 분리한다.
cols = ['평균공부시간(1주일)', '평균수면시간(1일)', '학점(4.5만점)']
# 학습 데이터
X_train, y_train = train_df[cols[:-1]], train_df[cols[-1:]]
# 검증 데이터
X_valid, y_valid = valid_df[cols[:-1]], valid_df[cols[-1:]]
# 평가 데이터
X_test, y_test = test_df[cols[:-1]], test_df[cols[-1:]])
```

## 4. 모델 완성

```
# 학습한 모델을 평가한다.
model.evaluate(X_test, y_test)
```

```
# 학습한 모델을 저장한다.
model.save('모델이름')
```

## 2. 모델 구축

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
inputs = Input(shape=(2,))
output = Dense(1)(inputs)
model = Model(inputs, output)
```

```
# 모델 학습 과정을 정의한다.
from tensorflow.keras.losses import MeanSquaredError
model.compile(loss=MeanSquaredError())
```

## 3. 모델 학습(fit)

```
model.fit(X_train, y_train,
          validation_data=(X_valid, y_valid),
          epochs=1000)
```

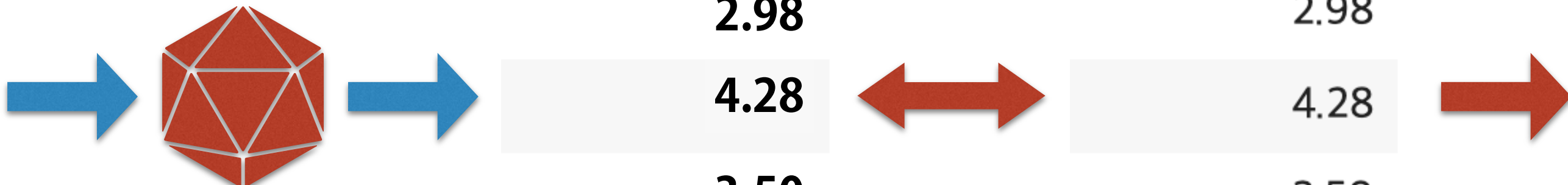
## 5. 모델 활용

```
# 일주일 평균 10시간을 공부하고 하루 평균 7시간을 잔다면 내 성적은?
print(model.predict([[10, 7]]))
```



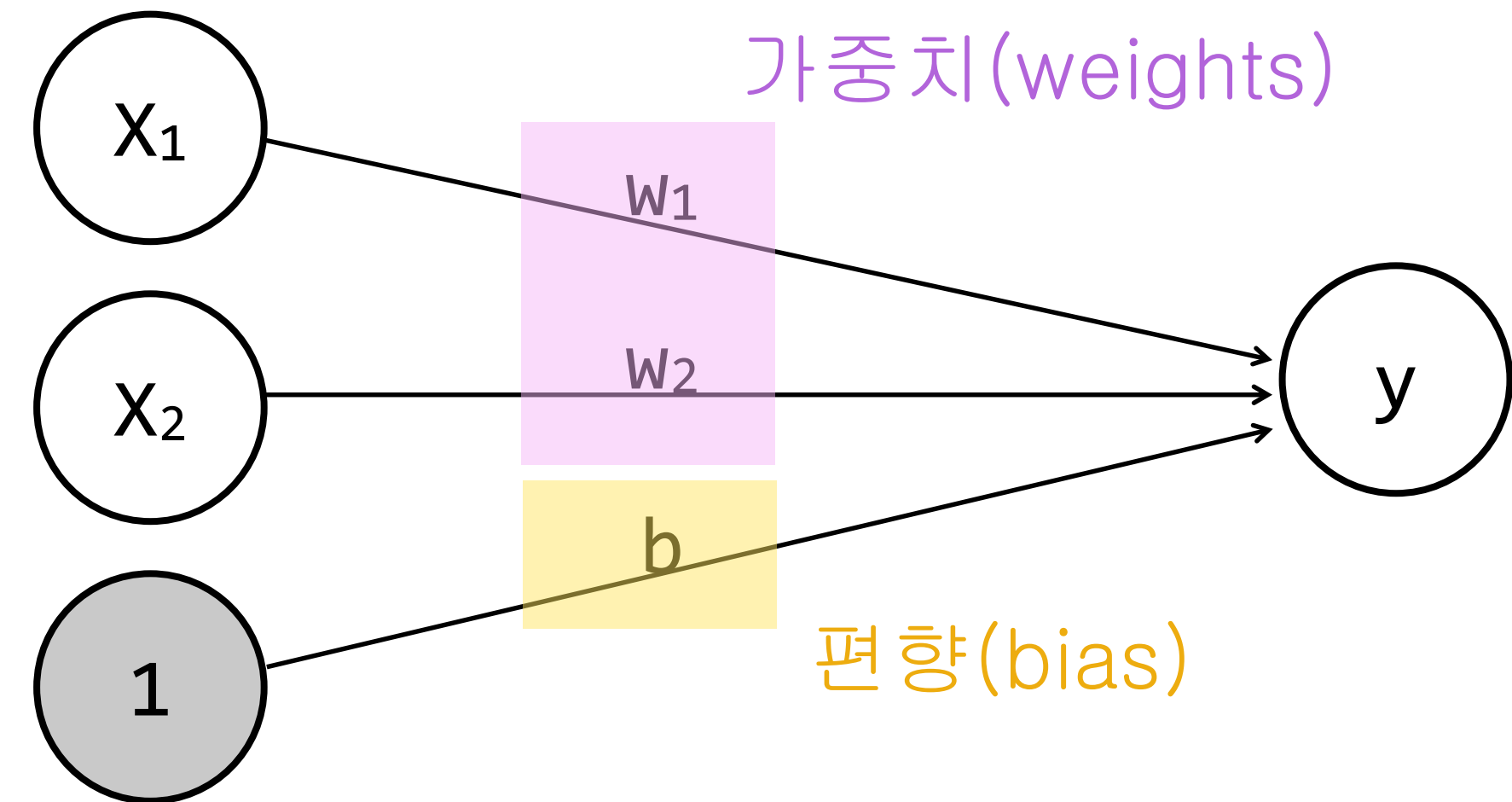
# 모델 학습에 대한 이해 : Loss

Epoch 1/10			
49/49	[=====]	- 0s 4ms/step	- loss: 236.4532 - val_loss: 215.9012
Epoch 2/10			
49/49	[=====]	- 0s 1ms/step	- loss: 213.0675 - val_loss: 194.5170
Epoch 3/10			
49/49	[=====]	- 0s 6ms/step	- loss: 191.2497 - val_loss: 174.3265
Epoch 4/10			
49/49	[=====]	- 0s 2ms/step	- loss: 171.2926 - val_loss: 155.5755
Epoch 5/10			
49/49	[=====]	- 0s 2ms/step	- loss: 152.1404 - val_loss: 138.3329

평균공부시간 (1주일)	평균수면시간 (1일)		예측 학점	학점 (4.5만점)	(예측 - 실제값) <sup>2</sup>
9.0	7.0		3.60	3.60	0.0000
0.0	6.5		2.98	2.98	0.0000
8.0	7.0		4.28	4.28	0.0000
3.0	5.0		3.50	3.50	0.0000
6.0	6.0		2.93	2.93	0.0000

평균: 0

평균공부시간 (1주일)	평균수면시간 (1일)		예측 학점
9.0	7.0	➔	3.60
0.0	6.5		2.98
8.0	7.0		4.28
3.0	5.0		3.50
6.0	6.0		2.93



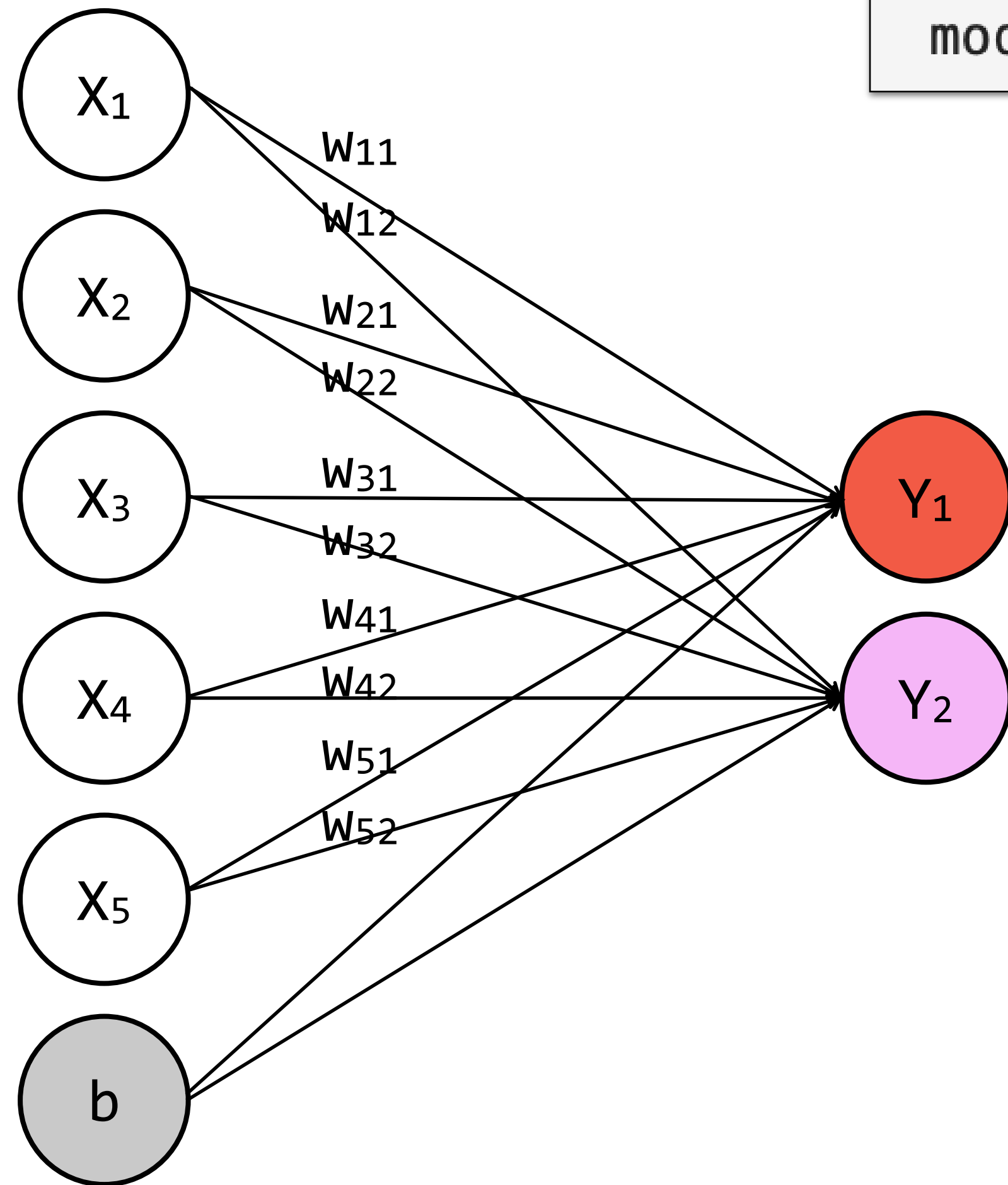
```

inputs = Input(shape=(2,))
output = Dense(1)(inputs)
model = Model(inputs, output)
  
```

$$y = w_1 X_1 + w_2 X_2 + b$$

만약 데이터가 독립변수가 5개이고 종속변수가 2개라고 가정한다면 모델을 어떻게 만들어야 할까?

```
inputs = Input(shape=(5,))  
output = Dense(2)(inputs)  
model = Model(inputs, output)
```



$$Y_1 = w_{11}X_1 + w_{21}X_2 + w_{31}X_3 + w_{41}X_4 + w_{51}X_5 + b$$

$$Y_2 = w_{12}X_1 + w_{22}X_2 + w_{32}X_3 + w_{42}X_4 + w_{52}X_5 + b$$



# 원-핫 인코딩(one-hot encoding)

평균공부시간(1주일)	평균수면시간(1일)	학점
1	7	D
5	6	B
3	8	C
7	7	B
12	6	A

```
df = pandas.get_dummies(df)
```

평균공부시간(1주일)	평균수면시간(1일)	학점_A	학점_B	학점_C	학점_D
1	7	0	0	0	0
5	6	0	0	0	0
3	8	0	0	0	0
7	7	0	0	0	0
12	6	0	0	0	0

- 평균 제곱 오차(mean squared error, MSE)

- 회귀 문제에서 자주 사용하는 손실 함수
  - [참고] 분류 문제에서 사용하는 손실 함수와 다름
- [https://en.wikipedia.org/wiki/Mean\\_squared\\_error](https://en.wikipedia.org/wiki/Mean_squared_error)

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

- 평균 절대 오차(mean absolute error, MAE)

- 많이 사용하는 회귀 지표
  - [참고] 분류 문제에서 사용하는 평가 지표와 다름
- [https://en.wikipedia.org/wiki/Mean\\_absolute\\_error](https://en.wikipedia.org/wiki/Mean_absolute_error)

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

- 정규화

- 수치 입력 데이터의 특징(열, 변수)이 여러 가지 범위를 가질 때 동일한 범위가 되도록 각 특징의 스케일을 독립적으로 조정해야 함

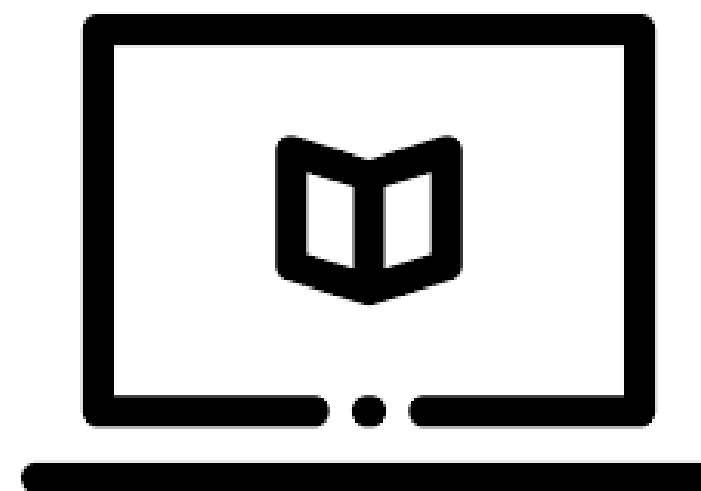
- 과적합

- 학습 데이터가 많지 않다면 과적합을 피하기 위해 은닉층의 개수가 적은 소규모 네트워크를 선택하는 방법이 좋음
- 조기 종료(Early stopping)은 과적합을 방지하기 위한 좋은 방법



# 분류 문제

Lab Exercises



종속 변수
성적(GAP)
주택 가격
주식 가격
매출 금액

연속형



회귀  
regression

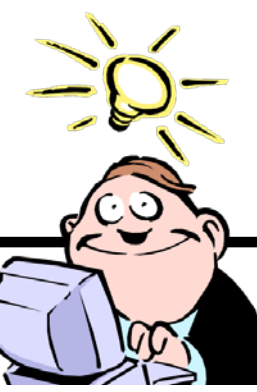
종속 변수
성적('A', 'B' , 'C', 'D' )
합격 여부
꽃의 품종
사물의 종류

범주형



분류  
classification





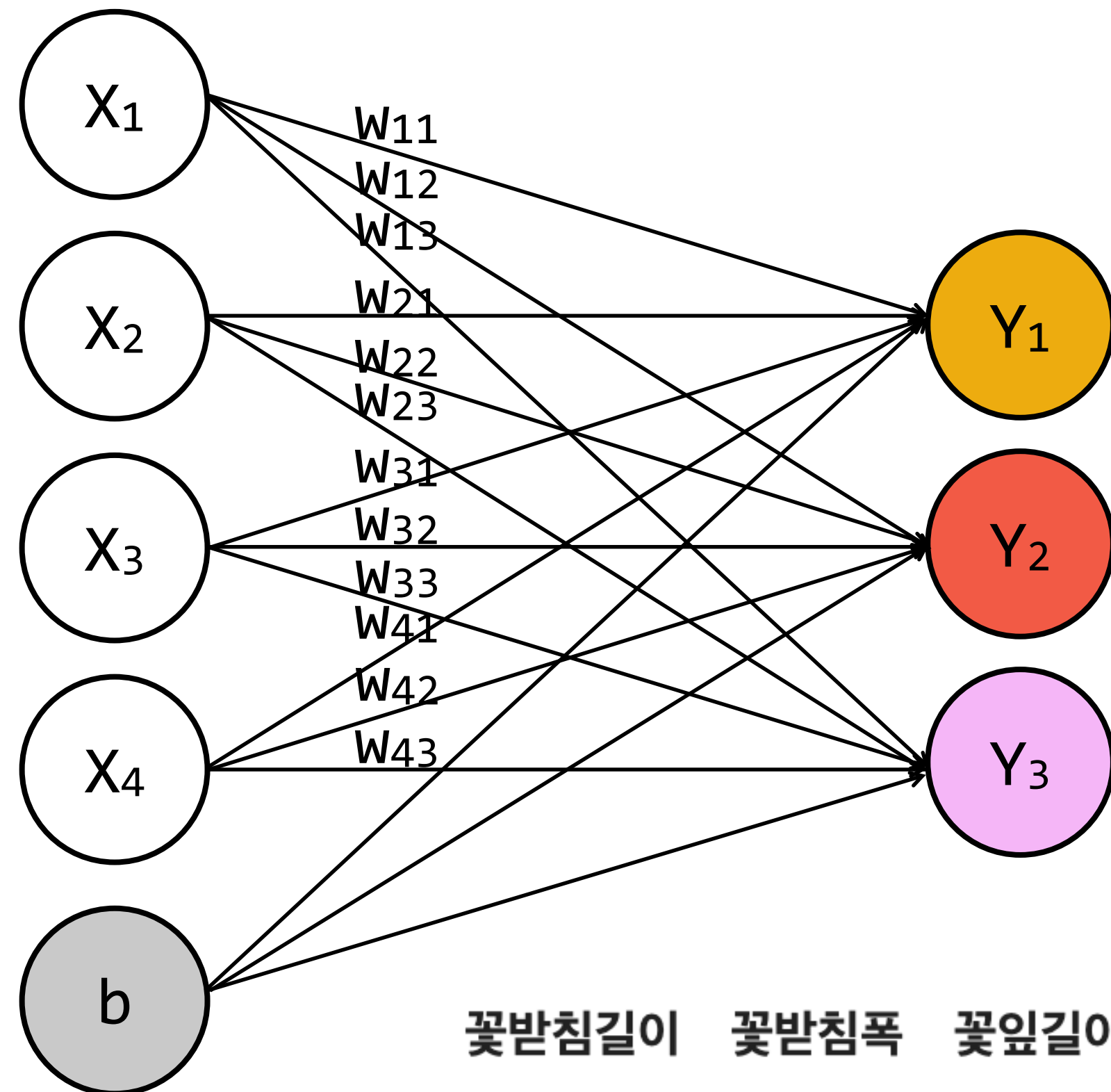
```
from tensorflow.keras.losses import CategoricalCrossentropy
```

```
inputs = Input(shape=(4,))
outputs = Dense(3, activation=softmax)(inputs)
model = Model(inputs, outputs)
```

```
model.compile(loss=CategoricalCrossentropy())
```

```
# 꽃받침 길이가 5.8cm, 꽃받침 폭이 2.7cm,
# 꽃잎 길이가 3.9cm, 꽃잎 폭이 1.2 cm인
# 붓꽃의 품종은?
y = model.predict([[5.8, 2.7, 3.9, 1.2]])
```





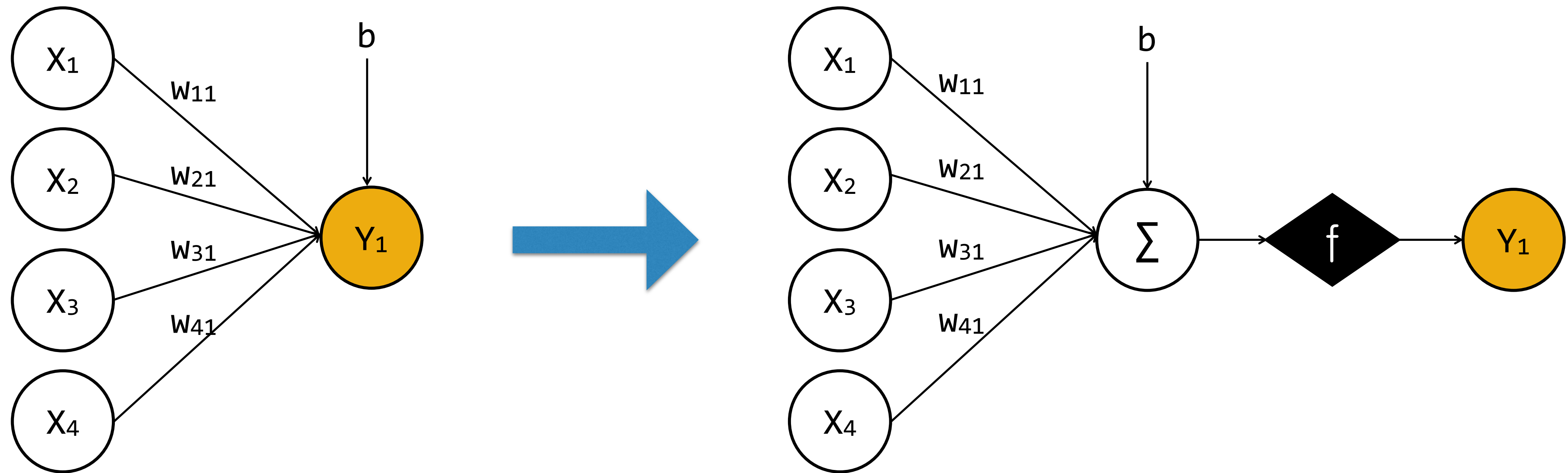
```
inputs = Input(shape=(4,))
outputs = Dense(3, activation=softmax)(inputs)
model = Model(inputs, outputs)
model.compile(loss=CategoricalCrossentropy())
```

$$Y_1 = \text{softmax}(w_{11}X_1 + w_{21}X_2 + w_{31}X_3 + w_{41}X_4 + b)$$

$$Y_2 = \text{softmax}(w_{12}X_1 + w_{22}X_2 + w_{32}X_3 + w_{42}X_4 + b)$$

$$Y_3 = \text{softmax}(w_{13}X_1 + w_{23}X_2 + w_{33}X_3 + w_{43}X_4 + b)$$

꽃받침길이	꽃받침폭	꽃잎길이	꽃잎폭	품종_Iris-setosa	품종_Iris-versicolor	품종_Iris-virginica
5.8	2.7	3.9	1.2	0	0.9	0.1
6.1	2.6	5.6	1.4	0	0.3	0.7
5.8	2.8	5.1	2.4	0	0.2	0.8
4.4	3.2	1.3	0.2	1	0	0
7.2	3.6	6.1	2.5	0.1	0.3	0.6



$$Y_1 = f(w_{11}X_1 + w_{21}X_2 + w_{31}X_3 + w_{41}X_4 + b)$$

## activation function

- 회귀 모델 : 항등(identity) 함수
- 분류 모델
  - 다중 분류 : 소프트맥스(softmax) 함수
  - 이진 분류 : 시그모이드(sigmoid) 함수

회귀 모델

```
model.compile(loss=MeanSquaredError())
```

분류 모델

```
model.compile(loss=CategoricalCrossentropy())
```

다중 분류 문제 (원-핫 인코딩)

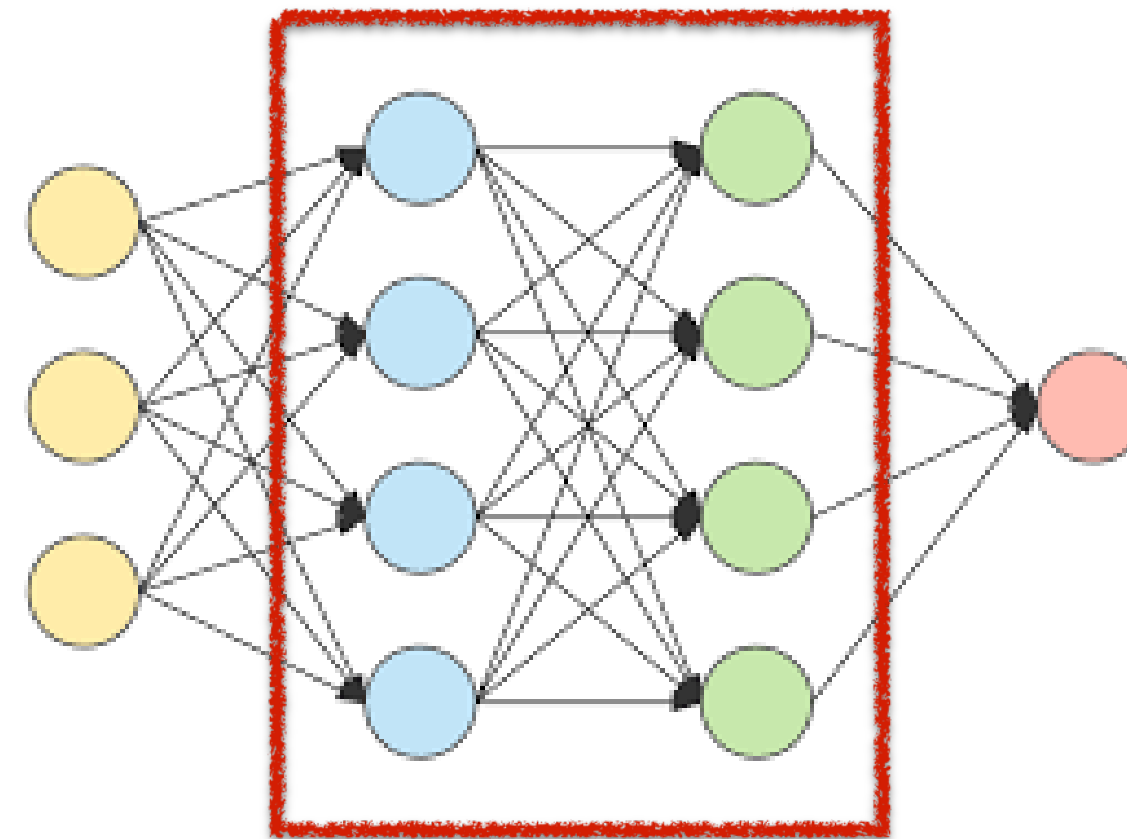
```
model.compile(loss=SparseCategoricalCrossentropy())
```

다중 분류 문제

```
model.compile(loss=BinaryCrossentropy())
```

이진 분류 문제

문제 유형	출력층 활성화 함수	오차 함수
회귀	항등 함수	제곱오차 (MeanSquaredError)
이진 분류	시그모이드(또는 로지스틱 함수) (sigmoid)	교차엔트로피 (BinaryCrossentropy)
다중 분류	소프트맥스 함수 (softmax)	교차엔트로피 (SparseCategoricalCrossentropy)
다중 분류 (one-hot-encoding)	소프트맥스 함수 (softmax)	교차엔트로피 (CategoricalCrossentropy)



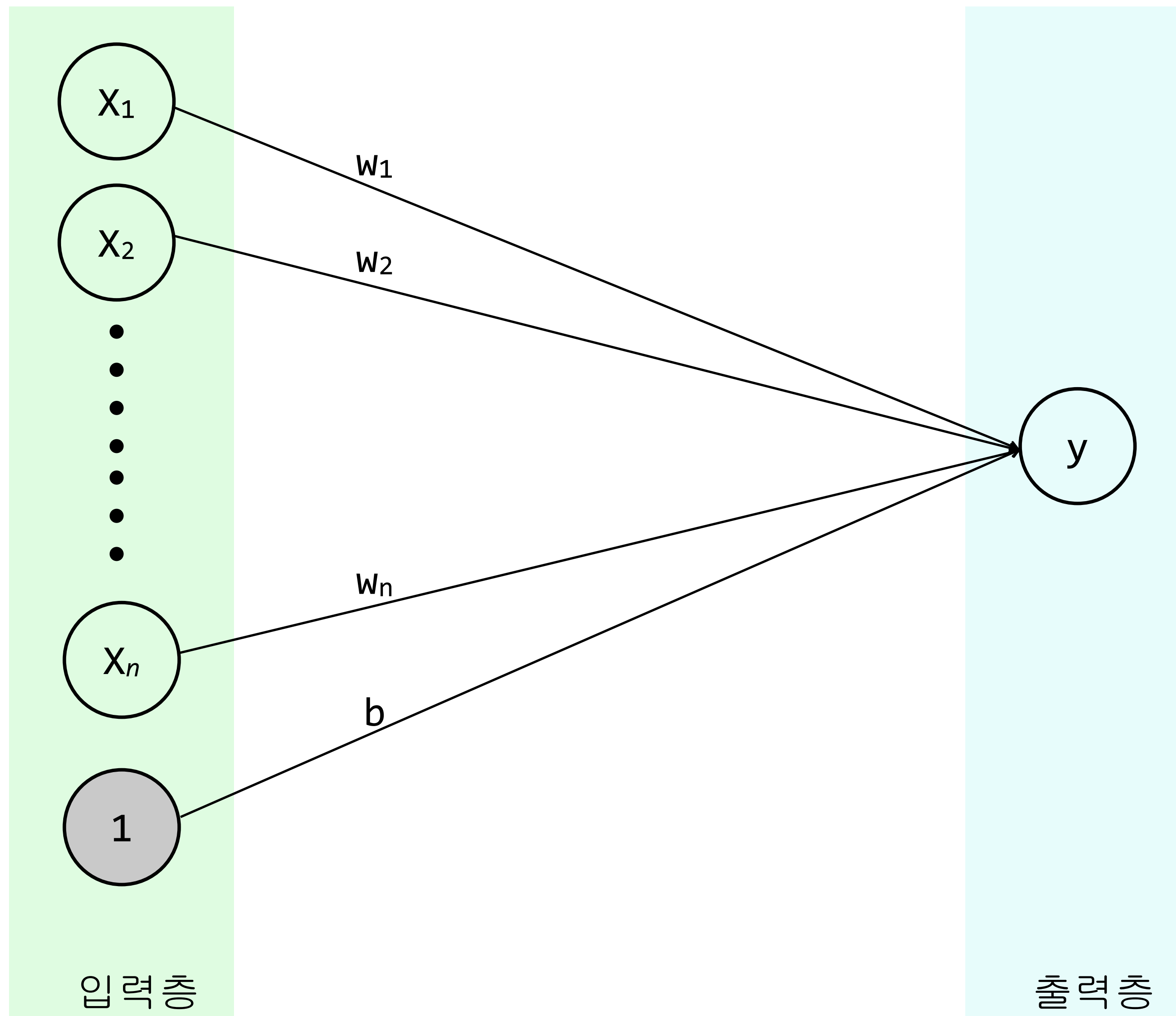
# 은닉층

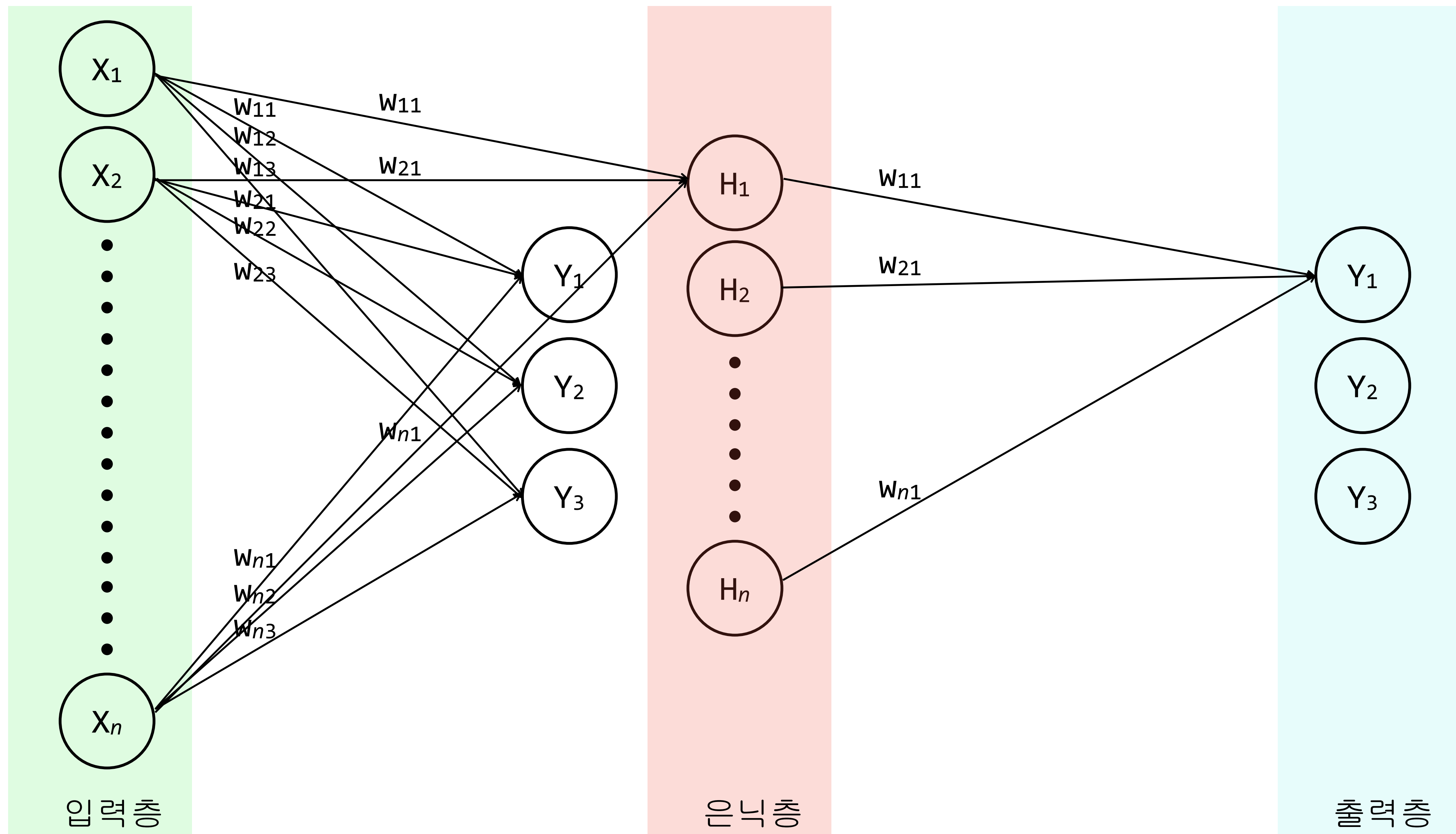
---

## Lab Exercises









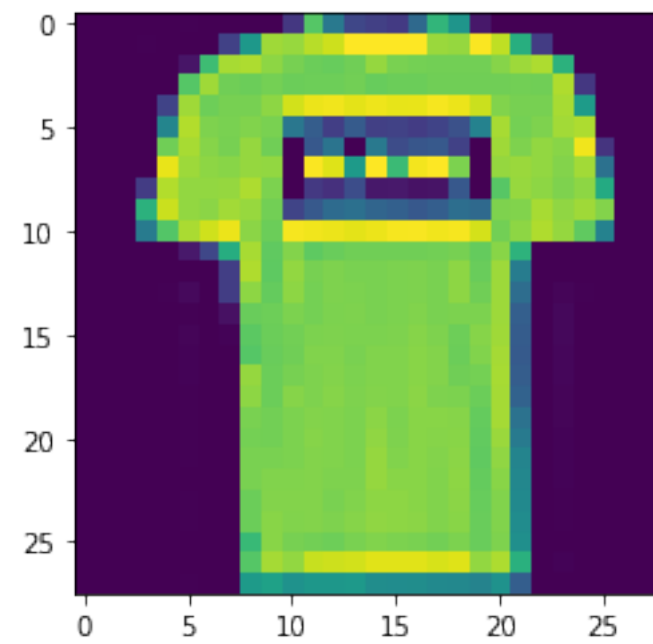
```

from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.activations import relu, softmax

inputs = Input(shape=(64,))
outputs = Dense(32, activation=softmax)(inputs)
hidden = Dense(32, activation=relu)(hidden)
model = Model(inputs=inputs, outputs=outputs)

```

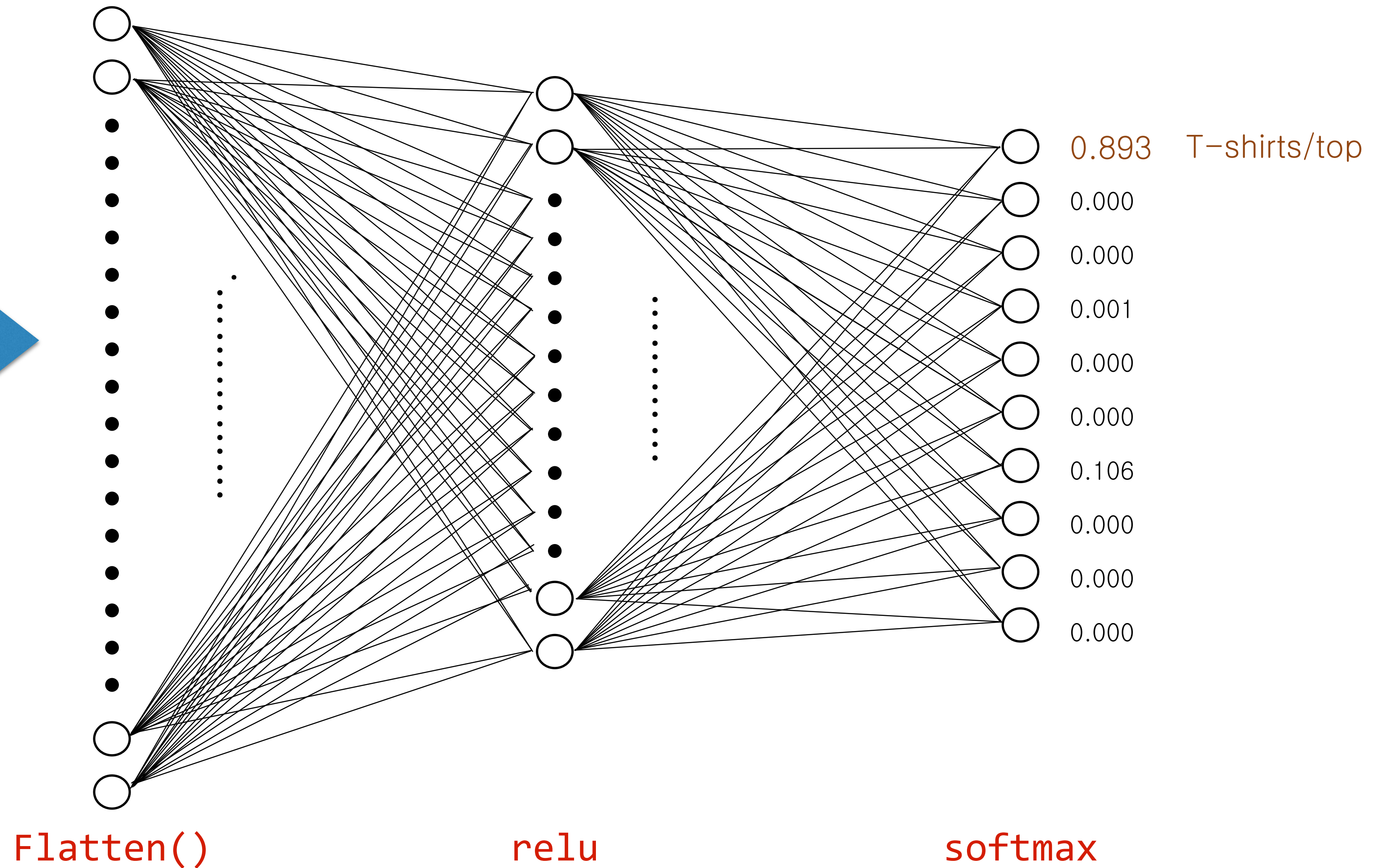
데이터 형상  
(28, 28)



입력층  
(784,)

은닉층  
(32,)

출력층  
(10,)



```

1  from tensorflow.keras.models      import Model
2  from tensorflow.keras.layers      import Input, Flatten, Dense
3  from tensorflow.keras.activations import relu, softmax
4
5  inputs = Input(shape=(28, 28))
6  flatten = Flatten()(inputs)
7  hidden = Dense(32, activation=relu)(flatten)
8  outputs = Dense(10, activation=softmax)(hidden)
9  model = Model(inputs, outputs)
10 model.summary()

```

Model: "model"

Layer (type)	Output Shape	Param #
=====		
input_1 (InputLayer)	[(None, 28, 28)]	0
flatten_1 (Flatten)	(None, 784)	0
dense_1 (Dense)	(None, 32)	25120
dense_2 (Dense)	(None, 10)	330
=====		
Total params: 25,450		
Trainable params: 25,450		
Non-trainable params: 0		





끝

수고 하셨습니다