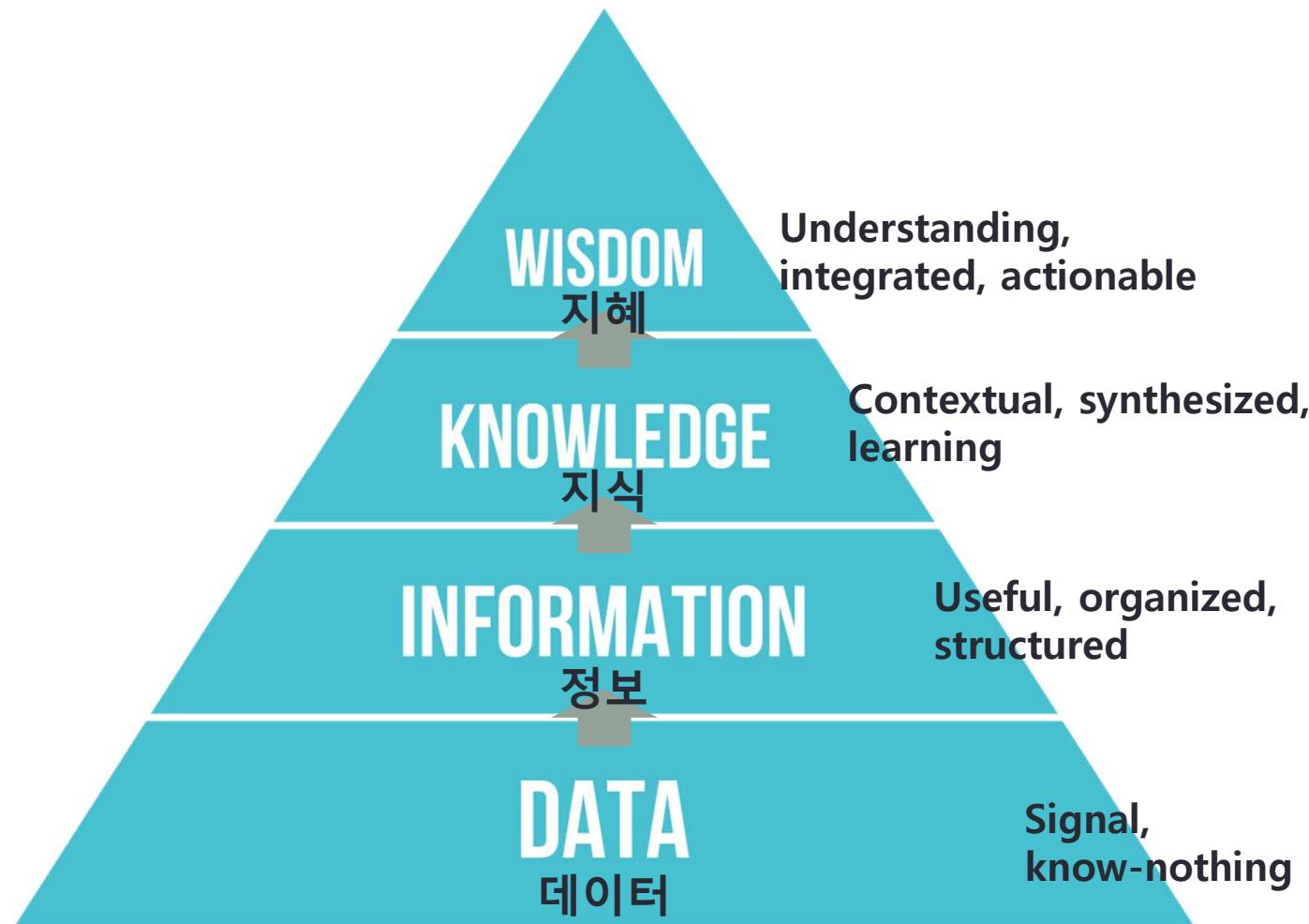


# 데이터사이언스 (DataScience)

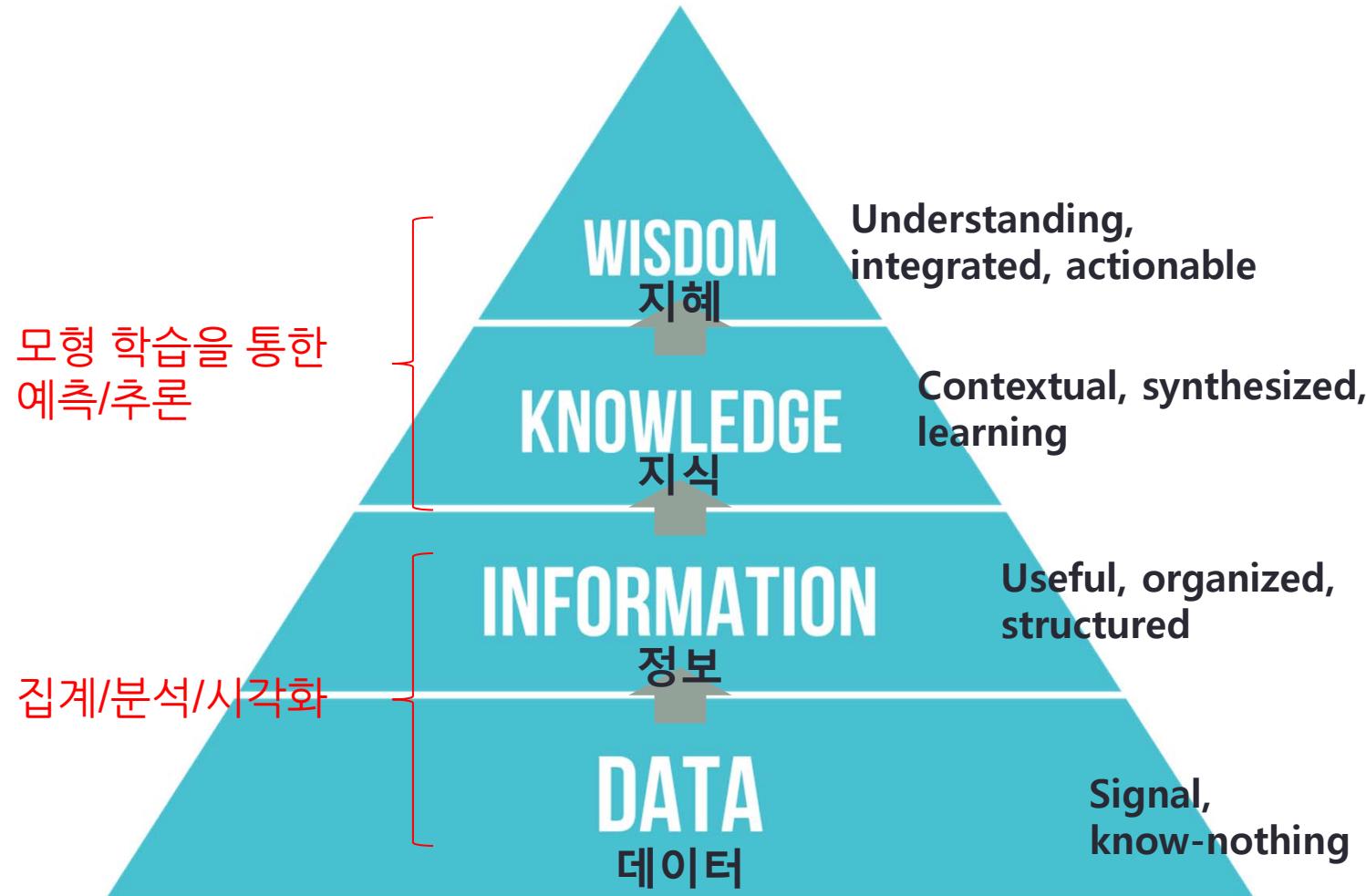
---

시스템경영공학부  
이지환 교수

# 데이터로부터 지혜를 얻기까지..

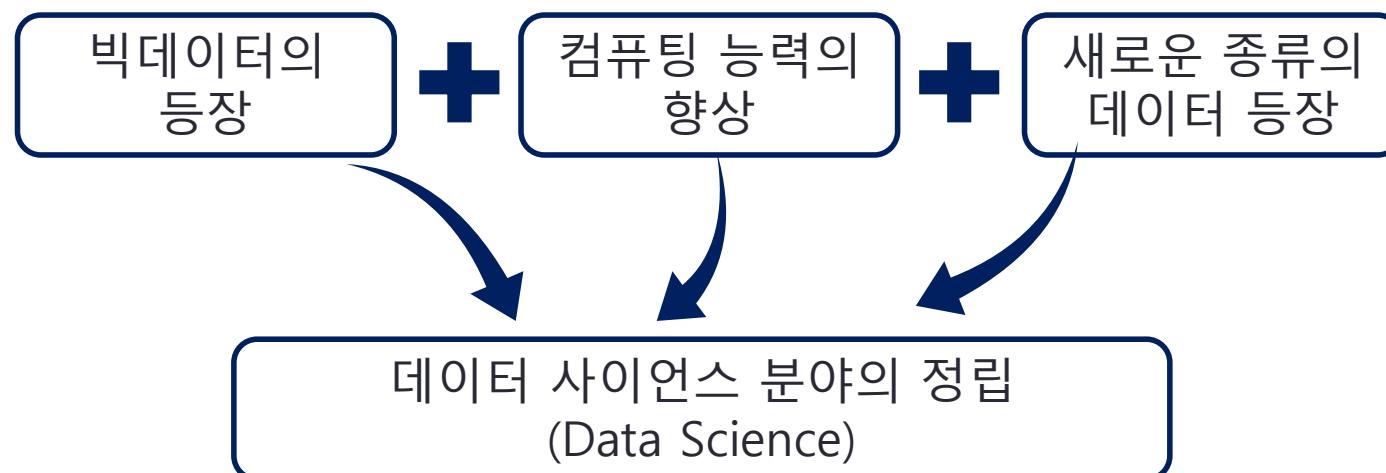


# 데이터로부터 지혜를 얻기까지..



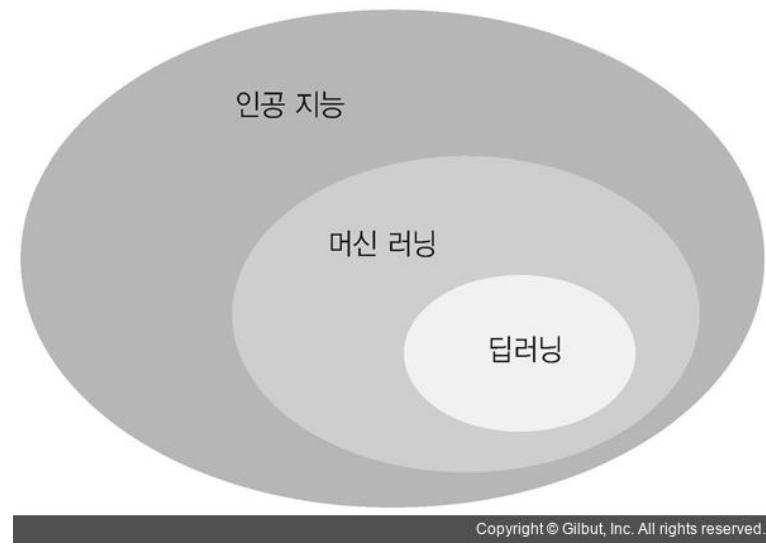
# 데이터 사이언스

- 데이터 과학(data science)이란 정형, 비정형 형태를 포함한 다양한 데이터로부터 지식과 인사이트를 추출하는데 과학적 방법론, 프로세스, 알고리즘, 시스템을 동원하는 융합분야다.



# 머신러닝, 딥러닝, 인공지능

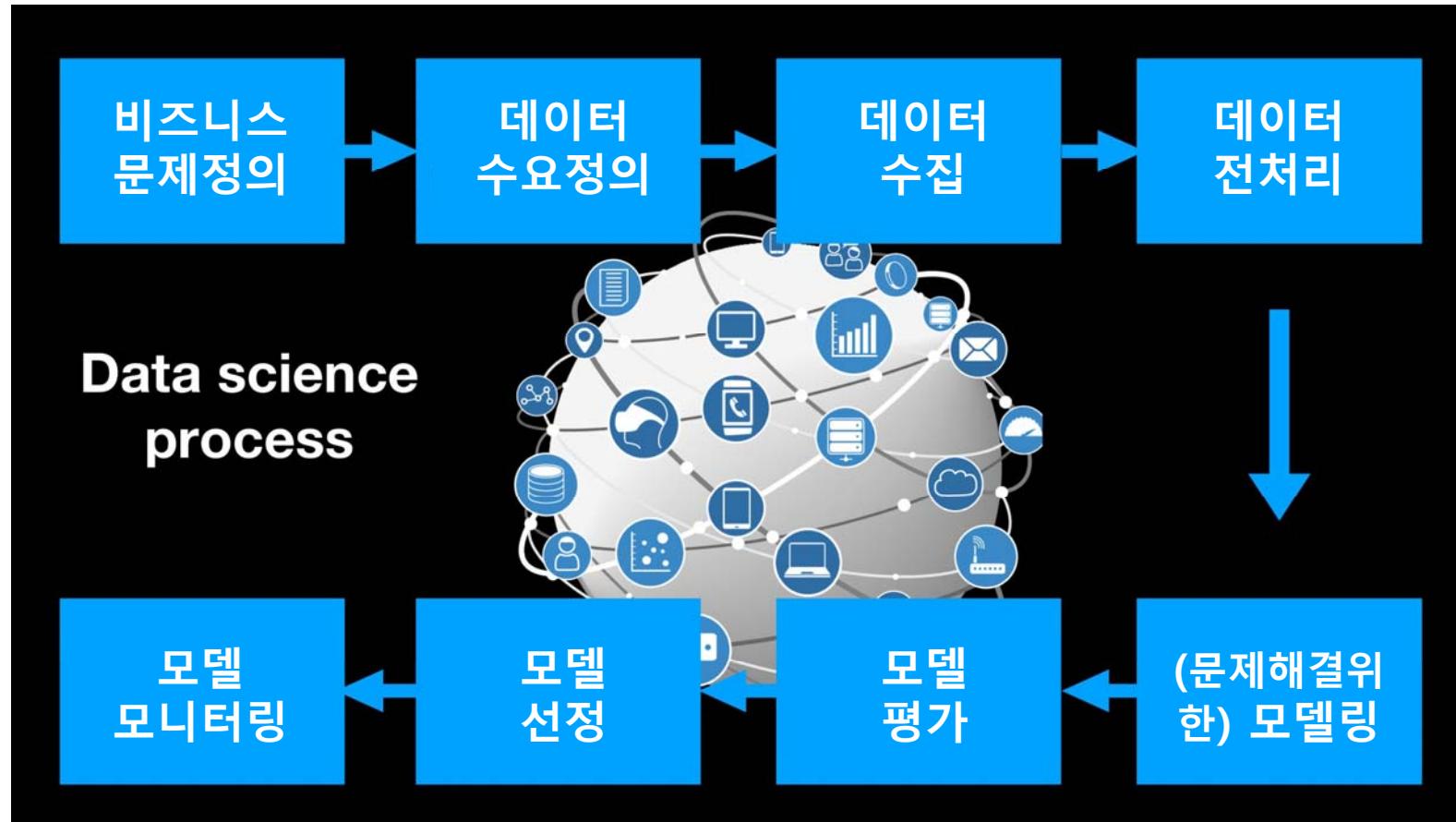
- 딥러닝
  - Extract patterns from data using neural network
  - 인공신경망 구조를 사용하여 데이터로부터 패턴을 추출
- 머신러닝
  - Ability to learn without explicitly being programmed
  - 컴퓨터가 알아서 데이터로부터 규칙을 발견하도록 하는 것
- 인공지능
  - Any techniques that enables computers to mimic human behavior
  - 컴퓨터로 하여금 사람의 생각과 행동을 모사(mimic)하도록 하는 모든 기술



Copyright © Gilbut, Inc. All rights reserved.

# 목표

- 머신러닝 기법을 적용하는 일반적인 절차의 습득

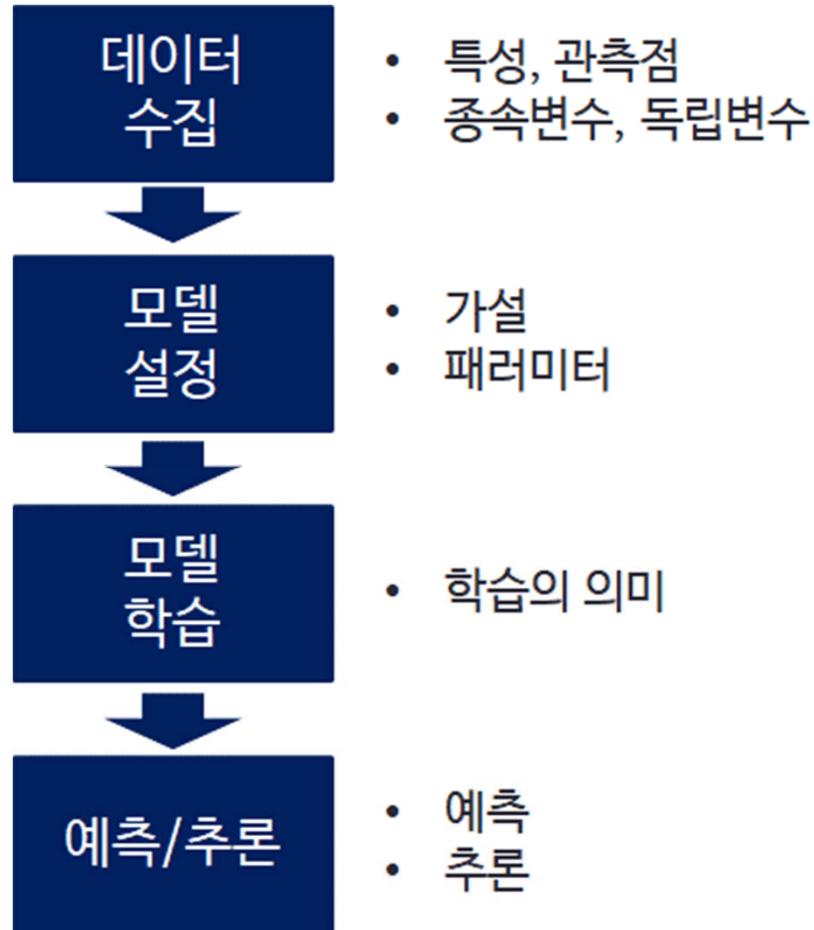


# 데이터사이언스 프로세스

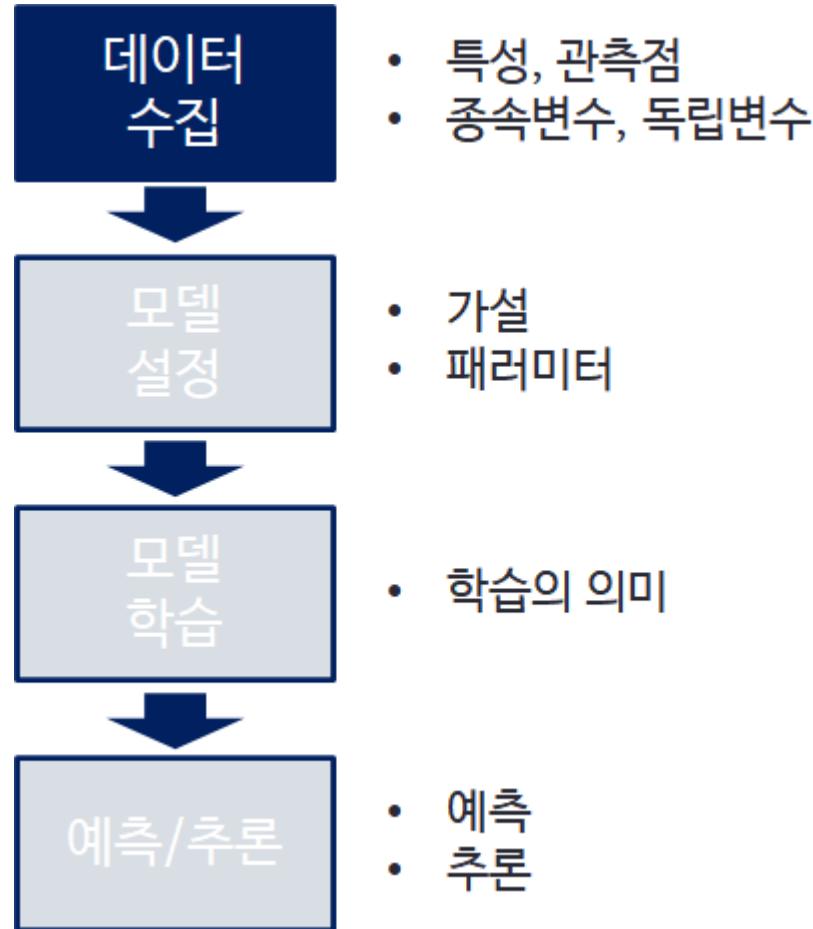
---

시스템경영공학부  
이지환 교수

# 데이터사이언스 주요 프로세스



# 데이터사이언스 주요 프로세스



# 데이터

- 과거에 일어난 사실들을 특정 형식에 맞게 기록해놓은 자료
- 데이터의 예시
  - 어떤 기업내 사원 50명에 대한 데이터를 다음과 같이 수집하였다

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000

# 데이터

- 특성(Features)
  - 데이터의 특성을 구분 지어 설명할 수 있는 것 (=변수, 열)
- 관측점(Observation)
  - 특성에 따라 기록되어있는 서로 다른 객체들 (=행, data point)

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000

# 독립변수와 종속변수

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000

- 종속변수(dependent variable)
  - 독립변수에 영향을 받는 특성
  - (예시) 연봉
  - (소문자)  $y$
  - A.k.a target variable, output variable, label
- 독립변수(independent variable)
  - 종속변수에 영향을 주는 것으로 여겨지는 특성들
  - (예시) 경력
  - (대문자)  $X$
  - a.k.a predictor, input variable, regressor

# 데이터를 가지고 해볼 수 있는 질문들: 예측

- 예측 (Prediction)
  - 독립변수가 주어져 있는 상태에서, 종속변수의 값을 추측
  - Predict the outcome for new data point.
  - (예시) 경력이 3년, 6년, 9년인 사람에게 각각 얼마의 연봉을 주어야 할까?

데이터 (과거)

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000



예측 (일어나지 않음)

3	??
6	??
9	??

# 데이터를 가지고 해볼 수 있는 질문들: 추론

- 추론 (Inference)
  - 독립변수와 종속변수간의 설명가능한 관계를 파악
  - Learn about data generation process
  - (예시) 1년 연봉이 오를때마다 20,000\$씩 연봉이 오른다.

데이터 (과거)

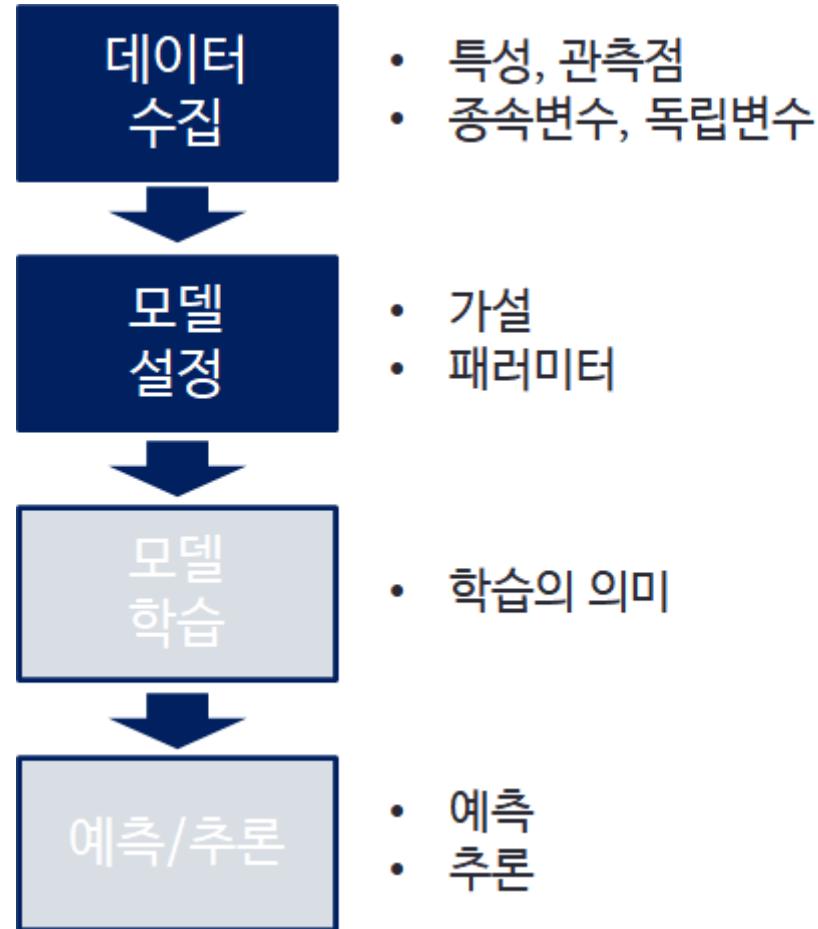
	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000



예측 (일어나지 않음)

3	??
6	??
9	??

# 데이터 사이언스 주요 프로세스



# 모형(Model)

- {예측}과 {추론}을 위해서는 데이터를 활용하여 {모형}을 만들어야 한다
- 모형(Model)
  - 독립변수와 종속변수의 관계에 대한 가설을 수학적으로 표현한 것
  - A machine learning model can be a mathematical representation of a relationship between X and y
- 모형의 종류
  - X와 y의 관계에 대한 가설에 따라 다양한 종류의 모형 존재
    - 회귀모형
    - 로지스틱회귀모형
    - K-Nearest Neighbor
    - Support Vector Machine
    - Random Forest
    - 딥뉴럴네트워크

# 모형의 예시

- 같은 데이터라 할 지라도 다양한 X와 y의 관계에 대한 다양한 모형을 생각해 볼 수 있음

가설1: 연봉이 경력에  
비례하여 증가할 것이다

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000

가설2: 연봉이 경력의 제곱에  
비례하여 증가할 것이다

가설n: 연봉이 경력의 n차식으로  
표현될 것이다.

# 모형의 예시

- 같은 데이터라 할 지라도 다양한 X와 y의 관계에 대한 다양한 모형을 생각해 볼 수 있음

가설1: 연봉이 경력에  
비례하여 증가할 것이다

$$y = \beta_0 + \beta_1 x$$

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000

가설2: 연봉이 경력의 제곱에  
비례하여 증가할 것이다

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

가설n: 연봉이 경력의 n차식으로  
표현될 것이다.

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n$$

# 파라미터 (parameter)

- 모델안에서 독립변수(X)와 종속변수(y)간의 관계를 표현하기 위해 조절할 수 있는 매개변수

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000

가설1: 연봉이 경력에  
비례하여 증가할 것이다

$$y = \beta_0 + \beta_1 x$$

파라미터



$$\beta_0, \beta_1$$

가설2: 연봉이 경력의 제곱에  
비례하여 증가할 것이다

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$



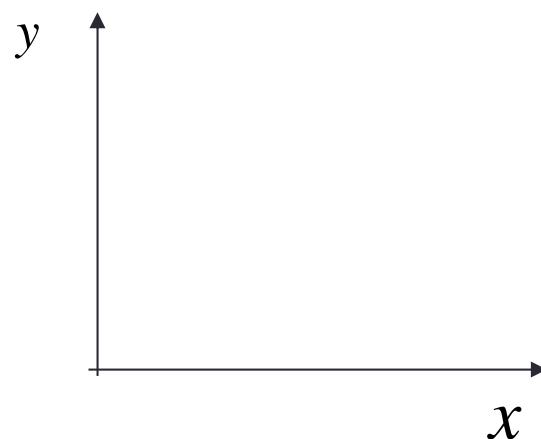
$$\beta_0, \beta_1, \beta_2$$

가설n: 연봉이 경력의 n차식으로  
표현될 것이다.

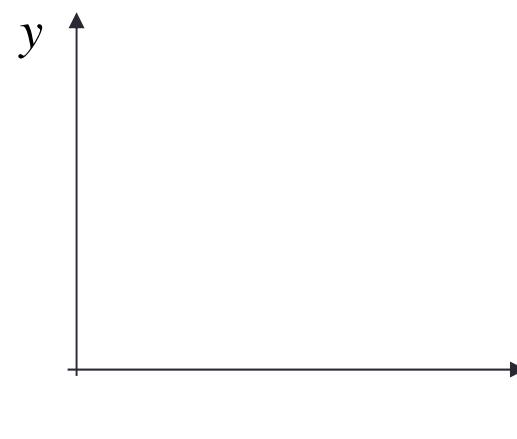
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n \rightarrow \beta_0, \beta_1, \dots, \beta_n$$

# 파라미터 (parameter)의 역할

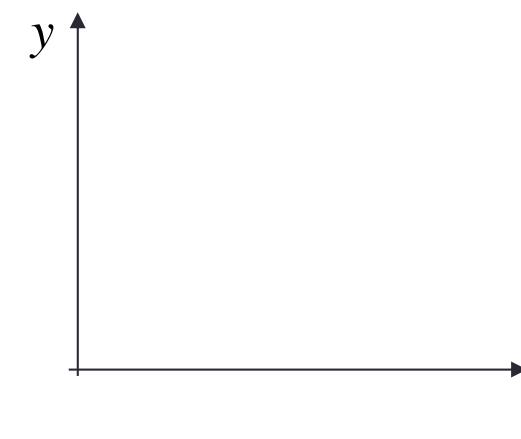
- 파라미터를 어떻게 잡느냐에 따라 X와 y의 관계를 다르게 표현할 수 있다!



$$\begin{aligned}\beta_0 &= 2 \\ \beta_1 &= 0\end{aligned}$$

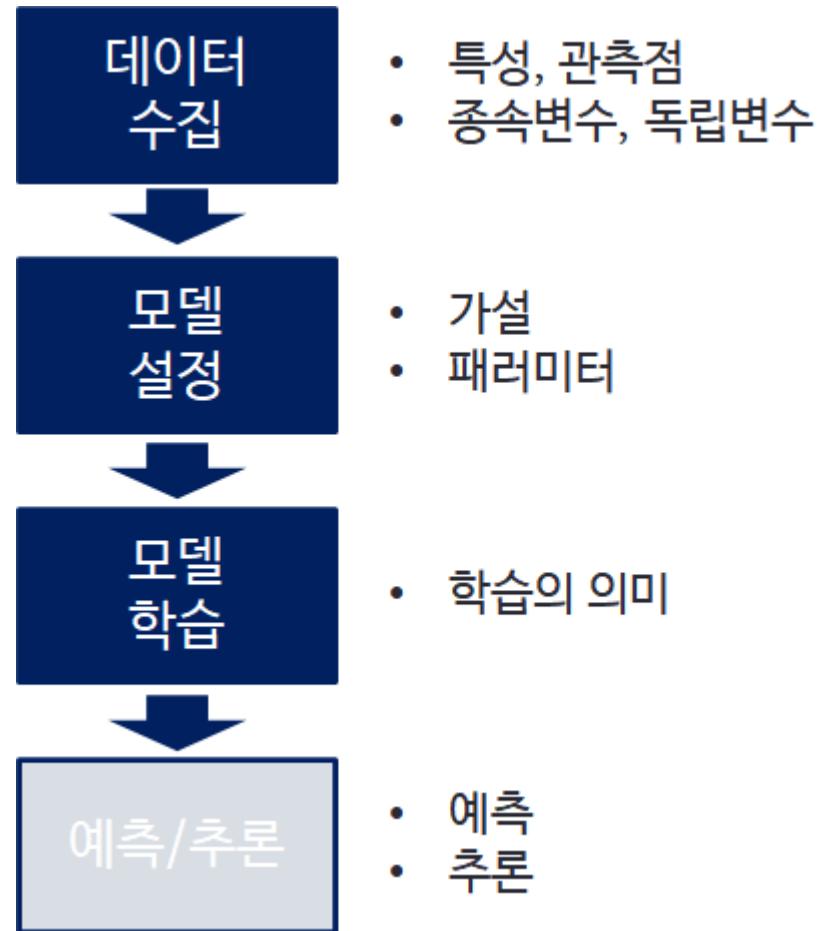


$$\begin{aligned}\beta_0 &= 2 \\ \beta_1 &= 1.5\end{aligned}$$



$$\begin{aligned}\beta_0 &= 2 \\ \beta_1 &= -0.5\end{aligned}$$

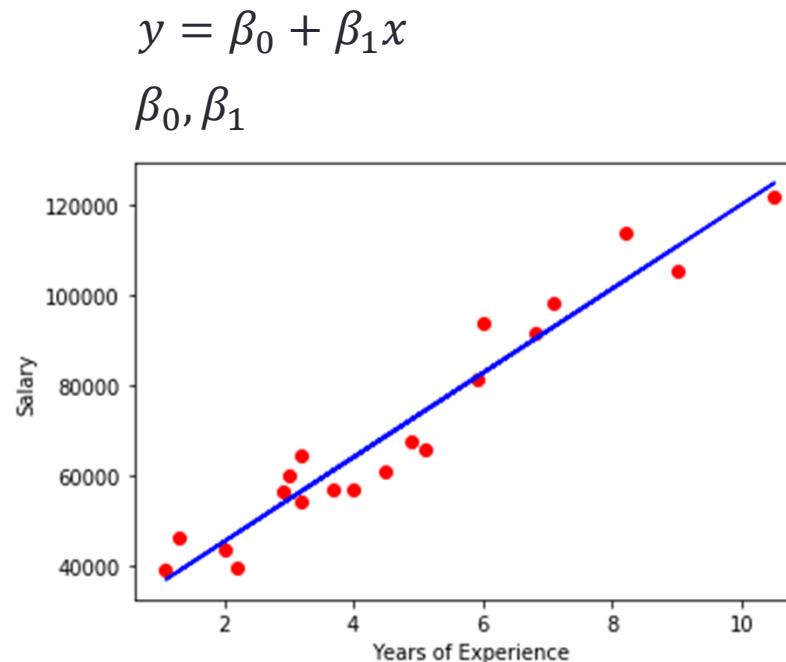
# 데이터 사이언스 주요 프로세스



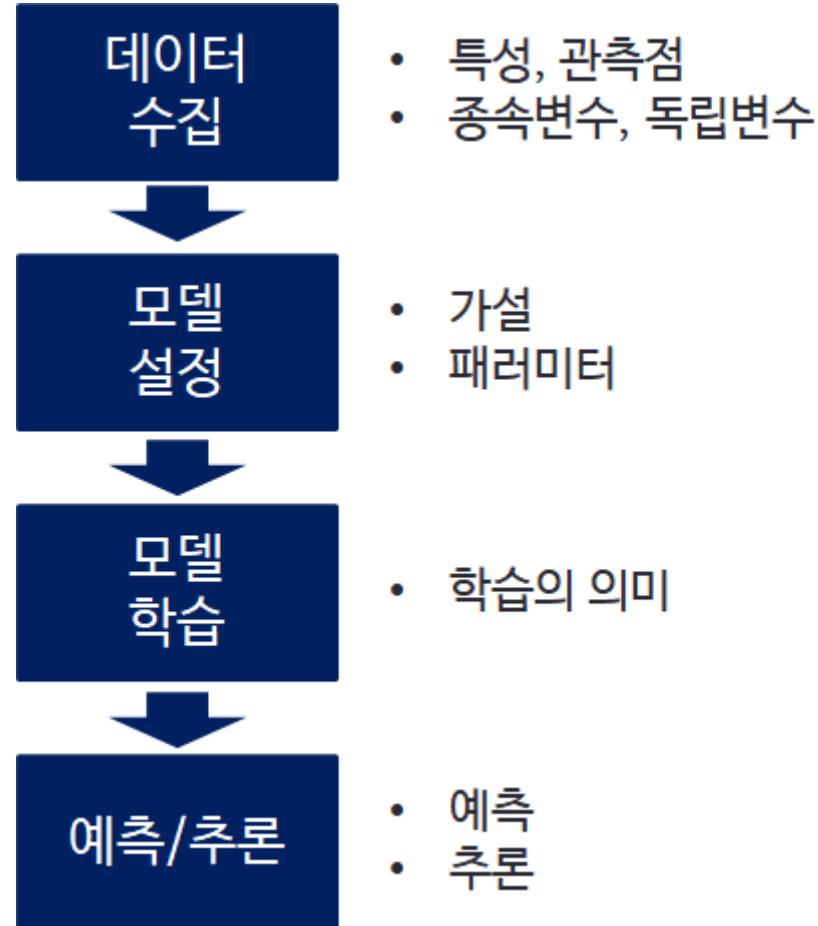
# 모델의 학습(Learning)

- 모델: X와 y에 대한 수학적인 관계
- 학습: 주어진 데이터를 가장 잘 나타내는 패러미터를 찾아나가는 과정

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000

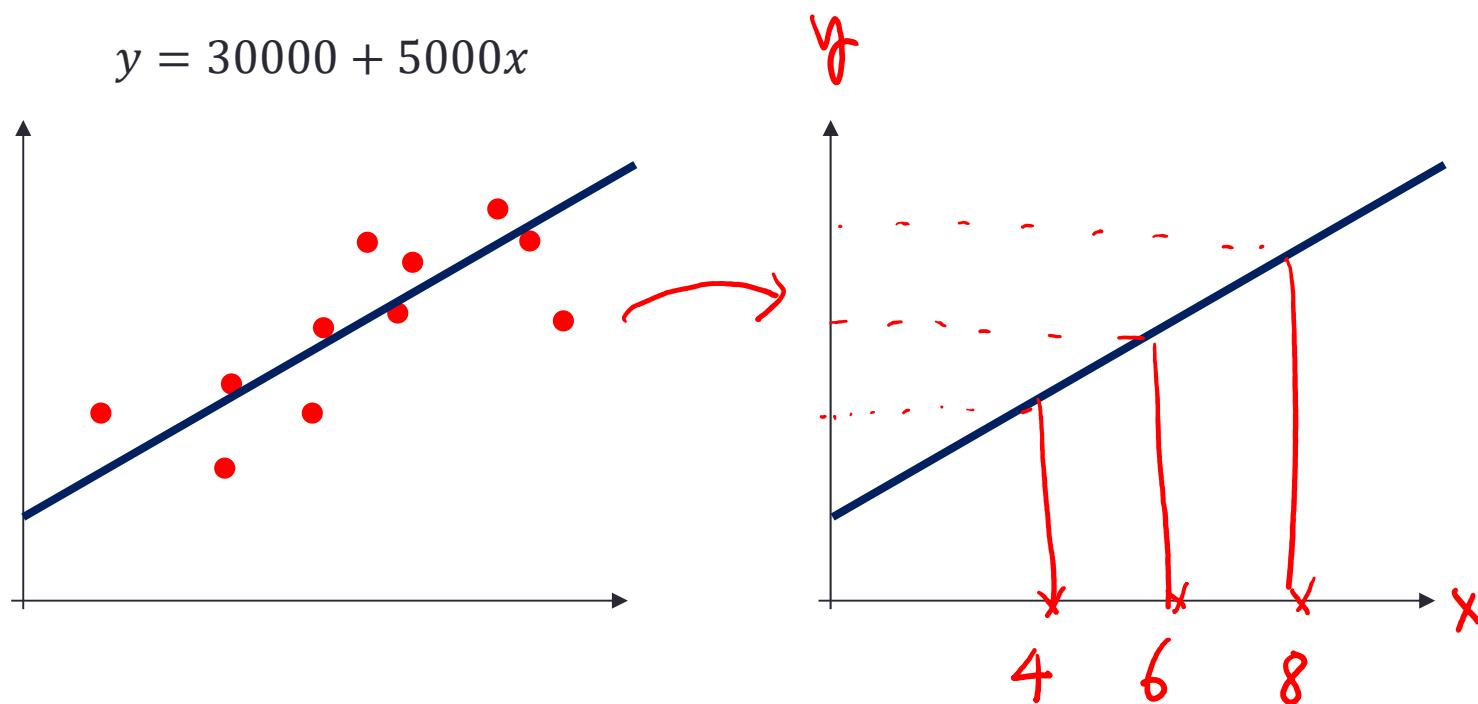


# 데이터 사이언스 주요 프로세스



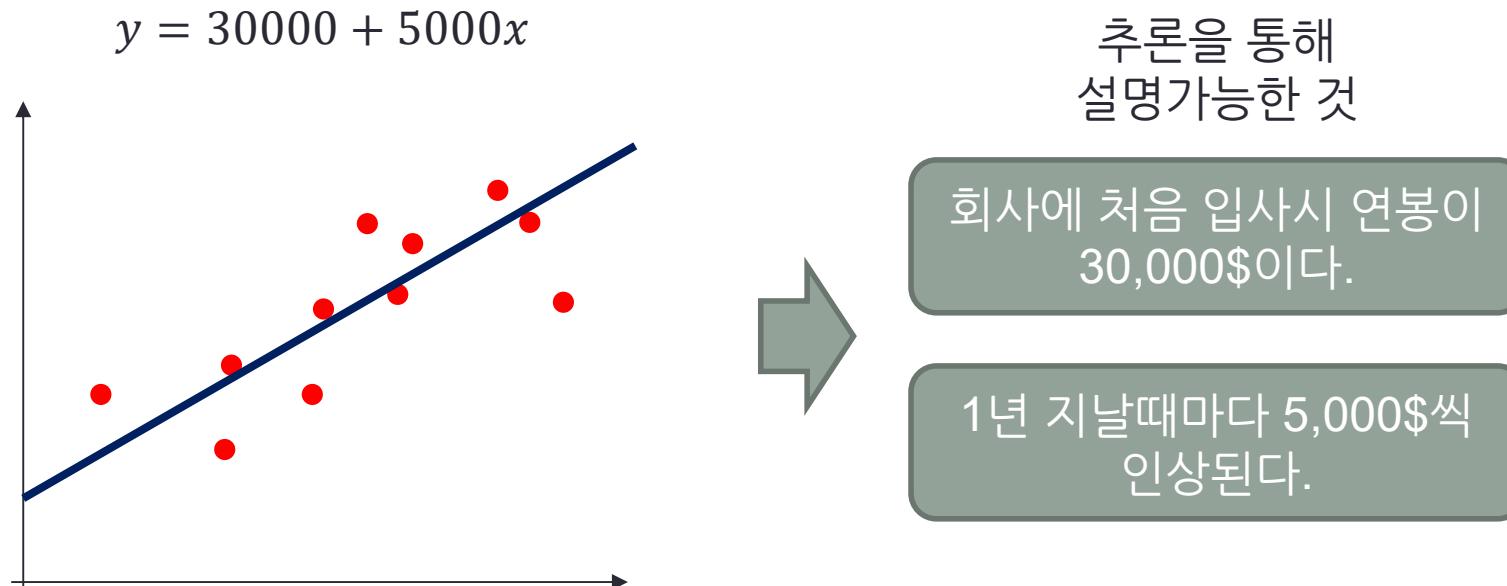
# 모델의 이용: 예측(Prediction)

- 새로운 데이터의  $x$ 값이 주어질 때, 모델에서 학습된 패러미터를 이용하여  $y$ 값을 예측
- 경력이 4년, 6년, 8년인 직원의 연봉은 각각 얼마인가?



# 모델의 이용: 추론(Prediction)

- $y$ 값을 예측하는데 있어  $X$ 의 중요한 특성(feature)이 무엇인지, 그리고 그 방향성에 대하여 설명하는 것
- 데이터의 {생성과정을 설명}할 수 있는 명확한 모델이 있을 때 가능
  - 모든 모델에서 추론이 가능하진 않음



# 머신러닝 모형의 분류

---

시스템경영공학부  
이지환 교수

# 머신러닝 모형의 분류

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression )
  - (No) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

# 머신러닝 모형의 분류

- 종속변수가 숫자형 값인가? 범주형 값인가?
  - (숫자형) 회귀(Regression)
  - (범주형) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

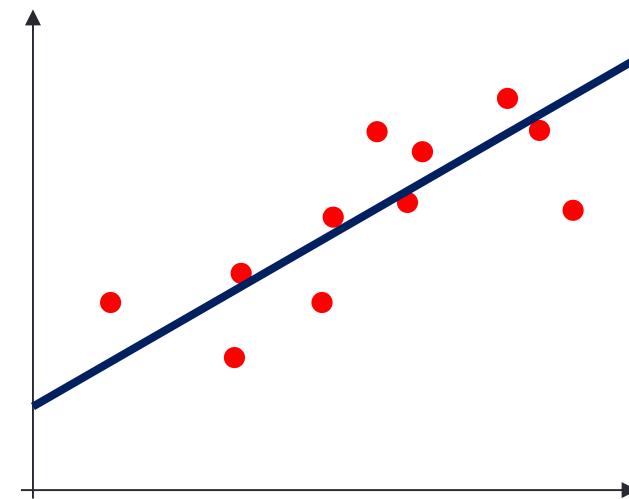
# 변수의 종류

- 숫자형 변수 (Numerical Variable)
  - 실수나 정수로 표현할 수 있음
    - 예시) 나이: {56, 24, …}, 온도: {27.2, 24.0, …},
  - 값의 차이가 의미 있음
- 범주형 변수 (Categorical Variable)
  - 실수로 표현할 수 없음
  - 분절된 서로 다른 값을 가짐
    - 예시) 성별: {남자, 여자} // 브랜드: {현대, 도요타, 기아, 포드} // 결혼여부: {yes, no}
  - (설령) 서로 다른 값에 숫자를 부여한다 해도 숫자간의 차이는 무의미

# 회귀(Regression)

- 종속변수가 숫자형인 경우
- 경력이 8년인 사람의 연봉은 얼마인가? → 실수값을 예측

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000



# 분류(Classification)

- 종속변수가 범주형(Categorical Variable)인 경우
  - 다음과 같은 50명의 기존 데이터가 있다

	나이	성별	직업	브랜드
1	24	남	군인	삼성
2	23	여	디자이너	애플
3	40	남	디자이너	애플
49	50	여	회사원	삼성
50	27	여	요리사	삼성

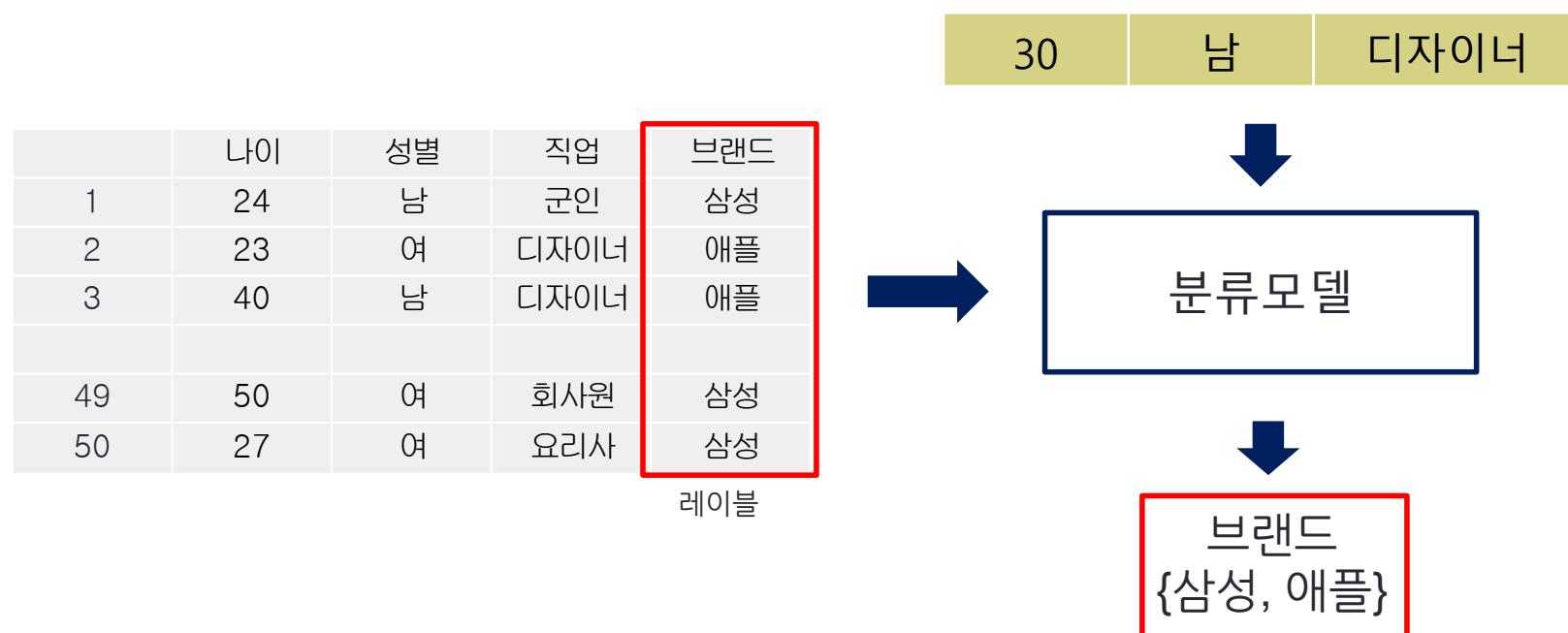
- 나이, 성별, 직업이 주어진 어떤 사람이 {삼성, 애플} 두 핸드폰 중 무엇을 고를지 예측하고 싶다.
  - 브랜드는 전형적인 범주형 변수로 분류문제에 해당됨

# 모델의 분류

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression )
  - (No) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

# 지도학습 (Supervised Learning)

- 학습하는 데이터에 예측하고자 하는 값이 이미 존재하는 경우
- 예시) 나이, 성별, 직업을 통해 핸드폰 브랜드를 예측하고 싶다.
  - 데이터에 실제 사람들이 어떤 브랜드를 선택했는지 있다.



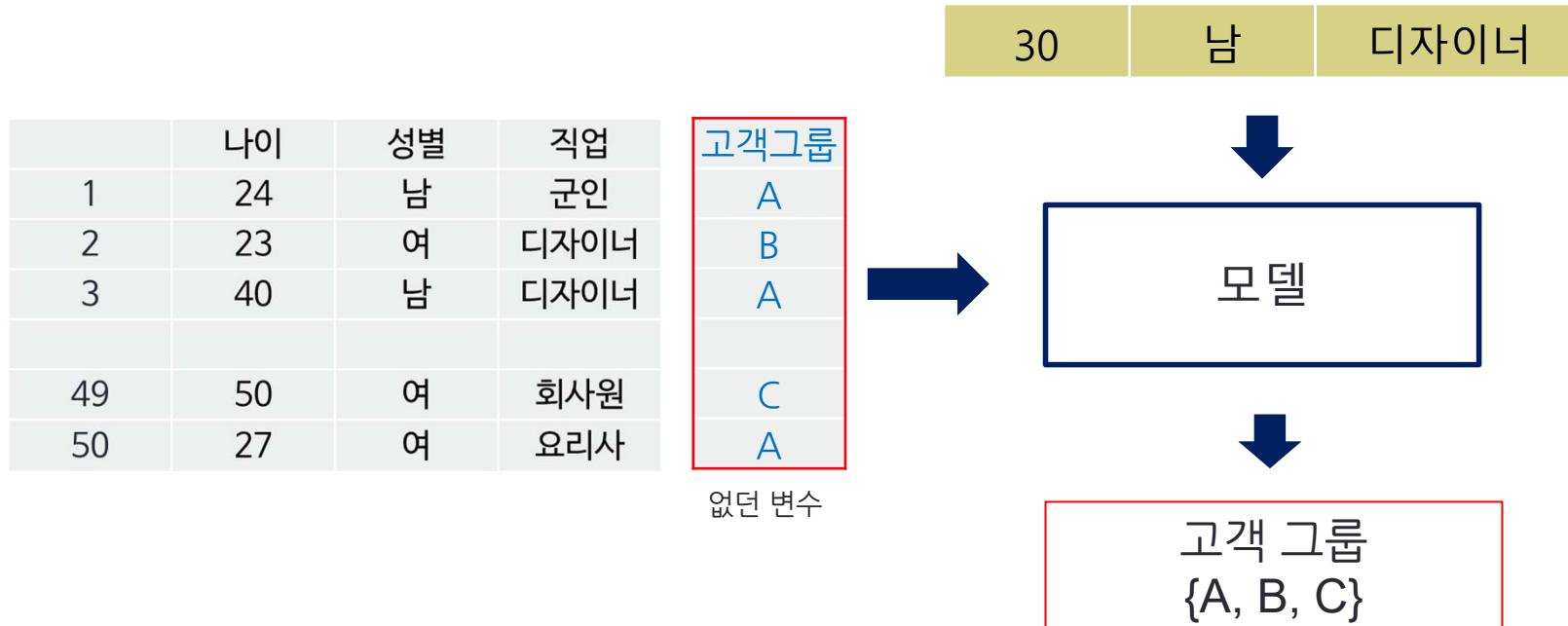
# 비지도학습 (Unsupervised Learning)

- 학습하는 데이터내에 예측하고자 하는 값이 없는 경우
- 예시) 나이, 성별, 직업의 유사도에 따라 3개의 그룹으로 나누고 싶다.



# 비지도학습 (Unsupervised Learning)

- 학습하는 데이터내에 예측하고자 하는 값이 없는 경우
- 예시) 나이, 성별, 직업의 유사도에 따라 3개의 그룹으로 나누고 싶다.
  - 데이터를 설명하는 내재적인 구조를 학습해야 함
- 군집분석(Clustering Analysis)



# 비지도학습 (Unsupervised Learning)

- 학습하는 데이터내에 예측하고자 하는 값이 없는 경우
- 예시) 시간에 따라 기록된 5,000개의 공정 데이터가 존재한다. 새로운 데이터가 기존의 데이터에서 얼마나 벗어나 있는지 측정하고자 한다.

	전압	온도	압력	습도
1	XX	XX	XX	XX
2	XX	XX	XX	XX
3	XX	XX	XX	XX
4999	XX	XX	XX	XX
5000	XX	XX	XX	XX

10v | 32c | 200p | 30%



모델



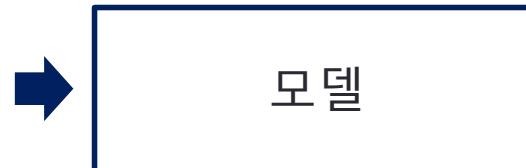
# 비지도학습 (Unsupervised Learning)

- 학습하는 데이터내에 예측하고자 하는 값이 없는 경우
- 예시) 시간에 따라 기록된 5,000개의 공정 데이터가 존재한다. 새로운 데이터가 기존의 데이터에서 얼마나 벗어나 있는지 측정하고자 한다.
- 이상치 탐지(anomaly detection)

	전압	온도	압력	습도
1	xx	xx	xx	xx
2	xx	xx	xx	xx
3	xx	xx	xx	xx
4999	xx	xx	xx	xx
5000	xx	xx	xx	xx

이상치 점수
0.05
0.04
0.2
0.01
0.2

10v 32c 200p 30%



모델



이상치 점수  
{1.68}

# 지도학습 vs 비지도 학습

## 지도학습

(Supervised Learning)

### 데이터: $(X, y)$

- $X$  데이터

- $y$  레이블(정답)

### 목표

- $X \rightarrow y$ 의 관계를 맵핑하는 함수를 찾는 것

### 예시

- 회귀, 분류, CNN, 객체탐지, Sequence 모델

## 비지도학습

(Unsupervised Learning)

### 데이터: $X$

- $X$  데이터

- $y$ 는 존재하지 않음

### 목표

- 데이터를 설명하는 내재적인 구조를 학습

### 예시

- 군집분석(Clustering)
- 차원축소 (Dimensionality Reduction)
- 이상치 탐지

# 모델의 분류

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression)
  - (No) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

# Parametric Model

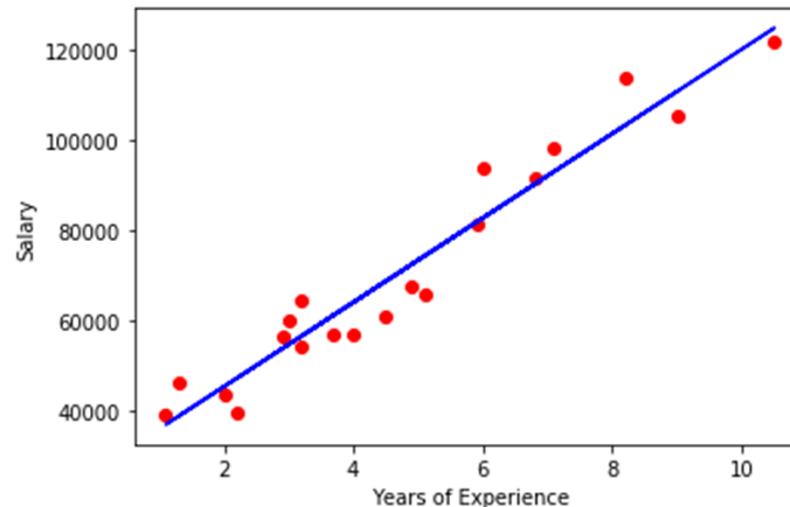
- X와 y사이의 함수적 형태를 가정
- 독립변수와 종속변수의 관계를 고정된 수의 패러미터로 표현할 수 있음

	경력	연봉(\$)
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000



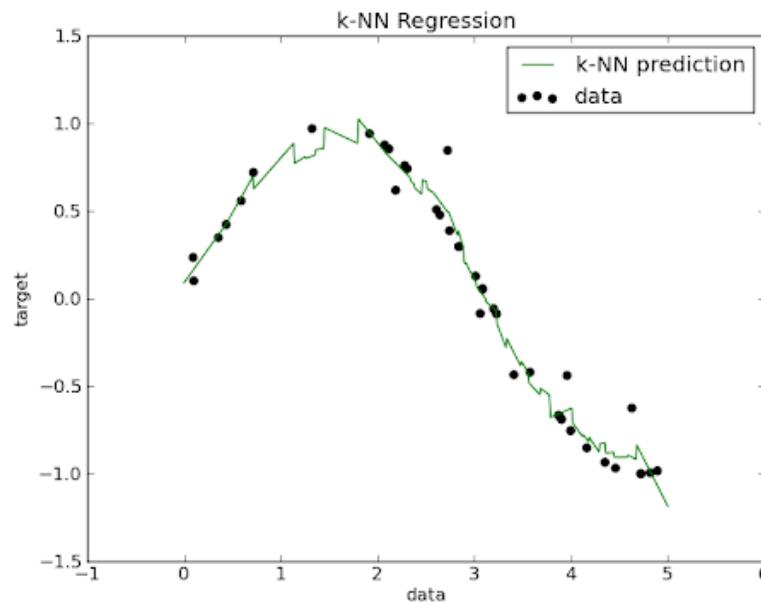
$$y = \beta_0 + \beta_1 x$$

$$\beta_0, \beta_1$$



# Non-Parametric Model

- 패러미터의 수가 고정되지 않고 데이터에 따라 달라짐
  - X와 y 사이의 함수적 형태를 가정하지 않음
- 
- 예시) KNN Regression
    - (주변에서 가장 가까운 K개의 데이터를 y값을 평균내어 예측)



# 선형 회귀모형 (Linear Regression)

---

시스템경영공학부  
이지환 교수

# 모형의 분류

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression )
  - (No) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

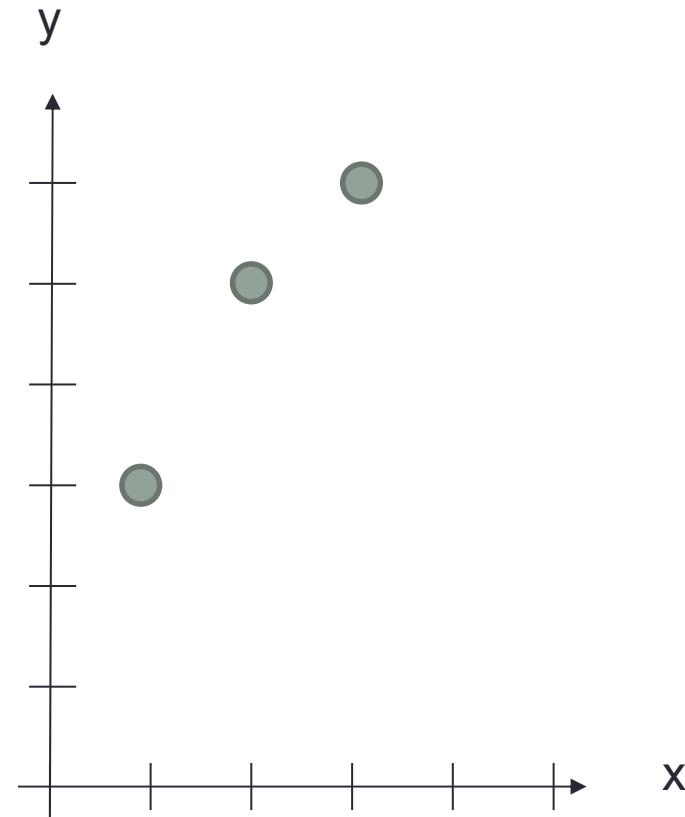
# 모형의 분류: 선형회귀모형

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression)
  - (No) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

# 예시

- 데이터셋
  - 독립변수: 학습시간
  - 종속변수: 성적

$x$	$y$
학습시간	성적
1 $x^{(1)}$	0 $y^{(1)}$
2 $x^{(2)}$	2 $y^{(2)}$
5 $x^{(3)}$	4 $y^{(3)}$



# 선형회귀모형의 가설

- 선형회귀모형의 가설
  - X와 y의 관계를 1차 다항식으로 표현

학습시간	성적
1	3
2	5
5	6

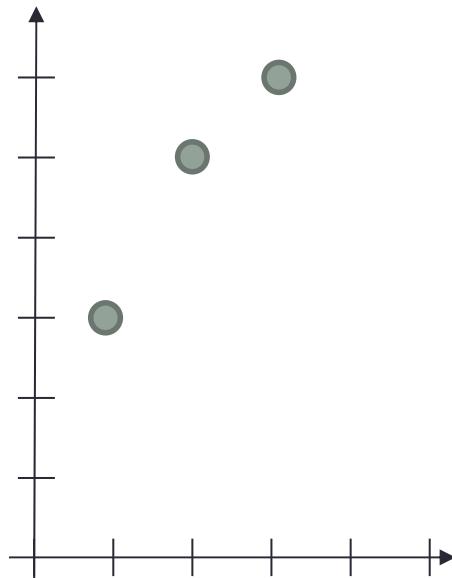
모델의 가정:  
학습시간(X)과 성적(y) 사이에는  
선형관계가 있을 것이다.



$$y = \theta_0 + \theta_1 x$$

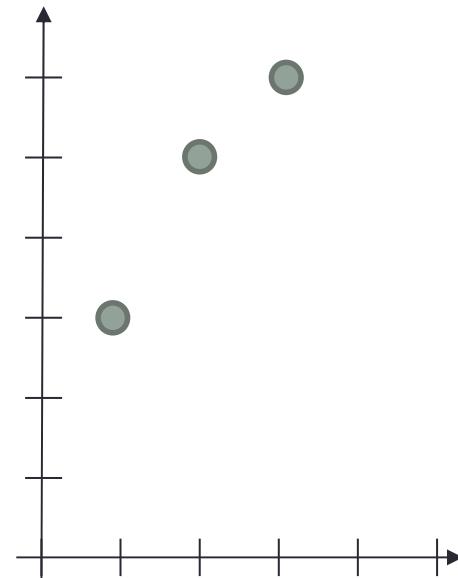
# 선형회귀모형의 학습

- 학습의 목적: X와 y의 관계를 잘 나타내는 패러미터를 찾는 것
- {X와 y의 관계를 잘 나타내는 정도}를 어떻게 {측정} 할 수 있을까?



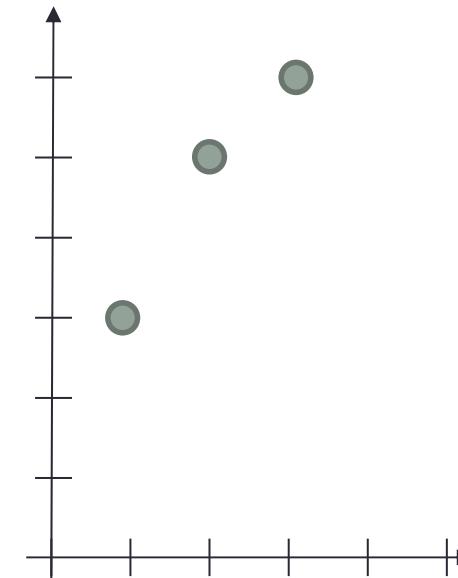
$$\theta_0 = 4$$

$$\theta_1 = 0$$



$$\theta_0 = 2$$

$$\theta_1 = 1.5$$



$$\theta_0 = 1$$

$$\theta_1 = 0.5$$

# 비용(Cost)

- 비용: 현재 패러미터로 주어진 데이터를 얼마나 잘 표현할 수 있는지 측정하는 함수
- 비용의 정의  $\text{cost} = f(y, \hat{y})$ 
  - $y$ : 데이터의 실제 레이블 값
  - $\hat{y}$ : 현재 패러미터를 가지고 예측한 레이블 값

# 비용(Cost)

- 비용: 현재 패러미터로 주어진 데이터를 얼마나 잘 표현할 수 있는지 측정하는 함수
- 비용의 정의  $\text{cost} = f(y, \hat{y})$ 
  - $y$ : 데이터의 실제 레이블 값
  - $\hat{y}$ : 현재 패러미터를 가지고 예측한 레이블 값

$$\text{cost} = f(y, \hat{y}) = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

학습시간	성적
1	3
2	5
5	6

$y$	$\hat{y}$	$y - \hat{y}$	$(y - \hat{y})^2$

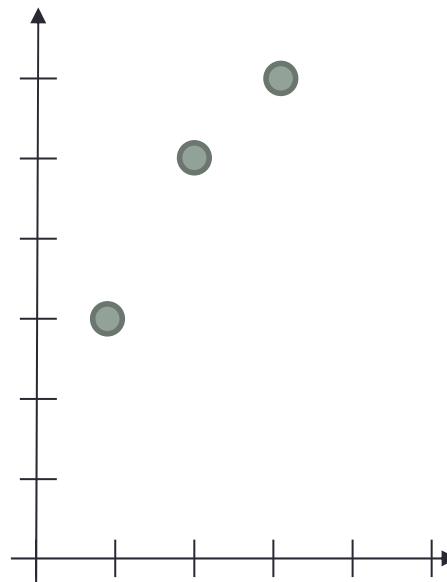
# 비용(Cost) 계산 예시

- 선형회귀에서의 비용함수

$$\text{cost} = f(y, \hat{y}) = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

- 비용함수 계산

- 패러미터  $\theta_0 = 4, \theta_1 = 0$



학습시간	성적
1	3
2	5
5	6

$y$	$\hat{y}$	$y - \hat{y}$	$(y - \hat{y})^2$

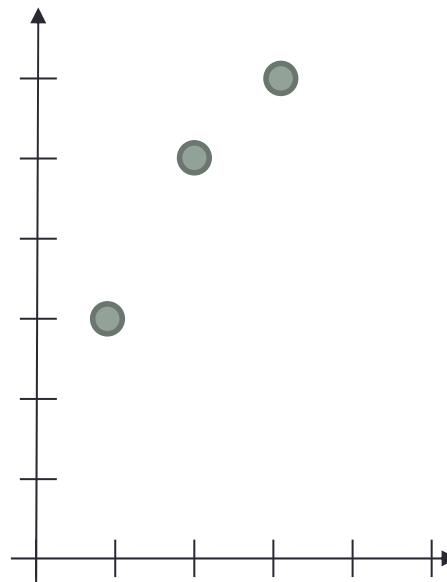
# 비용(Cost) 계산 예시

- 선형회귀에서의 비용함수

$$\text{cost} = f(y, \hat{y}) = \sum_{i=1}^n (y^{(i)} - \hat{y}^{(i)})^2$$

- 비용함수 계산

- 패러미터  $\theta_0 = 3, \theta_1 = 1$



학습시간	성적
1	3
2	5
5	6

$y$	$\hat{y}$	$y - \hat{y}$	$(y - \hat{y})^2$

# 비용 함수 (Cost Function)

- 비용함수는 패러미터의 함수로 표현된다.
  - 데이터: 주어진 상수
  - 패러미터: 우리가 결정해야 하는 변수

- 모델  $y = \theta_0 + \theta_1 x$

- 패러미터  $\theta_0, \theta_1$

- 비용  $\hat{y}^{(i)}, y^{(i)}$

$$x^{(i)} \quad \theta_0 + \theta_1 x^{(i)} \quad y^{(i)}$$

:

$$x^{(n)} \quad \theta_0 + \theta_1 x^{(n)} \quad y^{(n)}$$

- 비용함수

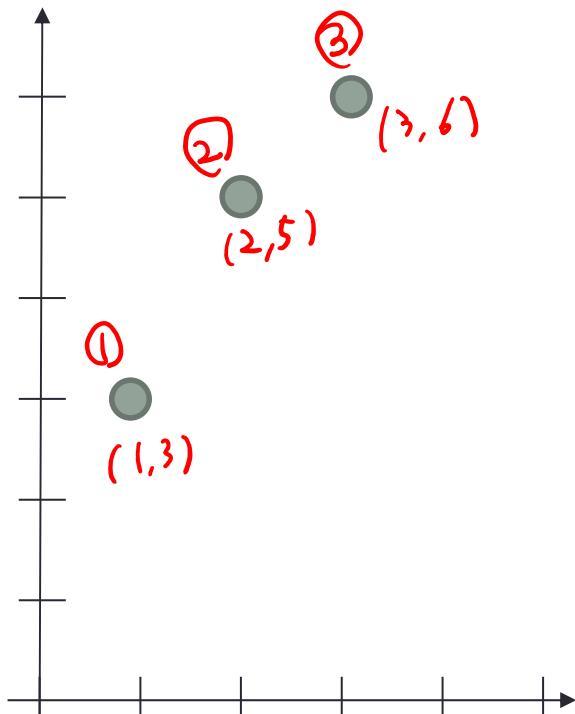
$$J(\theta_0, \theta_1)$$

$$= (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 + \dots + (\hat{y}_n - y_n)^2$$

$$= (\theta_0 + \theta_1 x^{(1)} - y^{(1)})^2 + \dots + (\theta_0 + \theta_1 x^{(n)} - y^{(n)})^2$$

↳  $\theta_0, \theta_1$ 에 대한  
2차 항수.

# 비용함수 예시

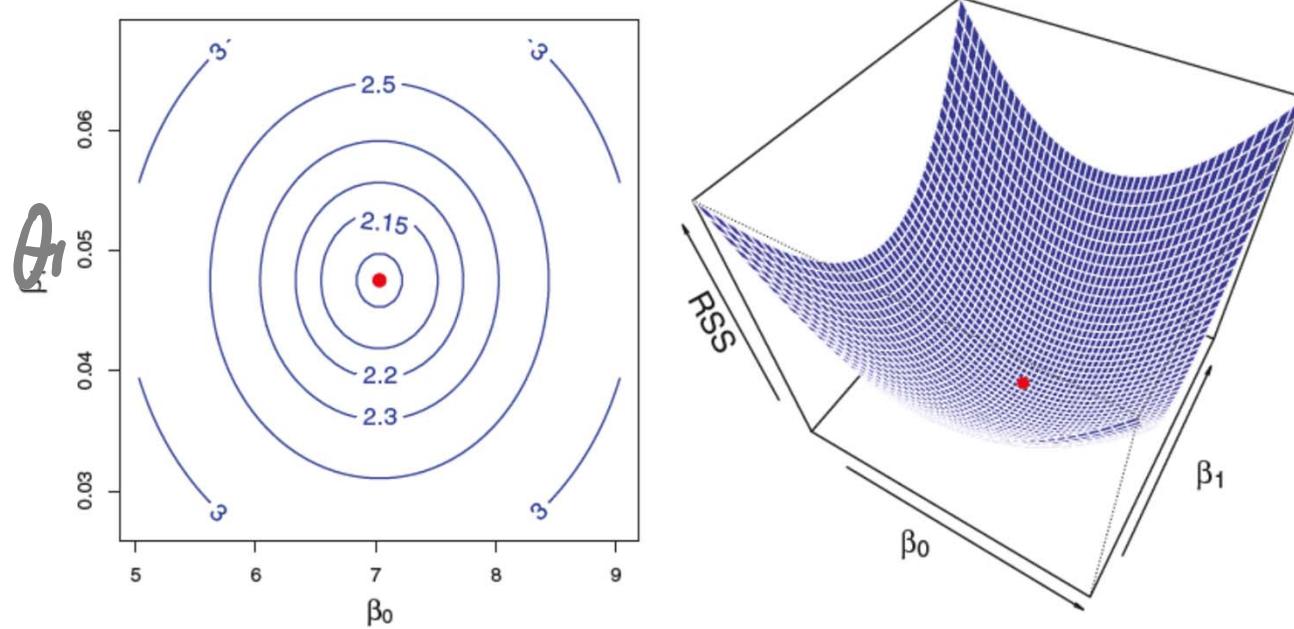


$$\begin{aligned} J(\beta_0, \beta_1) &= (y^{(1)} - \hat{y}^{(1)})^2 + (y^{(2)} - \hat{y}^{(2)})^2 + (y^{(3)} - \hat{y}^{(3)})^2 \\ &= (\theta_0 + \theta_1 x^{(1)} - y^{(1)})^2 + (\theta_0 + \theta_1 x^{(2)} - y^{(2)})^2 \\ &\quad + (\theta_0 + \theta_1 x^{(3)} - y^{(3)})^2 \\ &= (\theta_0 + \theta_1 - 3)^2 + (\theta_0 + 2\theta_1 - 5)^2 \\ &\quad + (\theta_0 + 3\theta_1 - 6)^2 \\ &= \vdots \\ \eta_0 + 3\theta_0^2 + 14\theta_1^2 - 28\theta_0 - 62\theta_1 + 12\theta_0\theta_1 \end{aligned}$$

# 학습

- 학습
  - 비용함수  $J(\theta_0, \theta_1)$  를 최소화 시킬 수 있는 패러미터를 찾는것이 목표!

$$J(\theta_0, \theta_1) = 70 + 3\theta_0^2 + 14\theta_1^2 - 28\theta_0 - 62\theta_1 + 12\theta_0\theta_1$$



# 학습

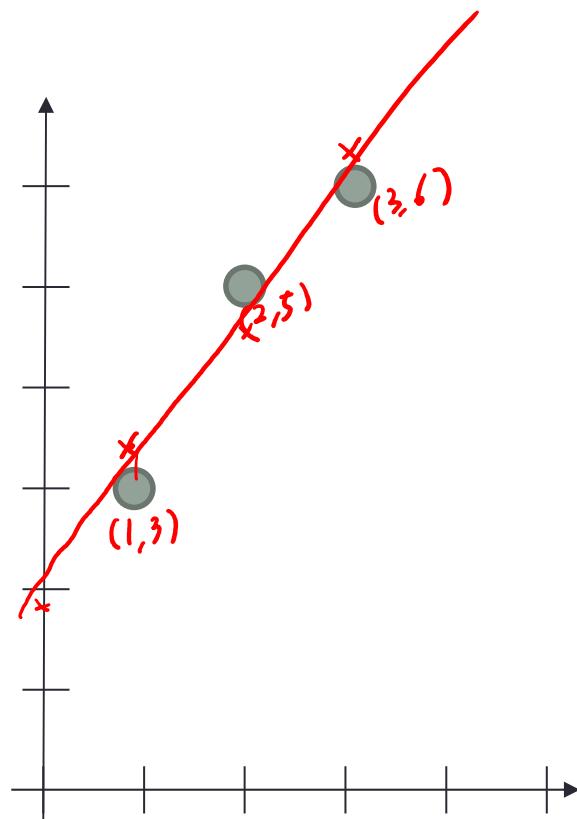
$$J(\theta_0, \theta_1) = 70 + 3\theta_0^2 + 14\theta_1^2 - 28\theta_0 - 62\theta_1 + 12\theta_0\theta_1$$

$$\left\{ \begin{array}{l} \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_0} = 6\theta_0 - 28 + 12\theta_1 = 0 \\ \frac{\partial J(\theta_0, \theta_1)}{\partial \theta_1} = 28\theta_1 - 62 + 12\theta_0 = 0 \end{array} \right.$$

연립방정식

$$\theta_1 = \frac{3}{2} \quad \theta_0 = \frac{5}{3}$$

# 학습완료: 최적 패러미터 도출



$$\theta_0 = \frac{5}{3} \quad \theta_1 = \frac{3}{2}$$

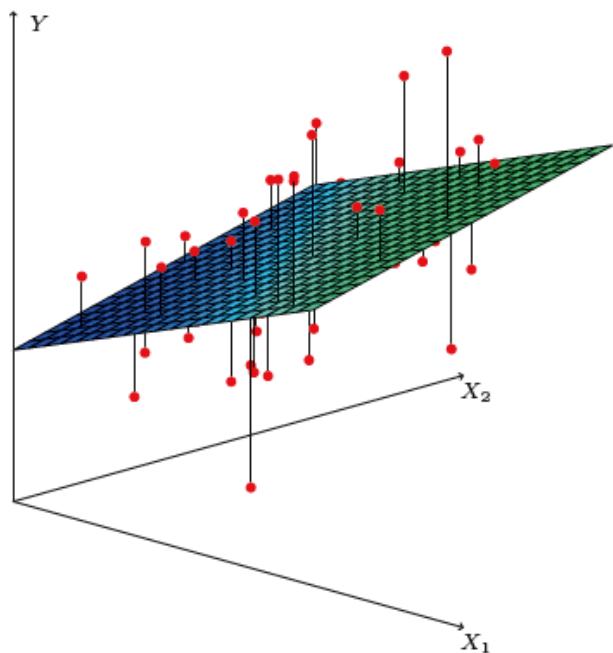
$$y = \underline{\underline{\frac{5}{3} + \frac{3}{2}x}}$$

↳ 학습된 최적 패러미터!

# 선형회귀의 확장 - 다중회귀모형

- 다중선형회귀
  - 독립변수의 개수 확장 가능
- P개의 독립변수가 있는 다중회귀모형

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$



# 선형회귀의 확장

- 기존 선형회귀모형의 가정
  - 서로 다른 변수간에는 독립적이다 (독립성)
  - 독립변수의 증가에 따라 종속변수가 비례하여 바뀐다. (선형성)

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

- 독립성 가정을 제거하여 변수간의 상호작용을 표현할 수 있다.

# 선형회귀의 확장 - Interaction

- 기존 모형

$$\widehat{\text{sales}} = \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times \text{newspaper}$$

- 해석: TV, 라디오, 신문 광고의 효과는 상호간에 영향을 주지 않는다.

- 독립성 가정을 제외한 모형

$$\text{sales} = \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times (\text{radio} \times \text{TV})$$

# 선형회귀의 확장 - Interaction

- 기존 모형

$$\widehat{\text{sales}} = \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio}$$

- 해석: TV, 라디오 광고의 효과는 상호간에 영향을 주지 않는다.

- 독립성 가정을 제외한 모형

$$\begin{aligned}\text{sales} &= \beta_0 + \beta_1 \times \text{TV} + \beta_2 \times \text{radio} + \beta_3 \times (\text{radio} \times \text{TV}) \\ &= \beta_0 + (\beta_1 + \beta_3 \times \text{radio}) \times \text{TV} + \beta_2 \times \text{radio}\end{aligned}$$

- 해석: 라디오 광고에 돈을 많이쓰면, TV광고의 효과가 더 커진다!
- 즉, 변수간의 시너지 효과를 모델링할 수 있다.

# 선형회귀의 확장

- 기존 선형회귀모형의 가정
  - 서로 다른 변수간에는 독립적이다 (독립성)
  - 독립변수의 증가에 따라 종속변수가 비례하여 바뀐다. (선형성)

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon,$$

- 선형성 가정을 제거하여 다차식 관계를 표현할 수 있다.

# 선형회귀의 확장 - Non linearity

- 연비는 마력에 비례하여 바뀐다.

$$\text{mpg} = \beta_0 + \beta_1 \times \text{horsepower}$$

- 연비와 마력사이에는 2차 다항식의 관계가 있다.

$$\text{mpg} = \beta_0 + \beta_1 \times \text{horsepower} + \beta_2 \times \text{horsepower}^2$$

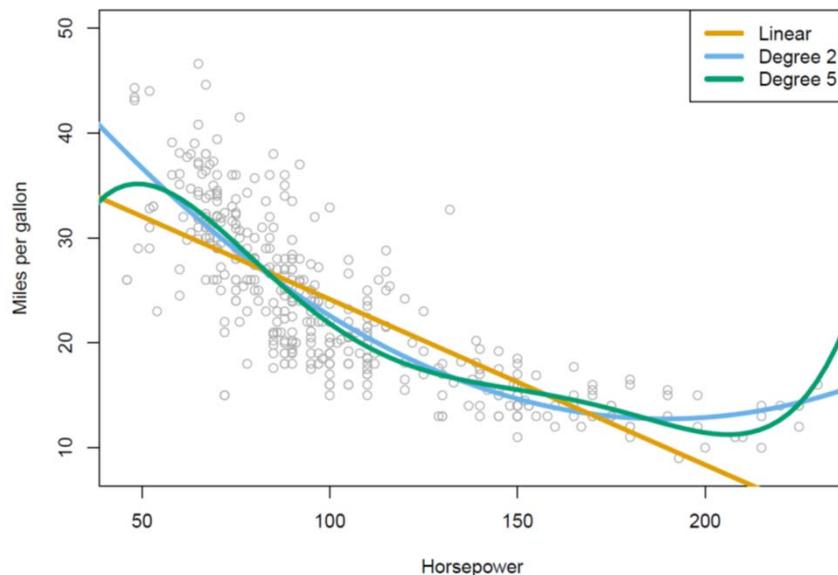
# 선형회귀의 확장 - Non linearity

- 연비는 마력에 비례하여 바뀐다.

$$\text{mpg} = \beta_0 + \beta_1 \times \text{horsepower}$$

- 연비와 마력사이에는 2차 다항식의 관계가 있다.

$$\text{mpg} = \beta_0 + \beta_1 \times \text{horsepower} + \beta_2 \times \text{horsepower}^2$$



# 범주형 변수의 처리

Example: investigate differences in credit card balance between males and females, ignoring the other variables. We create a new variable

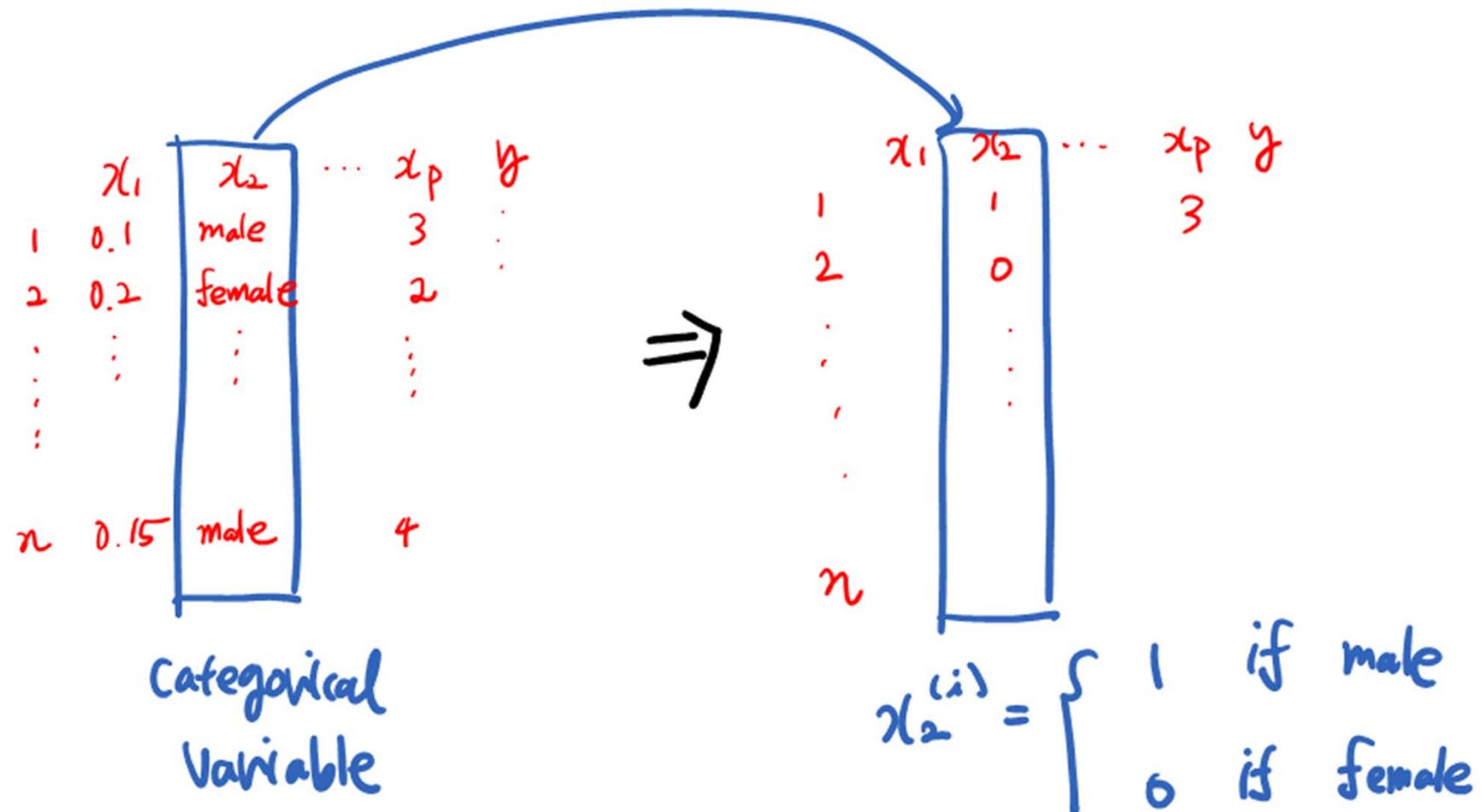
$$x_i = \begin{cases} 1 & \text{if } i\text{th person is female} \\ 0 & \text{if } i\text{th person is male} \end{cases}$$

Resulting model:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is female} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is male.} \end{cases}$$

Intrepretation?

# 범주형 변수의 처리



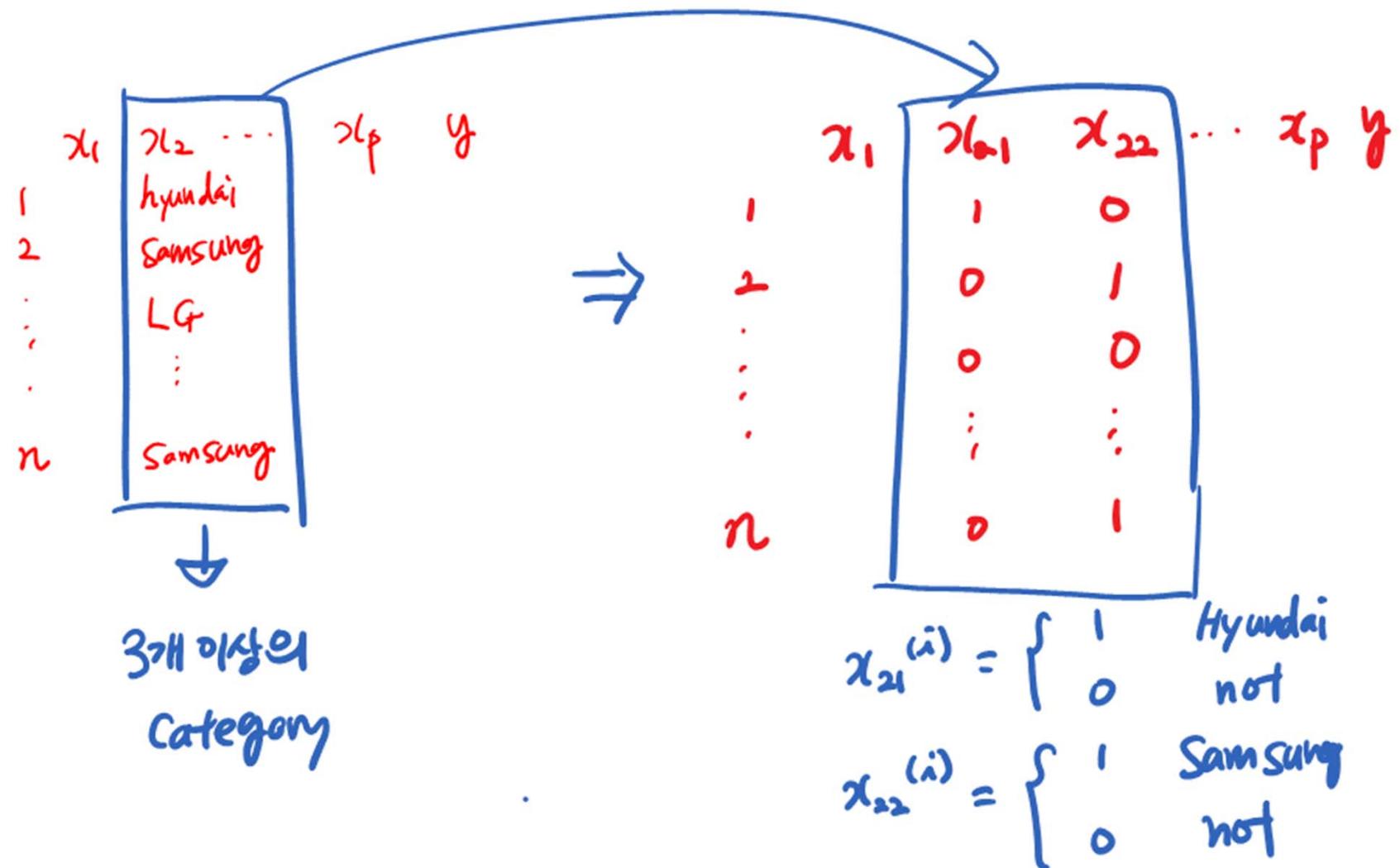
# 2개 이상의 범주형 변수 처리

- Then both of these variables can be used in the regression equation, in order to obtain the model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i = \begin{cases} \beta_0 + \beta_1 + \epsilon_i & \text{if } i\text{th person is Asian} \\ \beta_0 + \beta_2 + \epsilon_i & \text{if } i\text{th person is Caucasian} \\ \beta_0 + \epsilon_i & \text{if } i\text{th person is AA.} \end{cases}$$

- There will always be one fewer dummy variable than the number of levels. The level with no dummy variable — African American in this example — is known as the *baseline*.

## 2개 이상의 범주형 변수 처리



# 로지스틱 회귀모형

# Logistic Regression

---

시스템경영공학부  
이지환 교수

# 모형의 분류

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression )
  - (No) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

# 모형의 분류: 로지스틱 회귀모형

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression)
  - (No) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

# 예시데이터

- 대학원 진학생 4명의 평균학점과 진학결과를 다음과 같이 수집하였다.
- 이 데이터를 사용하여, 평균학점을 통해 대학원 진학여부를 예측하는 모델을 만들고자 한다.

평균학점	결과
4.3	합격
1.7	불합격
2.5	불합격
3.3	합격

# 분류문제에 선형회귀 적용

- 예측하고자 하는 범주 각각에 숫자를 부여하고 회귀분석 학습

$$y \in \{0, 1\}$$

0: negative class (불합격)  
1: positive class (합격)

- 선형회귀 모형을 사용하면 학습이 가능함

$$y = \theta_0 + \theta_1 x$$

평균학점	결과	평균학점	결과
4.3	합격	4.3	1
0.7	불합격	0.7	0
2.5	불합격	2.5	0
3.3	합격	3.3	1

# 분류문제에 선형회귀 적용

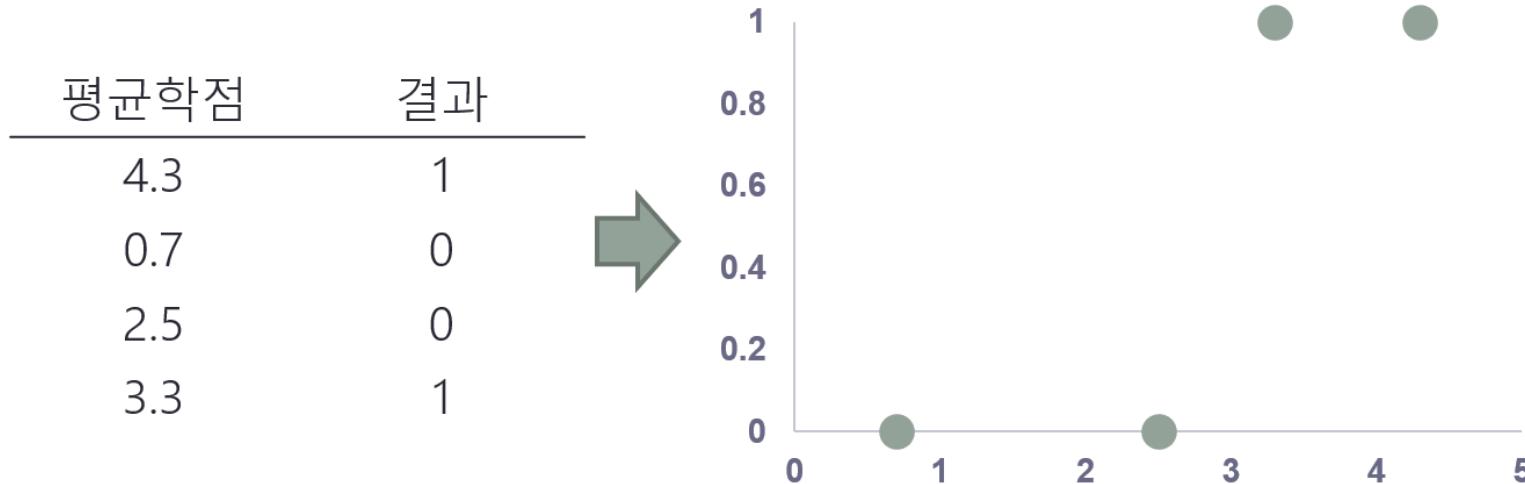
- 예측하고자 하는 범주 각각에 숫자를 부여하고 회귀분석 학습

$$y \in \{0, 1\}$$

0: negative class (불합격)  
1: positive class (합격)

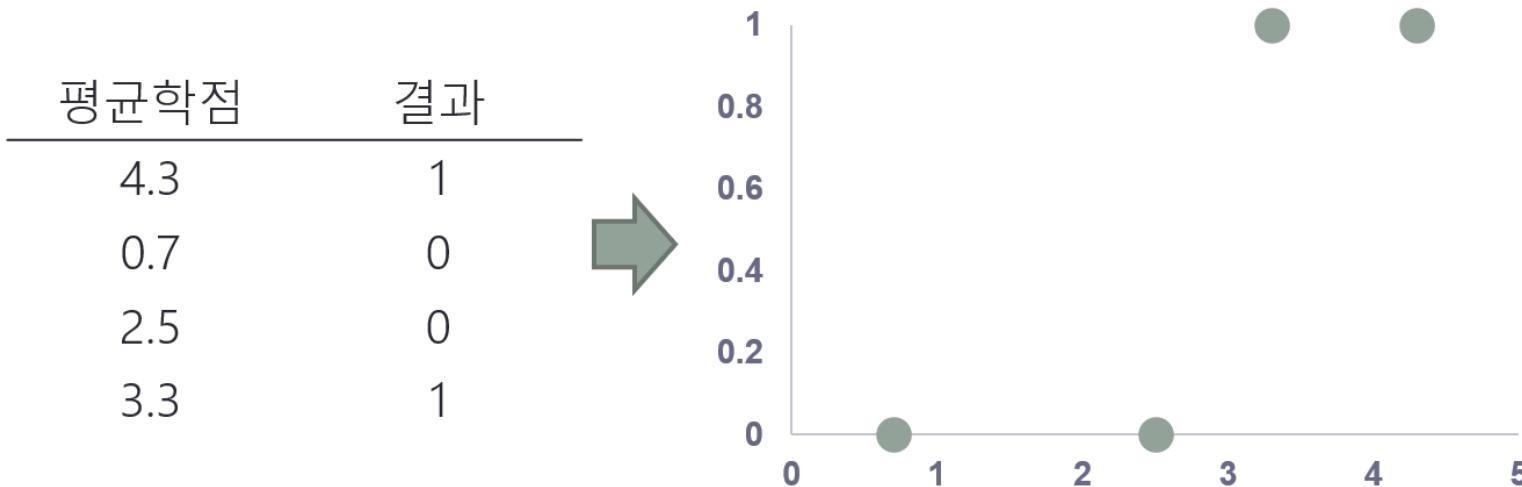
- 선형회귀 모형을 사용하면 학습이 가능함

$$y = \theta_0 + \theta_1 x$$



# 분류문제에 선형회귀 적용

- 예측된 값의 해석: 합격할 확률
  - 0.9 → 합격할 확률이 90%다
  - 0.1 → 합격할 확률이 0%
- 예측된 확률을 이용하여 **분류 의사결정** 가능
  - 만약 합격기준을 50%로 설정했다면 (변경가능)
    - $>=0.5$  : 합격
    - $<0.5$  : 불합격

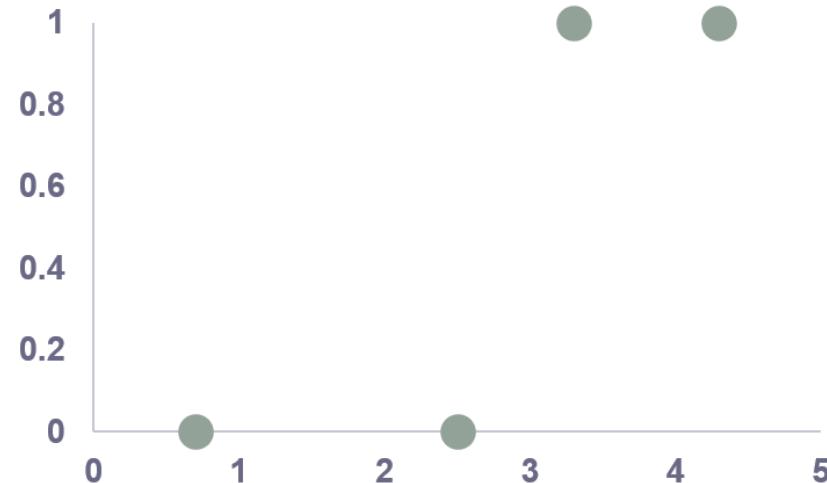


# 분류문제에 선형회귀 적용시 문제점

- 범주형 종속변수를 {0,1}로 바꾸고 선형회귀 학습시 문제점
- 예측된 값을 확률로 간주할수 없다.
  - 0~1 사이의 값을 벗어나는 경우 존재

$$y = \theta_0 + \theta_1 x$$

평균학점	결과
4.3	1
0.7	0
2.5	0
3.3	1



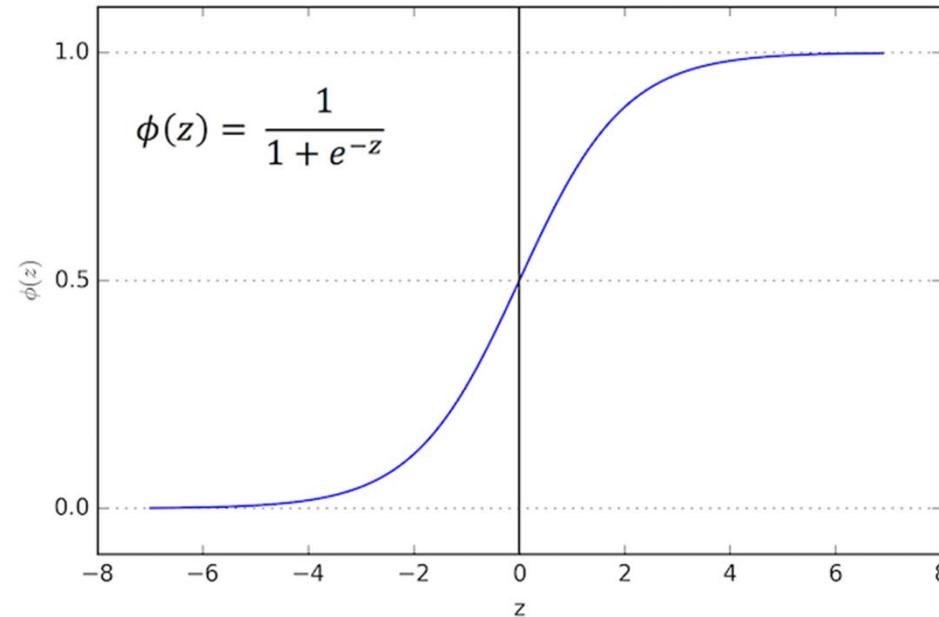
# 문제 해결을 위한 요구사항

- 종속변수(y)가 특정 레이블로 분류될 확률을 계산해야 함!
- 어떤 경우에라도 종속변수는 0~1사이의 값을 가지도록 모델링 되어야 함!

# 로지스틱 함수

- 로지스틱 함수

$$f(x) = \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$



- 성질
  - 음의 무한대 : 0에 가까워짐
  - 양의 무한대 : 1에 가까워짐
- 로지스틱회귀 → 로지스틱 함수를 선형회귀와 결합하여 특정 레이블로 분류될 확률을 예측하는 모형

# 선형회귀 모형의 가정(remind)

모델의 가정:  
학습시간(X)과 성적(y) 사이에는  
선형관계가 있을 것이다.



$$y = \theta_0 + \theta_1 x$$

# 로지스틱 회귀 모형의 가정

(1) i번째 데이터  $x^{(i)}$ 에 대하여  $y^{(i)} = 1$  일 확률을 다음과 같이 계산

$$f(x^{(i)}) = p(y^{(i)} = 1 | x^{(i)}) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x^{(i)})}}$$

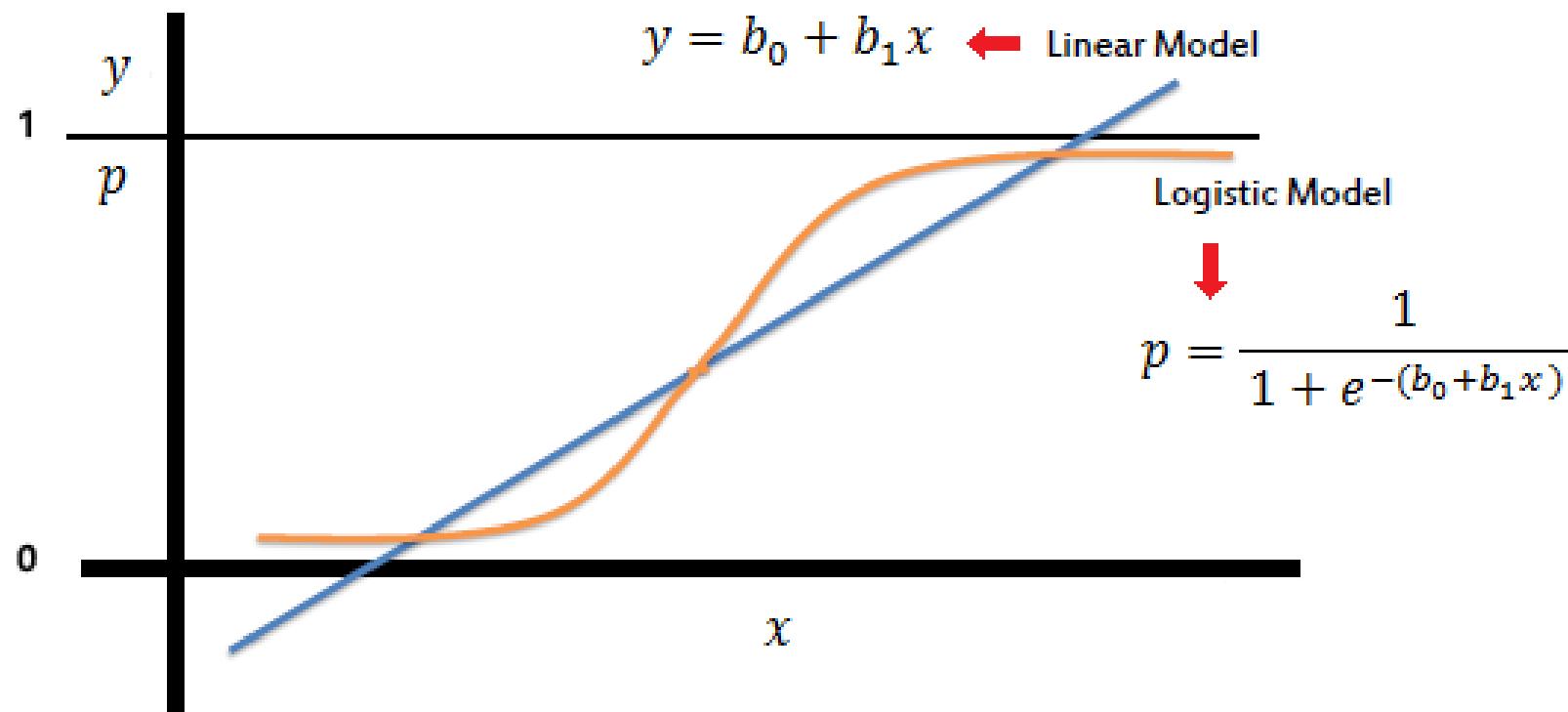
(2) 구해진 확률을 의사결정 기준  $th$ (예시 0.5) 비교하여 다음과 같이 예측

- 만약 확률이 0.5보다 크면 1로 분류
- 만약 확률이 0.5보다 작다면 0으로 분류

$$\widehat{y}^{(i)} = \begin{cases} 1 & \text{if } f(x^{(i)}) > 0.5 \\ 0 & \text{if } f(x^{(i)}) \leq 0.5 \end{cases}$$

# 로지스틱회귀 모형

- 예측되는 값을 온전한 확률값으로 표현할 수 있음



# 로지스틱 회귀 예측 예시

- 다음과 같이 패러미터가 주어져 있을때
- 주어진 데이터에 대하여 합격확률과 예측치를 계산해보고 실제 레이블(합격여부)와 비교해보시오.

패러미터  
 $\theta_0 = -2.5$   
 $\theta_1 = 1$

학점	회귀식	합격확률	예측	레이블
$x$	$\theta_0 + \theta_1 x^{(i)}$	$f(x^{(i)}) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$	$\hat{y}^{(i)}$	$y^{(i)}$
4.3	4.45			1
0.7	-0.95			0
2.5	1.75			0
3.3	2.95			1

# 로지스틱 회귀 학습 - 비용(cost)의 정의

- 로지스틱 회귀의 현재 패러미터가 X와 y사이의 관계를 얼마나 잘 나타내고 있는가?

패러미터  
 $\theta_0 = -2.5$   
 $\theta_1 = 1$

학점	회귀식	합격확률	예측치	레이블
$x$	$\theta_0 + \theta_1 x^{(i)}$	$f(x^{(i)}) = \frac{1}{1+e^{-(\theta_0+\theta_1x)}}$	$\hat{y}^{(i)}$	$y^{(i)}$
4.3	4.45	0.98	1	1
0.7	-0.95	0.28	0	0
2.5	1.75	0.85	1	0
3.3	2.95	0.95	1	1

# 로지스틱 회귀 학습 - 비용(cost)의 정의

- 로지스틱 회귀모형은 합격확률을 예측하는 모형
  - 실제 레이블이 1인(합격) → 예측된 합격확률이 높을수록
  - 실제 레이블이 0인(불합격) → 예측된 불합격 확률이 높을수록
  - 좋은 모형이라고 할수 있다 → 비용이 낮다

파라미터  
 $\theta_0 = -2.5$   
 $\theta_1 = 1$

합격확률	레이블
$f(x^{(i)}) = \frac{1}{1+e^{-(\theta_0+\theta_1x)}}$	$y^{(i)}$
0.98	1
0.28	0
0.85	0
0.95	1

# 로지스틱 회귀 학습 - 비용(cost)의 정의

- 로지스틱 회귀모형은 합격확률을 예측하는 모형
  - 실제 레이블이 1인(합격) → 예측된 합격확률이 높을수록
  - 실제 레이블이 0인(불합격) → 예측된 불합격 확률이 높을수록
  - 좋은 모형이라고 할수 있다 → 비용이 낮다

파라미터  
 $\theta_0 = -2.5$   
 $\theta_1 = 1$

합격확률	레이블
$f(x^{(i)}) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$	$y^{(i)}$
0.98	1
0.28	0
0.85	0
0.95	1

(0.98)

(1-0.28)

(1-0.85)

(0.95)

# 로지스틱 회귀모형의 비용함수

- 로지스틱 회귀모형은 합격확률을 예측하는 모형
  - 실제 레이블이 1인데이터(합격) → 예측된 합격확률이 높을수록
  - 실제 레이블이 0인데이터(불합격) → 예측된 불합격 확률이 높을수록
  - 좋은 모형이라고 할수 있다 → 비용이 낮다

$$f(x^{(i)}) = p(y^{(i)} = 1 | x^{(i)}) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x^{(i)})}}$$

- 비용함수

$$J(\theta_0, \theta_1) = -\frac{1}{m} \sum_{i=1}^m \left[ (y^{(i)} \log f(x^{(i)}) + (1 - y^{(i)}) \log(1 - f(x^{(i)})) \right]$$

# 비용함수 계산 예시

$$J(\theta_0, \theta_1) = -\frac{1}{m} \sum_{i=1}^m \left[ (y^{(i)} \log f(x^{(i)}) + (1-y^{(i)}) \log(1-f(x^{(i)})) \right]$$

파라미터  
 $\theta_0 = -2.5$   
 $\theta_1 = 1$

합격확률	레이블
$\hat{y} = \frac{1}{1+e^{-(\theta_0+\theta_1 x)}}$	$y^{(i)}$
0.98	1
0.28	0
0.85	0
0.95	1

(0.98)  
(1-0.28)  
(1-0.85)  
(0.95)

$$-\frac{1}{4} [0.98 + 0.72 + 0.15 + 0.95]$$

# 로지스틱 회귀함수의 확장

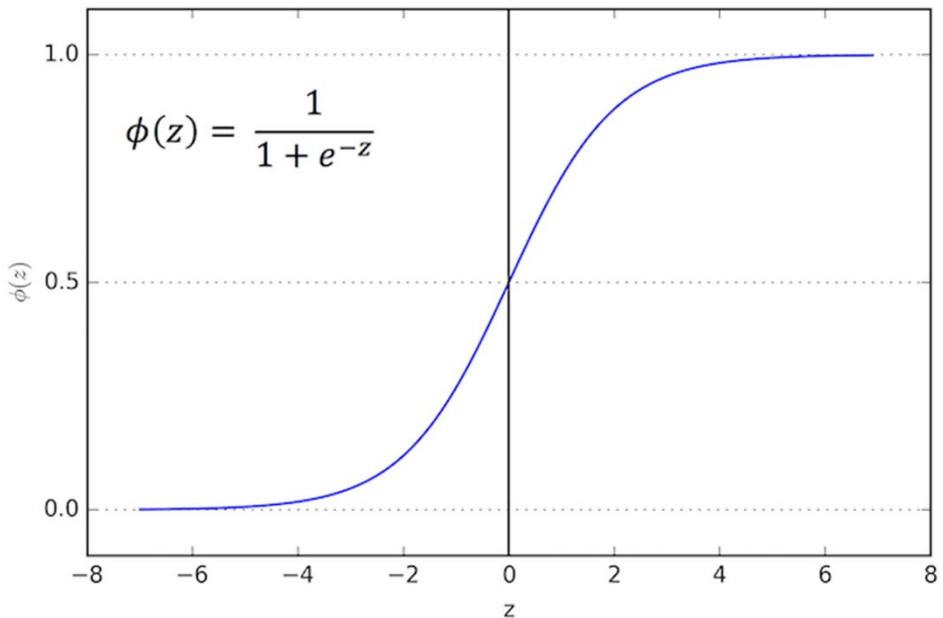
- 로지스틱회귀모형도 마찬가지로 선형회귀식이 사용되므로 다음과 같이 확장 가능
  - 독립변수의 변수 개수 확장 가능
  - 독립성 가정을 제거하여 변수 사이의 상호작용 표현
  - 선형성 가정을 제거하여 다차식 관계 표현

# 분류모형의 경계선

- 정리
  - 로지스틱회귀모형은 1의 값으로 분류될 확률을 예측
- 확률에 의한 의사결정
  - 만약 확률  $\hat{y}$ 가 0.5보다 크면 1로 분류
  - 만약 확률  $\hat{y}$ 가 0.5보다 작으면 0으로 분류
- 의사결정 기준이 0.5일때 경계선

$$\theta_0 + \theta_1 x = 0$$

$$[x = -\frac{\theta_0}{\theta_1}]$$



# 분류모형의 경계선

- 독립변수가 2개인 다중 로지스틱 회귀모형의 경계선 → 직선

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 > 0$$

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 < 0$$

- 2차식으로 이루어진 로지스틱 회귀모형의 경계선 → 원으로 표현

# 지도학습 모형의 평가

# Supervised Model

# Evaluation

---

시스템경영공학부  
이지환 교수

# 모델의 예측 정확도 평가

- 모델이 주어진 데이터를 얼마나 잘 맞추는가?
- (현재까지 측정방법)
  - Step 1: 데이터  $(X, y)$ 를 사용하여 모형을 학습
  - Step 2: 훈련에 사용된 데이터  $x^{(i)}$ 들을 다시 모형에 입력하여 예측값  $\hat{y}^{(i)}$  계산
  - Step 3: 정답  $y$ 와  $\hat{y}$ 사이의 차이를 측정
- 회귀모형의 정확도 평가
  - RMSE (Rooted Mean Squared Error)
  - R2 Score
- 분류모형의 정확도 평가 지표
  - Accuracy
  - Precision
  - Recall
  - ROC

# 회귀 모형의 정확도 평가 : RMSE

- 선형회귀 뿐 아니라 모든 다양한 회귀모형에 공통으로 사용 가능
- RMSE(Residual Mean Squared Error)가 사용됨

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

$x$	$y$	$\hat{y} (1.67 + 1.5x)$
1	3	3.17
2	5	4.67
3	6	6.17

}

RMSE

$= \sqrt{\frac{0.17^2 + 0.33^2 + 0.17^2}{3}} = ..$

Scale dependant

# 회귀 모형의 정확도 평가 : RMSE

- 선형회귀 뿐 아니라 모든 다양한 회귀모형에 공통으로 사용 가능
- RMSE(Residual Mean Squared Error)가 사용됨

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

- 예측값과 실제값이 평균적으로 얼마나 떨어져 있는지를 측정
- Scale dependent*

# 회귀모형의 정확도 평가: R2 Score

- 수학적 정의

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

$$TSS = \sum (y_i - \bar{y})^2 \quad RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2.$$

- TSS: 회귀모형이 없이 모든 데이터에 상관없이  $y$ 값의 평균으로 똑같이 예측한다고 했을 때 편차
- RSS: 회귀모형을 적용 후 실제 변수와 예측된 변수사이의 제곱 합

- R2Score
  - measures the proportion *of variability in Y that can be explained using X*.
  - 단위가 사라지고 비율로 표현된다
  - 어떤 문제던지 0~1사이의 점수로 측정 가능 (*Scale-Free*)

# 모형의 정확도 평가: R-squared

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

$$\text{TSS} = \sum (y_i - \bar{y})^2 \quad \text{RSS} = \sum_{i=1}^n (y_i - \hat{y}_i)^2.$$

↳ 모델 적용 전 편차                          ↳ 모델 적용 후 편차

x	y	$\hat{y}$ ( $1.67 + 1.5x$ )
1	3	3.17
2	5	4.67
3	6	6.17

Scale free

$$0 \leq R^2 \leq 1$$

$$\bar{y} = \frac{3+5+6}{3}$$

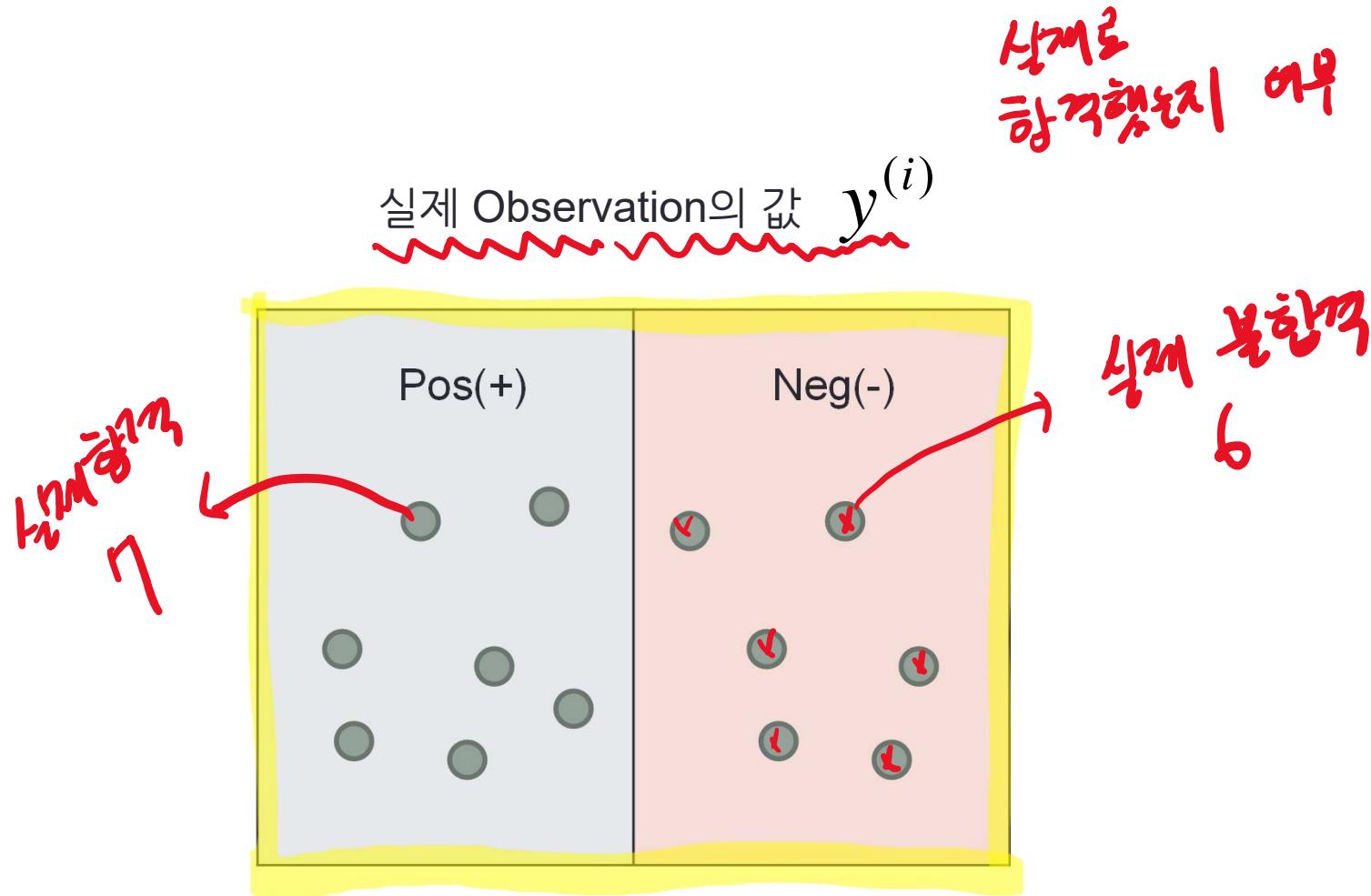
# R2score가 RMSE보다 항상 좋은가?

- RMSE:
  - Scale에 의존하며, 절대적인 지표
  - 단일 RMSE 측정만으로는 도출된 값이 좋은것인지 알 수 없다.
  - 다른 회귀모형들과 비교 필요
- R2Score
  - 상대적 지표며, Scale-free
  - 모든 상황에서 0~1로 점수화가 가능
  - 하지만 데이터 활용분야에 따라 R2 Score는 상대적으로 해석되어야 한다.
    - 정교한 물리실험에서 발생하는 데이터 R2 score가 낮다면 심각한 결과
    - 사회과학 등 비교적 비정확한 데이터가 생성되는 경우 → 낮은 R2 Score도 받아들여짐

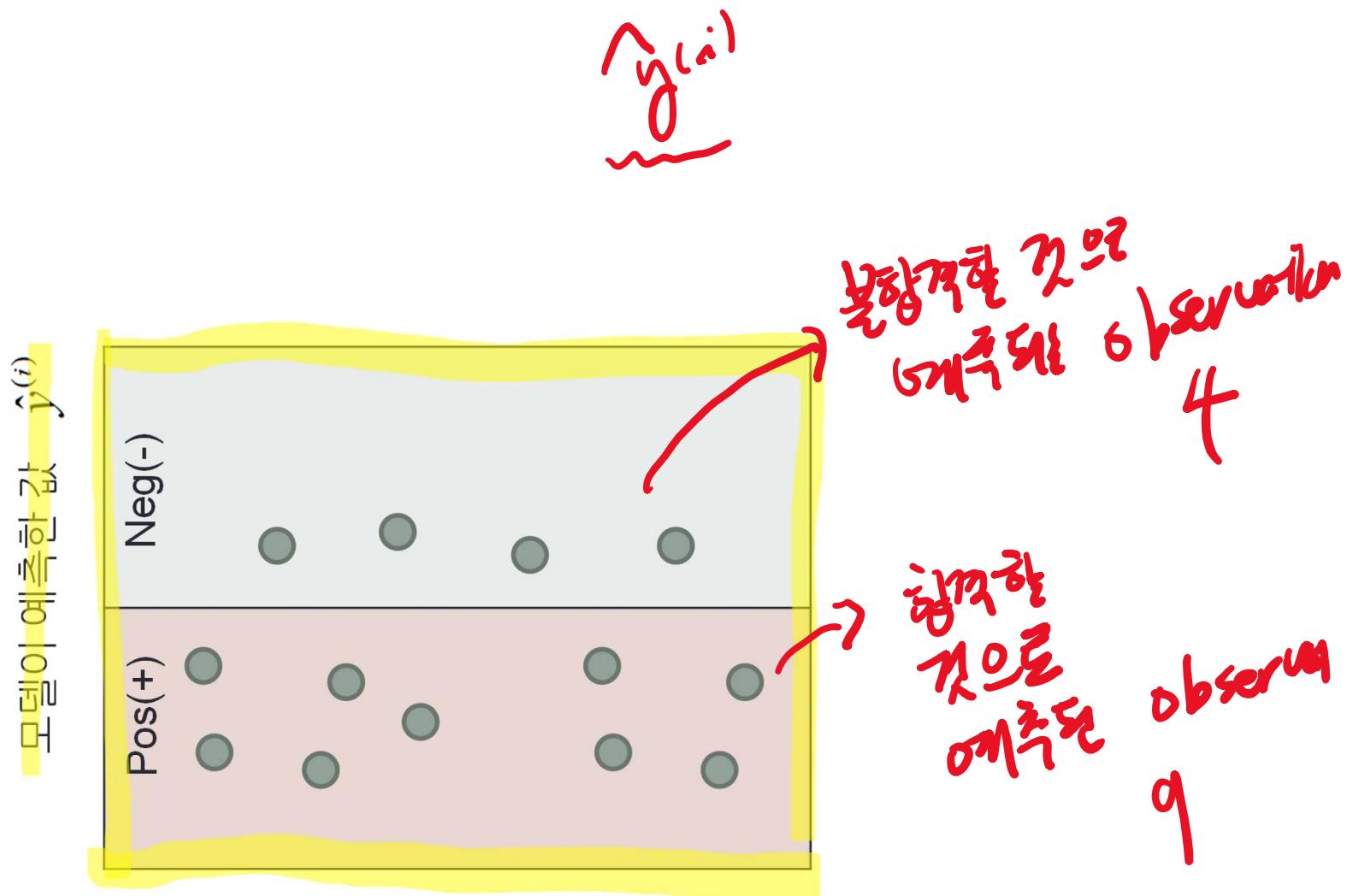
# 분류모델의 정확도

- Notation:
  - 1(Positive)
  - 0(Negative)
- 좋은 분류모형
  - 정답이 Positive인 데이터를 Positive로 예측
  - 정답이 Negative인 데이터를 Negative로 예측
- 분류모형의 평가에 사용되는 지표
  - Accuracy
  - Precision
  - Recall
  - ROC-curve
- 분류모델의 정확도 계산을 위해 먼저 Confusion Matrix를 작성해야한다.

# 실제 Observation의 값



# 모델이 예측한 값



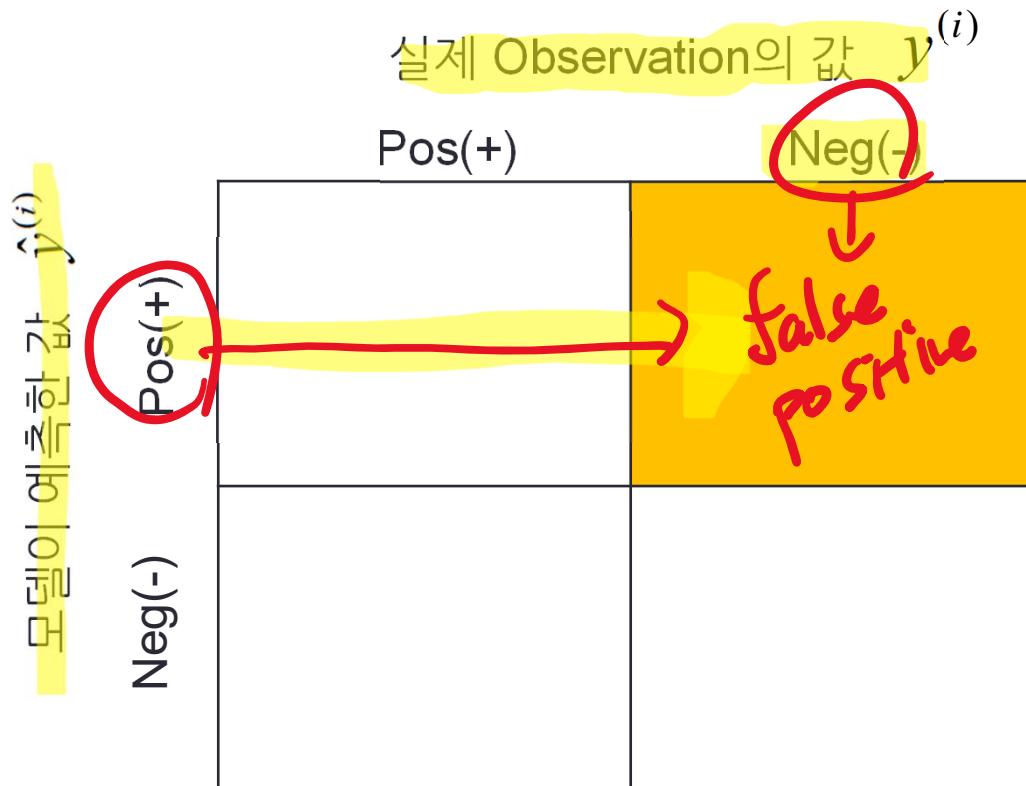
# True Positive

- 모델이 Positive이라고 예측하고, 실제로도 그 값이 Positive인 경우
  - True Positive (모델이 Positive라고 예측한 것이 True 였다.)

		$y^{(i)}$ 실제 Observation의 값
		Pos(+) ↗
$\hat{y}^{(i)}$ 모델이 예측한 값	Pos(+)	True positive
	Neg(-)	

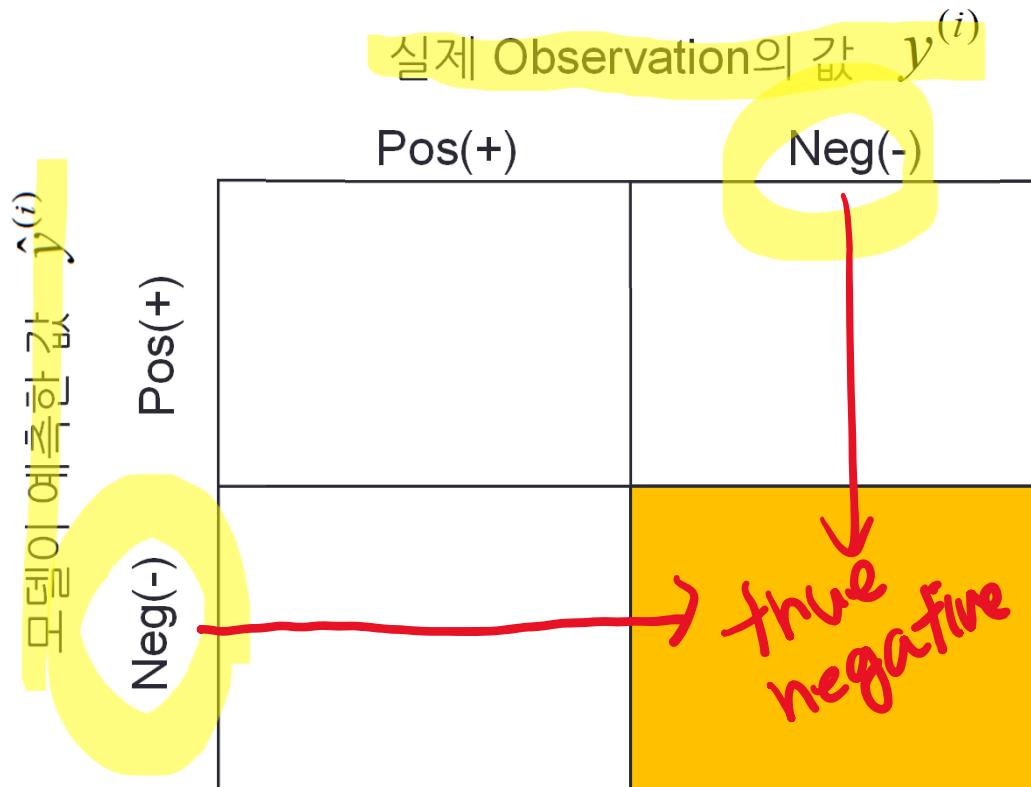
# False Positive

- 모델이 Positive이라고 예측했지만, 실제로는 Negative이었던 경우
  - False Positive (모델이 Positive라고 예측한 것이 False 였다.)



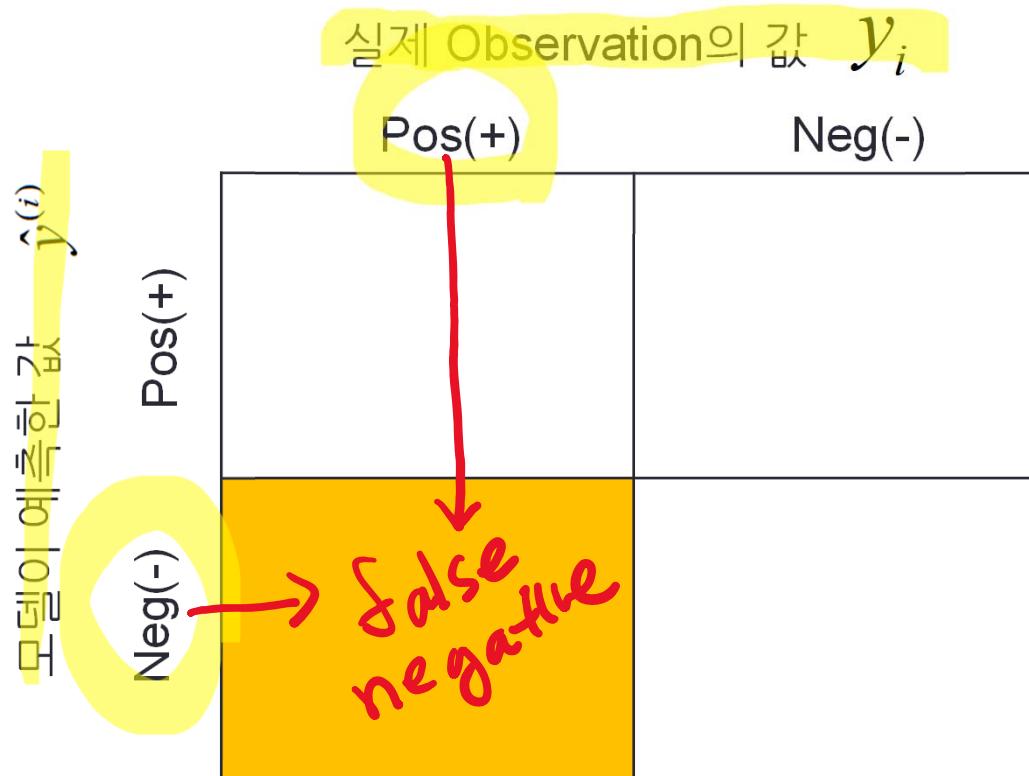
# True Negative

- 모델이 Negative라고 예측하고, 실제로도 Negative이었던 경우
  - True Negative (모델이 Negative라고 예측한 것이 True 였다.)



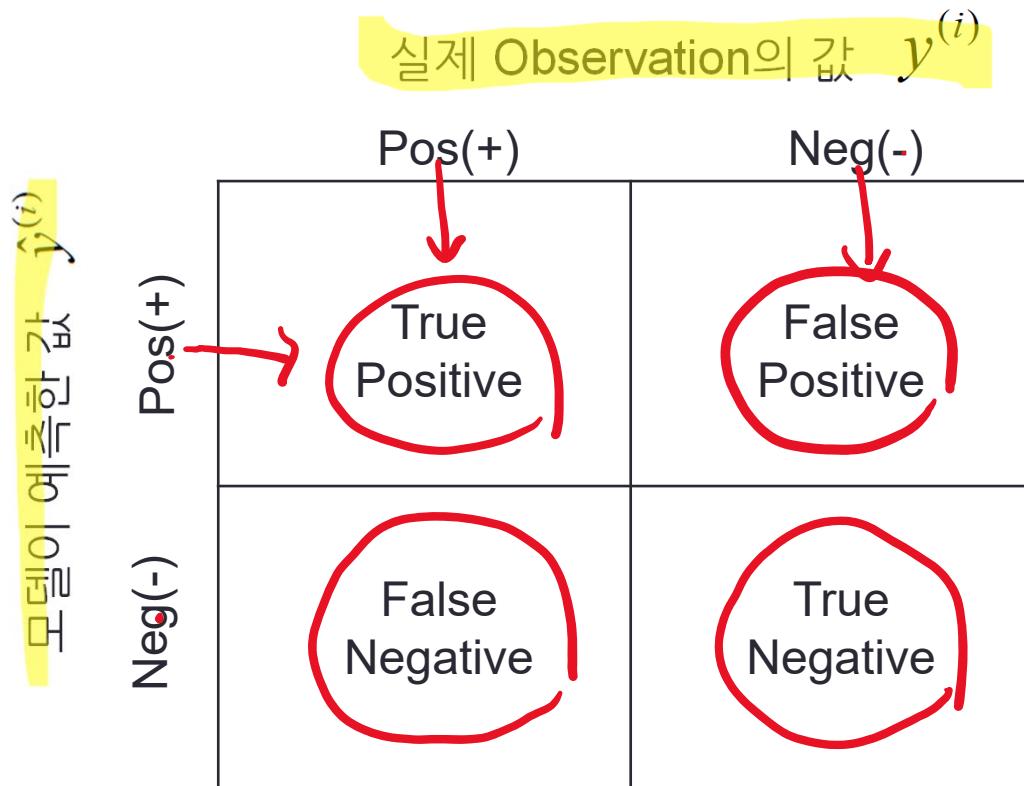
# False Negative

- 모델이 Negative라고 예측했지만, 실제로는 True이었던 경우
  - False Negative (모델이 Negative라고 예측한 것이 False 였다.)



# Confusion Matrix

- True Positive, True Negative, False Positive, False Negative에 해당하는 Observation 수를 계산



# 분류모델의 정확도 - Confusion Matrix

- Confusion Matrix는 의사결정 기준값에 따라 달라짐
- 의사결정 기준(threshold) 0.5에서의 confusion matrix

	$f(x^{(i)})$	$\hat{y}^{(i)}$	$y^{(i)}$
1	0.4	0	
2	0.9	1	
3	0.7	0	
4	0.7	1	
5	0.3	1	
6	0.4	0	
7	0.6	1	

# 분류모델의 정확도 - Confusion Matrix

0.5

0.5

	$f(x^{(i)})$	$\hat{y}^{(i)}$	$y^{(i)}$
1	0.4	0	0
2	0.9	1	1 ✓
3	0.7	1	✗ 0
4	0.7	1	1 ✓
5	0.3	0	✗ 1
6	0.4	0	0
7	0.6	1	1 ✓

실제 Observation의 값  $y^{(i)}$

		모델이 예측한 값 $\hat{y}^{(i)}$	
		Pos(+)	Neg(-)
Pos(+)	Pos(+)	true positive 3	false positive 1
	Neg(-)	false negative 1	true negative 2

# 분류모델의 정확도 평가지표 - 정확도

		실제 Observation의 값 $y^{(i)}$	
		Pos(+)	Neg(-)
모델이 맞춘 값 $\hat{y}^{(i)}$	Pos(+)	TP	FP
	Neg(-)	FN	TN

- 정확도(Accuracy)  
 $(TP+TN)/(TP+FP+FN+TN)$

= 모델이 맞춘 것수 (true)  
전체 데이터

- 오차율  
1-정확도

# 정확도 계산

		실제 Observation의 값 $y^{(i)}$	
		Pos(+)	Neg(-)
모델이 예측한 값 $\hat{y}^{(i)}$	Pos(+)	3 <i>true positive</i>	1
	Neg(-)	1	2 <i>true negative</i>

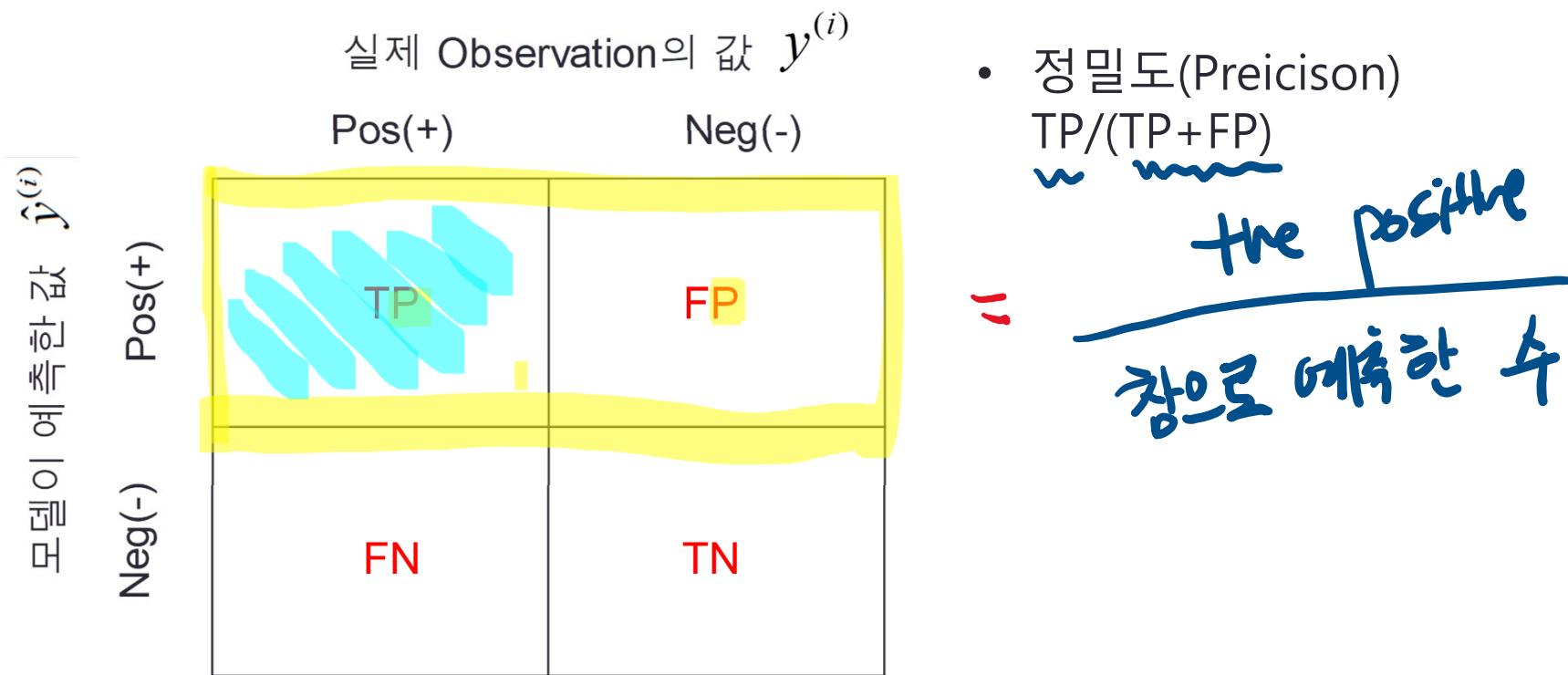
- 정확도(Accuracy)  
 $(TP+TN)/(TP+FP+FN+TN)$

$$= \frac{3+2}{3+1+1+2} = \frac{5}{7}$$

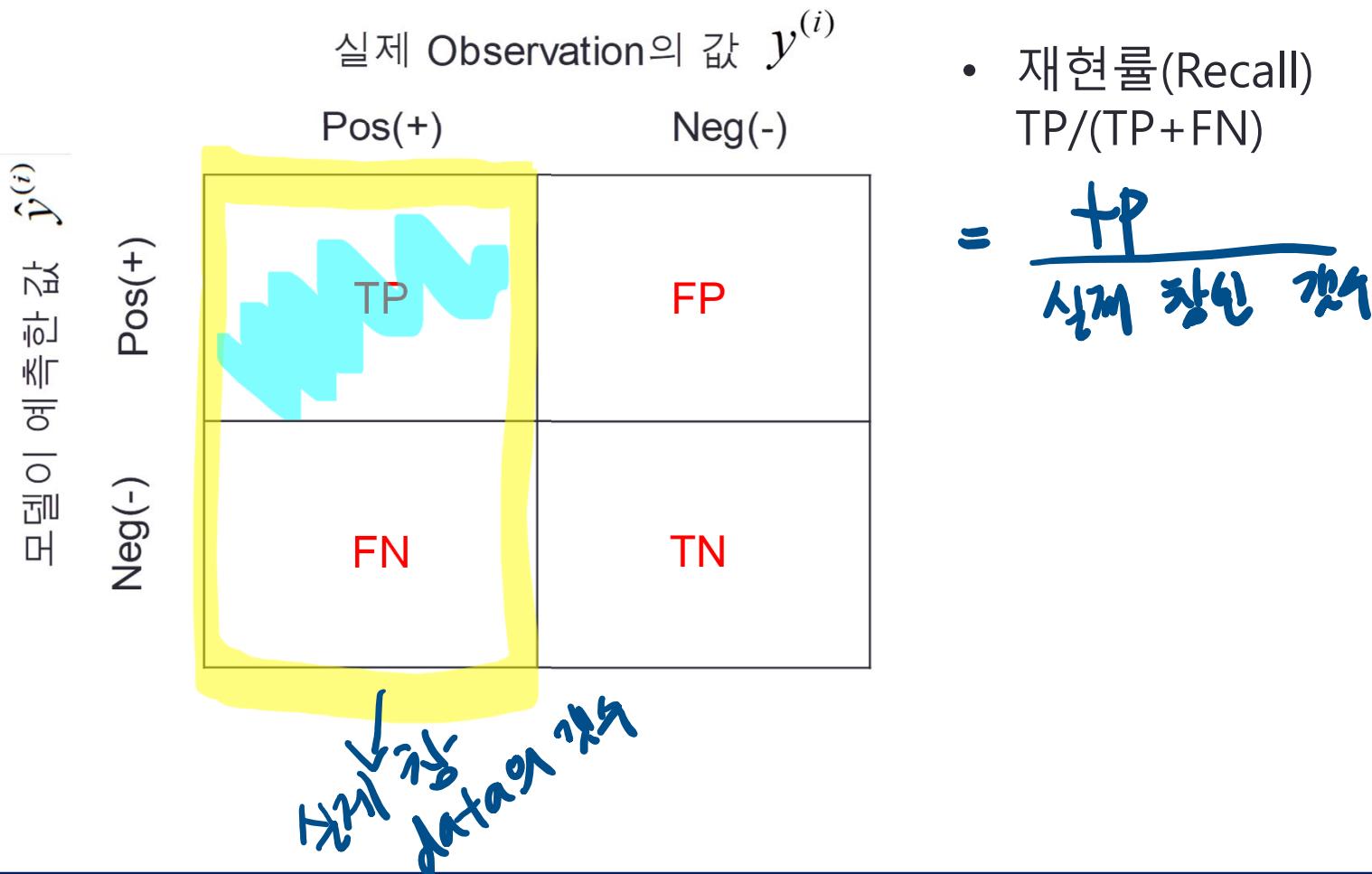
- 오차율  
1-정확도

$$= 1 - \frac{5}{7} = \frac{2}{7}$$

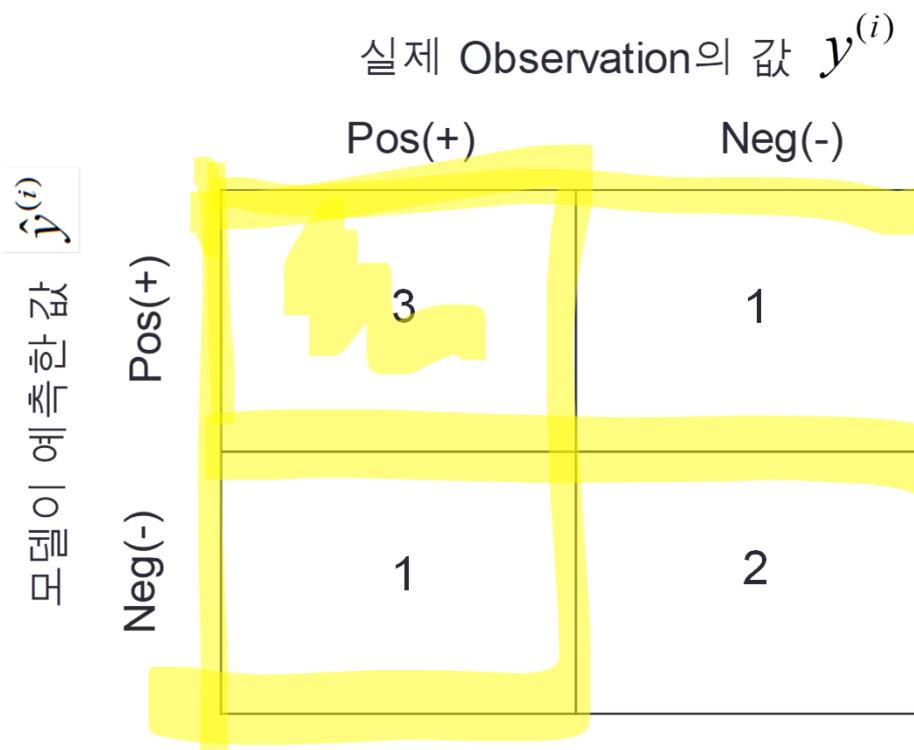
# 분류모델의 정확도 평가지표 - 정밀도



# 분류모델의 정확도 평가지표 - 재현률



# 정밀도 재현률 계산



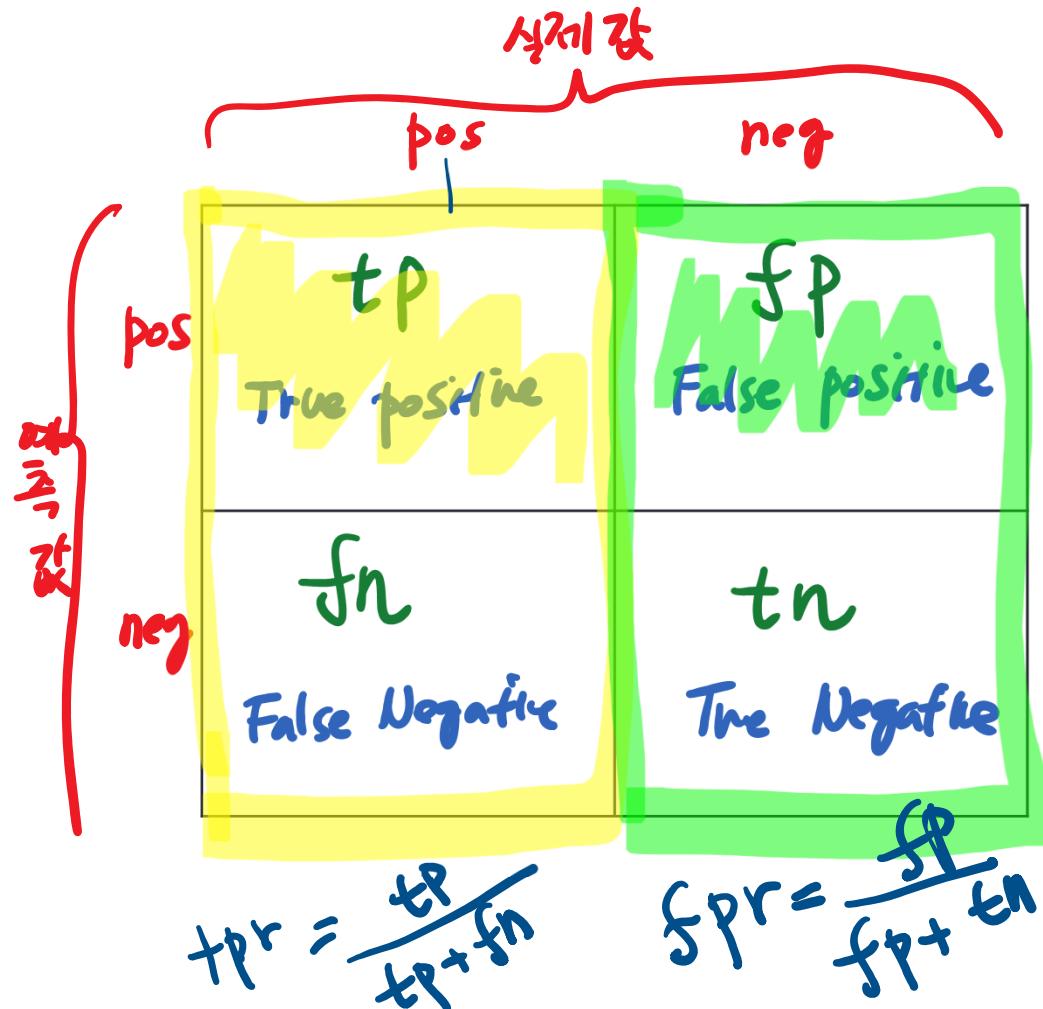
- 정밀도(Precision)  
 $TP/(TP+FP)$

$$\frac{3}{3+1} = \frac{3}{4}$$

- 재현률(Recall)  $\rightarrow$   
 $TP/(TP+FP)$

$$\frac{3}{3+1} = \frac{3}{4}$$

# 분류모형의 정확도 - TPR/FPR



True Positive Rate  
(recall)

→ 실제 positive 중  
잘 분류된 값의 비율

$$\frac{tp}{tp+fn}$$

False Positive Rate

→ 실제 negative 중  
잘못 분류된 값의 비율

$$\frac{fp}{fp+tn}$$

# TPR/FPR 사이의 관계

0.5 → 0.7

- Threshold가 낮아지면 무슨 일이 일어나나? → 쉽게 합격시켜줌
  - TPR 증가 → (합격해야 할 사람을 합격시키는 비율)
  - FPR 증가 → (불합격해야 할 사람을 합격시키는 비율)

		실제 값	
		pos	neg
예측 값	pos	tp True positive	fp False positive
	neg	fn False Negative	tn True Negative

True Positive

Rate

→ 실제 positive 중  
잘 분류된 값의 비율

$$\frac{tp}{tp+fn}$$

False Positive Rate

→ 실제 negative 중  
잘못 분류된 값의 비율

$$\frac{fp}{fp+tn}$$

# TPR/FPR 사이의 관계

- 0%
- Threshold가 높아지면 무슨일이 일어나나? → 잘 합격시켜주지 않음
    - TPR 감소 → (합격해야 할 사람을 합격시키는 비율)
    - FPR 감소 → (불합격해야 할 사람을 합격시키는 비율)

		실제 값	
		pos	neg
예측 값	pos	tp True positive	fp False positive
	neg	fn False Negative	tn True Negative

True Positive

Rate

→ 실제 positive 중  
잘 분류된 값의 비율

$$\frac{tp}{tp+fn}$$

False Positive Rate

→ 실제 negative 중  
잘못 분류된 값의 비율

$$\frac{fp}{fp+tn}$$

# 의사결정 경계값(Threshold)에 따른 변화

Threshold 0.2  
널널한 의사결정  
(웬만하면 Positive)

	$f(x^{(i)})$	$\hat{y}^{(i)}$	$y^{(i)}$
1	0.4	1	0 ✕
2	0.9	1	1
3	0.7	1	0 ✕
4	0.7	1	1
5	0.3	1	1
6	0.4	1	0 ✕
7	0.6	1	1

$$tpr = \frac{4}{4} = 1 > \uparrow$$

$$fpr = \frac{3}{3} = 1 > \uparrow$$

Threshold 0.8  
빡빡한 의사결정  
(웬만하면 Negative)

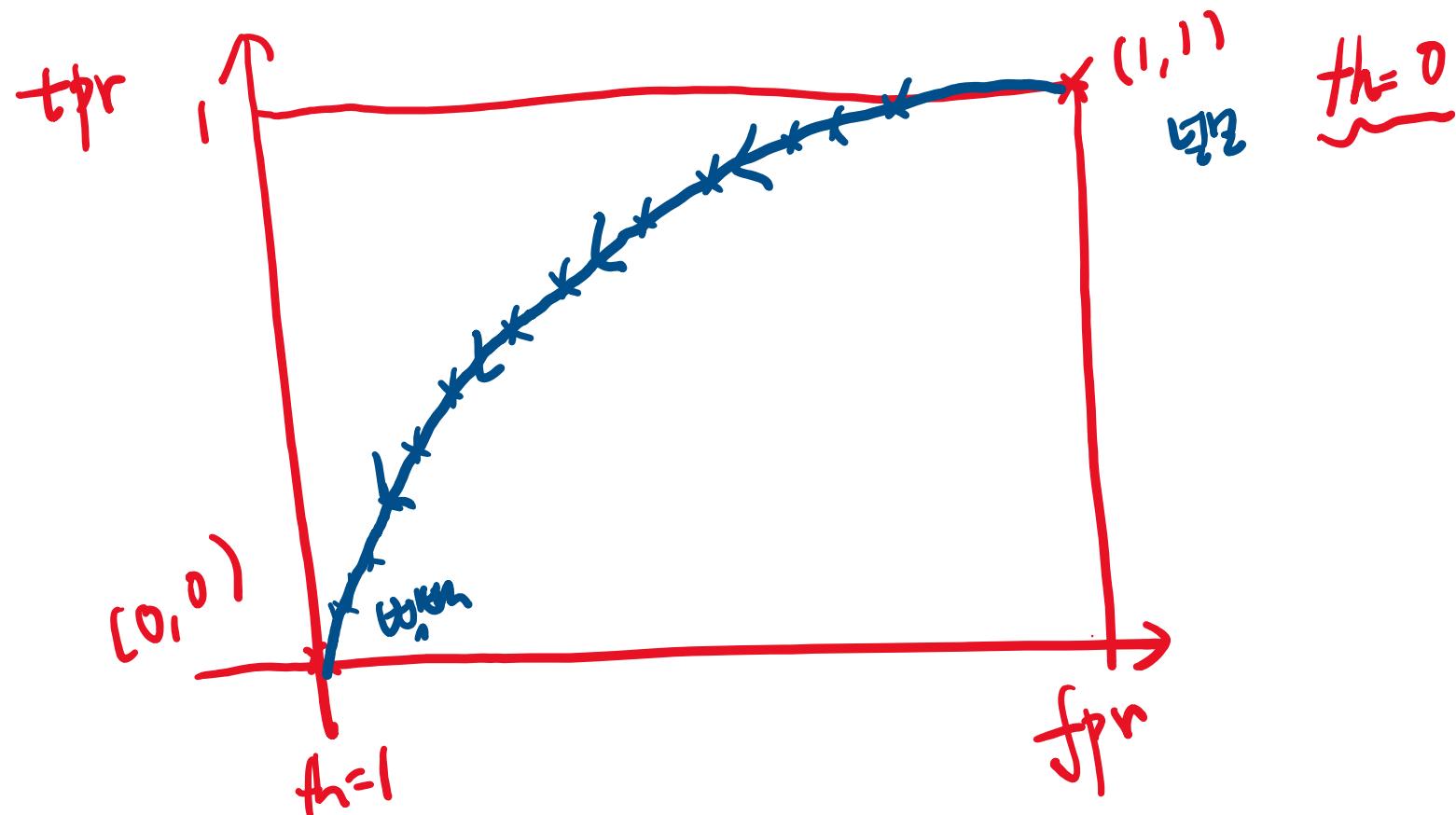
	$f(x^{(i)})$	$\hat{y}^{(i)}$	$y^{(i)}$
1	0.4	0	0 ✦
2	0.9	1	1
3	0.7	0	0 ✦
4	0.7	0	1
5	0.3	0	1
6	0.4	0	0 ✦
7	0.6	0	1

$$tpr = \frac{1}{4} = 0.25 > \downarrow$$

$$fpr = \frac{0}{3} = 0 > \downarrow$$

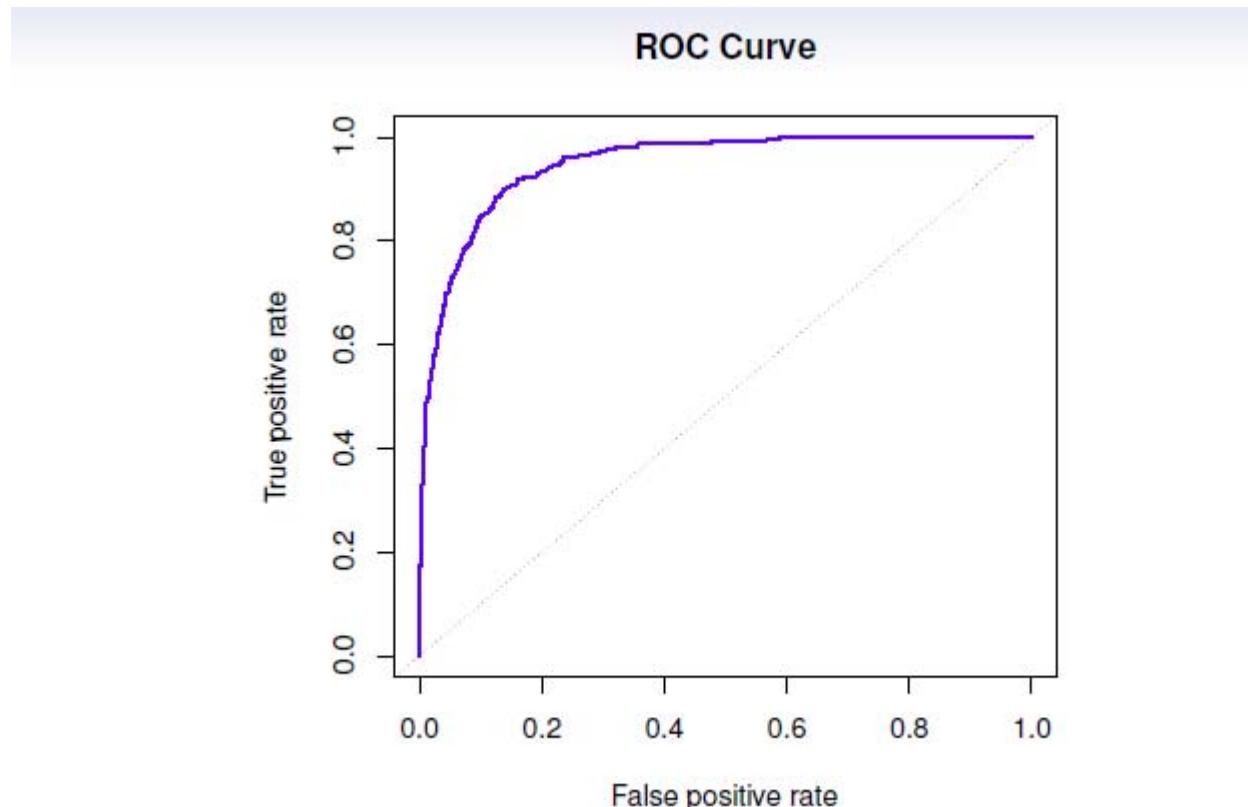
# ROC (Receiver Operating Characteristics)

- 가로축을 FPR로 세로축을 TPR로 하여 threshold를 바꾸어가며 그린 그래프



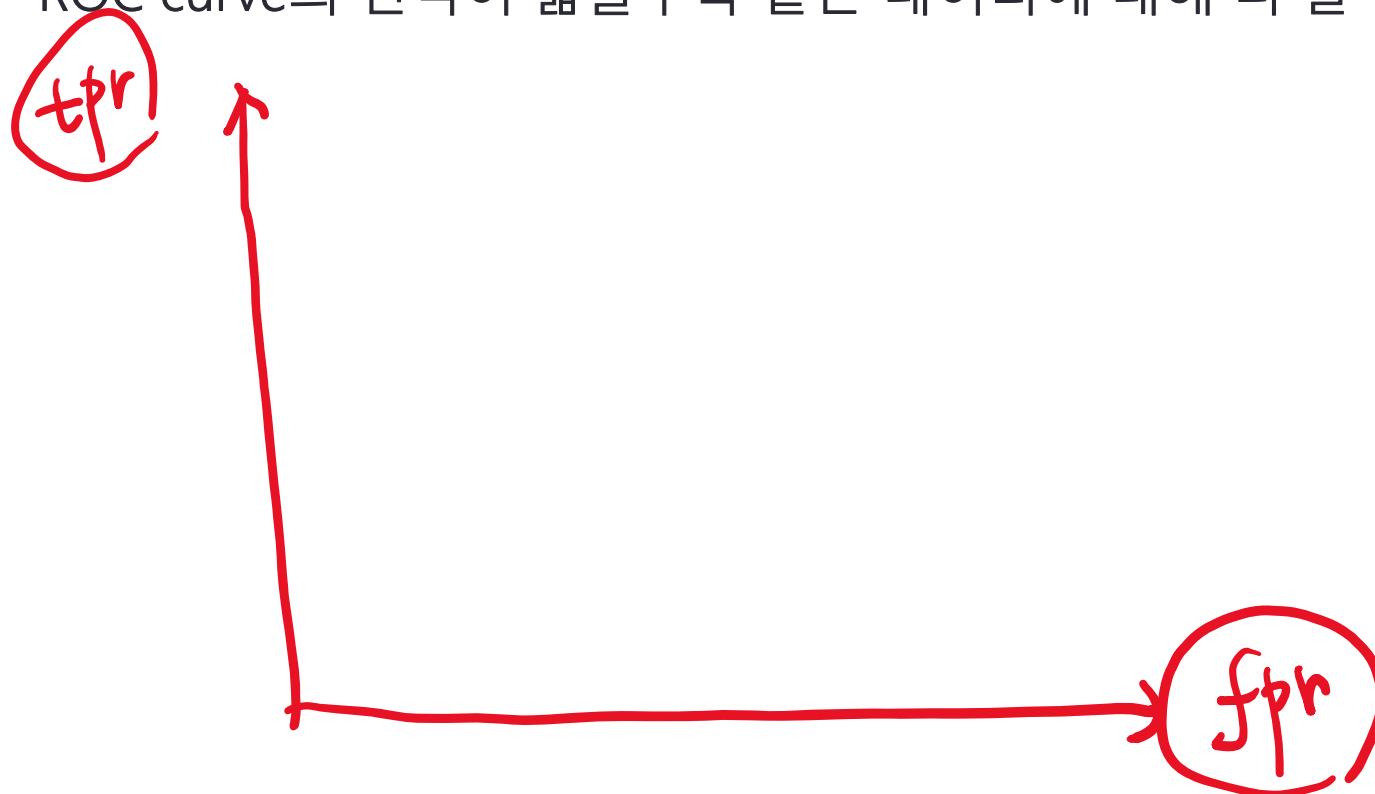
# ROC (Receiver Operating Characteristics)

- 가로축을 Specificity로 세로축을 Sensitivity로 하여 시각화 한 그래프



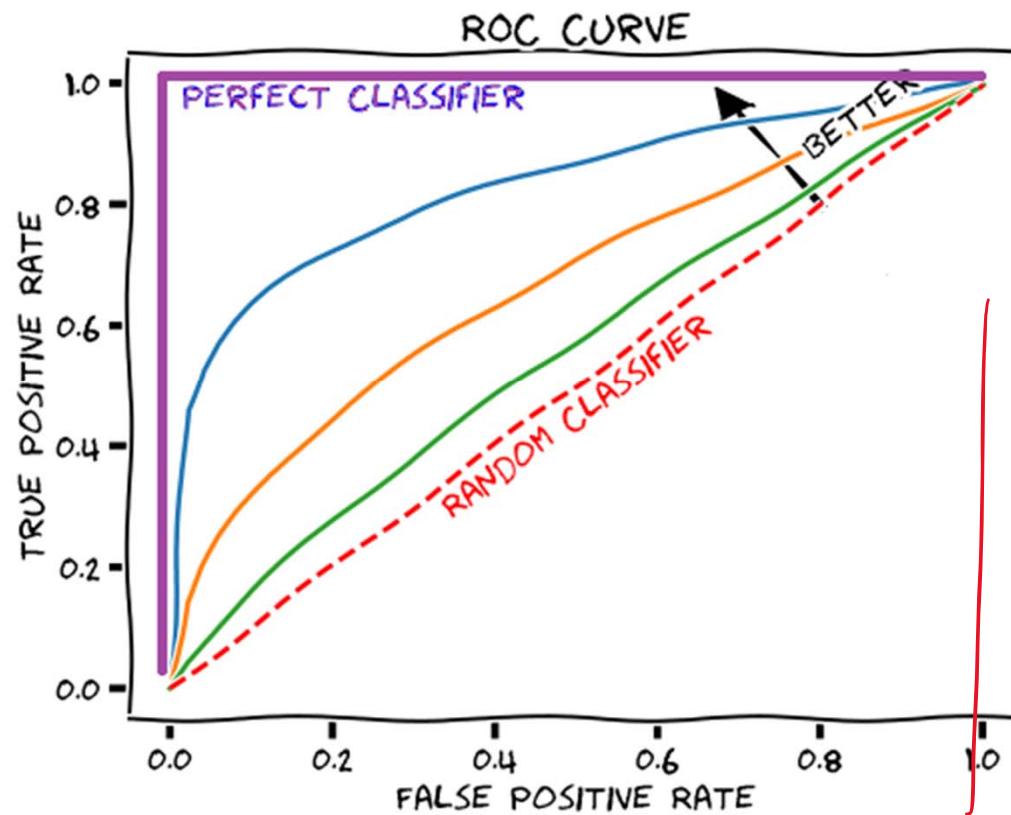
# ROC 커브의 활용

- Receiving Operator Characteristics
- 의사결정 기준을 바꾸어가며 fpr과 tpr을 그린 것
- ROC curve의 면적이 넓을수록 같은 데이터에 대해 더 잘 분류한 모형



# ROC 커브의 활용

- Receiving Operator Characteristics
- 의사결정 기준을 바꾸어가며 fpr과 tpr을 그린 것
- ROC curve의 면적이 넓을수록 같은 데이터에 대해 더 잘 분류한 모형



# 일반화와 과적합

---

시스템경영공학부  
이지환 교수

# 모델(Model)

- 예측과 추론을 위해서는 데이터를 활용하여 모델을 만들어야 한다
- 모델
  - 독립변수와 종속변수의 관계에 대한 가설을 수학적으로 표현한 것
  - A machine learning model can be a mathematical representation of a relationship between X and y
- 모델
  - 통계학습모델, 머신러닝모델

# 모델의 예시

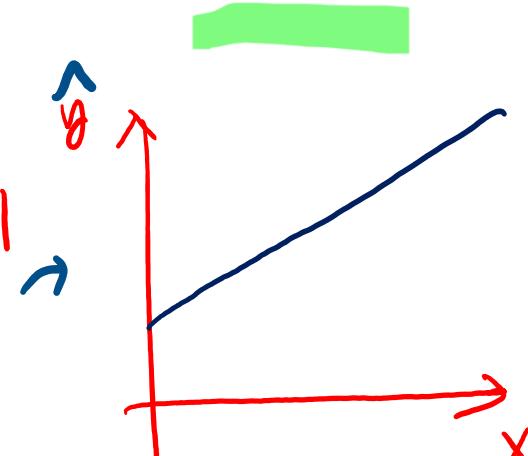
- 같은 데이터라 할 지라도  $y = f(x)$  관계에 대한 다양한 모델을 적용해 볼 수 있음

	X	y
1	0.3	40000
2	0.5	48000
3	5	70000
49	11	120000
50	6	65000

선형회귀

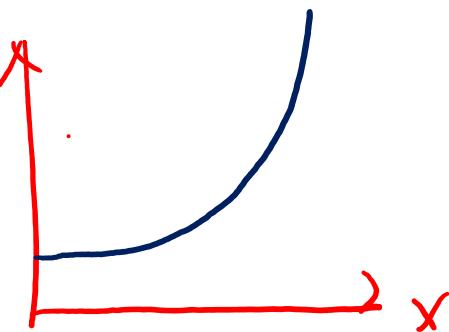
① 연봉이 경력에 비례

$$y = \beta_0 + \beta_1 X$$



② 연봉이 경력이 제곱에 비례 2차식

$$y = \beta_0 + \beta_1 X + \beta_2 X^2$$



③ deep learning  
Random forest

# 데이터

- 문제: 경력을 통해 연봉을 예측하고자 한다

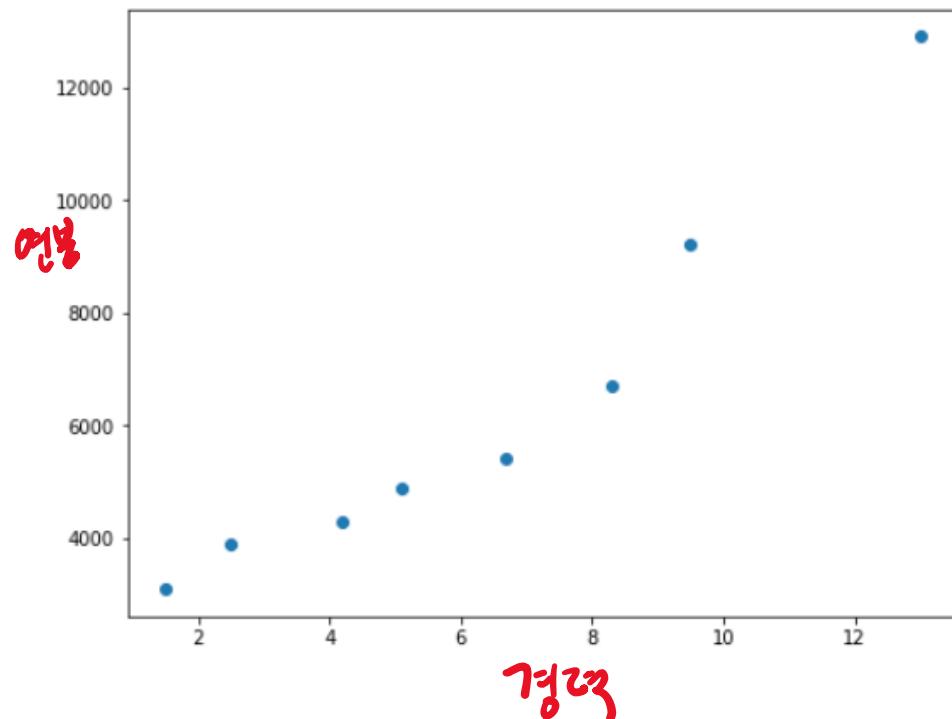


obs	X (경력)	y(연봉)
1	1.5	3100
2	2.5	3900
3	4.2	4300
4	5.1	4900
5	6.7	5400
6	8.3	6700
7	9.5	9200
8	13	12900

# 데이터

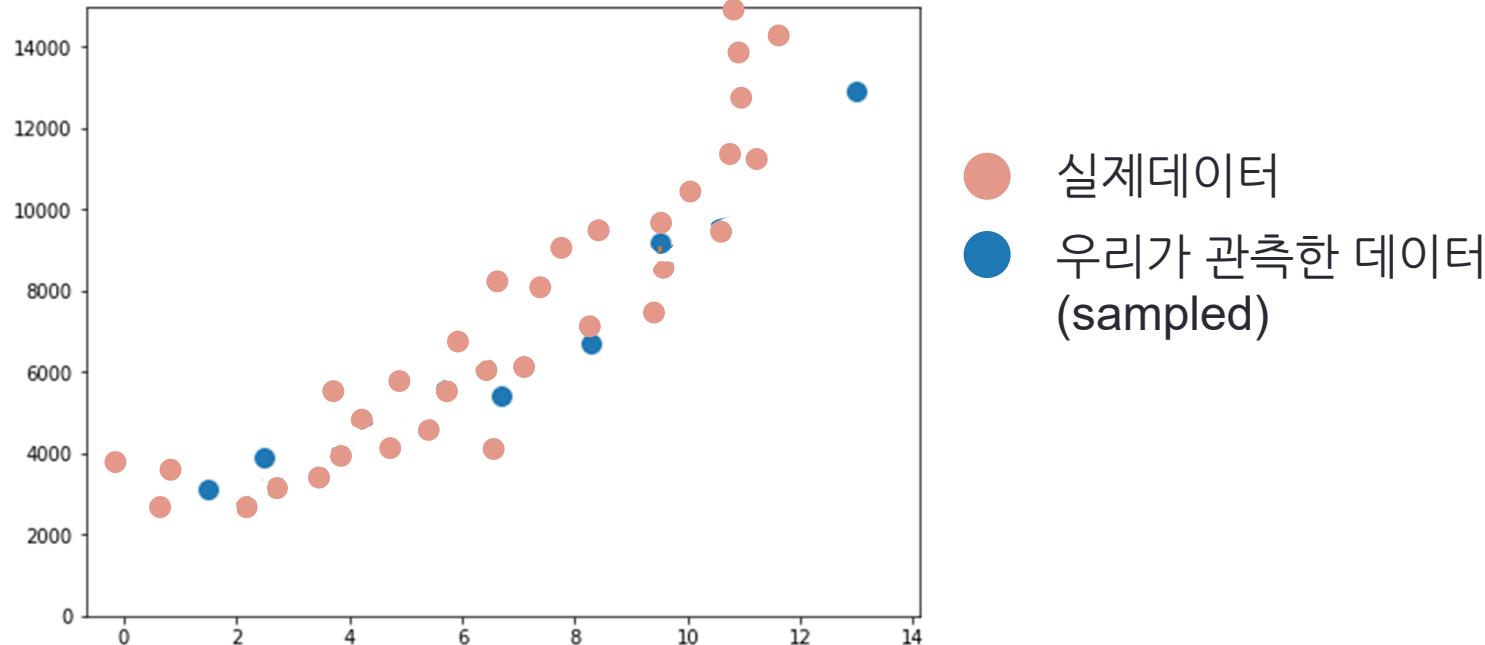
- 문제: 경력을 통해 연봉을 예측하고자 한다
- 회귀모형

obs	X (경력)	y(연봉)
1	1.5	3100
2	2.5	3900
3	4.2	4300
4	5.1	4900
5	6.7	5400
6	8.3	6700
7	9.5	9200
8	13	12900



# 실제 데이터 vs 관측된 데이터

- (모르지만) 실제 데이터: 이 세계에 존재하는 모든데이터
- 관측된 데이터: 우리가 확보한 데이터 (실제 데이터에서 샘플링됨)



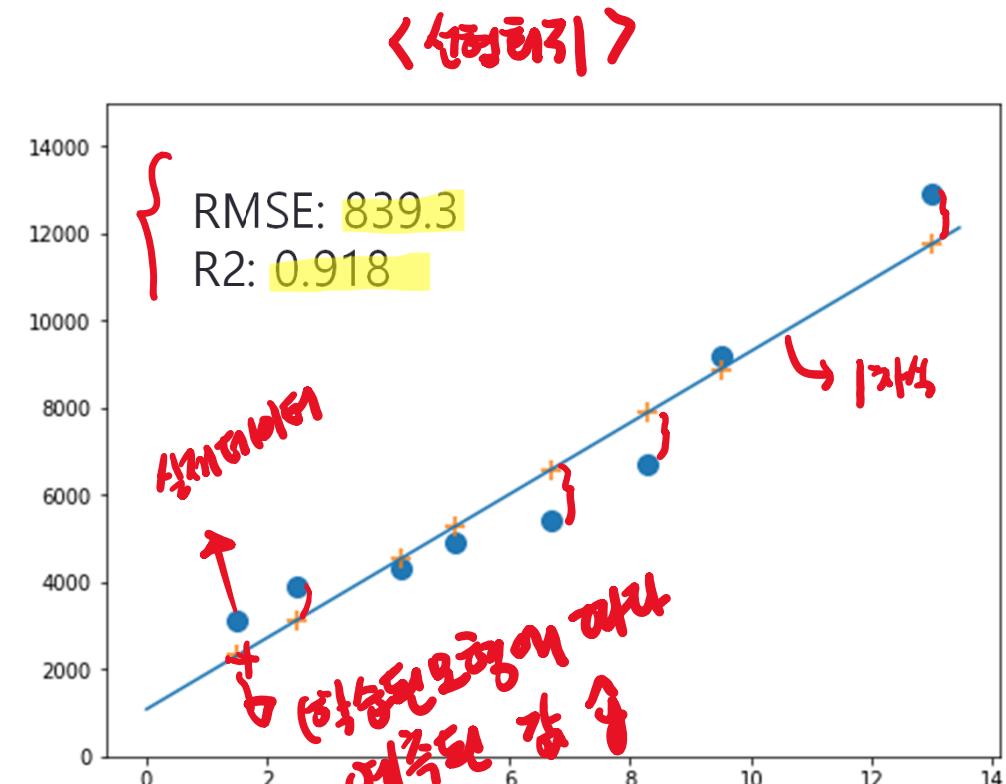
# 선형회귀모형

- 관측된 데이터만을 이용하여 학습할 수 밖에 없다.
- 선형회귀 모형 적용!

$$y = \beta_0 + \beta_1 x$$

obs	X (경력)	y(연봉)
1	1.5	3100
2	2.5	3900
3	4.2	4300
4	5.1	4900
5	6.7	5400
6	8.3	6700
7	9.5	9200
8	13	12900

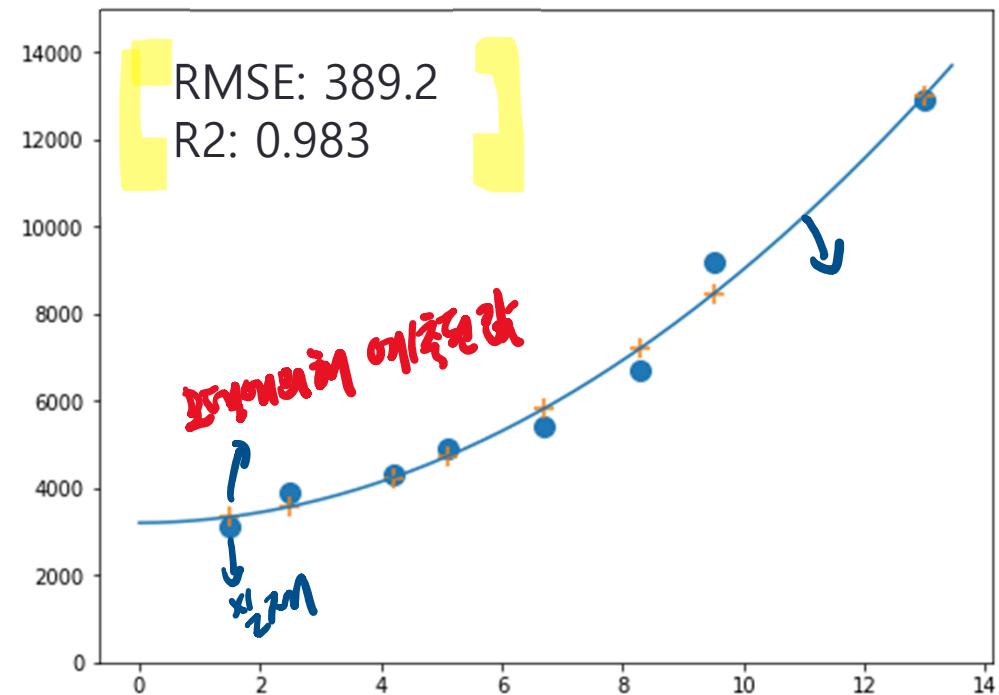
1차식 비례



## 2차 선형회귀모형

- 연봉(y)가 경력(x)의 제곱에 비례한다.
- $y = \beta_0 + \beta_1 x + \beta_2 x^2 \rightarrow$  파라미터 ( $\beta_0, \beta_1, \beta_2$ )

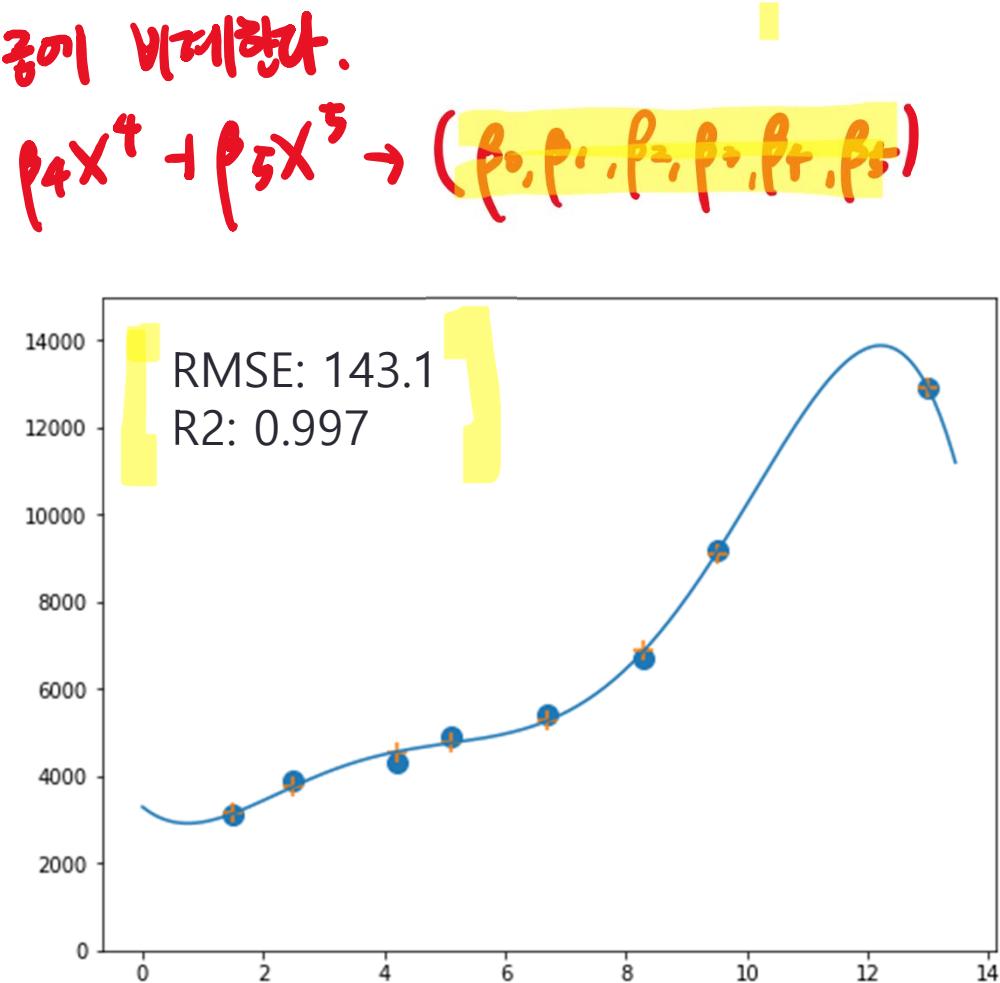
obs	X (경력)	y(연봉)
1	1.5	3100
2	2.5	3900
3	4.2	4300
4	5.1	4900
5	6.7	5400
6	8.3	6700
7	9.5	9200
8	13	12900



# 5차 선형회귀모형

- 연봉(y)과 경력(x)이 5 차중에 비례한다.
- $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4 + \beta_5 x^5 \rightarrow (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5)$

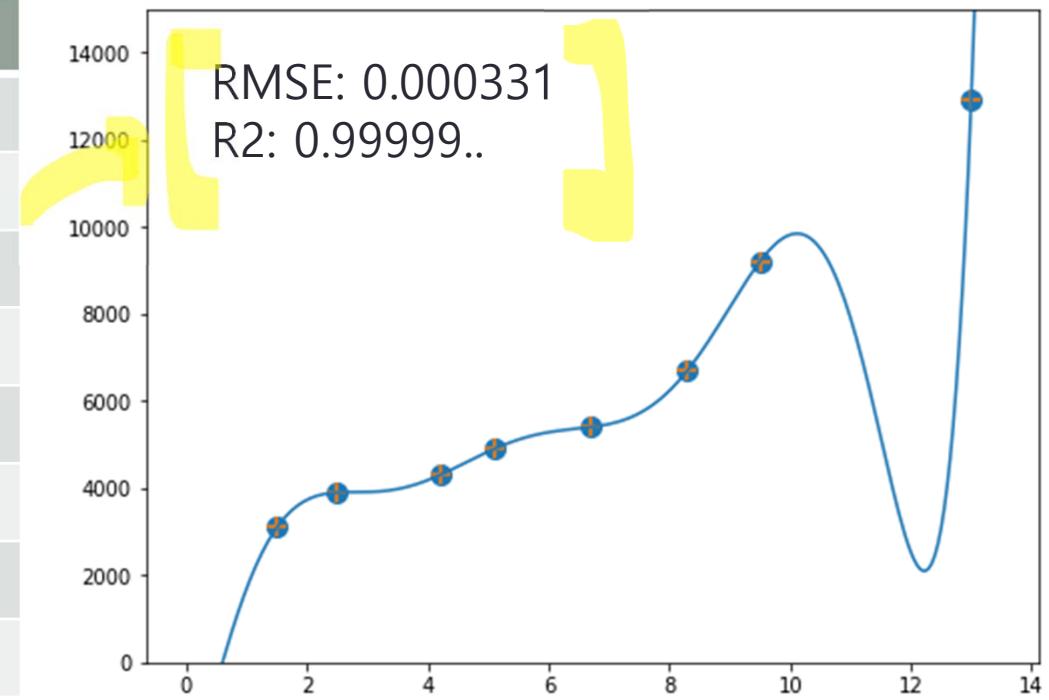
obs	X (경력)	y(연봉)
1	1.5	3100
2	2.5	3900
3	4.2	4300
4	5.1	4900
5	6.7	5400
6	8.3	6700
7	9.5	9200
8	13	12900



# 8차 선형회귀모형

•  $y = \beta_0 + \beta_1 x + \dots + \beta_8 x^8$

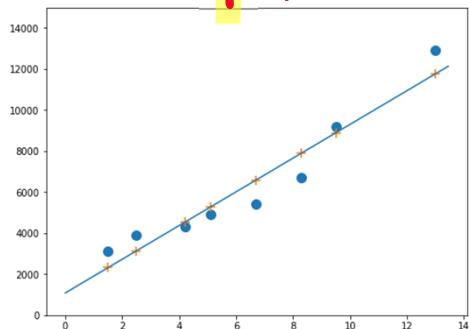
obs	X (경력)	y(연봉)
1	1.5	3100
2	2.5	3900
3	4.2	4300
4	5.1	4900
5	6.7	5400
6	8.3	6700
7	9.5	9200
8	13	12900



# 정리

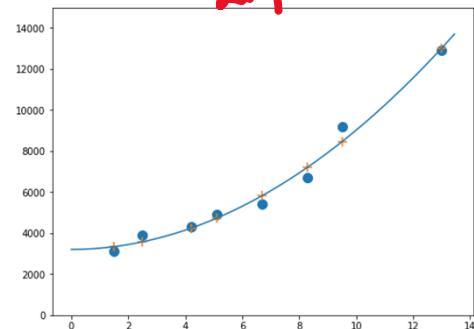
obs	X(경력)	y(연봉)
1	1.5	3100
2	2.5	3900
3	4.2	4300
4	5.1	4900
5	6.7	5400
6	8.3	6700
7	9.5	9200
8	13	12900

1차



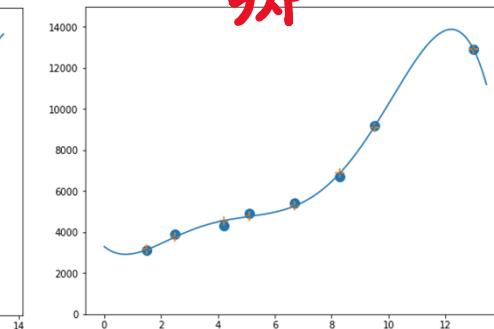
RMSE: 839.3  
R2: 0.918

2차



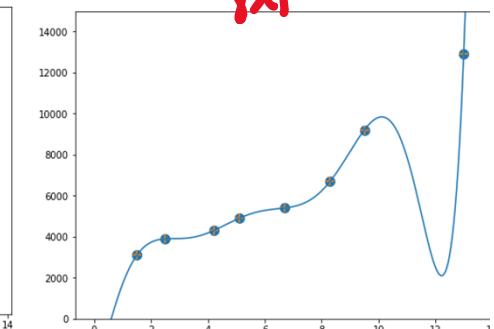
RMSE: 389.2  
R2: 0.983

5차



RMSE: 143.1  
R2: 0.997

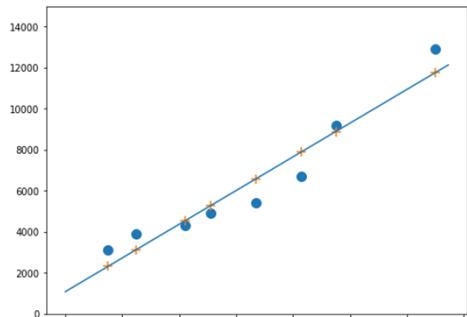
8차



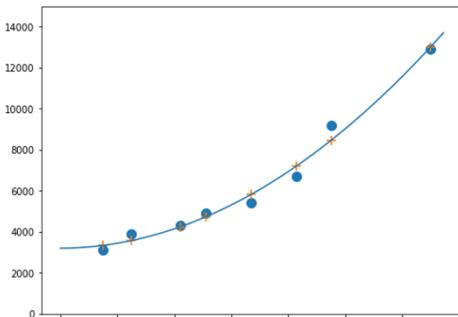
RMSE: 0.000331  
R2: 0.99999..

# 모형의 유연성

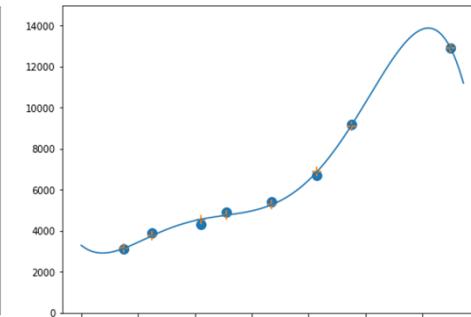
- 모델에서 가정하는 차수가 커질수록
  - 모델의 유연성이 커진다
  - 더 복잡한 패턴을 표현할 수 있다
  - [학습데이터]에 대한 정확도가 높아진다



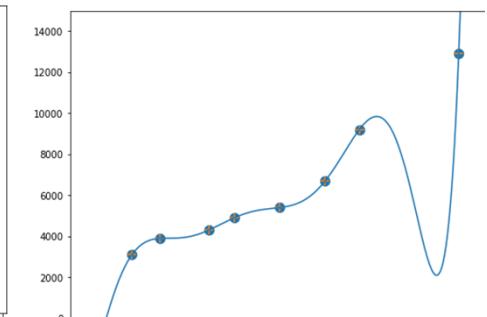
RMSE: 839.3  
R2: 0.918



RMSE: 389.2  
R2: 0.983

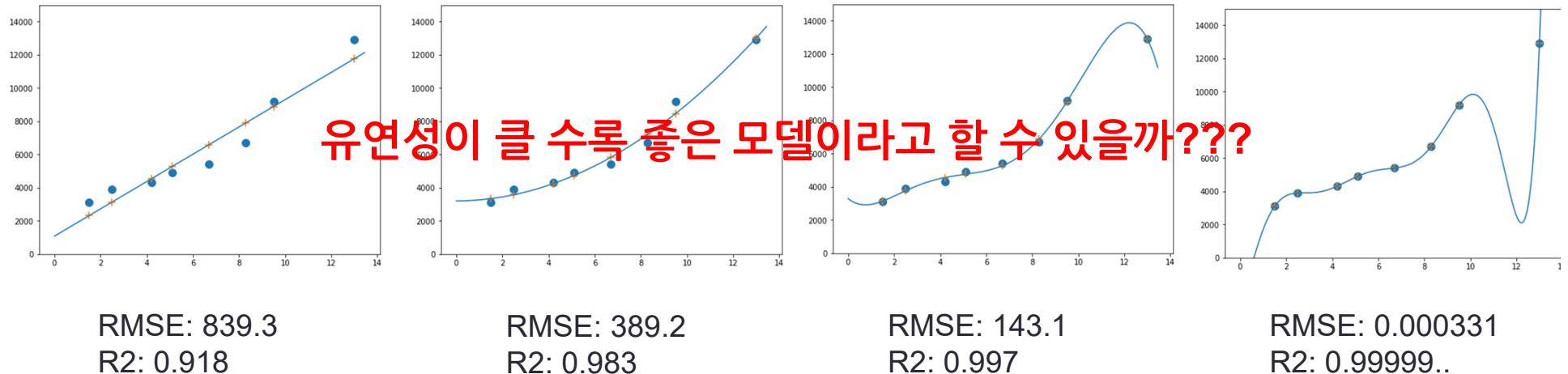
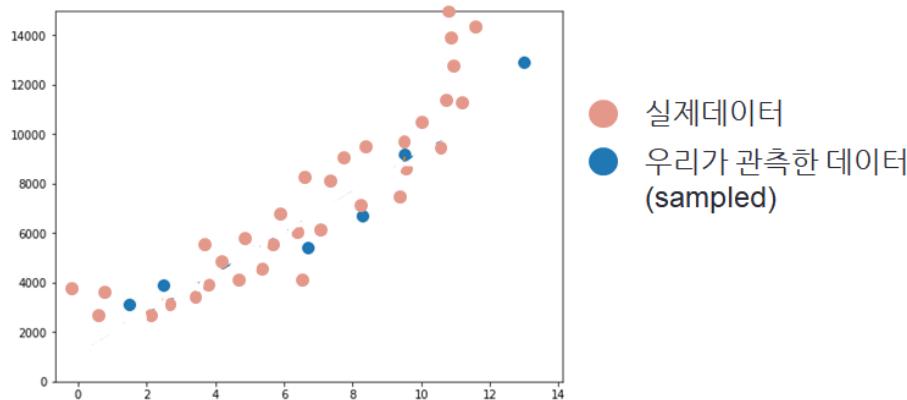


RMSE: 143.1  
R2: 0.997



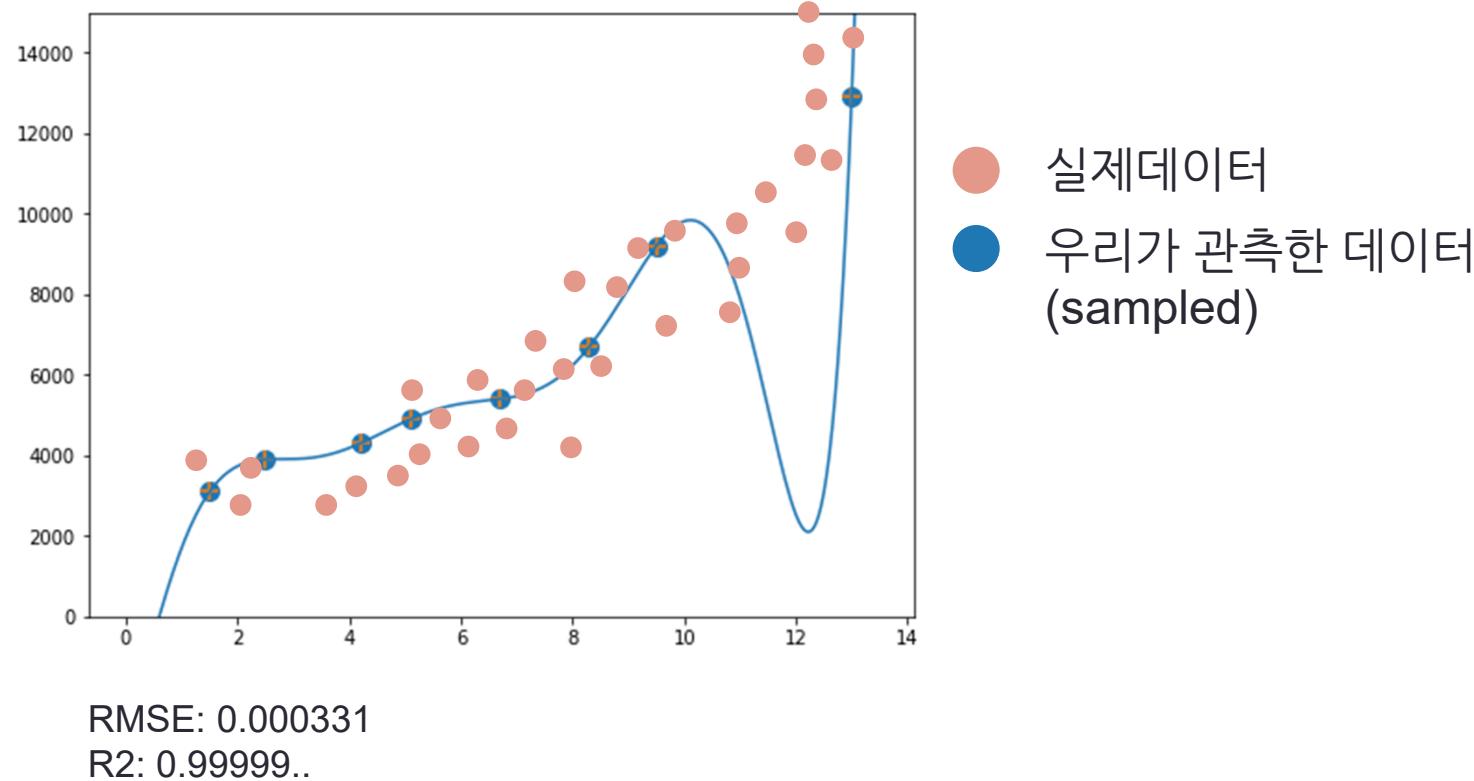
RMSE: 0.000331  
R2: 0.99999..

# 모형의 유연성



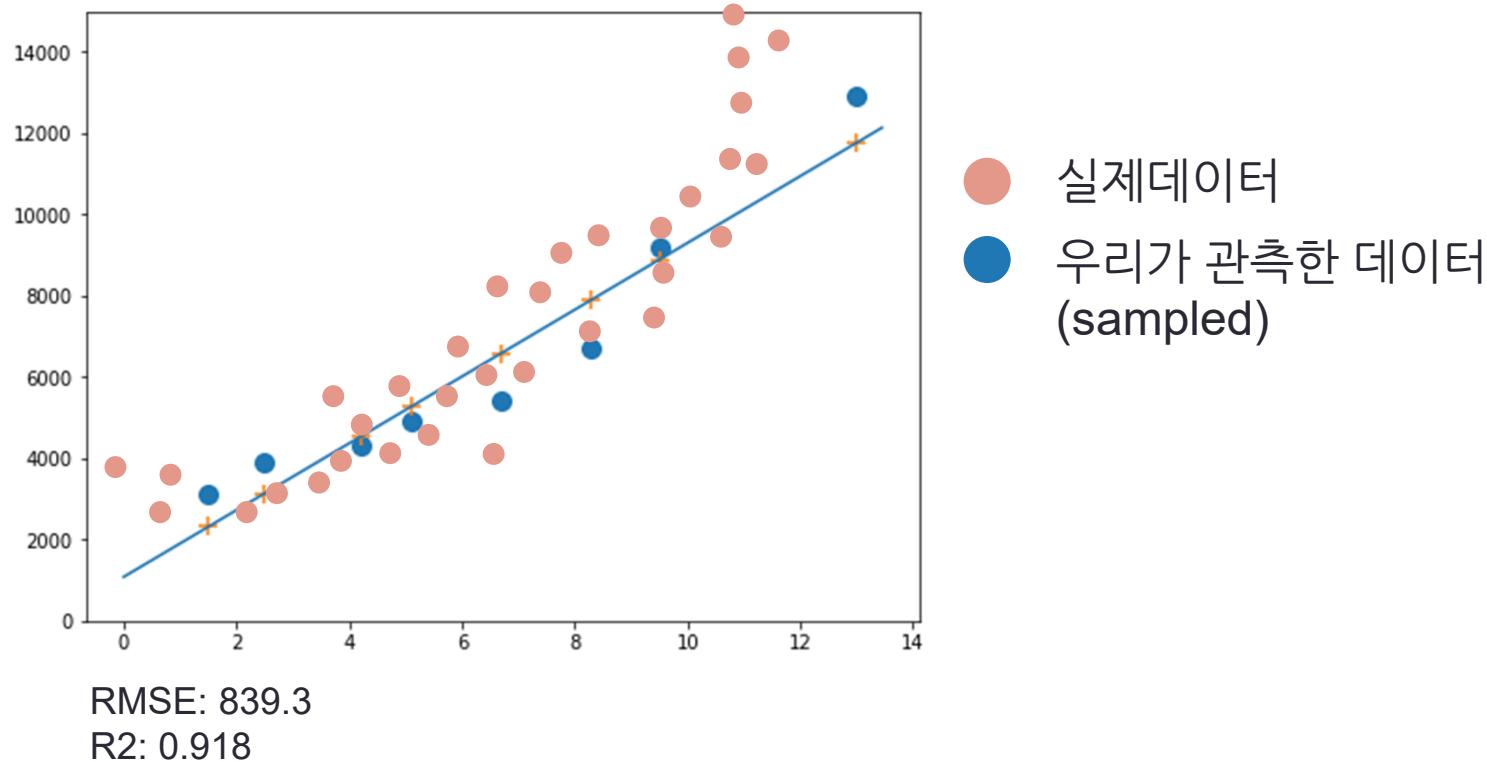
# Overfitting (과적합)

- 관측된 데이터에 대한 정확도가 가장 높았던 모델
  - 모델이 학습에 사용된 데이터를 지나치게 충실히 표현
  - (모르지만) 실제 데이터에 대한 일반화에 실패



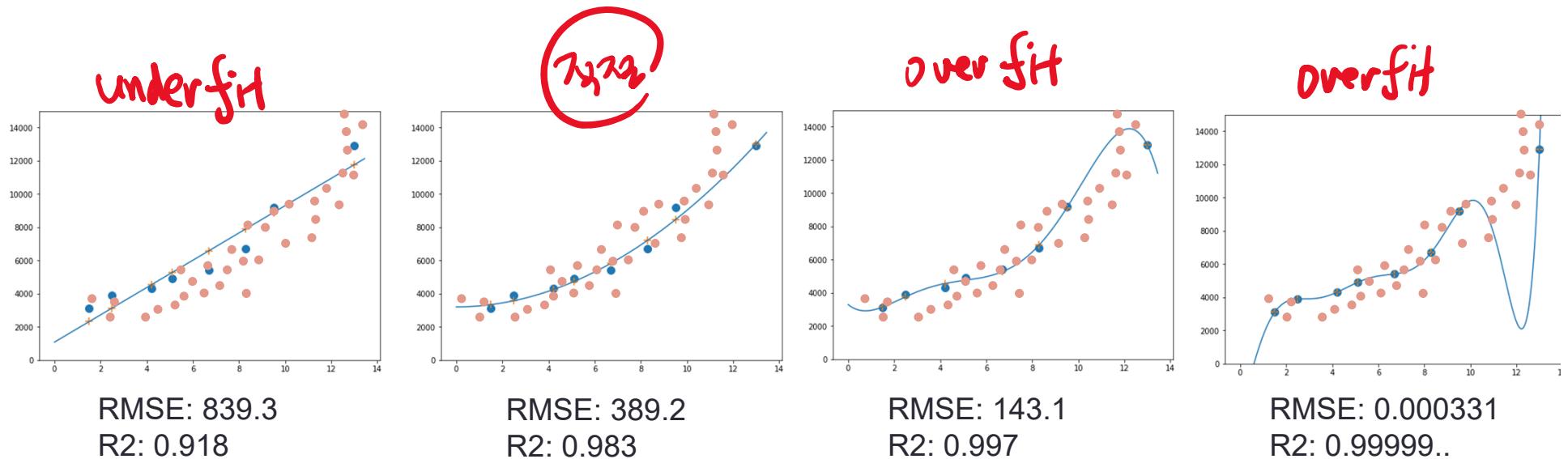
# Underfitting (과소적합)

- 관측된 데이터에 대한 정확도가 가장 낮았던 모델
  - 모델이 데이터를 지나치게 단순하게 표현
  - 실제 데이터에 대한 일반화에 실패



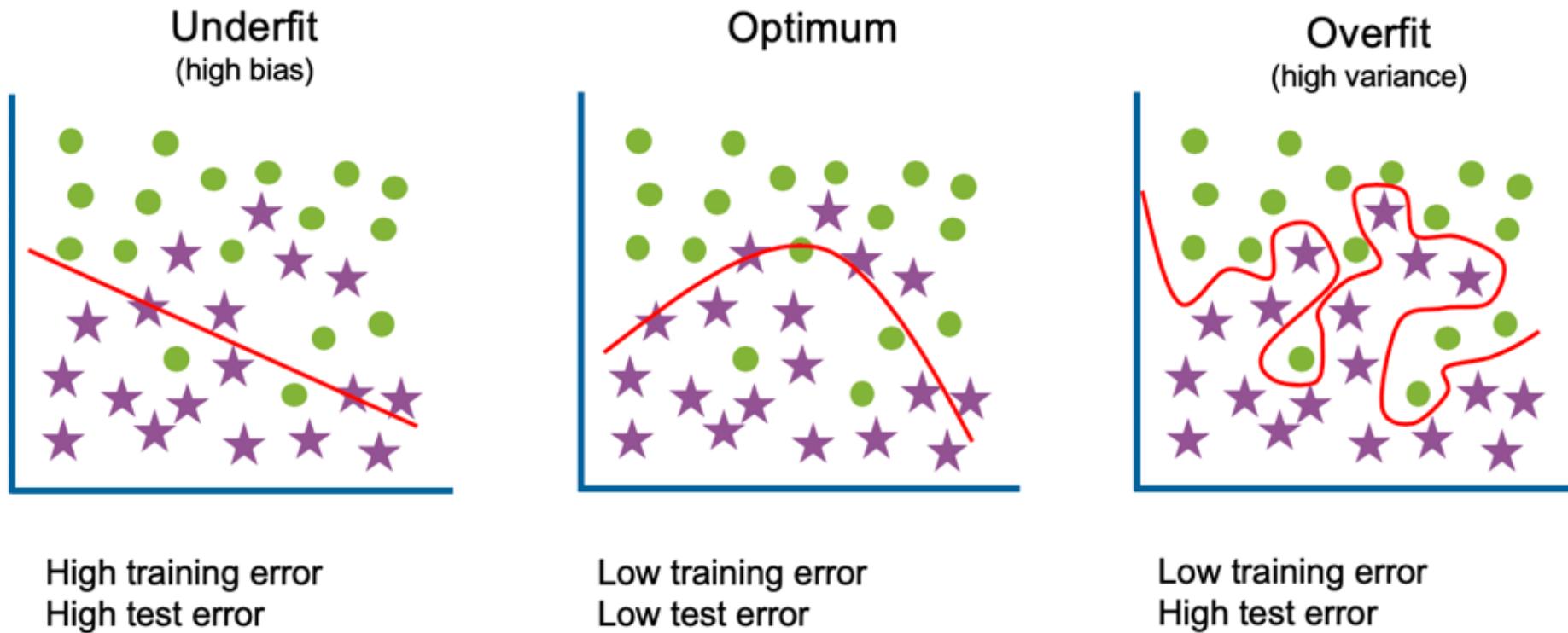
# 일반화 측면에서 모델의 비교

- 관측된 데이터에 대한 성능이 아닌, 실제 데이터에 대한 성능이 좋아야 함.
- 일반화 능력이 좋은 모델을 선택해야 함
  - Underfitting(X)
  - Overfitting(X)



# 분류문제에서의 과적합

- 서로 다른 label의 경계선이 울퉁불퉁할 수록 유연한 모델



# 모델의 일반화 능력 측정 방법

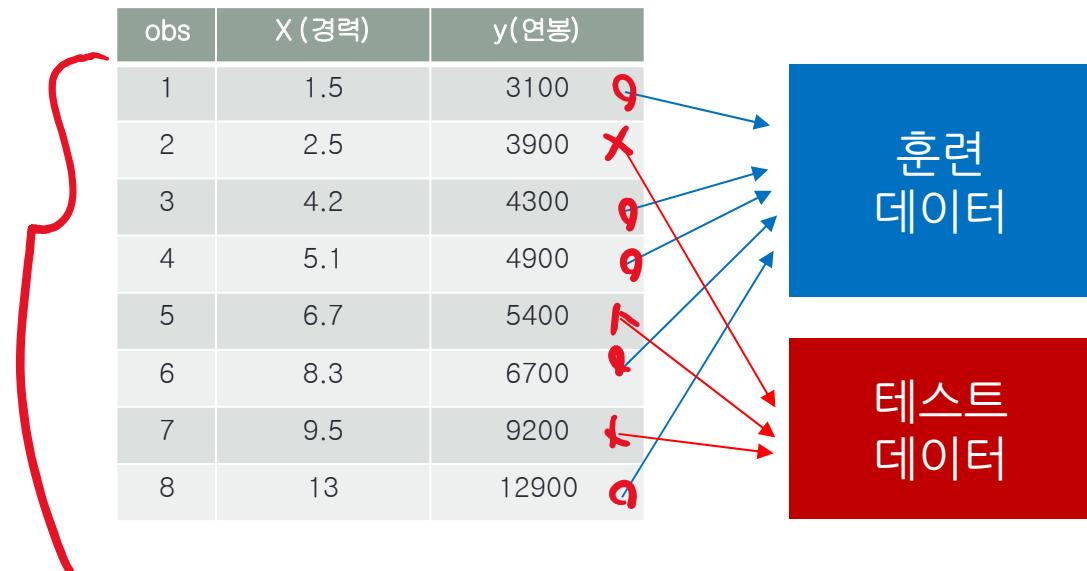
- 과적합과 과소적합을 피하기 위해 모델의 일반화 능력 측정 필요
- 하지만, 모델을 학습하는 시점에서 Unknown data를 수집할 수 없다

# 모델의 일반화 능력 측정 방법

- 과적합과 과소적합을 피하기 위해 모델의 일반화 능력 측정 필요
- 하지만, 모델을 학습하는 시점에서 Unknown data를 수집할 수 없다
- 대안
  - 관측된 데이터를 훈련데이터[Training data]와 테스트데이터[Testing data]로 나눔
  - 훈련데이터: 모델의 학습에 사용
  - 테스트데이터: 모델의 일반화 능력 검증에 사용

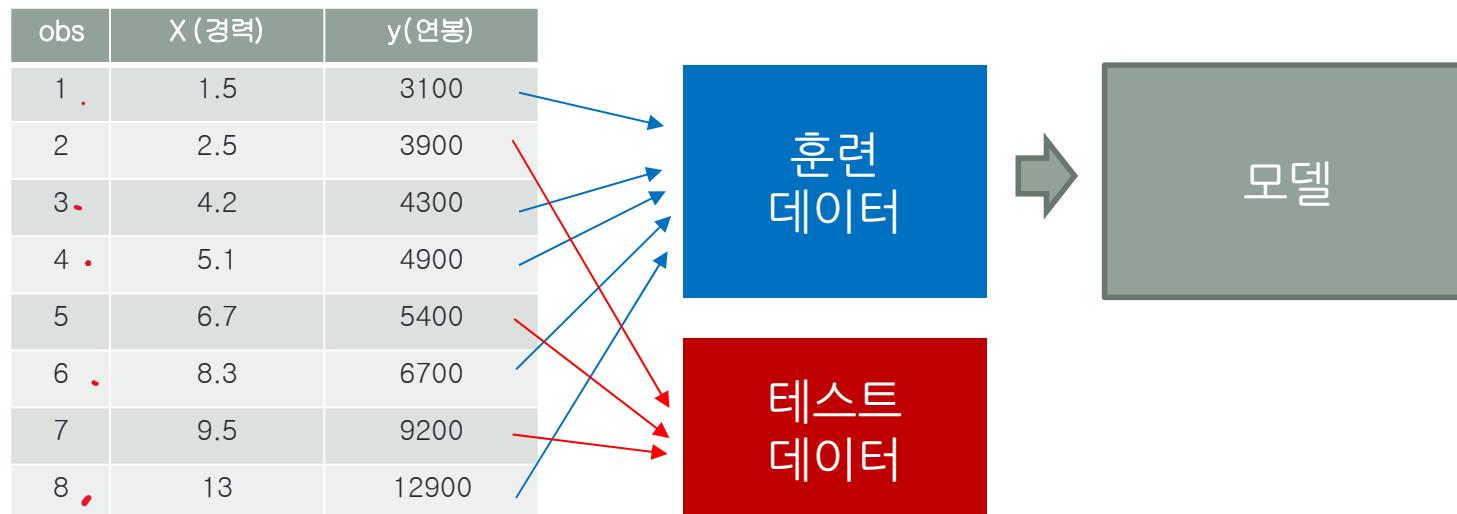
# Step 1. 데이터 나누기

- 우리가 가진 데이터를 훈련데이터와 테스트데이터로 나눔
  - 훈련데이터: 모델의 학습에 사용
  - 테스트데이터: 모델의 일반화 능력 검증에 사용



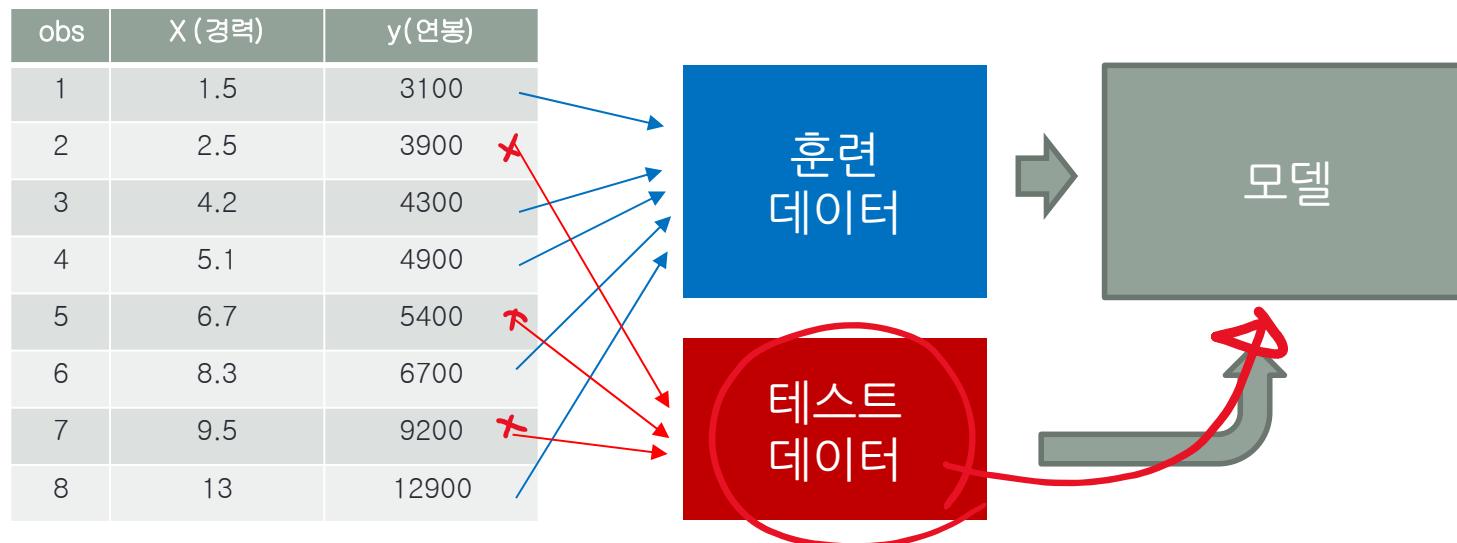
## Step 2. 훈련데이터로 모델 학습

- 우리가 가진 데이터를 훈련데이터와 테스트데이터로 나눔
  - 훈련데이터: 모델의 학습에 사용
  - 테스트데이터: 모델의 일반화 능력 검증에 사용
- 훈련데이터로 모델을 학습



# Step 3. 테스트데이터로 모델 정확도 측정

- 우리가 가진 데이터를 훈련데이터와 테스트데이터로 나눔
  - 훈련데이터: 모델의 학습에 사용
  - 테스트데이터: 모델의 일반화 능력 검증에 사용
- 훈련데이터로 모델을 학습
- 테스트 데이터로 모델 정확도 측정



# 계산예시 - 데이터 나누기

X	Y
시간	성적
1	1.5
1	1.2
2	2.5
3	2.5
3	3.3
4	4.5
4	3.6
5	4.8
5	4.5
6	5.5

시간	성적
1	1.5
2	2.5
3	2.5
4	4.5
5	4.5
6	5.5

훈련데이터

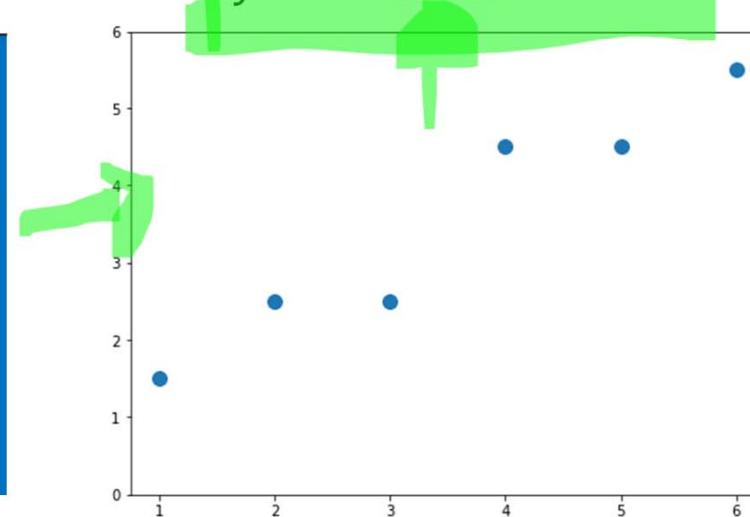
시간	성적
1	1.2
3	3.3
4	3.6
5	4.8

테스트데이터

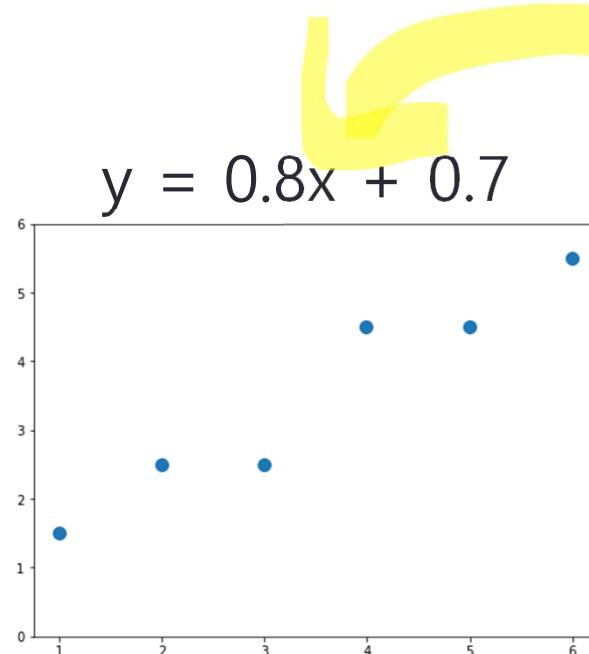
# 훈련데이터로 모델 학습

- 훈련데이터
  - 모델을 학습하는데 활용되는 데이터

시간	성적
1	1.5
2	2.5
3	2.5
4	4.5
5	4.5
6	5.5



# 테스트데이터로 정확도 평가



## • 테스트 데이터

- 모델의 일반화 능력을  
검증하는데 사용하는  
데이터

시간	성적	예측 성적
1	1.2	1.5
3	3.3	3.1
4	3.6	3.9
5	4.8	4.7

R<sup>2</sup>: R<sup>2</sup> SE

# 훈련 정확도 vs 테스트 정확도

( $R^2$ , RMSE)

$R^2 \cdot RMSE$

모델이 얼마나  
훈련데이터를  
정확히  
표현하는지

시간

성적

1.5

2.3

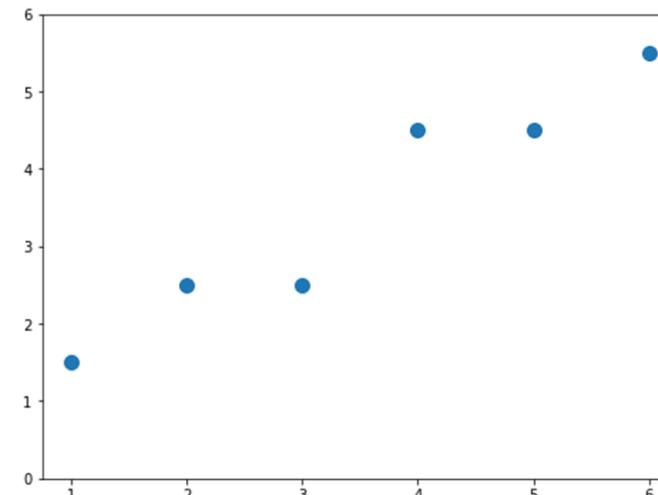
3.1

3.9

4.7

5.5

$$y = 0.8x + 0.7$$



↑

$R^2 \cdot RMSE$

모델이 얼마나  
테스트 데이터를  
정확히  
표현하는지

시간

성적

1.5

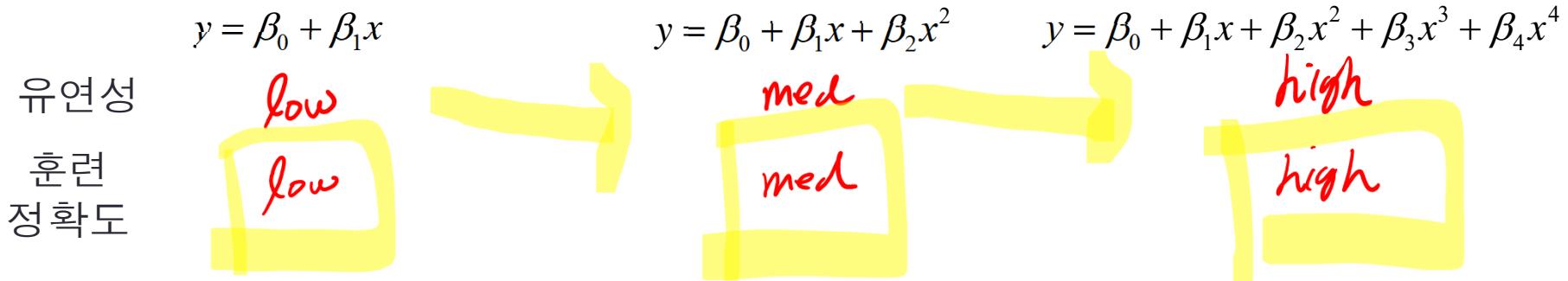
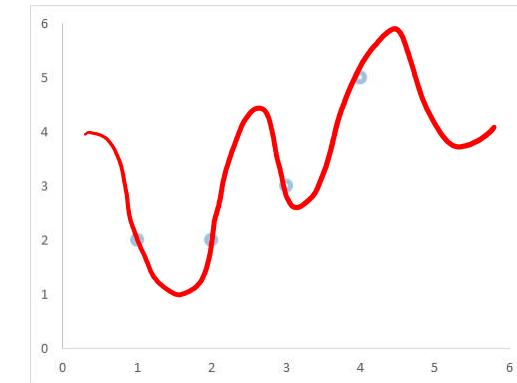
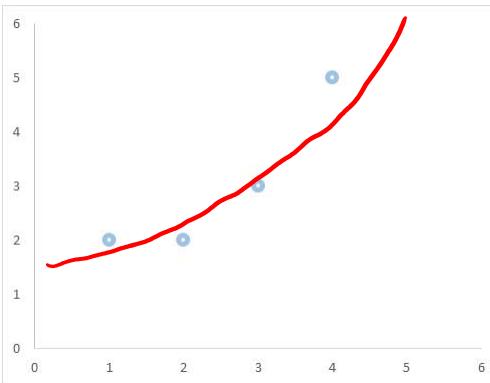
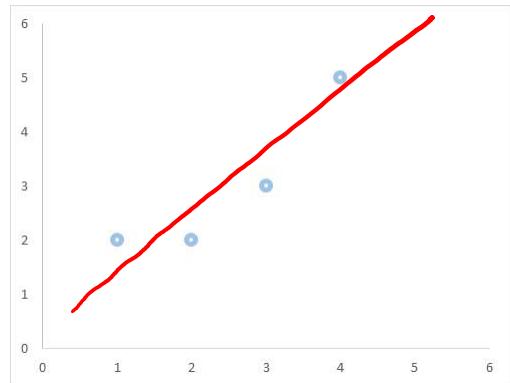
3.1

3.9

4.7

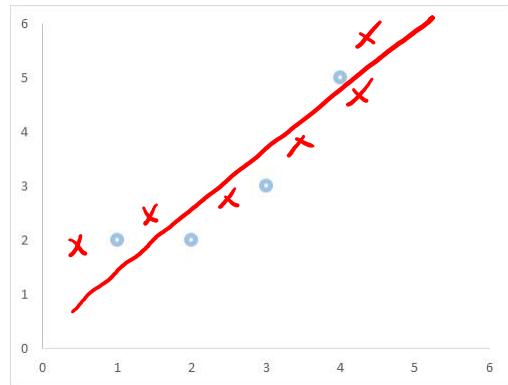
# 모델에 따른 훈련 정확도

- 같은 데이터에 대해 서로 다른 모델 가정 가능
- 모델의 유연성 = 모델의 표현력



# 모델에 따른 테스트 정확도

- 같은 데이터에 대해 서로 다른 모델 가정 가능
- 모델의 유연성 = 모델의 표현력



$$y = \beta_0 + \beta_1 x$$

유연성

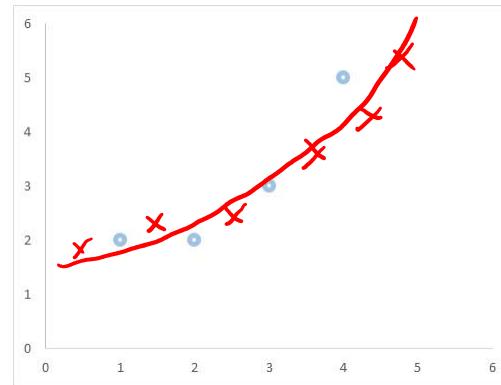
low

훈련  
정확도

med

테스트  
정확도

med

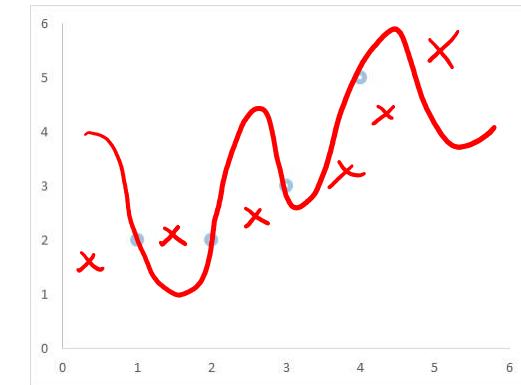


$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

med

good

good



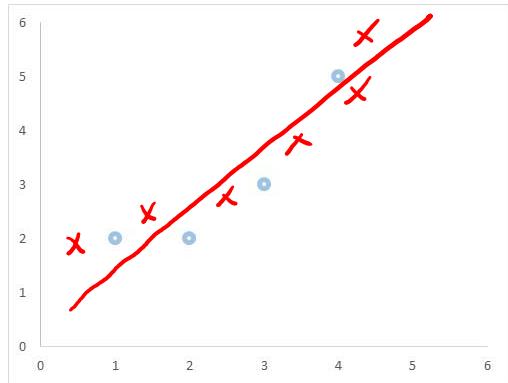
$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$

high

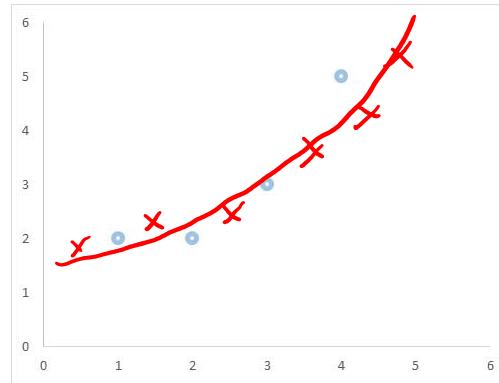
very good

bad

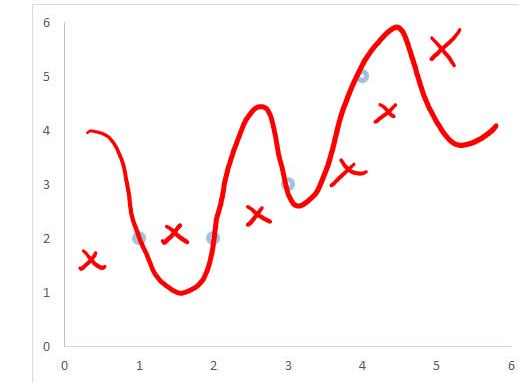
# Overfitting vs Underfitting



$$y = \beta_0 + \beta_1 x$$



$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$



$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$



# Underfitting



- High training error(트레이닝 데이터 정확도 떨어짐)
- High testing error (테스팅 데이터 정확도 떨어짐)
- 지나치게 단순한 모델을 적용하여 설명력이 부족 → High bias

# Overfitting



- Very low training error (트레이닝 데이터 정확도 매우 높다)
- But High testing error (테스팅 데이터 정확도가 떨어진다)
- 지나치게 복잡한 모델을 적용하여 일반화하기 어려움 → High variance

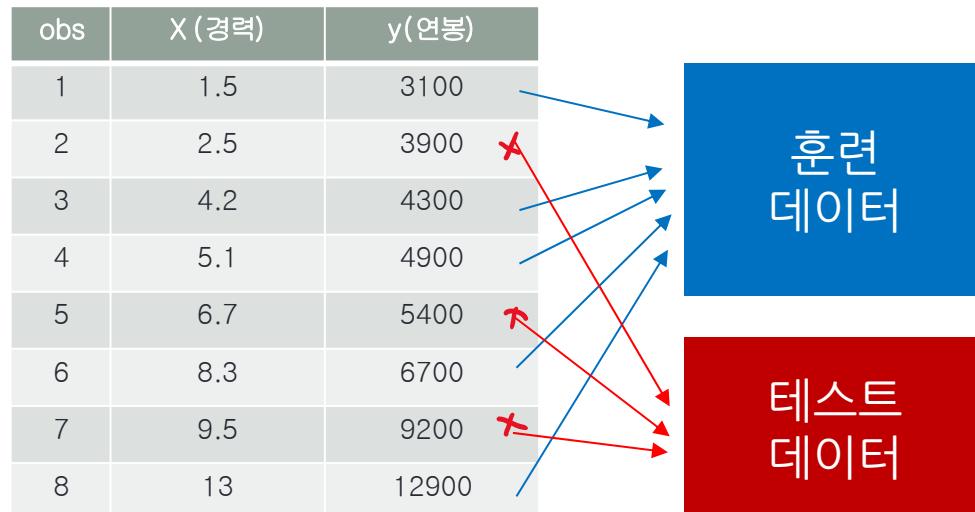
# 모형 선택을 위한 Resampling 기법들

---

시스템경영공학부  
이지환 교수

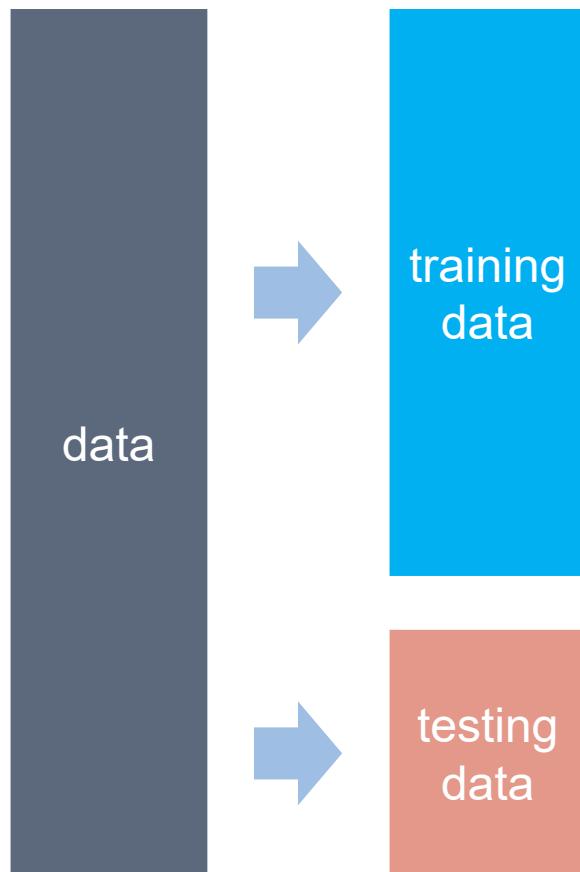
# 모델 선택

- 같은 데이터에 대하여 서로 다른 모델 적용 가능
- 이들을 평가하기 위해, 훈련데이터와 테스팅 데이터로 나누고, 테스팅 데이터의 정확도가 높은 모형을 골라야 함



# 모델선택 방법론: Train-test split

- 가장 기초적인 방법
- 해결책: 내가 가진 데이터 중 일부를 unknown data인것 처럼 간주하자



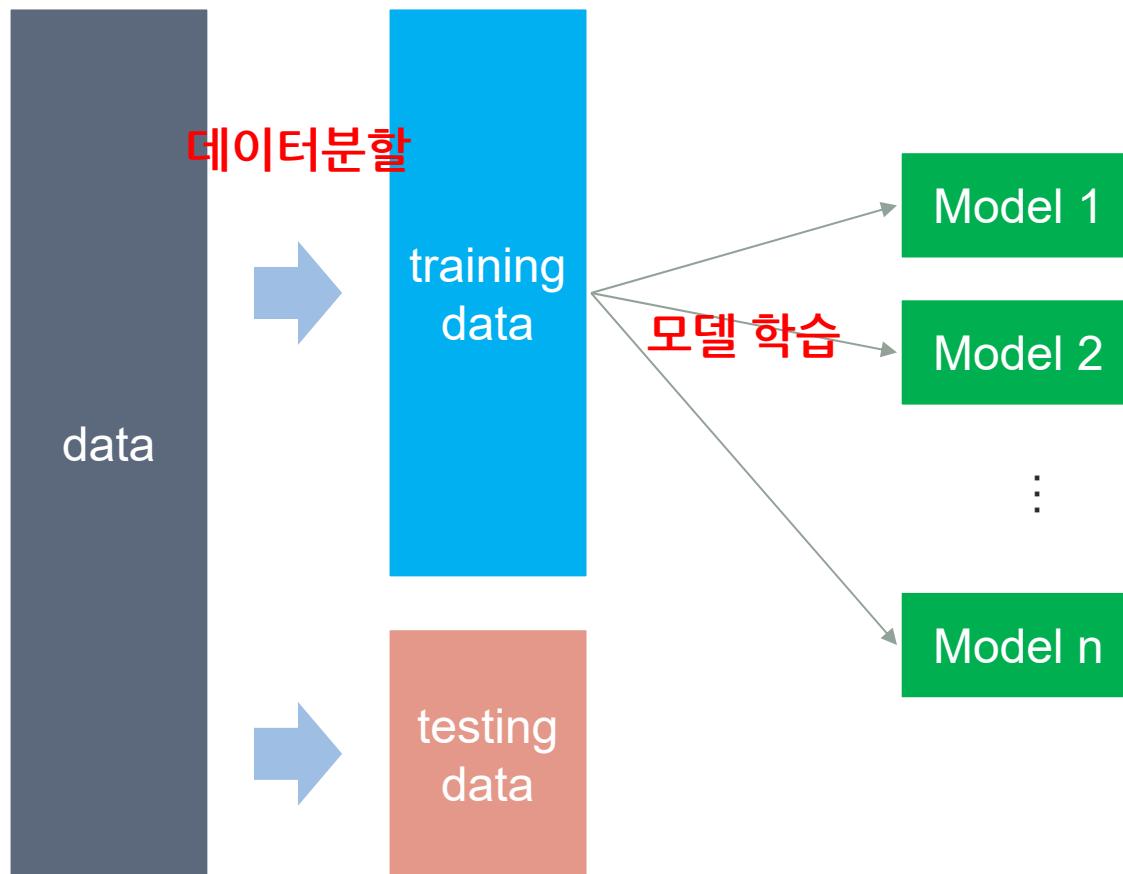
# 정리

- Step1 [데이터분할]
  - step1-1. 데이터 중  $p\%$ 를 무작위로 추출하여 [training data]로 선정
  - step1-2. 나머지  $(1-p)\%$ 의 데이터를 [testing data]로 선정
- Step2 [모델학습]
  - [training data]를 이용하여 모든 비교하고자 하는 모델  $M_1, \dots, M_n$  의 학습 수행
- Step3 [모델검증]
  - [testing data]를 이용하여 학습된 모델  $M_1, \dots, M_n$  각각의 testing 정확도를 측정
- Step4 [모델선정]
  - 모델  $M_1, \dots, M_n$  중 가장 높은 테스팅 정확도를 보인 모델을 최적의 모델로 선정

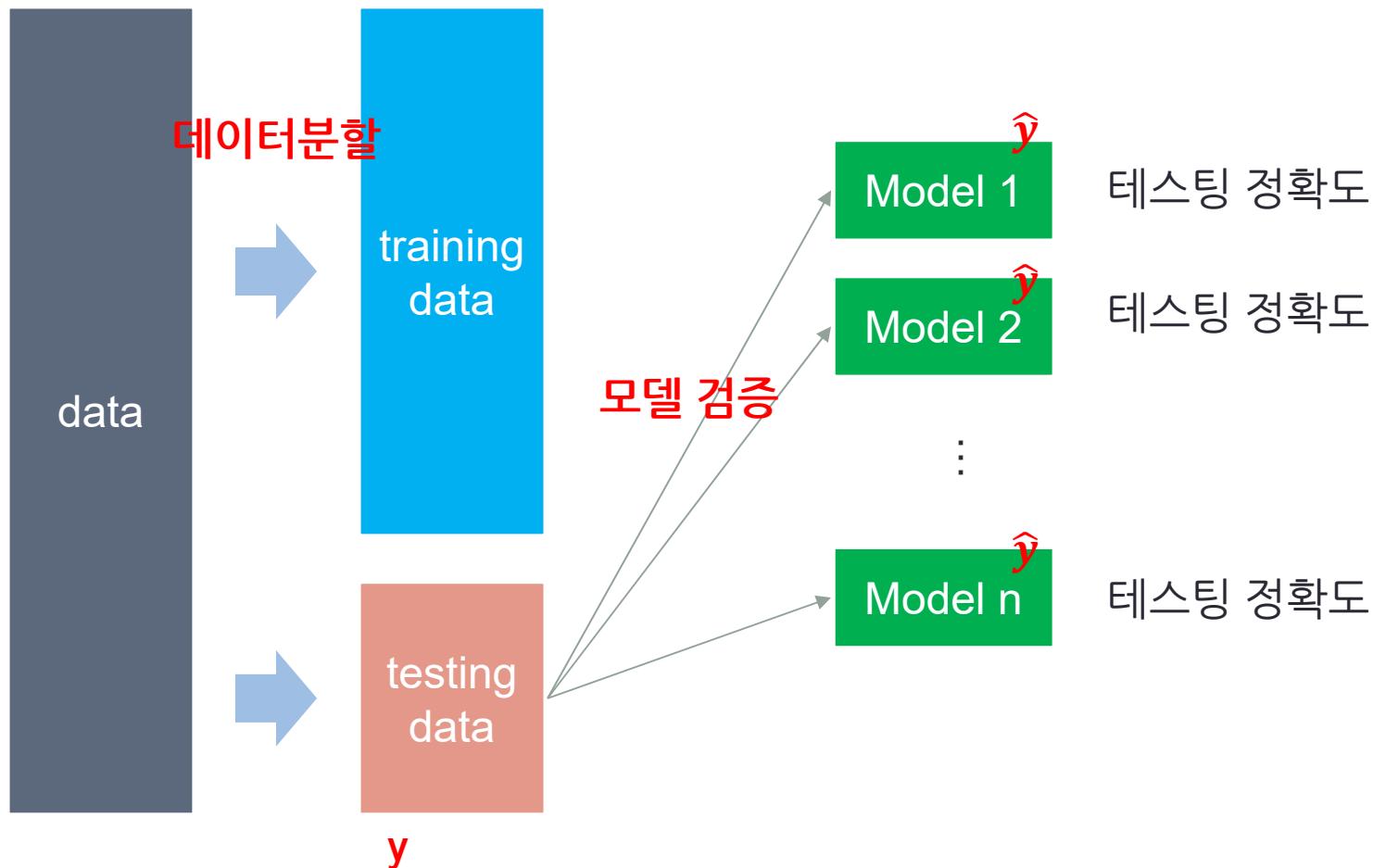
# Train-test split 알고리즘



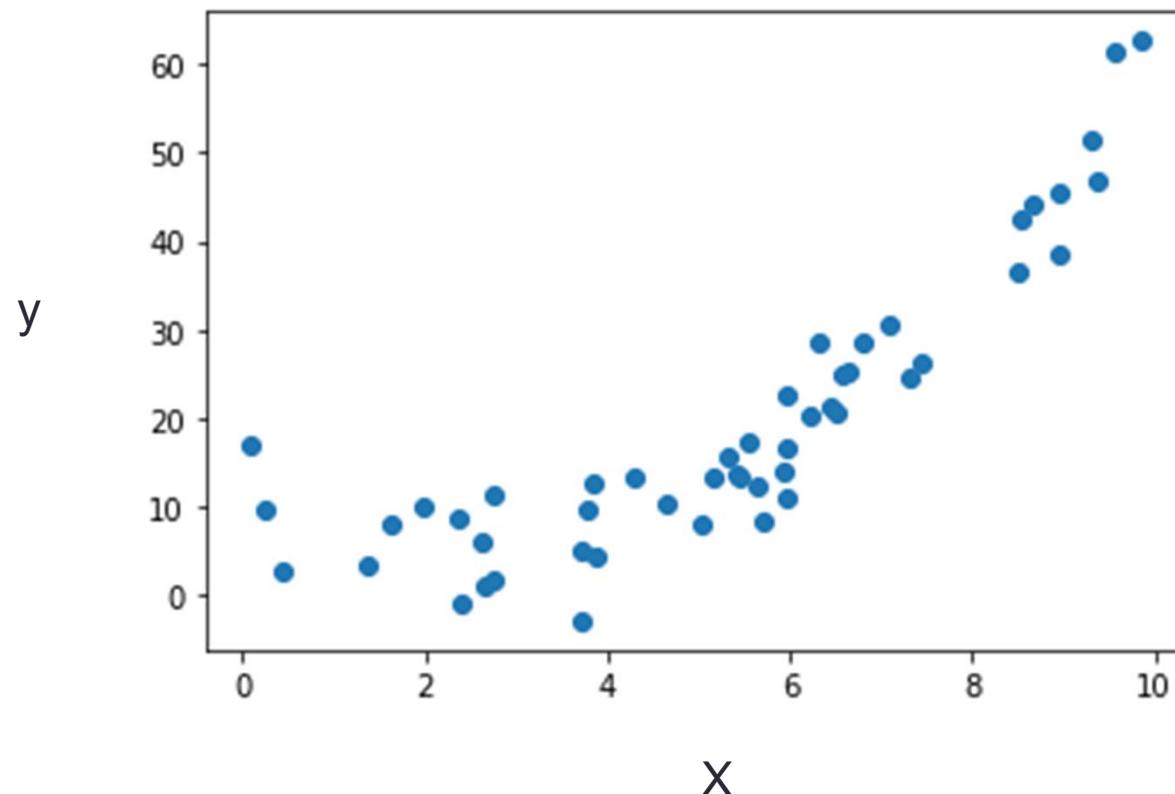
# Train-test split 알고리즘



# Train-test split 알고리즘



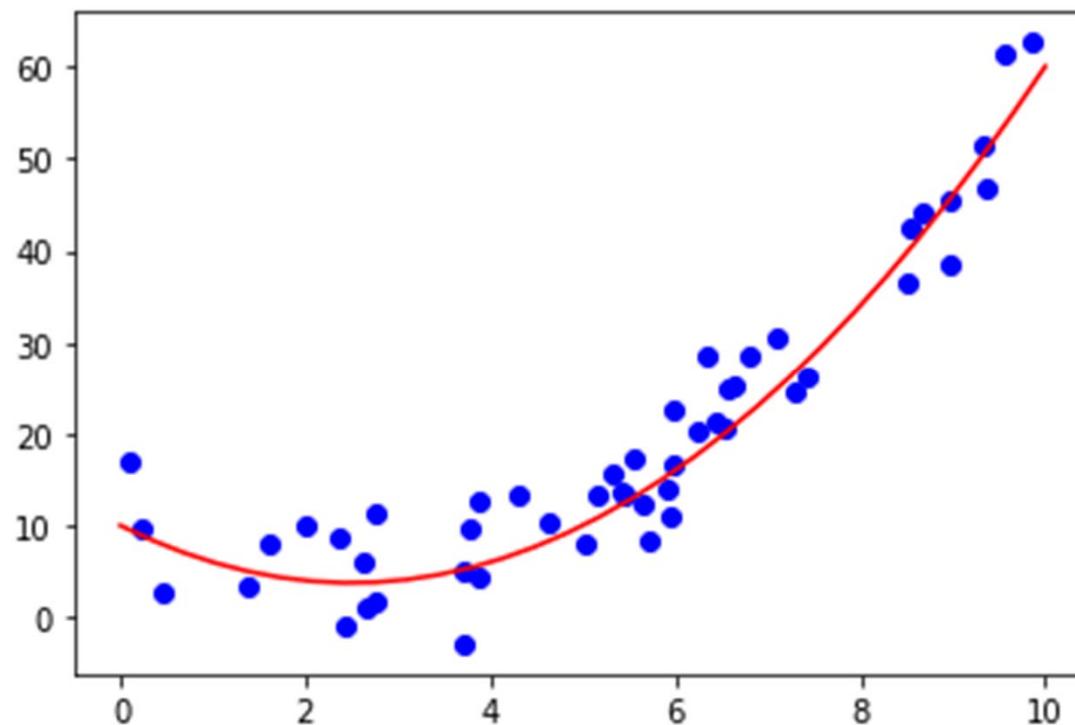
# Train-test split 예시



# 맞춰야할 정답

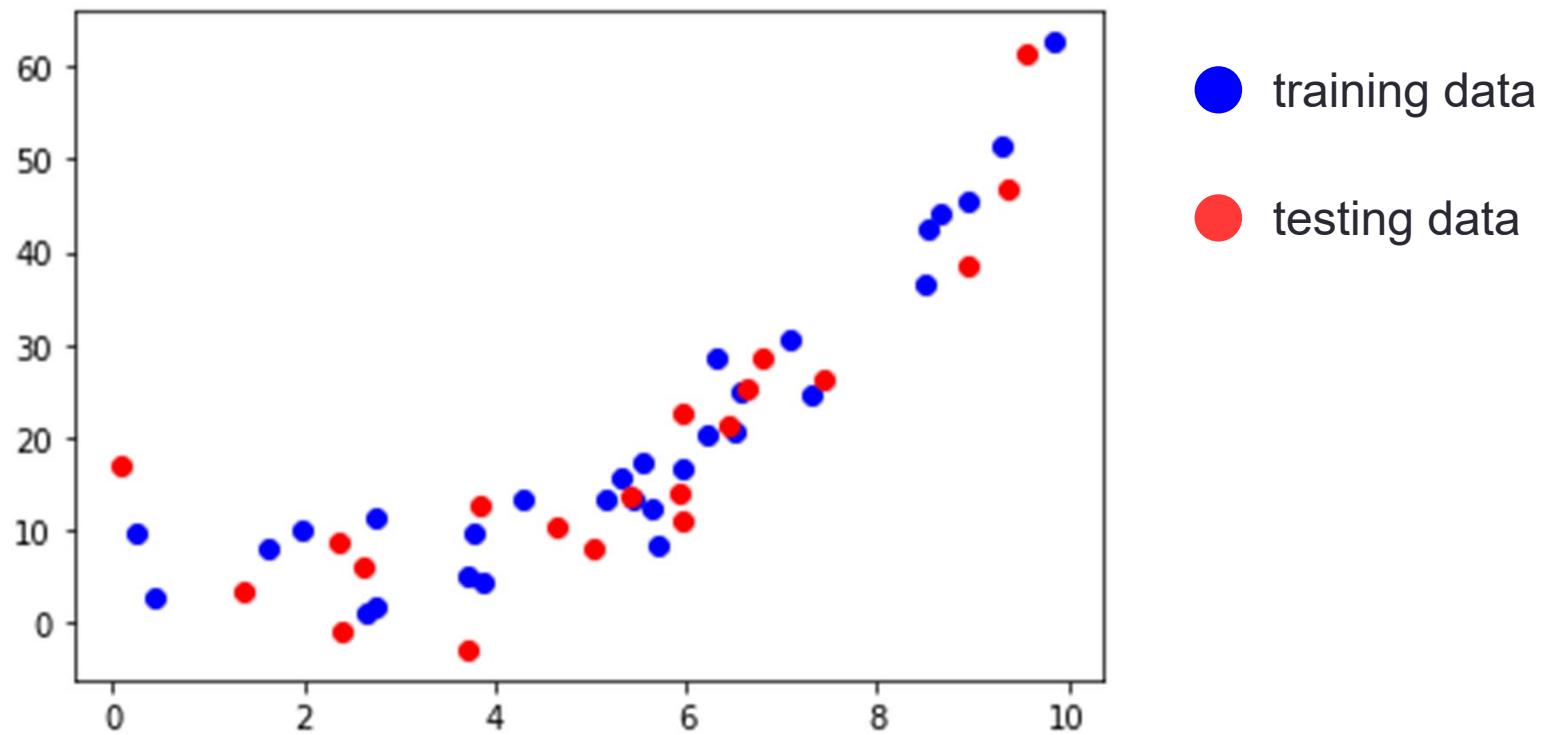
$$y = x^2 - 10x^2 + 10 + \varepsilon$$

$$\varepsilon \sim N(0, 5)$$

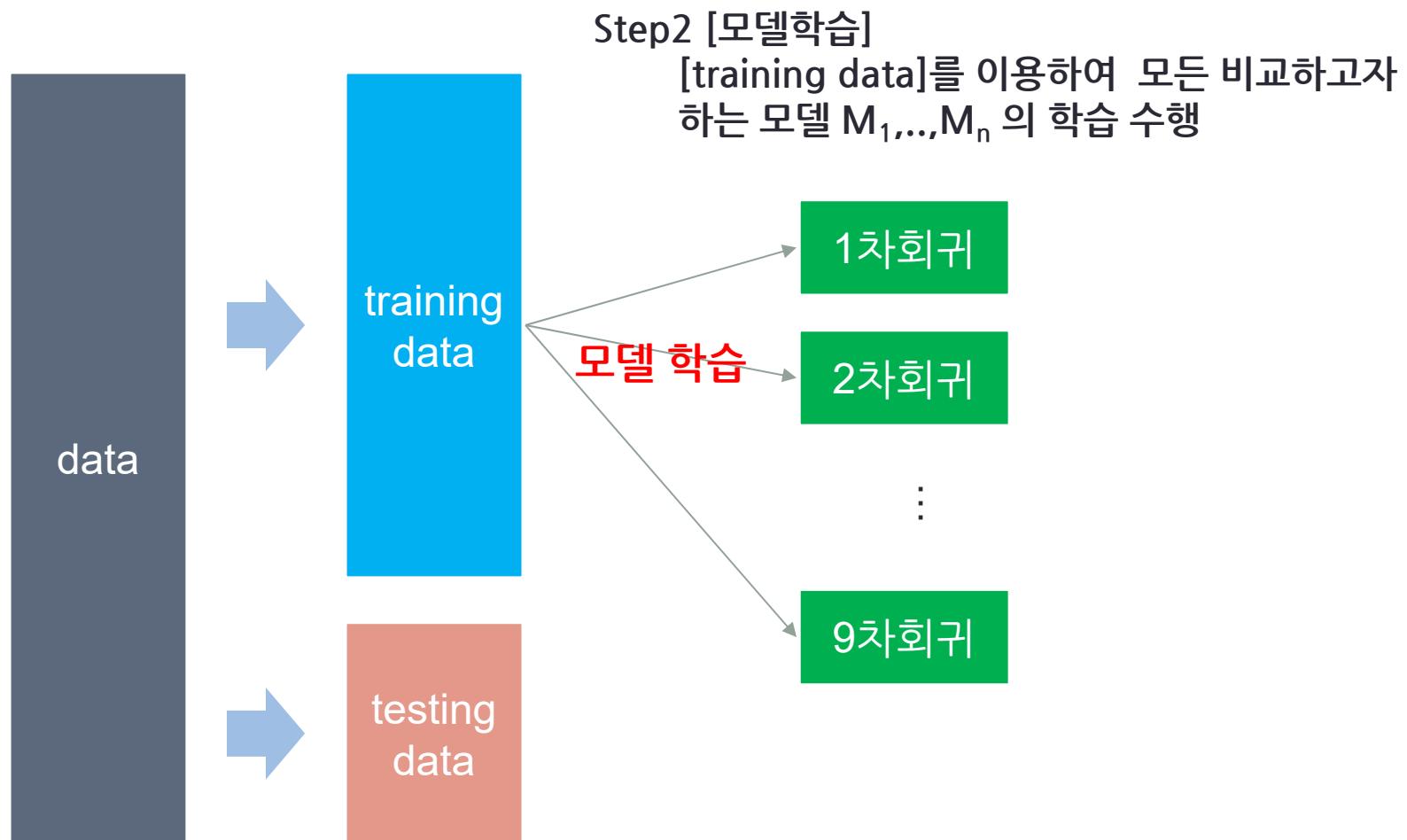


# 데이터 분할

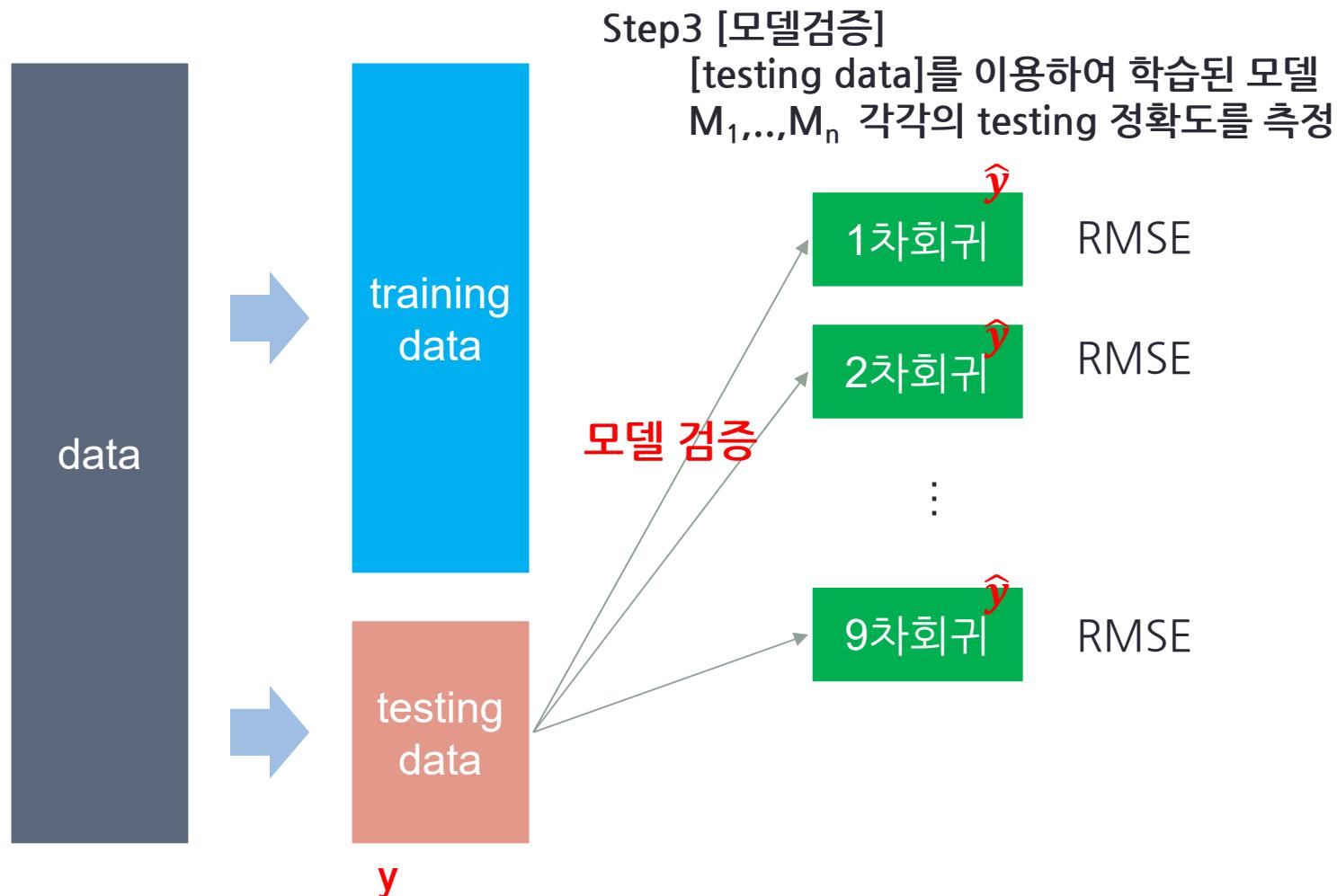
- Step1 [데이터분할]
  - step1-1. 데이터 중  $p\%$ 를 무작위로 추출하여 [training data]로 선정
  - step1-2. 나머지  $(1-p)\%$ 의 데이터를 [testing data]로 선정



# 모델 선택 예시 (모델 학습)



# 모델 선택 예시 (모델 검증)

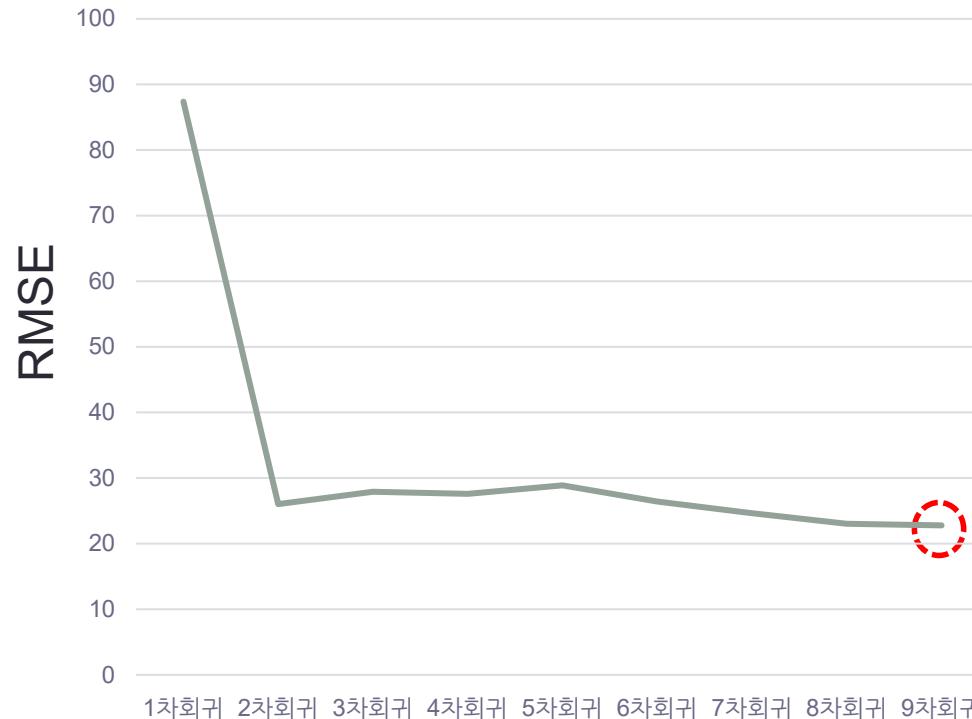


# 각 모델들의 테스팅 정확도

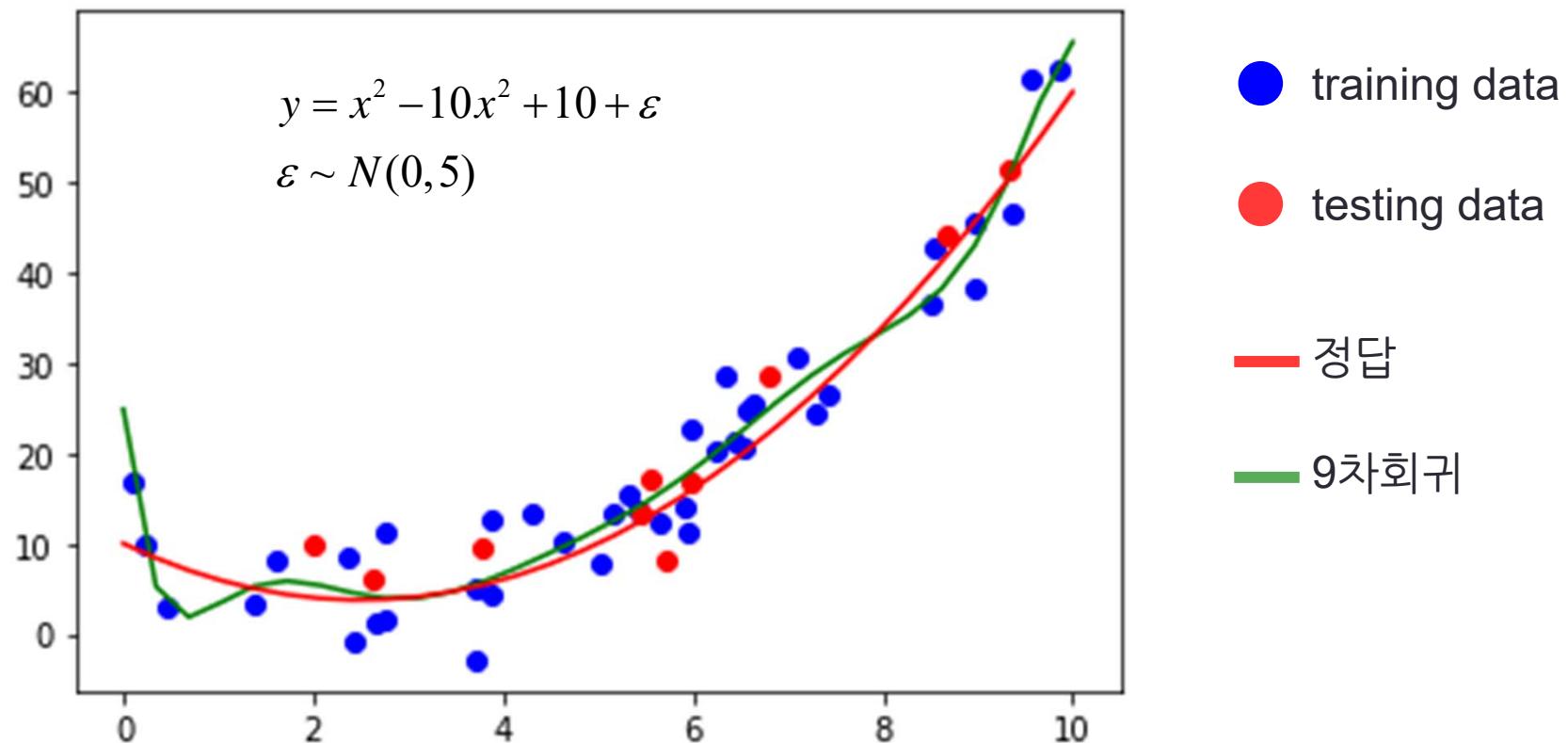
## Step4 [모델선정]

모델  $M_1, \dots, M_n$  중 가장 높은 테스팅 정확도를  
보인 모델을 최적의 모델로 선정

1차회귀	87.38192
2차회귀	26.02434
3차회귀	27.91851
4차회귀	27.58762
5차회귀	28.88298
6차회귀	26.42483
7차회귀	24.6514
8차회귀	23.02441
9차회귀	22.7894



# 구해진 모델과 정답 비교

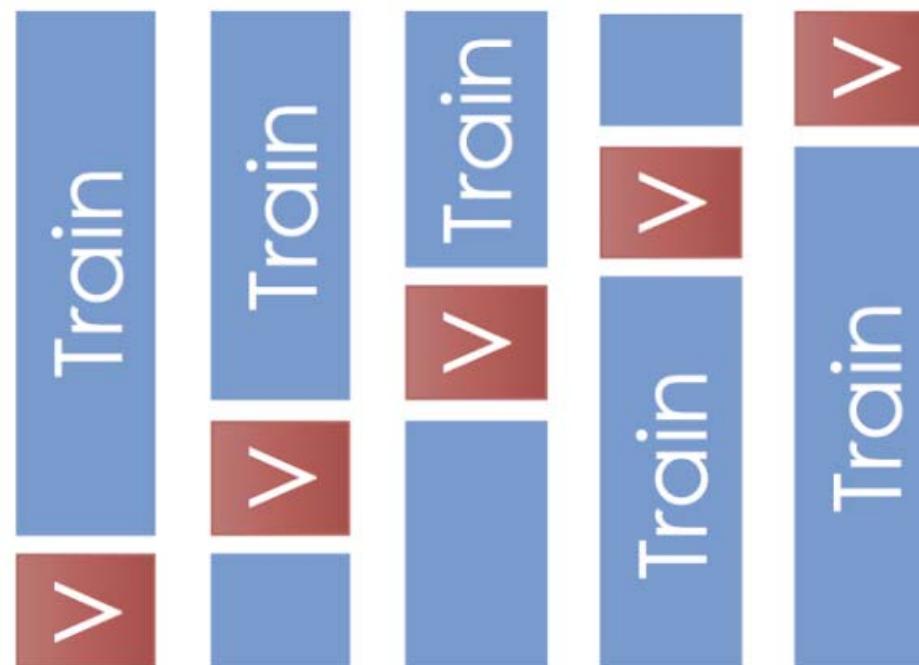


# 기존 방법의 문제점

- Step1 [데이터분할]
  - step1-1. 데이터 중  $p\%$ 를 무작위로 추출하여 [training data]로 선정
  - step1-2. 나머지  $(1-p)\%$ 의 데이터를 [testing data]로 선정
- 문제점
  - 데이터가 나뉘어지는 방식에 따라 모델 검증 결과에 큰 편차가 존재할 수 있음
  - training data와 testing data가 우연에 따라 치우친 상태로 나뉘 어졌을 경우 모델의 일반화 능력을 제대로 파악할 수 없다.

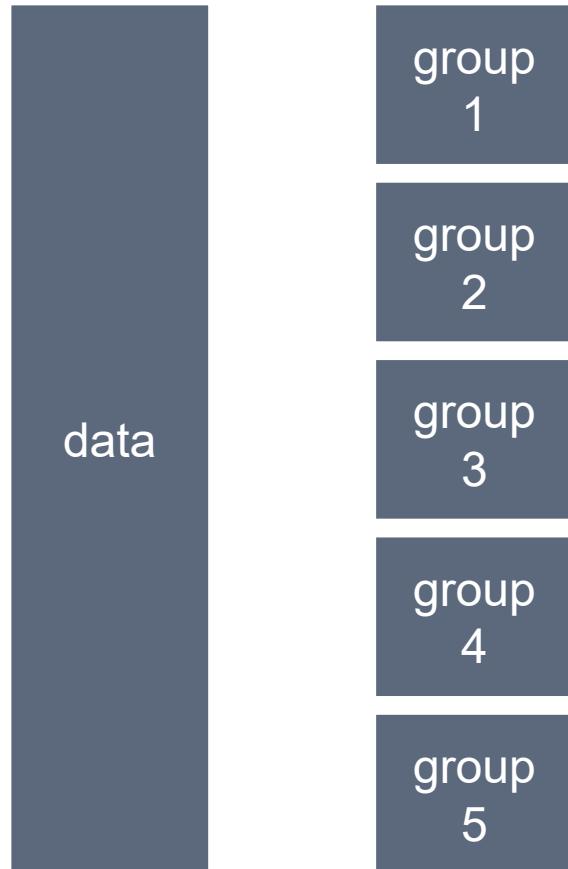
# 교차검증 (Cross-validation)

- Train-test split의 단점을 보완하기 위해 나온 방법
- 한번의 시도로 데이터를 나누다 보면 우연에 의해 치우칠 수 있으니, 데이터를 잘개 쪼개어 여러 번 train-test split을 해 보자



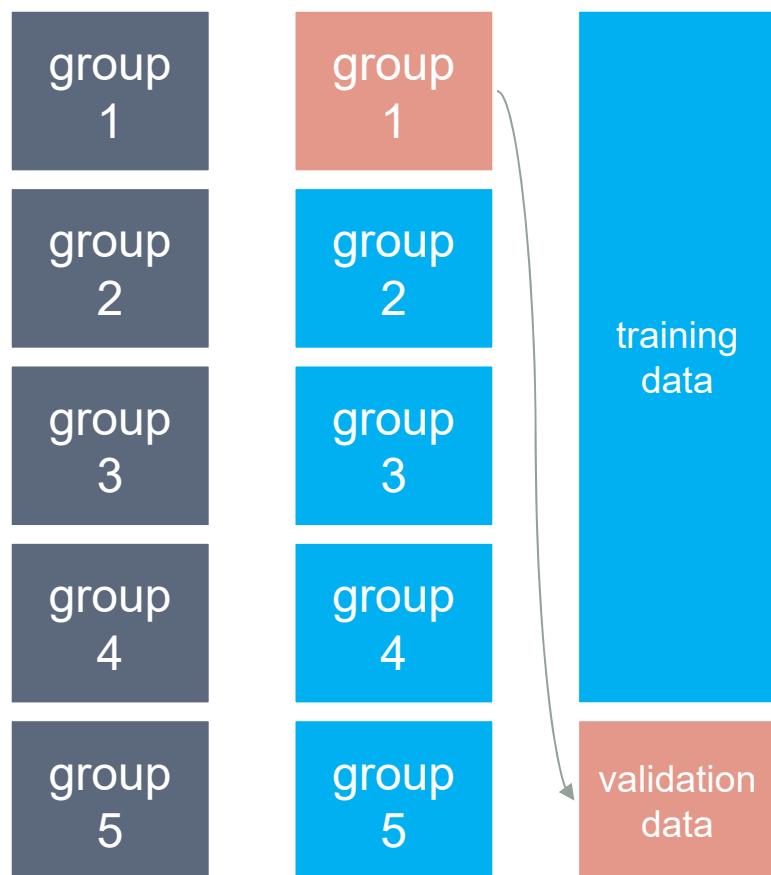
# 교차검증 알고리즘

- 데이터를 무작위로 섞고 같은 수를 가진 K개의 집단으로 나눔



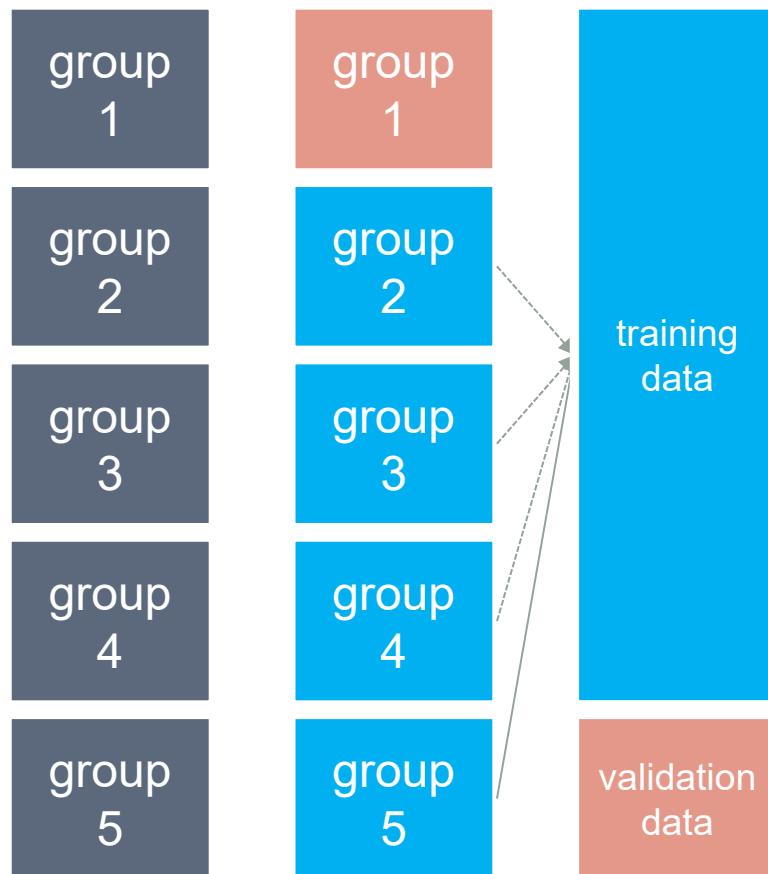
# 교차검증 알고리즘: 1<sup>st</sup> split

- 첫번째 그룹을 [Testing data]로 선정



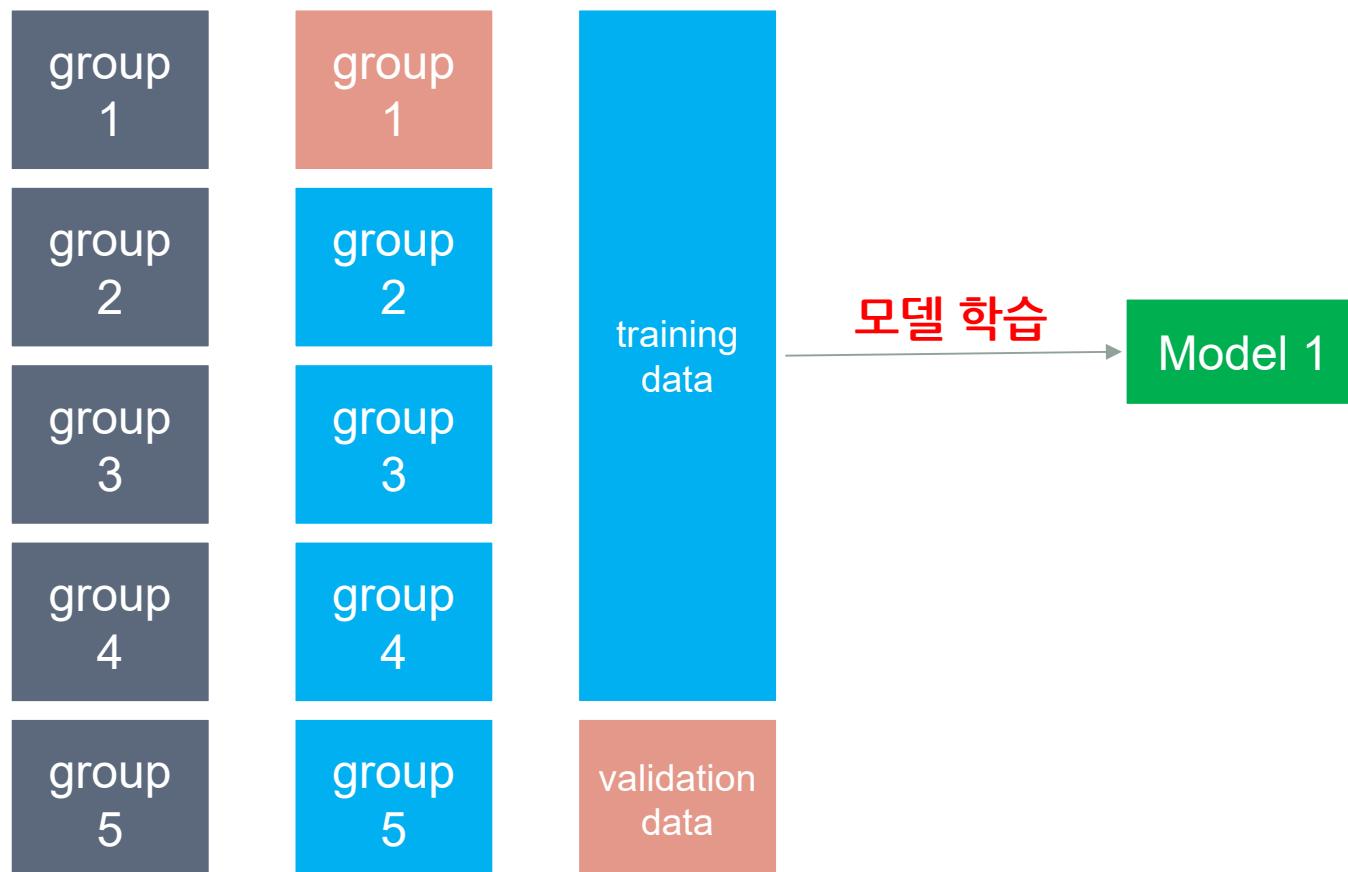
# 교차검증 알고리즘: 1<sup>st</sup> split

- 나머지 그룹을 [training data]로 선정



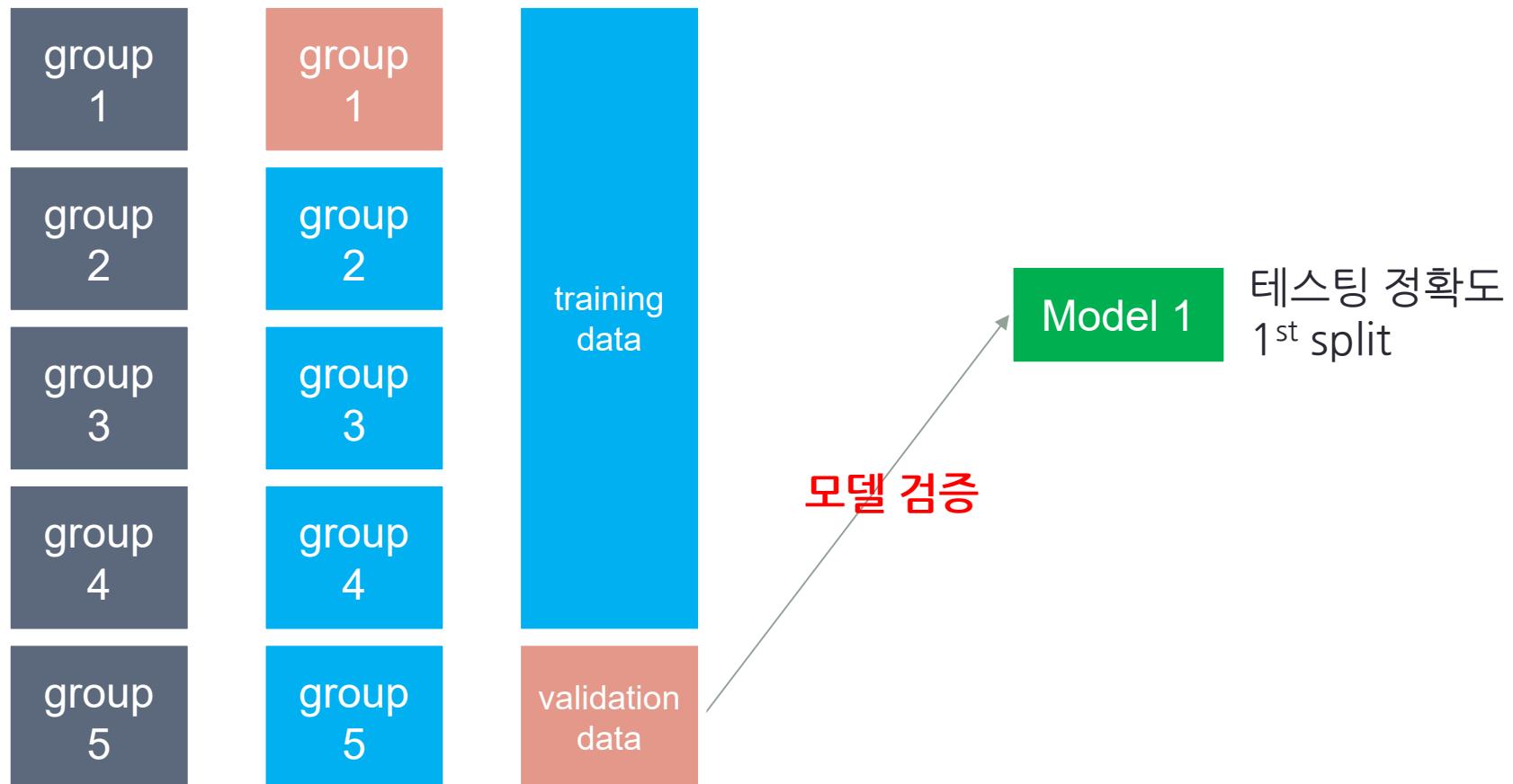
# 교차검증 알고리즘: 1<sup>st</sup> split

- [training data]를 사용하여 모델 학습



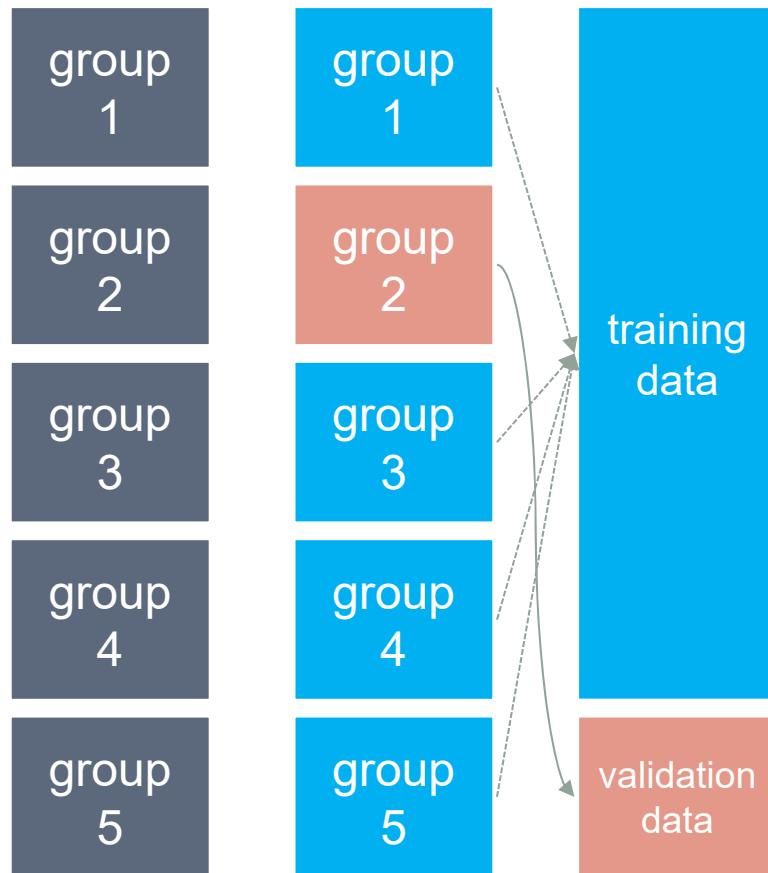
# 교차검증 알고리즘: 1<sup>st</sup> split

- [testing data]를 사용하여 모델 검증(테스팅 정확도 측정)



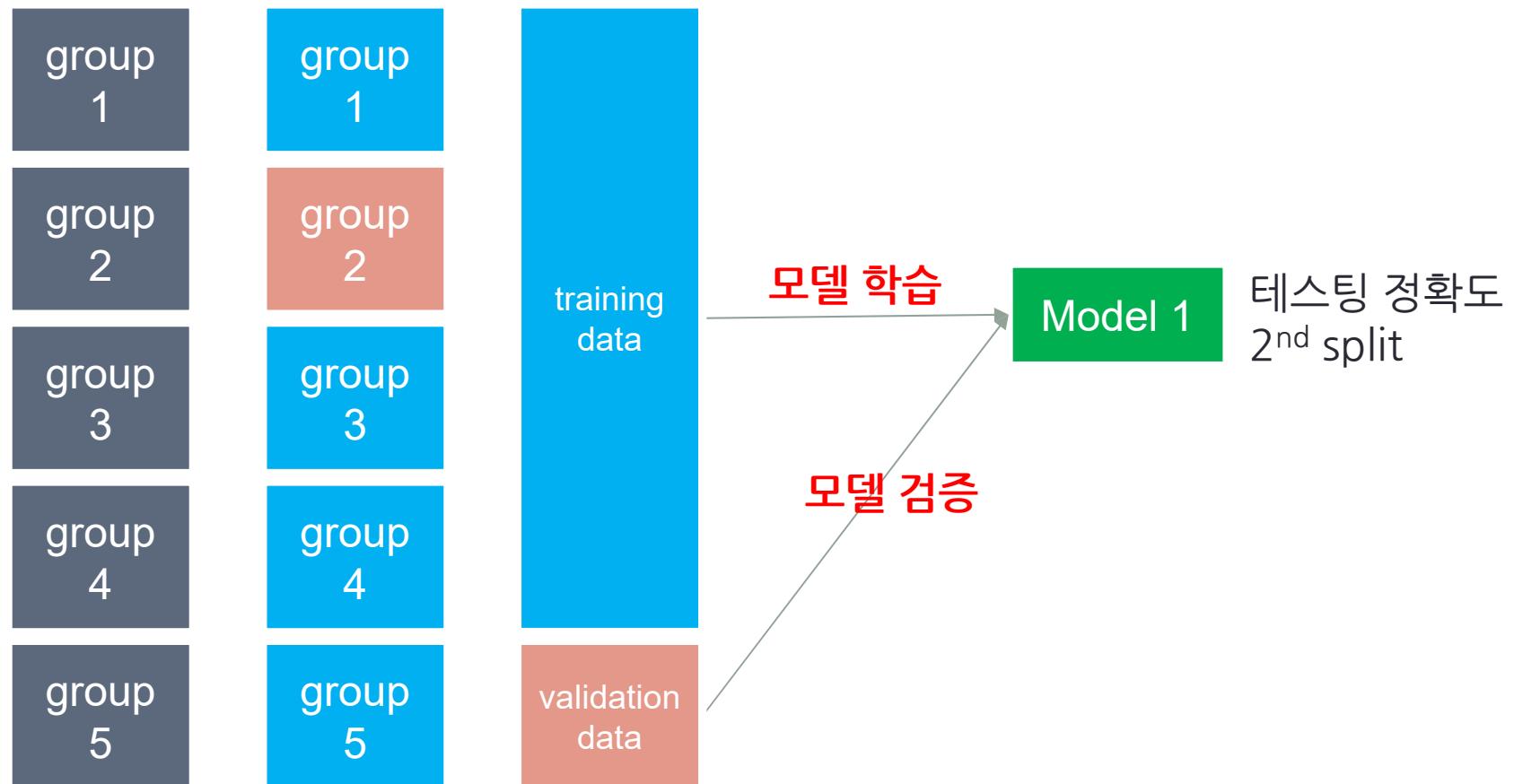
# 교차검증 알고리즘: 2<sup>nd</sup> split

- 두번째 그룹을 [testing data]로 나머지 데이터를 [training data]로 설정



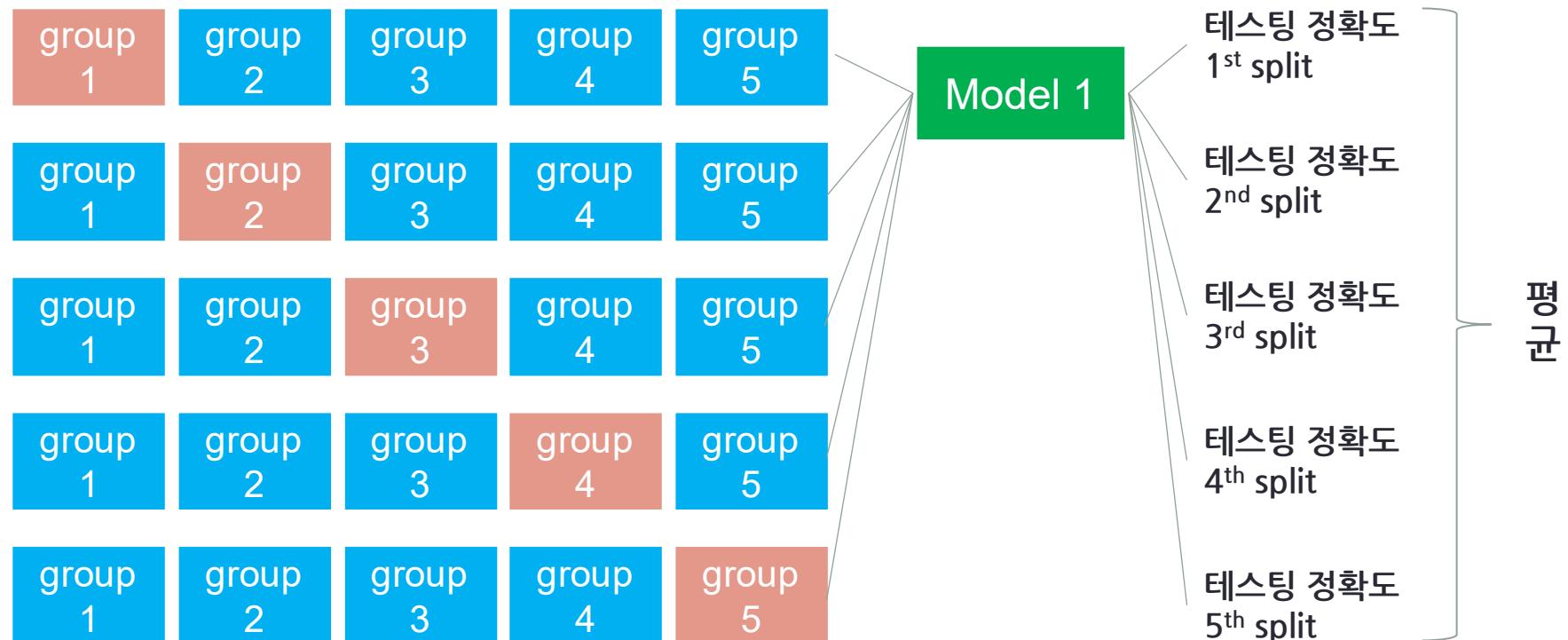
# 교차검증 알고리즘: 2<sup>nd</sup> split

- [training data]를 사용하여 모델 학습
- [testing data]를 사용하여 모델 검증(테스팅 정확도 측정)



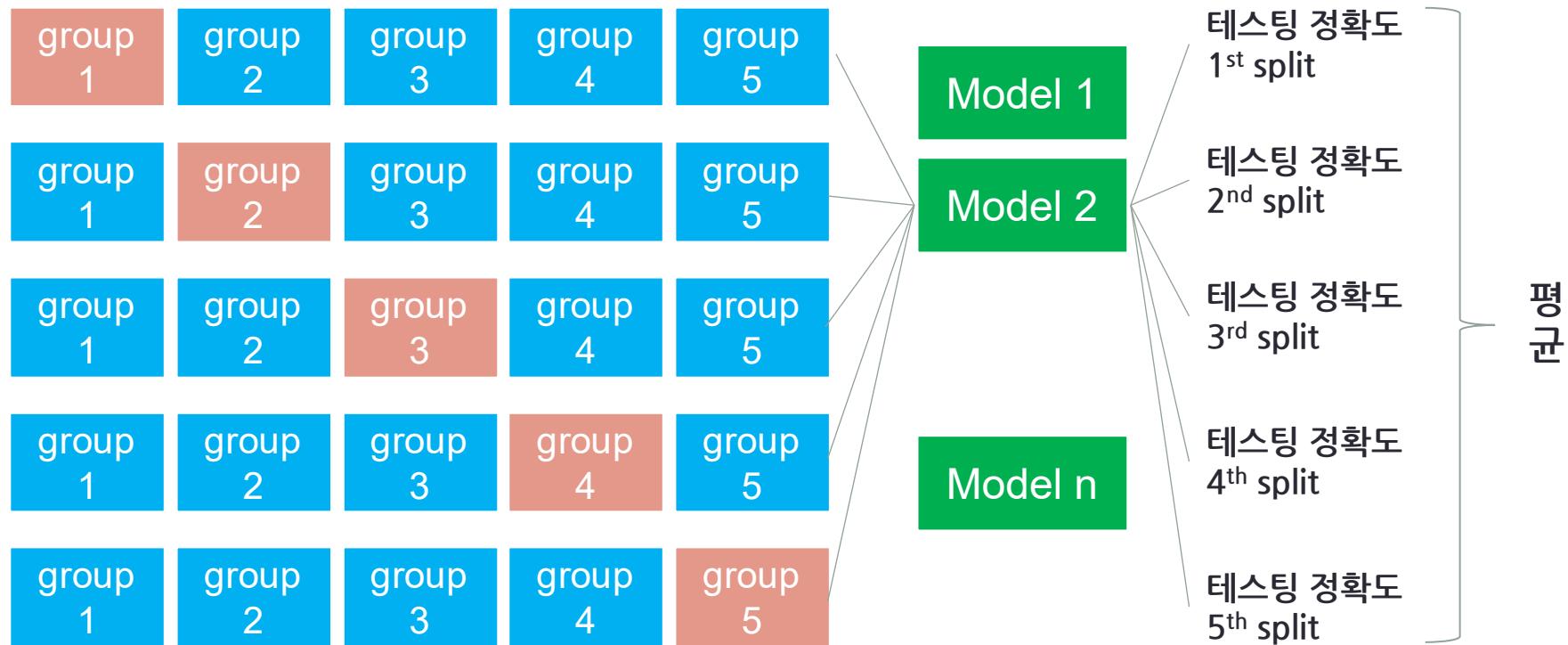
# 교차검증 알고리즘: 반복

- 위와 같은 방법으로 1부터 K번째 split에 대하여 같은 과정을 반복
- 각 과정에서 얻어진 K개의 테스팅 정확도의 평균 → 모델1의 정확도

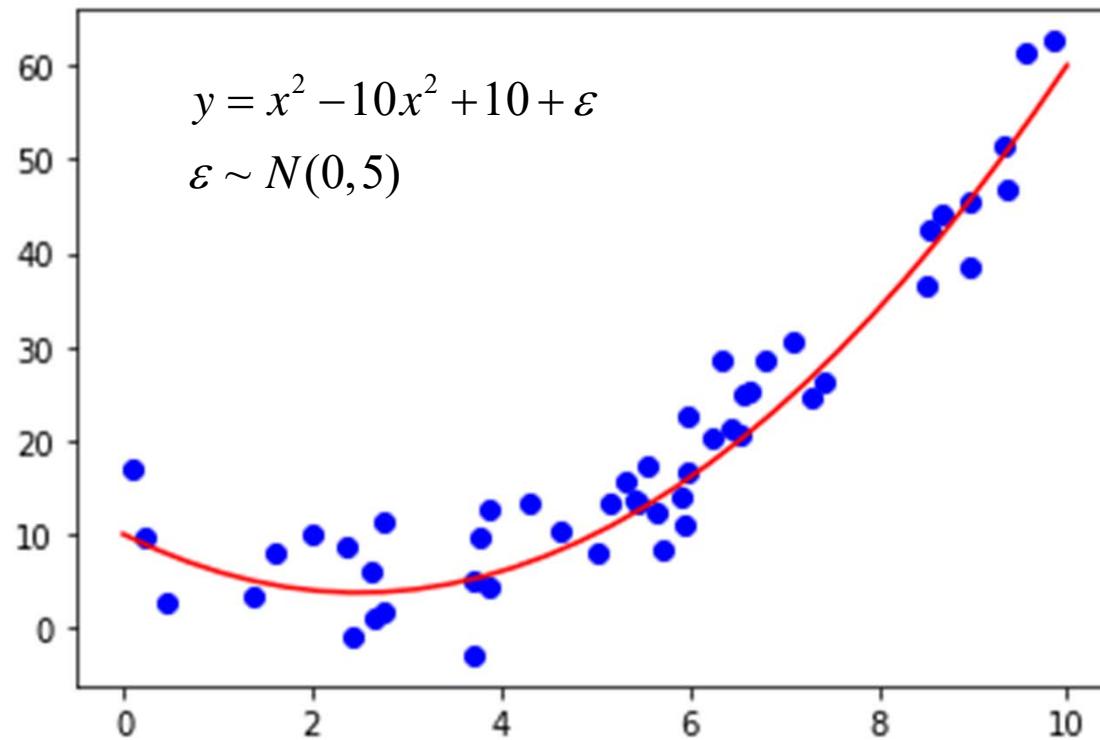


# 교차검증 알고리즘

- 비교하고자 하는 모델 1부터 n까지 동일한 과정을 반복
- 평균 테스팅 정확도가 가장 좋은 모델이 최적의 모델

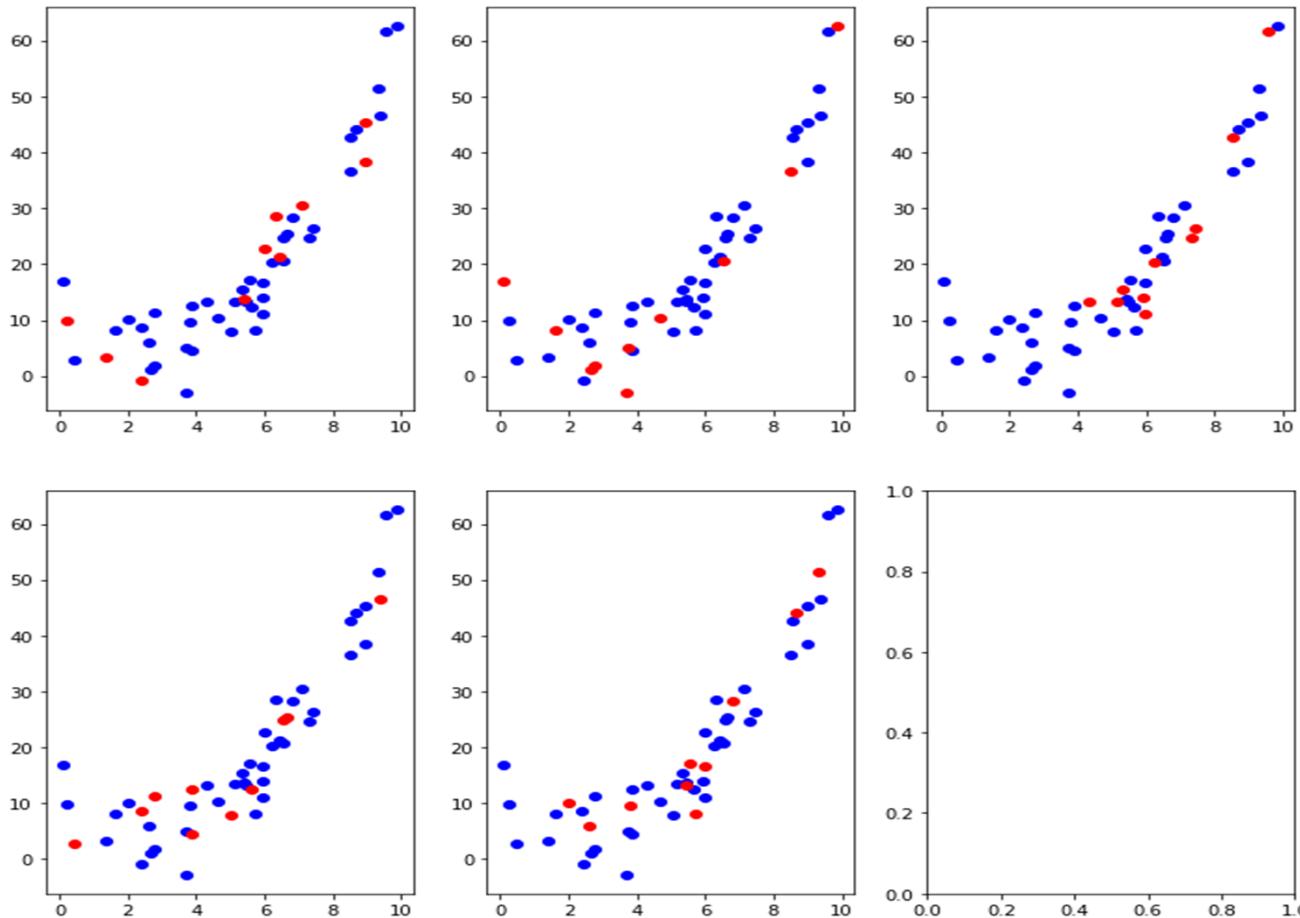


# 교차검증 예시



# 교차검증 예시

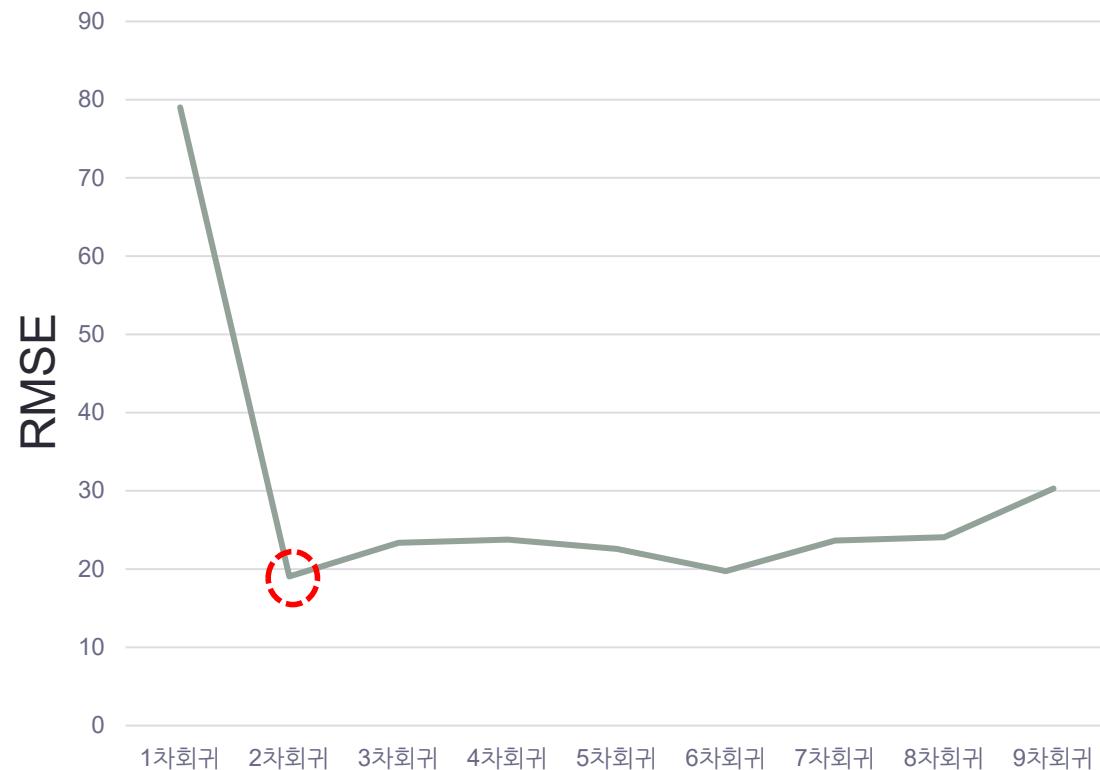
- 5개의 집단으로 데이터를 쪼개어 교차검증 실시 ( $K=5$ )



# 교차검증 예시

- 5개 집단으로 데이터를 쪼개어 교차검증 실시
- 교차검증 결과 최적의 모델: 2차회귀

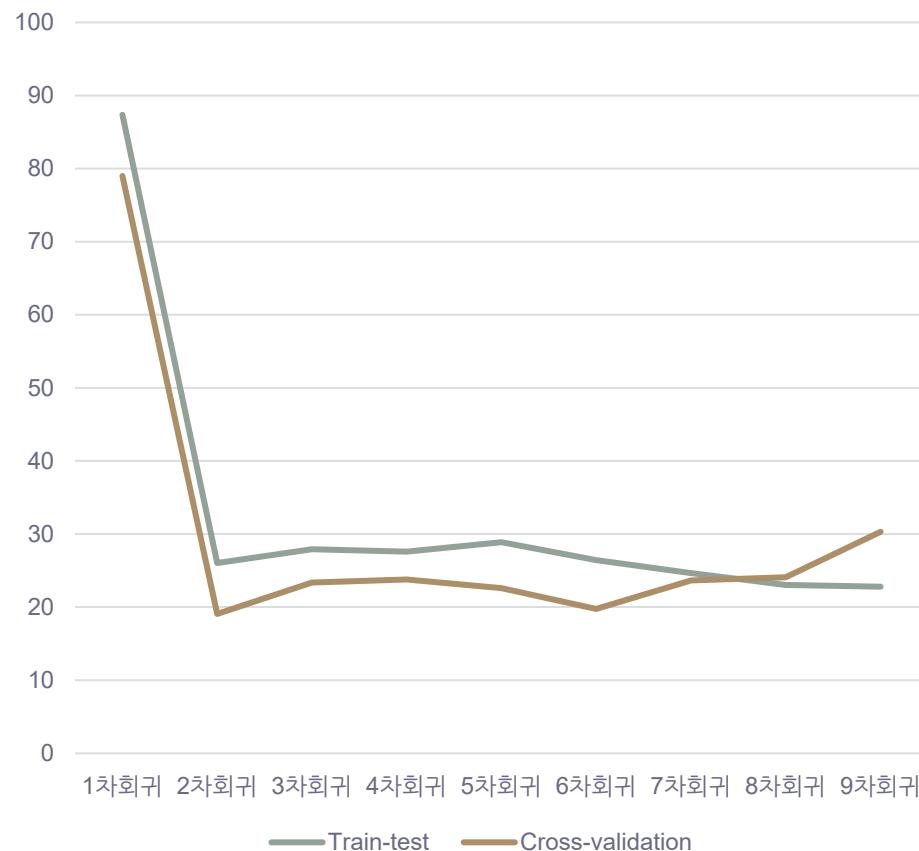
1차회귀	79.01455
2차회귀	19.07141
3차회귀	23.36856
4차회귀	23.76516
5차회귀	22.58806
6차회귀	19.73573
7차회귀	23.65242
8차회귀	24.08369
9차회귀	30.30078



# Train-test-split vs 교차검증

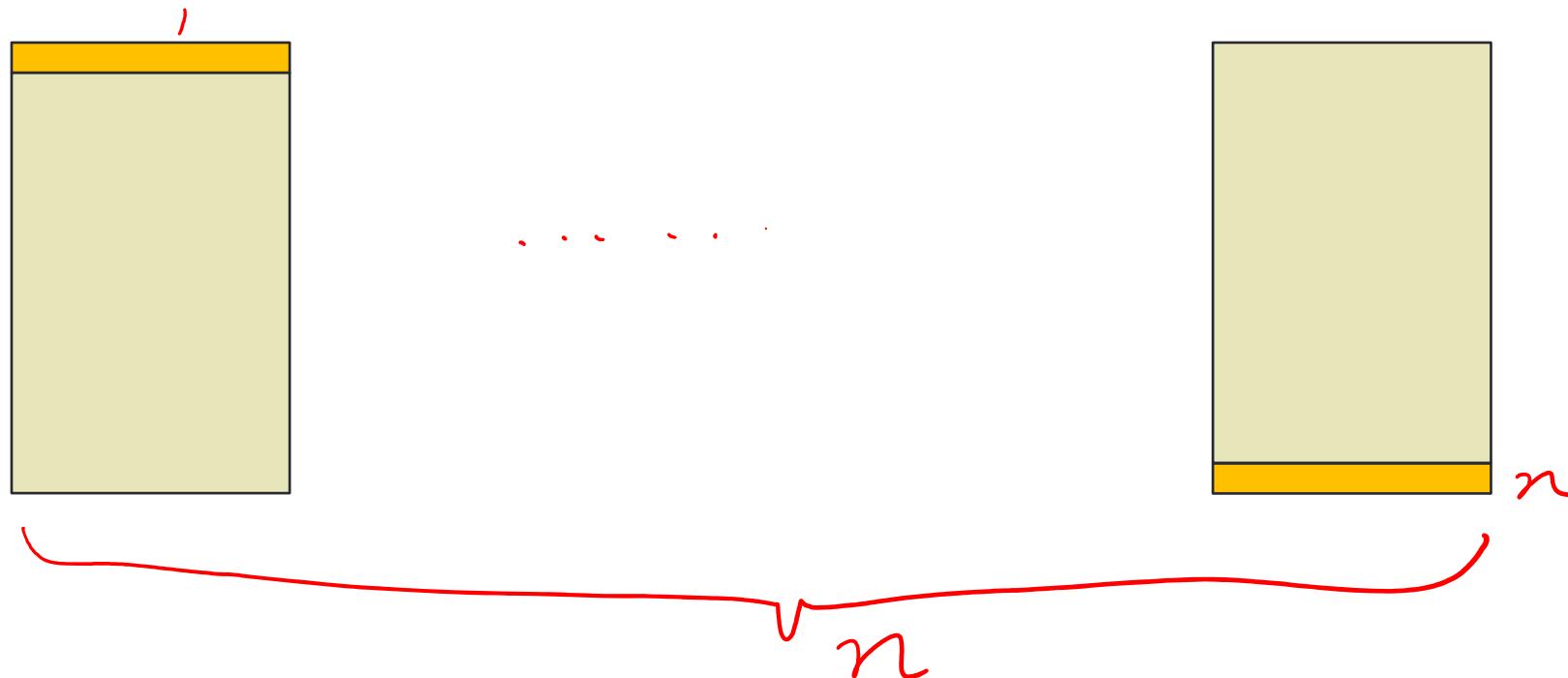
- 교차검증이 정답[2차회귀]을 더 잘 잡아내고 있는것을 확인할 수 있다.

	Train-test	Cross-validation
1차회귀	87.38192	79.01455
2차회귀	26.02434	19.07141
3차회귀	27.91851	23.36856
4차회귀	27.58762	23.76516
5차회귀	28.88298	22.58806
6차회귀	26.42483	19.73573
7차회귀	24.6514	23.65242
8차회귀	23.02441	24.08369
9차회귀	22.7894	30.30078



# LOOCV

- K-fold Cross Validation의 극단적 형태
- K=데이터의 갯수



# 하이퍼 패러미터 튜닝

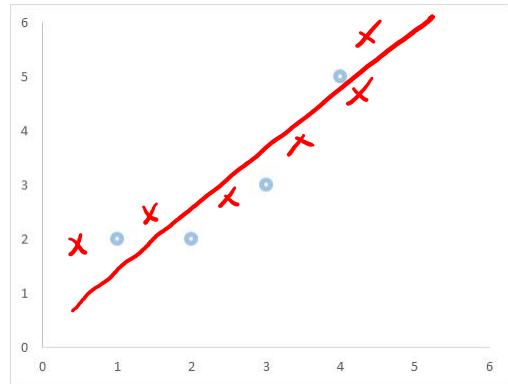
# Hyper Parameter Tuning

---

시스템경영공학부  
이지환 교수

# 모델의 유연성에 따른 일반화능력

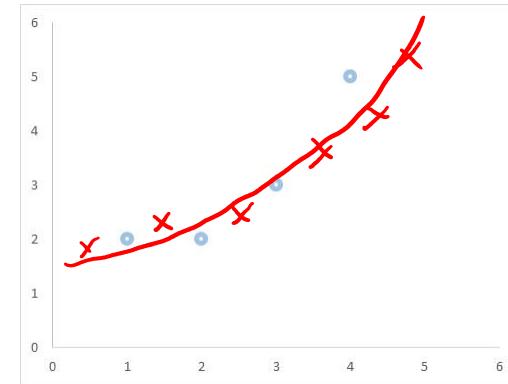
- 같은 데이터에 대해 서로 다른 모델 가정 가능
- 모델의 유연성 = 모델의 표현력



$$y = \beta_0 + \beta_1 x$$

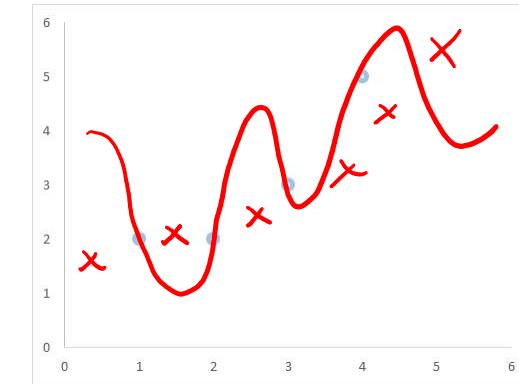
유연성

low



$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

med



$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$

high

very good

bad

훈련  
정확도

med

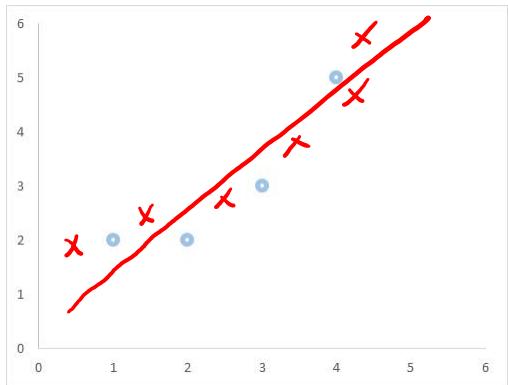
good

테스트  
정확도

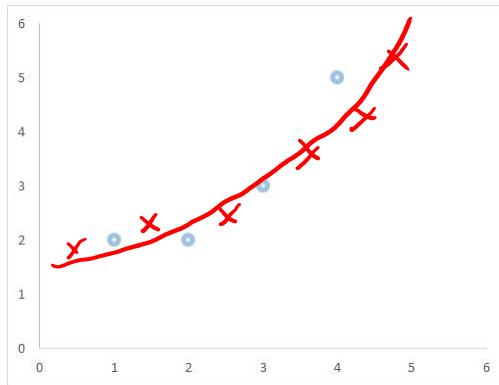
med

good

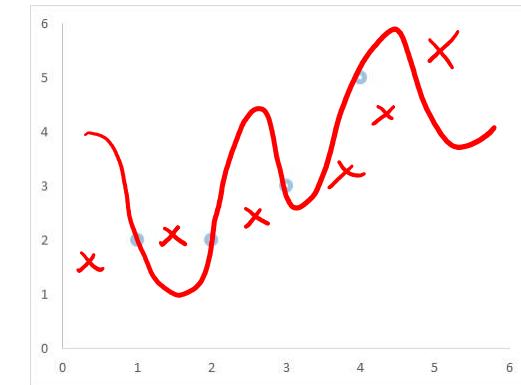
# Overfitting vs Underfitting



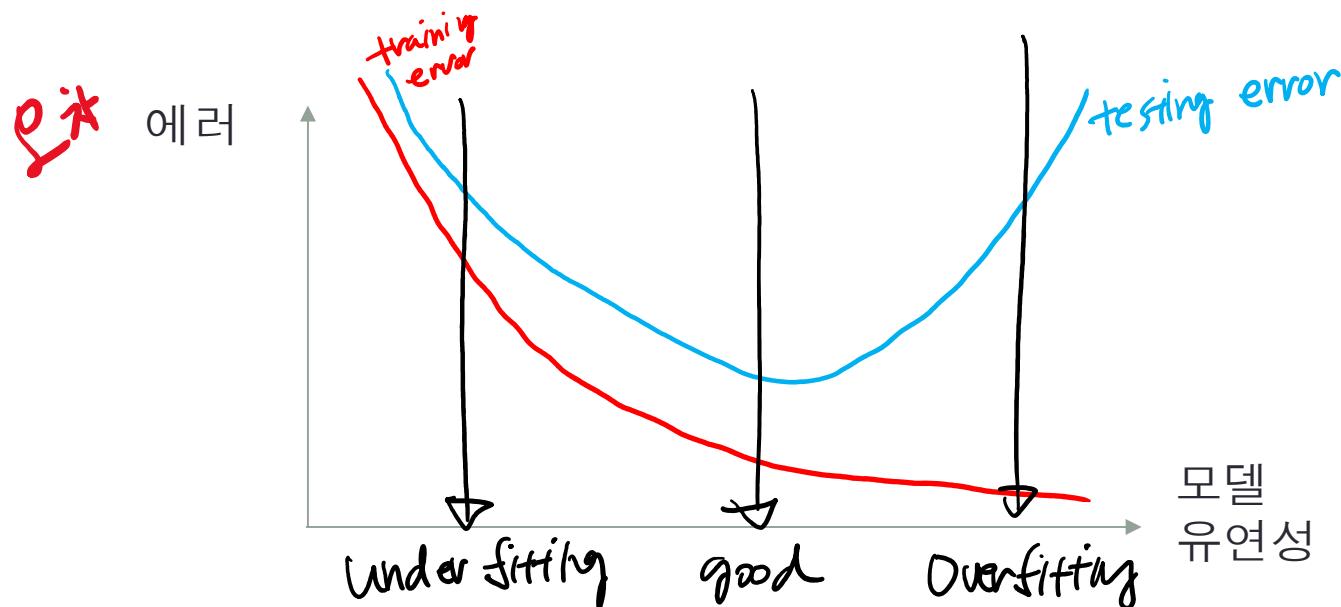
$$y = \beta_0 + \beta_1 x$$



$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$



$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$

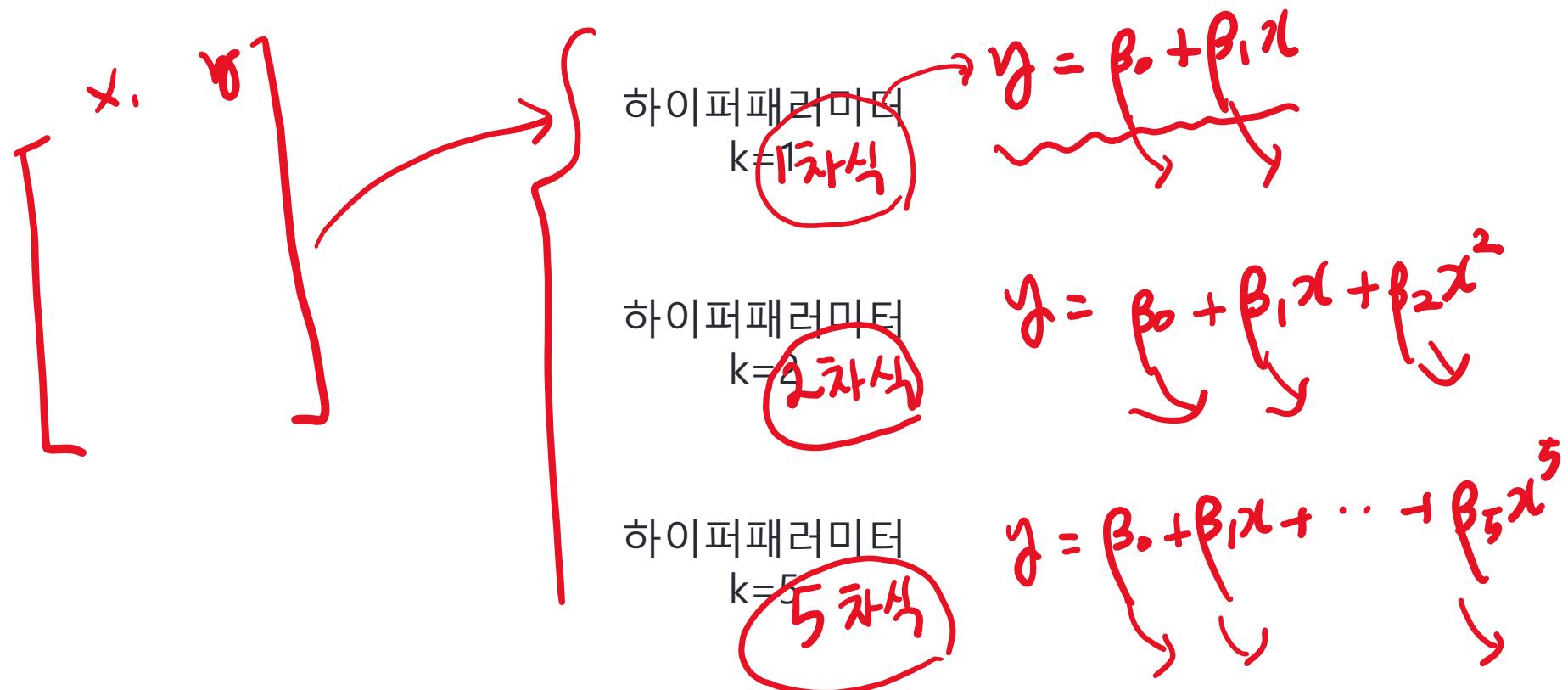


# 하이퍼 패러미터 (Hyper parameter)

- 모델의 일반화능력은 모델의 유연성과 깊은 관련성이 있음
- 하이퍼 패러미터 (Hyper parameter)
  - 모델의 유연성을 조정하는데 사용하는 패러미터
  - 학습 시작되기 전 지정됨
  - [사용자가] 사전적으로 지정하며 데이터로부터 학습되지 않음
- 패러미터 (Parameter)
  - 데이터로부터 학습됨

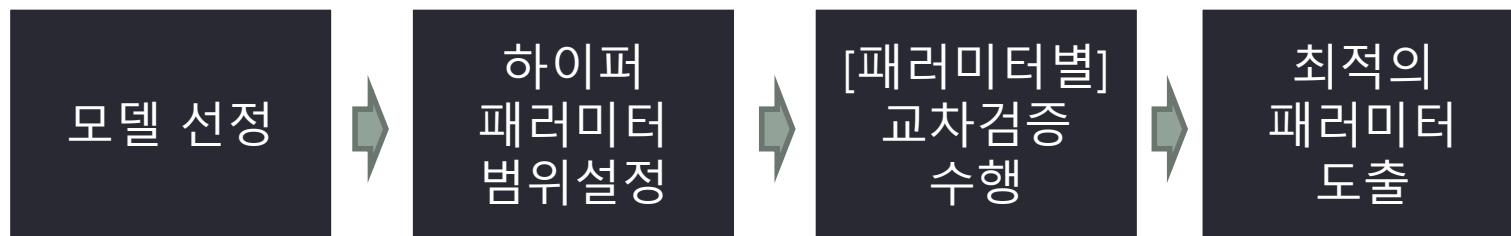
# 하이퍼 패러미터 예시

- 선형회귀식에 사용된 데이터의 다항식 차수 : K
- 예시)



# 하이퍼 패러미터 튜닝

- 하이퍼 패러미터의 조정을 통해 보다 정확한 모델을 찾아나기 위한 절차
- 하이퍼 패러미터 + 교차검증

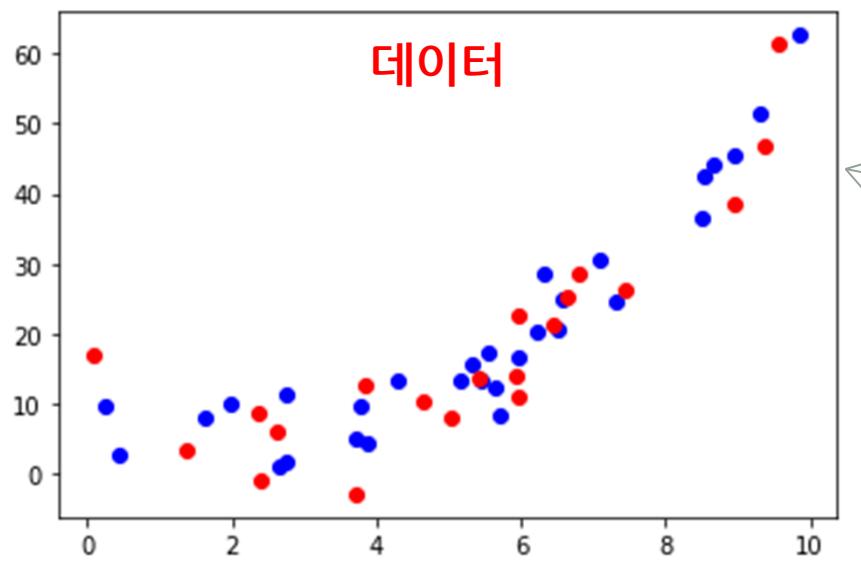


# 하이퍼 패러미터 튜닝 예시

- 선형회귀식에 사용된 데이터의 다행식 차수
- 하이퍼 패러미터: [1~9]

1) 모델선정: 선형회귀

2) 하이퍼 패러미터 범위설정



K=1

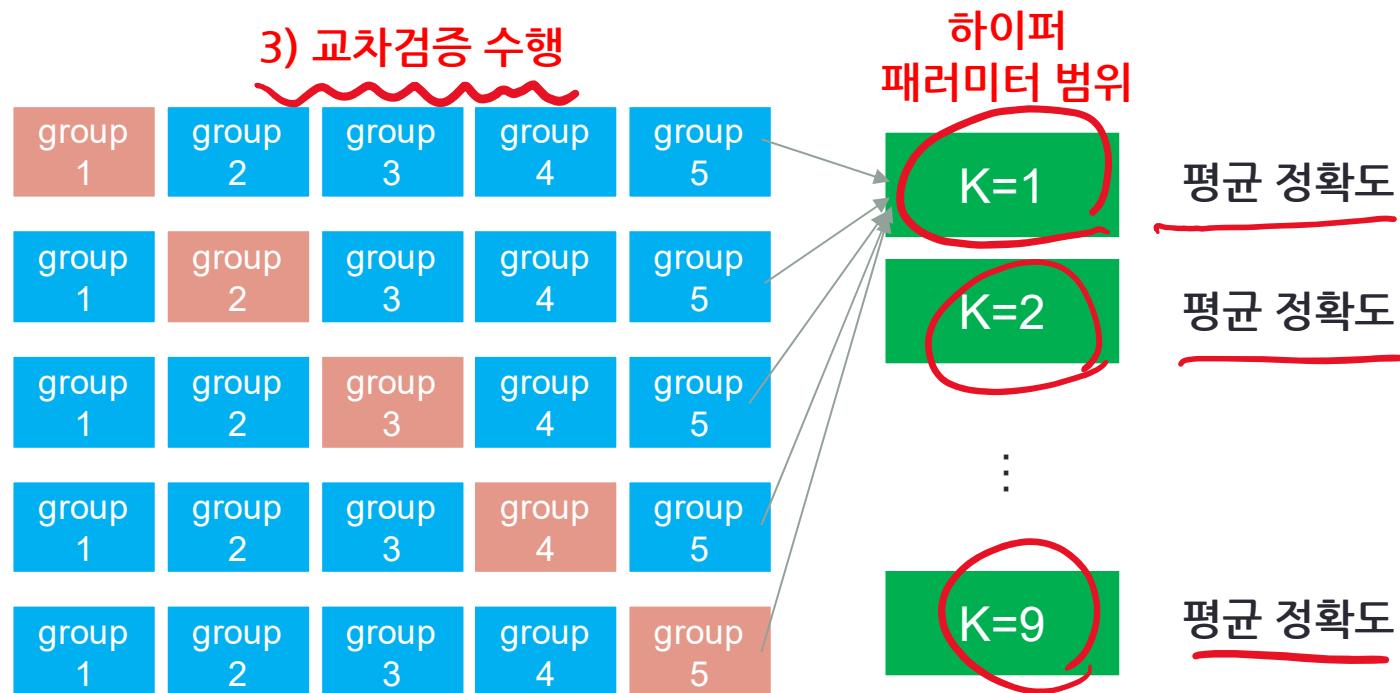
K=2

K=9

$K = [1 \sim 9]$

# 하이퍼 패러미터 튜닝 예시

- [train-test split] 한번의 테스팅 정확도 측정으로 인해 주어진 데이터에 지나치게 과적합된 모델을 찾을 가능성이 있음
- 교차검증을 통해 여러 번 데이터의 테스팅 정확도를 측정하여 과적합 방지 가능



# 하이퍼 패러미터 튜닝 예시

- 5개 집단으로 데이터를 쪼개어 교차검증 실시
- 교차검증 결과 최적의 모델: 2차회귀

1차회귀	79.01455
2차회귀	19.07141
3차회귀	23.36856
4차회귀	23.76516
5차회귀	22.58806
6차회귀	19.73573
7차회귀	23.65242
8차회귀	24.08369
9차회귀	30.30078

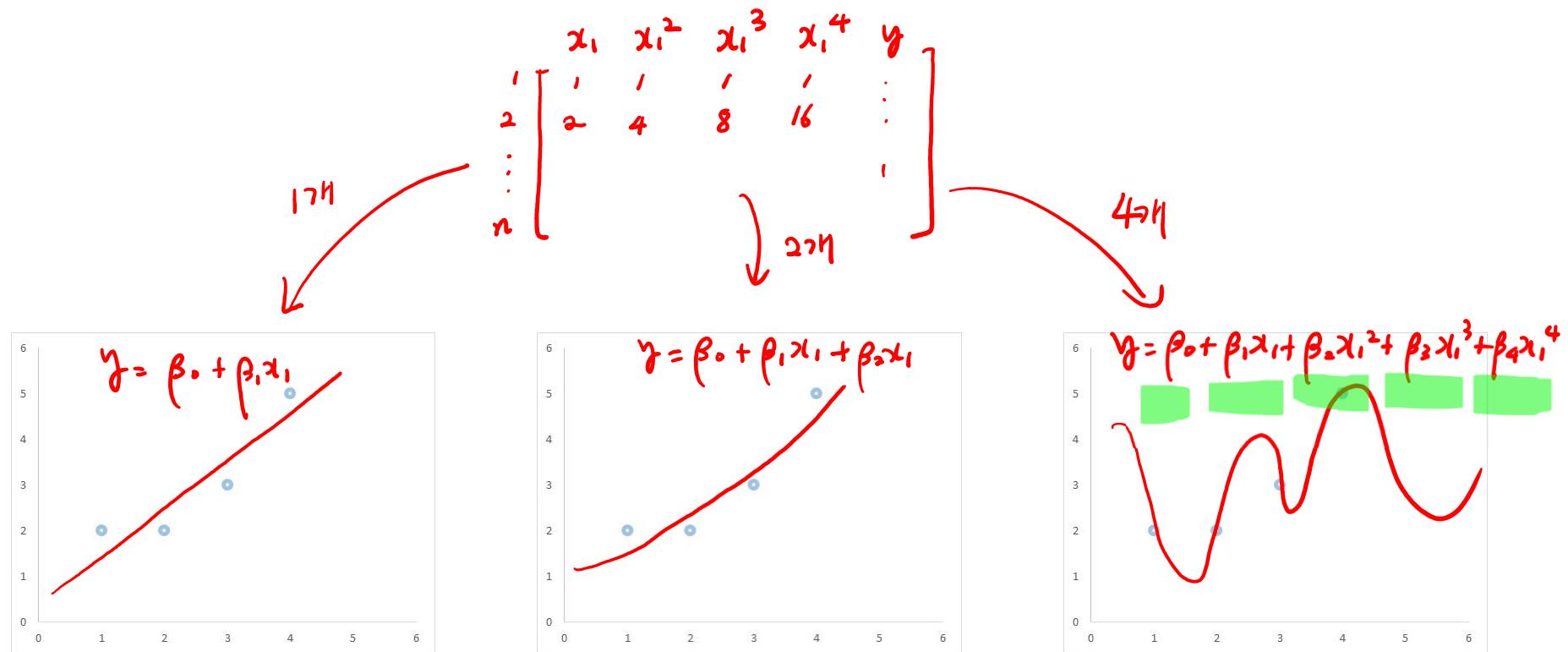


# 모델규제(model regularization)

- 선형회귀 모델의 비용함수에 추가적 패널티를 부과함으로써 과적합을 방지
- 패널티의 정도를 결정하는 계수  $\lambda \rightarrow$  하이퍼 패러미터
- 모델 규제 하이퍼 패러미터
  - 하이퍼 패러미터 계수에 따라 penalty의 정도를 조정 가능
  - 큰  $\lambda \rightarrow$  강력한 규제 (오버피팅 방지)
  - 작은  $\lambda \rightarrow$  느슨한 규제 (언더피팅 방지)

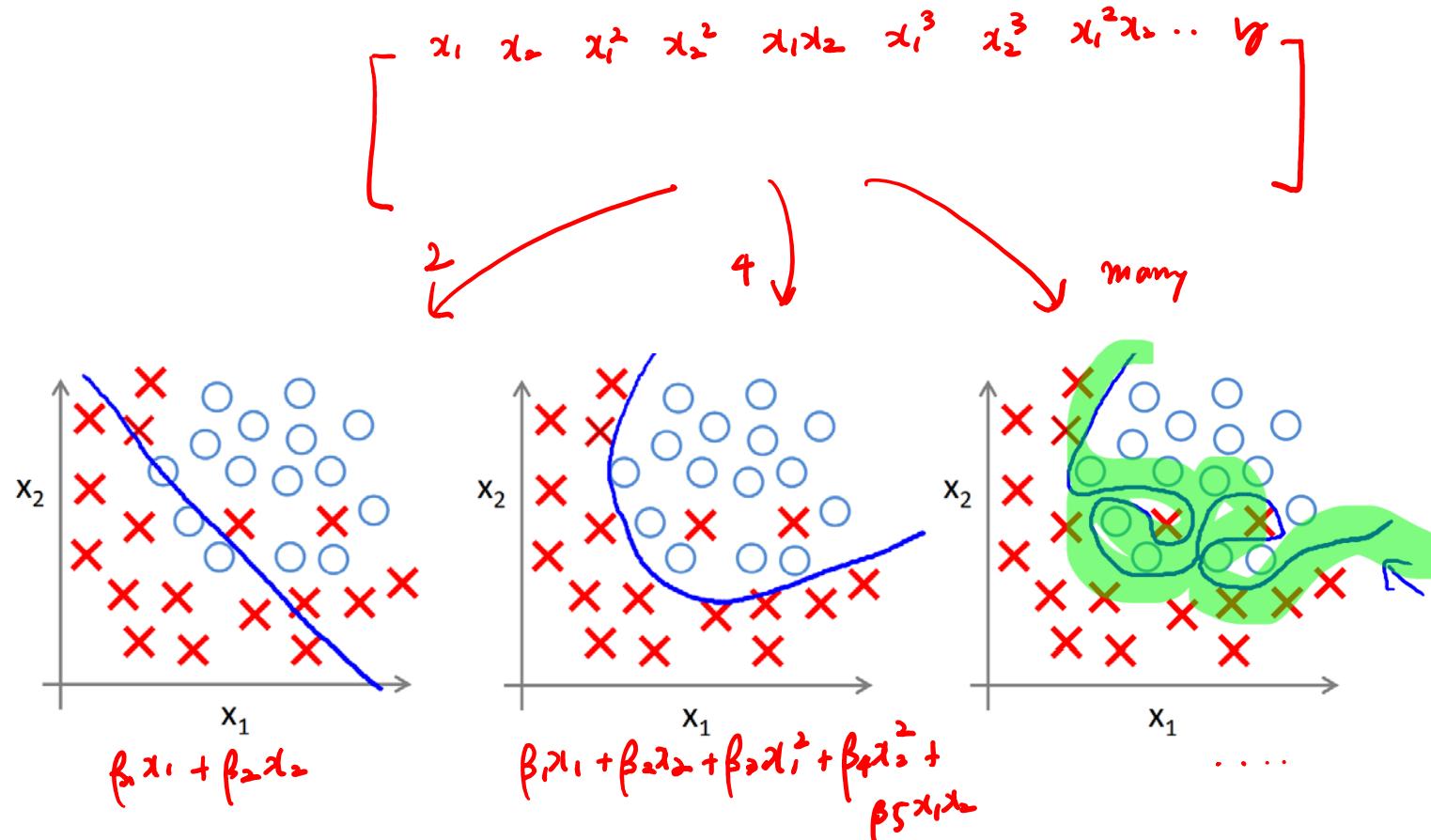
# 과적합의 원인: Feature의 수

- 더 많은 수의 Feature  $\rightarrow$  더 유연한 모델을 만들 수 있다.
- 더 유연한 모델  $\rightarrow$  Overfitting 발생



# 과적합의 원인: Feature의 수

- 더 많은 수의 Feature  $\rightarrow$  더 유연한 모델을 만들 수 있다.
- 더 유연한 모델  $\rightarrow$  Overfitting 발생



# 모델규제(model regularization)

- 선형회귀 모델의 비용함수에 추가적 패널티를 부과함으로써 과적합을 방지
- 패널티의 정도를 결정하는 계수  $\lambda$  → 하이퍼 패러미터
- 모델 규제 하이퍼 패러미터
  - 하이퍼 패러미터 계수에 따라 penalty의 정도를 조정 가능
  - 큰  $\lambda$  → 강력한 규제 (오버피팅 방지)
  - 작은  $\lambda$  → 느슨한 규제 (언더피팅 방지)

# 선형회귀의 비용함수

{  
     $y_{(i)}$  :  $i$  번째 데이터 정답  
     $f_{\beta}$  : 학습하고자하는 모델  
     $\hat{y}_{(i)} = f_{\beta}(x_{(i)})$   
                : 모델이 의해 예측값}

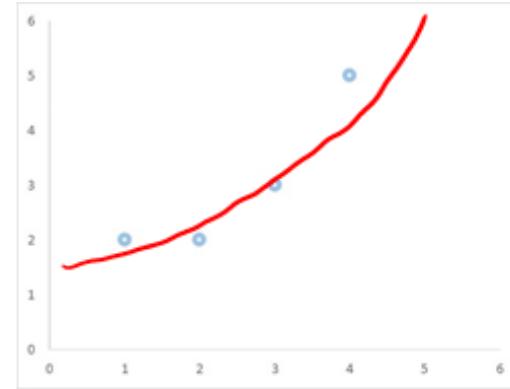
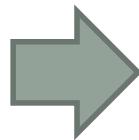
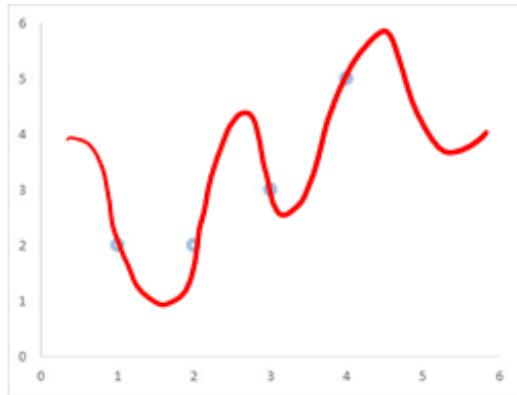
Cost  
Function

$$\beta = (\beta_0, \beta_1, \dots, \beta_n)$$
$$J(\beta) = \frac{1}{2n} \sum_{i=1}^n (\hat{y}_{(i)} - y_{(i)})^2 = \frac{1}{2n} \sum_{i=1}^n (f_{\beta}(x_{(i)}) - y_{(i)})^2$$

Learning

$$\min_{\beta} \frac{1}{2n} \sum_{i=1}^n (f_{\beta}(x_{(i)}) - y_{(i)})^2$$

# 모델 규제의 아이디어



$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 x^4$$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2$$

$$\min_{\beta} \left( \frac{1}{2n} \sum_{i=1}^n (f_p(x_i) - y_i)^2 + 1000\beta_3^2 + 1000\beta_4^2 \right)$$

loss  
정규화 항  
추가정수

$\beta_3 \approx 0 \quad \beta_4 \approx 0$

# 모델규제를 위한 비용함수

- Small values for parameters
  - Simpler hypothesis
  - Less prone to overfitting
- Examples:

→ Ridge Regression

features  $x_1 x_2 \dots x_{100}$

parameters  $\beta_0 \beta_1 \dots, \beta_{100}$

$$J(\beta) = \frac{1}{2n} \sum_{i=1}^n (\hat{y}_{(i)} - y_{(i)})^2 + \boxed{\lambda \sum_{j=1}^p \beta_j^2}$$



# 모델 규제 하이퍼 패러미터

- In regularized regression, we choose betas to minimizes

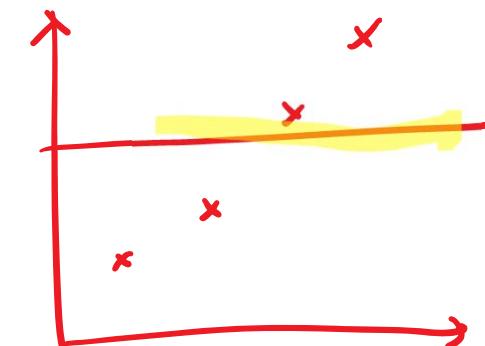
$$J(\beta) = \frac{1}{2n} \left[ \sum_{i=1}^n (f_\beta(x_i) - y_i)^2 + \sum_{j=1}^p \lambda \beta_j^2 \right]$$

- What if  $\lambda$  is set to extremely large value??

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_1^3 + \beta_4 x_1^4$$

$$J(\beta) = \frac{1}{2n} \left[ \sum_{i=1}^n (f_\beta(x_i) - y_i)^2 + \lambda \beta_1^2 + \lambda \beta_2^2 + \lambda \beta_3^2 + \lambda \beta_4^2 \right]$$

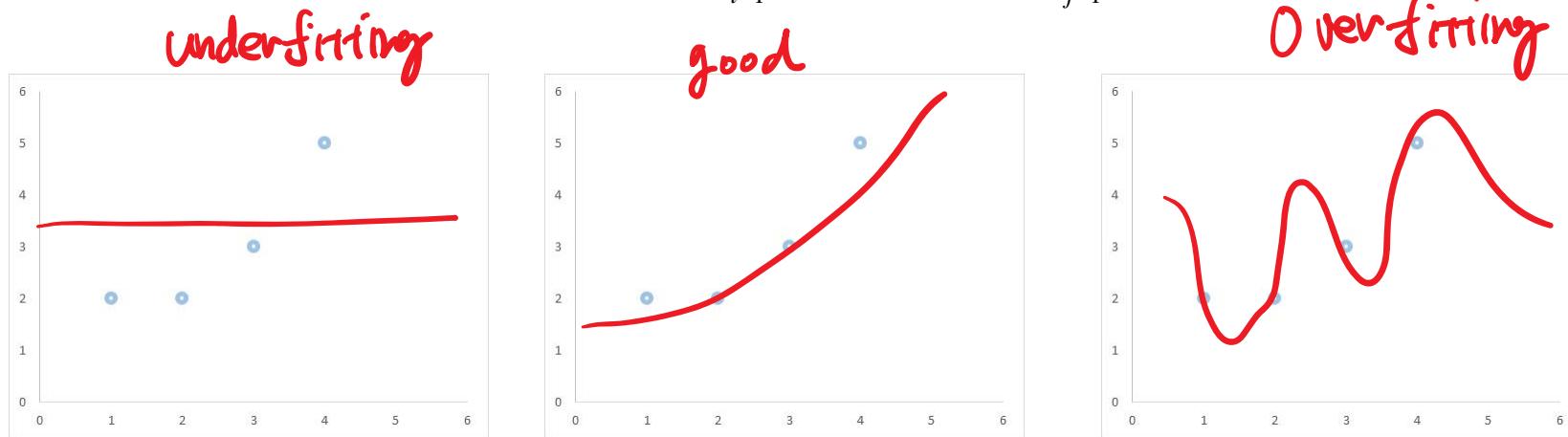
$$\min_{\beta} J(\beta) \rightarrow \beta_1 \approx \beta_2 \approx \beta_3 \approx \beta_4 \approx 0$$



# 하이퍼 패러미터에 따른 유연성의 조절

Model:  $y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_1^3 + \beta_4 x_1^4$

Cost Function:  $J(\beta) = \frac{1}{2n} \left[ \sum_{i=1}^n (f_\beta(x_i) - y_i)^2 - \sum_{j=1}^p \lambda \beta_j^2 \right]$



Large  $\lambda$

ex)  
 $\lambda = 1000$   
 $\beta_i \approx 0$

Intermediate  $\lambda$

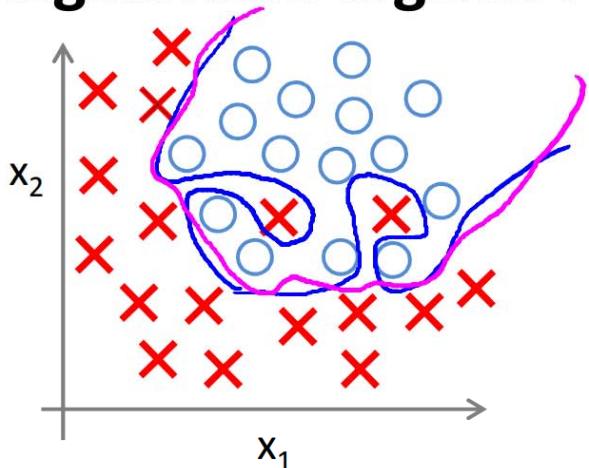
적절한수준

small  $\lambda$

$\lambda = 0$   
 $\beta_i \neq 0$

# Logistic Regression

Regularized logistic regression.



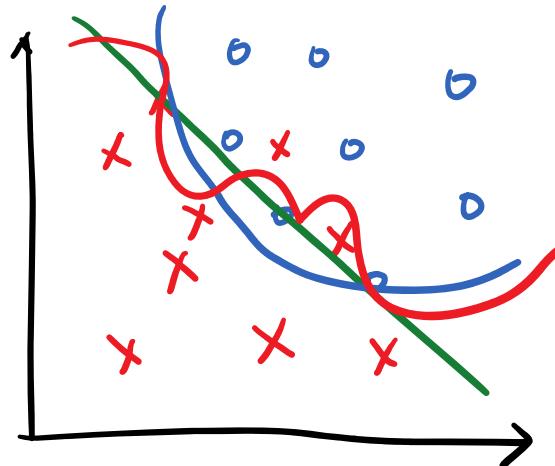
$$h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

Cost function:

$$\rightarrow J(\theta) = - \left[ \frac{1}{m} \sum_{i=1}^m y^{(i)} \log h_{\theta}(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

$\theta_1, \theta_2, \dots, \theta_n$

# Logistic Regression



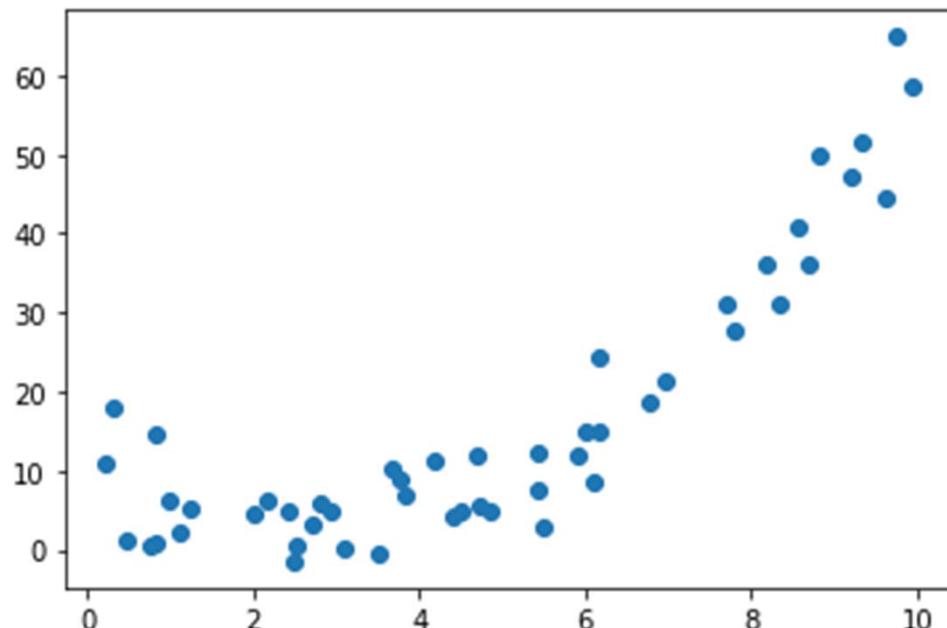
decision line

$$\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2 + \beta_4 x_1^2 + \beta_5 x_2^2 \\ + \dots$$

$$J(\beta) = \frac{1}{n} \left[ y_i \log f_{\beta}(x_i) + (1 - y_i) \log (1 - f_{\beta}(x_i)) \right] \\ + \frac{\lambda}{2n} \sum_{j=1}^p \lambda \beta_j^2$$

# 모델 규제 하이퍼 패러미터 튜닝

데이터



선형회귀모형  
→ 10차식  
( $k=10$ )

하이퍼  
패러미터 범위

$\lambda = 0$

$\lambda = 0.1$

$\lambda = 1$

$\lambda = 10$

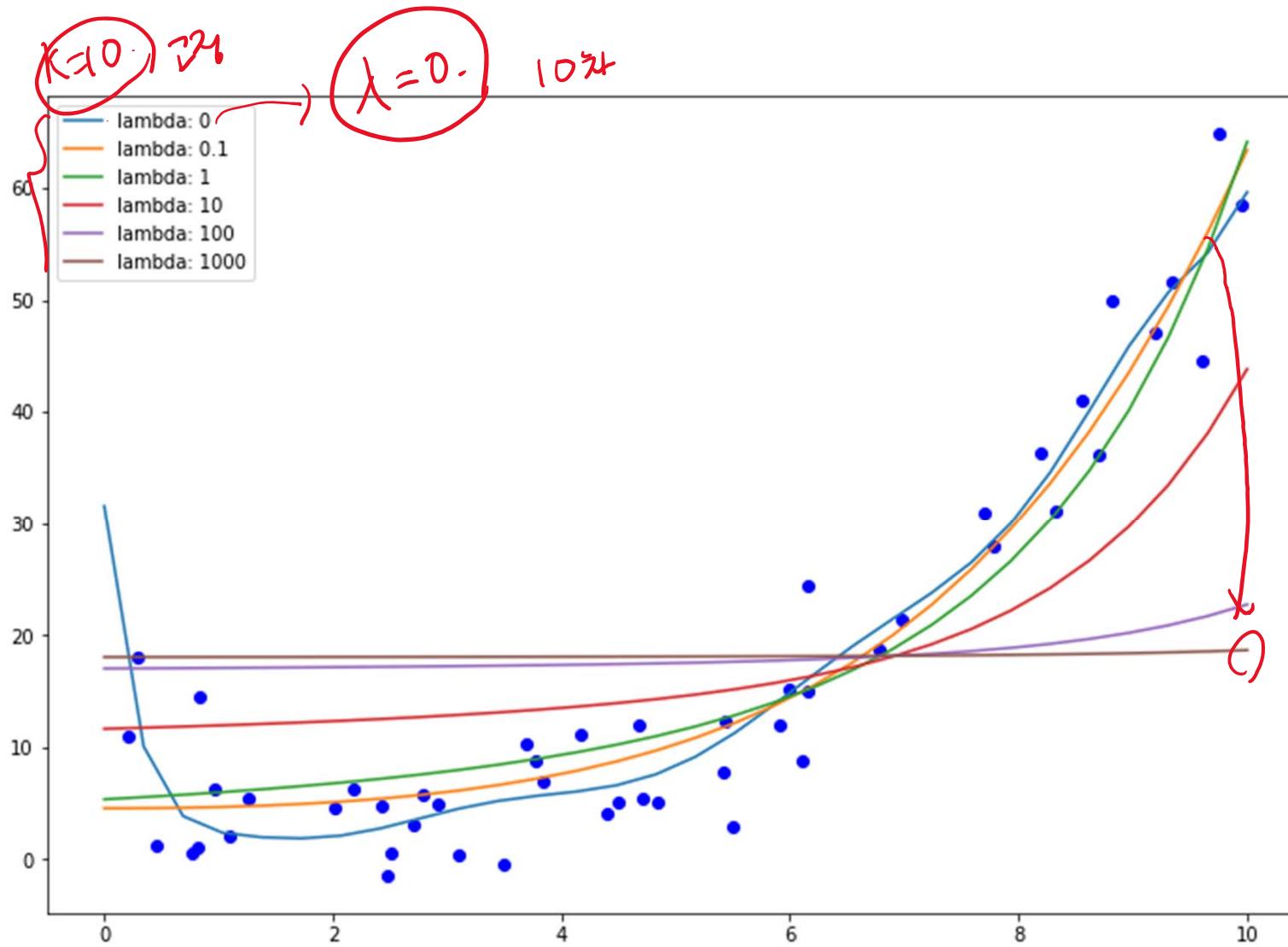
$\lambda = 100$

$\lambda = 1000$

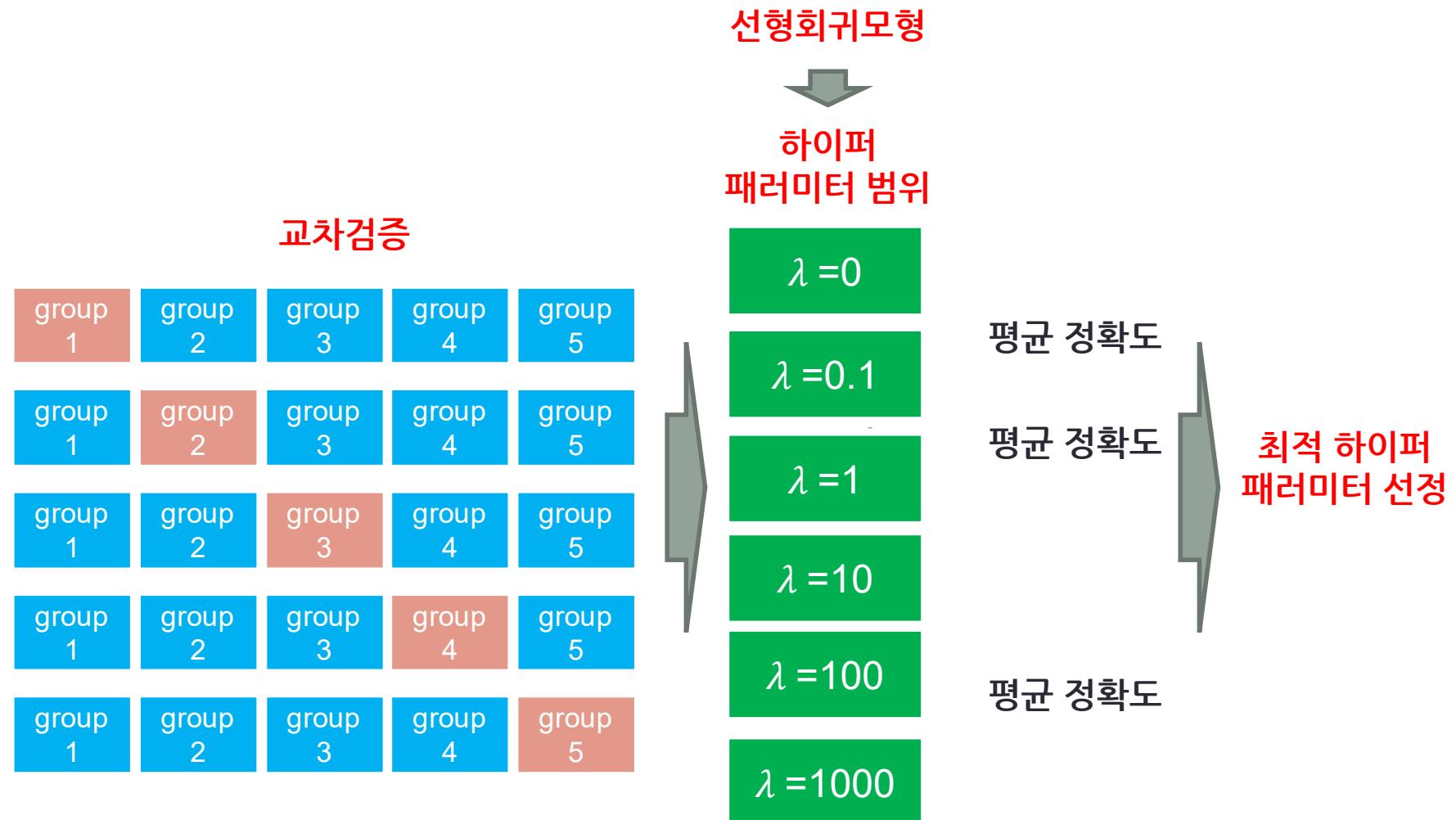
유연성 +

유연성 -

# 하이퍼 패러미터에 따라 학습된 모형



# 모델 규제에서 하이퍼 패러미터 튜닝



# 교차검증을 통한 최적 패러미터 선정



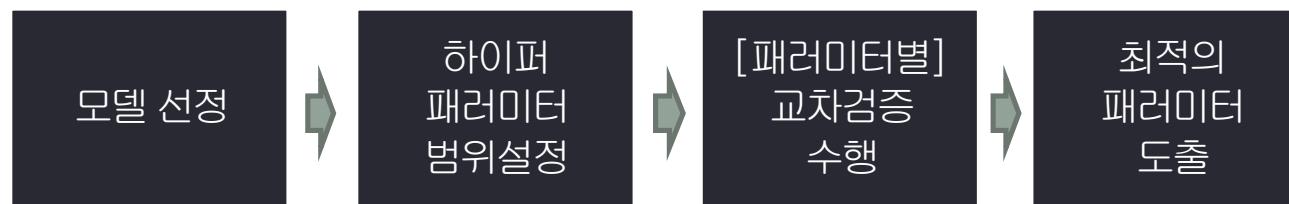
# 정리

- 하이퍼 패러미터 (Hyper parameter)
  - 모델의 유연성을 조정하는데 사용하는 패러미터
  - 학습 시작되기 전 지정됨
  - [사용자가] 사전적으로 지정하여 데이터로부터 학습되지 않음
- 선형회귀 모형에서 조정할 수 있는 패러미터
  - K: 다항식 차수
  - $\lambda$ : 모델규제 비용함수의 계수
  - 변수간 상호작용 삽입 여부



# 정리

- 하이퍼 패러미터 (Hyper parameter)
  - 모델의 유연성을 조정하는데 사용하는 패러미터
  - 학습 시작되기 전 지정됨
  - [사용자가] 사전적으로 지정하여 데이터로부터 학습되지 않음
- 모델마다 조정할 수 있는 다양한 패러미터 존재
  - 선형회귀/로지스틱회귀 → ( $\lambda$ , 차수, 상호작용여부)
  - Decision Tree → (Depth, Min-sample\_size)
  - KNN → K의 수
  - 딥 러닝 → (최적화 알고리즘 종류, Epochs, 노드의 수, 깊이, …)



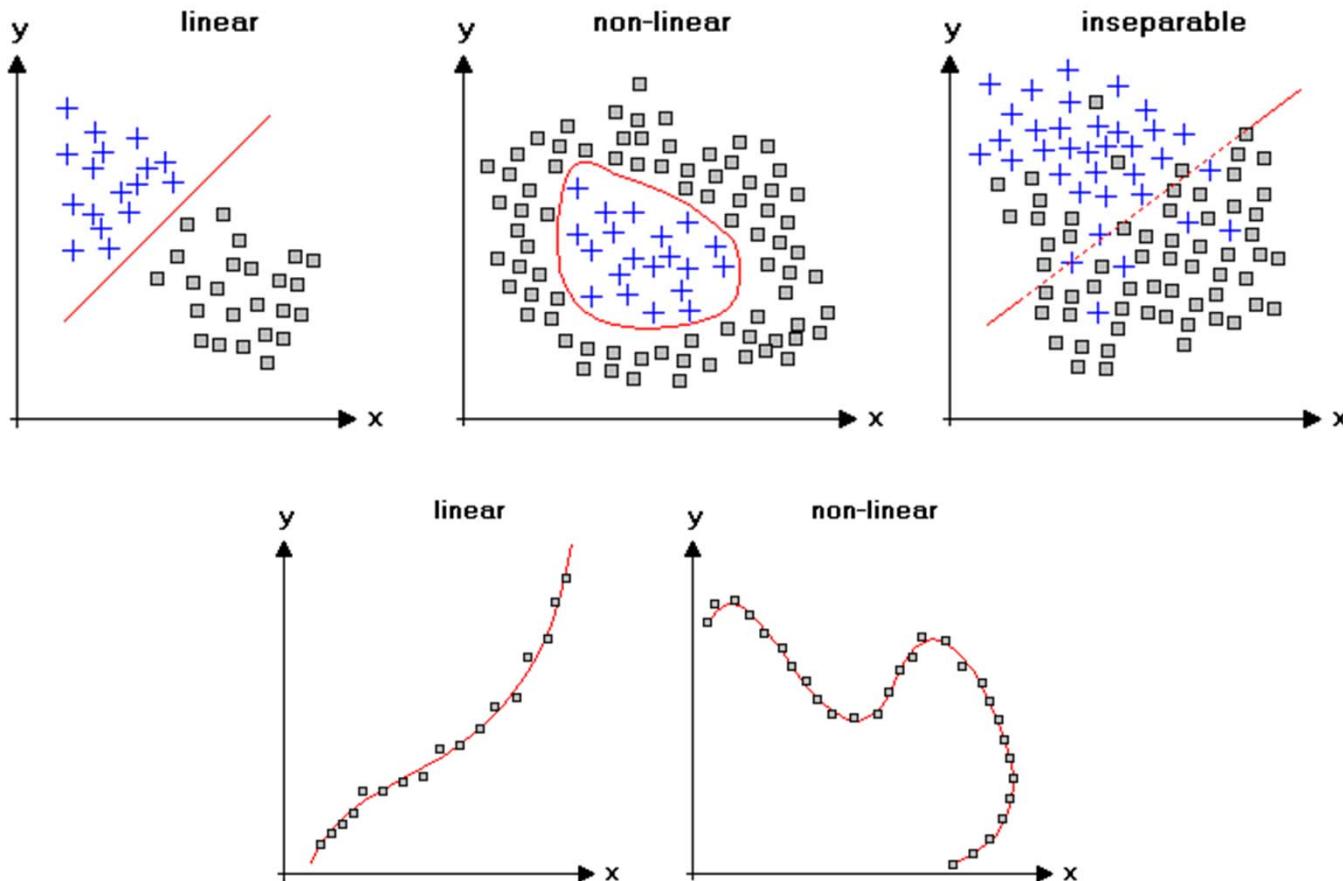
# 선형회귀모형을 넘어 (KNN)

---

시스템경영공학부  
이지환 교수

# Complex patterns of real-world dataset

- 선형 모형들의 중요 가정
  - 변수의 선형조합으로  $y$ 값을 예측
  - 하지만 실제 세계의 데이터는 선형적인 패턴을 가지고 있지 않은 경우가 많음

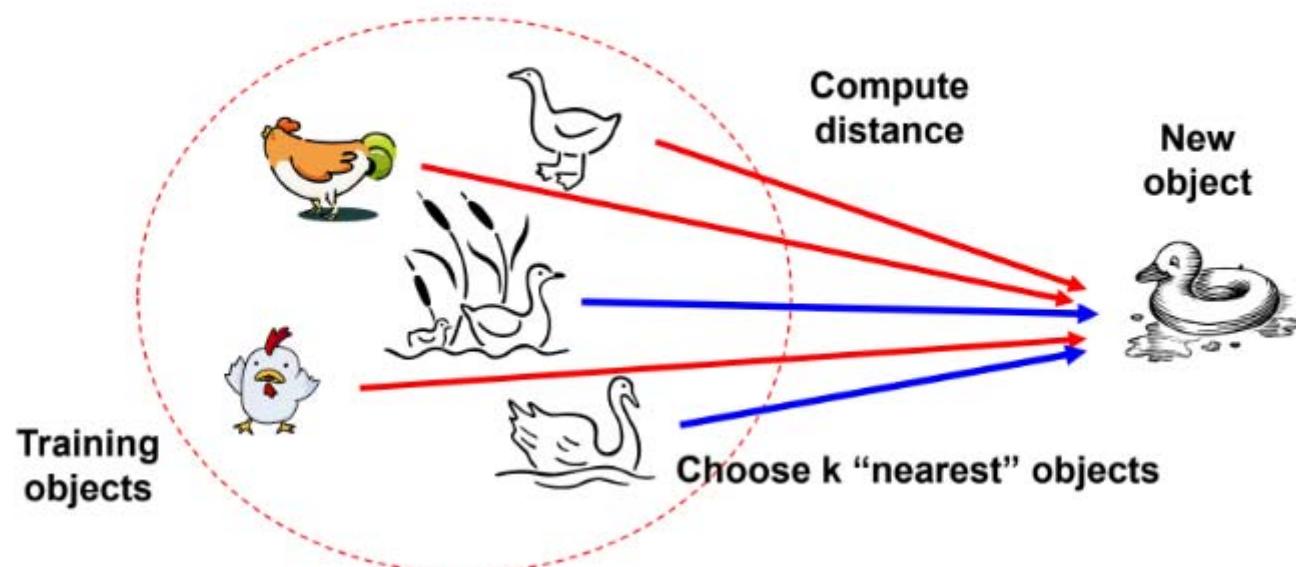


# 머신러닝 모형의 분류

- 예측해야하는 값이 실수인가?
  - (Yes) 회귀(Regression )
  - (No) 분류(Classification)
- 정답을 알고있는 데이터인가?
  - (Yes) 지도학습(Supervised Learning)
  - (No) 비지도학습(Unsupervised Learning)
- 패러미터로 모델을 표현할 수 있는가?
  - (Yes) Parametric Method
  - (No) Non-parametric Method

# KNN - Idea

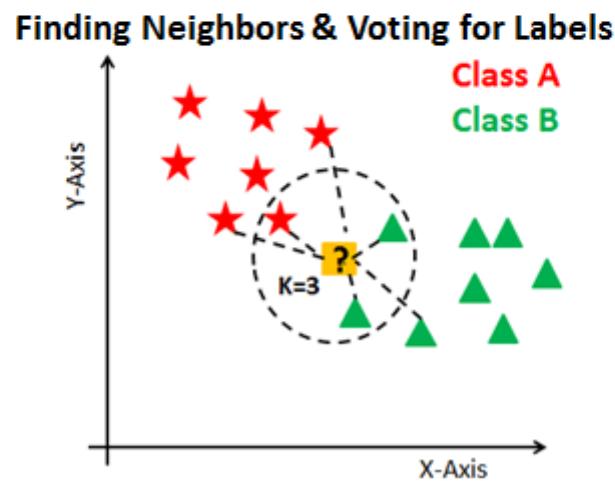
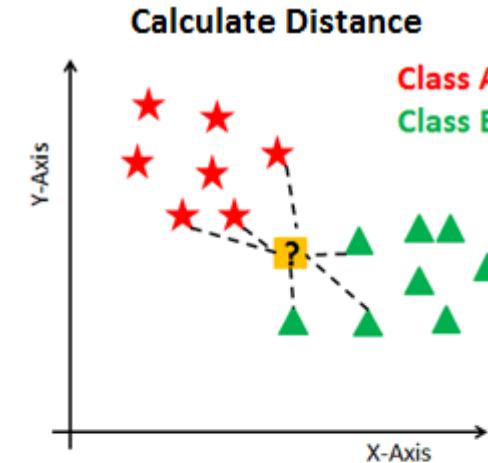
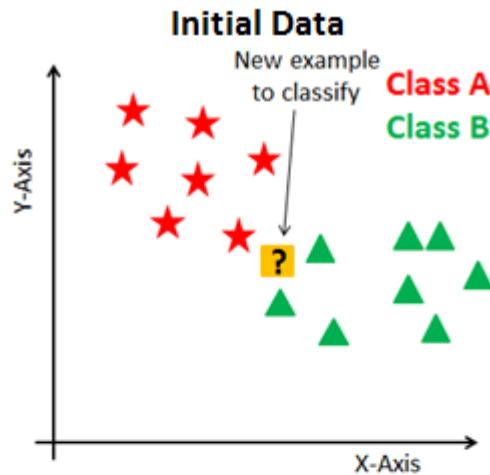
- If it walks like a duck, quacks like a duck, then it's probably a duck



# KNN - Idea

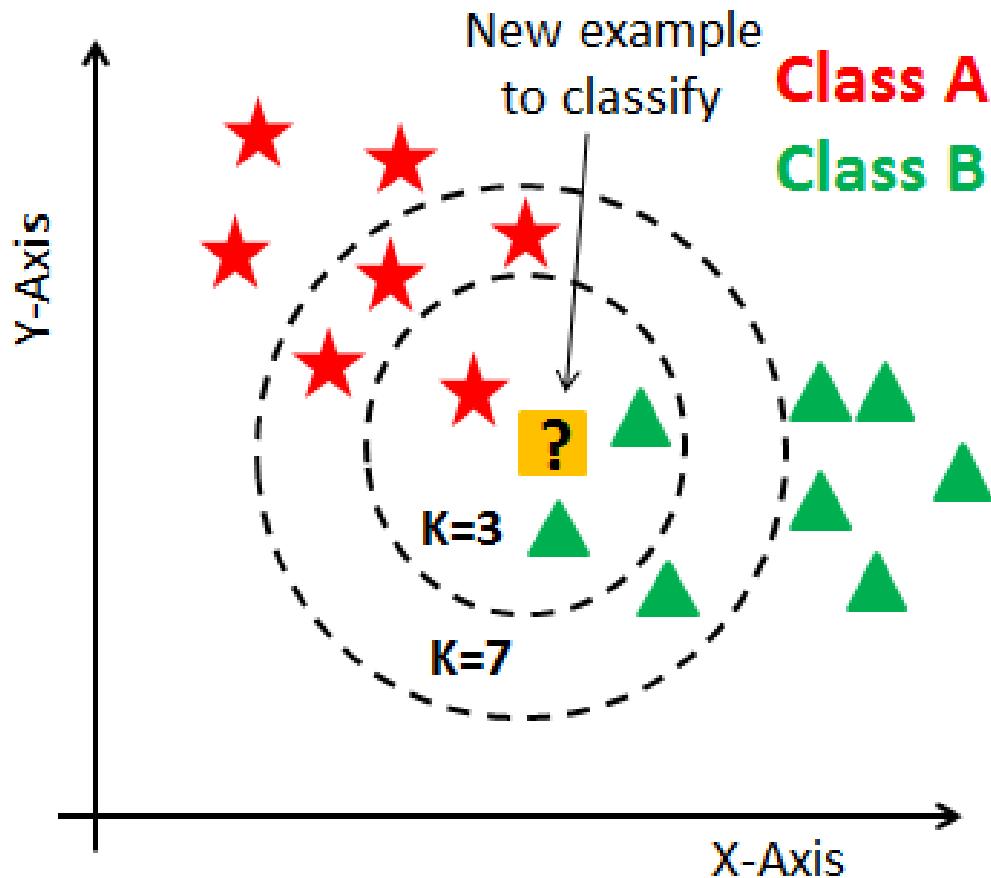
- 모델의 학습
  - 학습 필요 없음, 단순히 데이터를 저장 → 실시간 업데이트
  - 하이퍼 패러미터 K를 설정
- 모델의 예측
  - 1) 예측하고자 하는 데이터  $x$ 와 과거 데이터 사이의 거리를 구한다.
  - 2)  $x$ 와 가장 거리가 가까운  $K$ 개의 데이터를 찾는다.
  - 3) (분류)  $x$ 와 근접한  $K$ 개의 데이터의 Label들 중 가장 많은 Label로  $\hat{y}$  예측
  - 3) (회귀)  $x$ 와 근접한  $K$ 개의 데이터의 Label의 평균값

# KNN의 계산과정

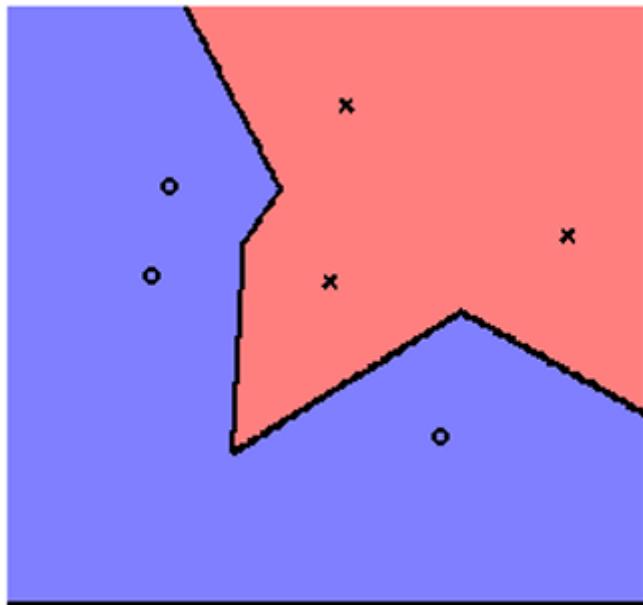


# KNN의 하이퍼 패러미터: K

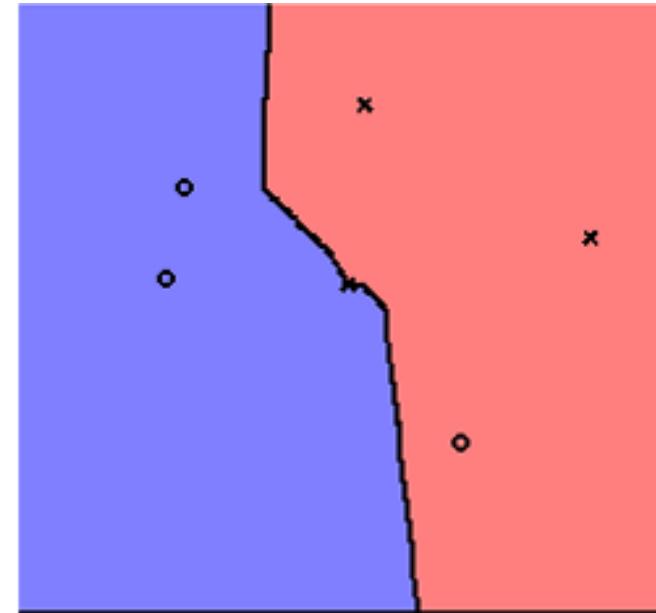
- 인접한 이웃의 개수 K에 따라 예측의 결과가 달라짐



# 분류문제에서 KNN의 경계선 (K의 영향)



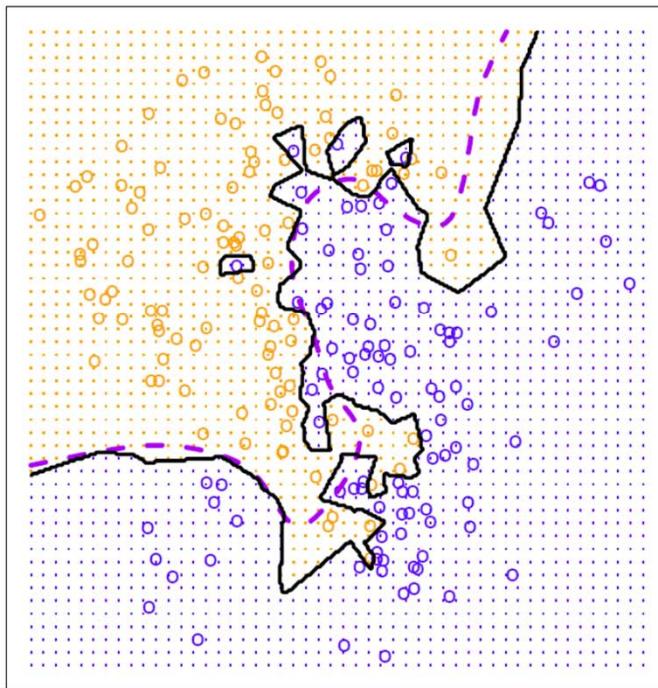
- K=1일 경우 두 카테고리사이의 경계선



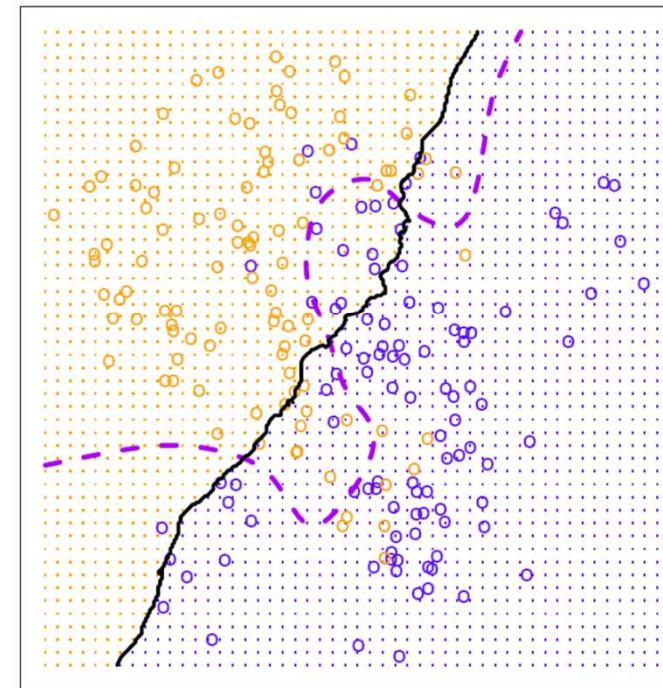
- K=3일 경우 두 카테고리사이의 경계선

# KNN - Hyper Parameter

KNN: K=1



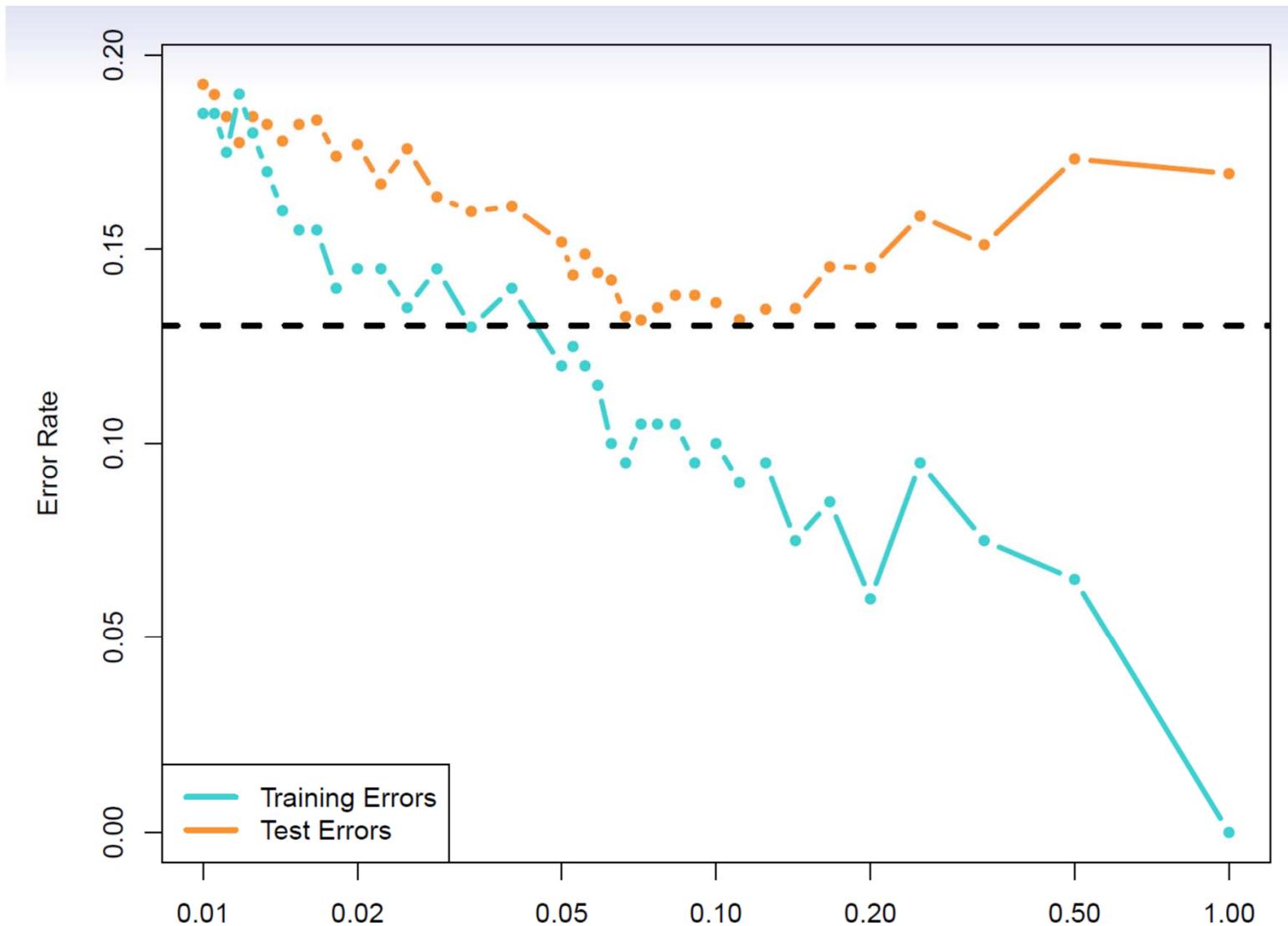
KNN: K=100



- 경계선이 근처 1개의 점에 의해 결정
- 경계선이 주변 점에 매우 **민감** (Much Flexible)

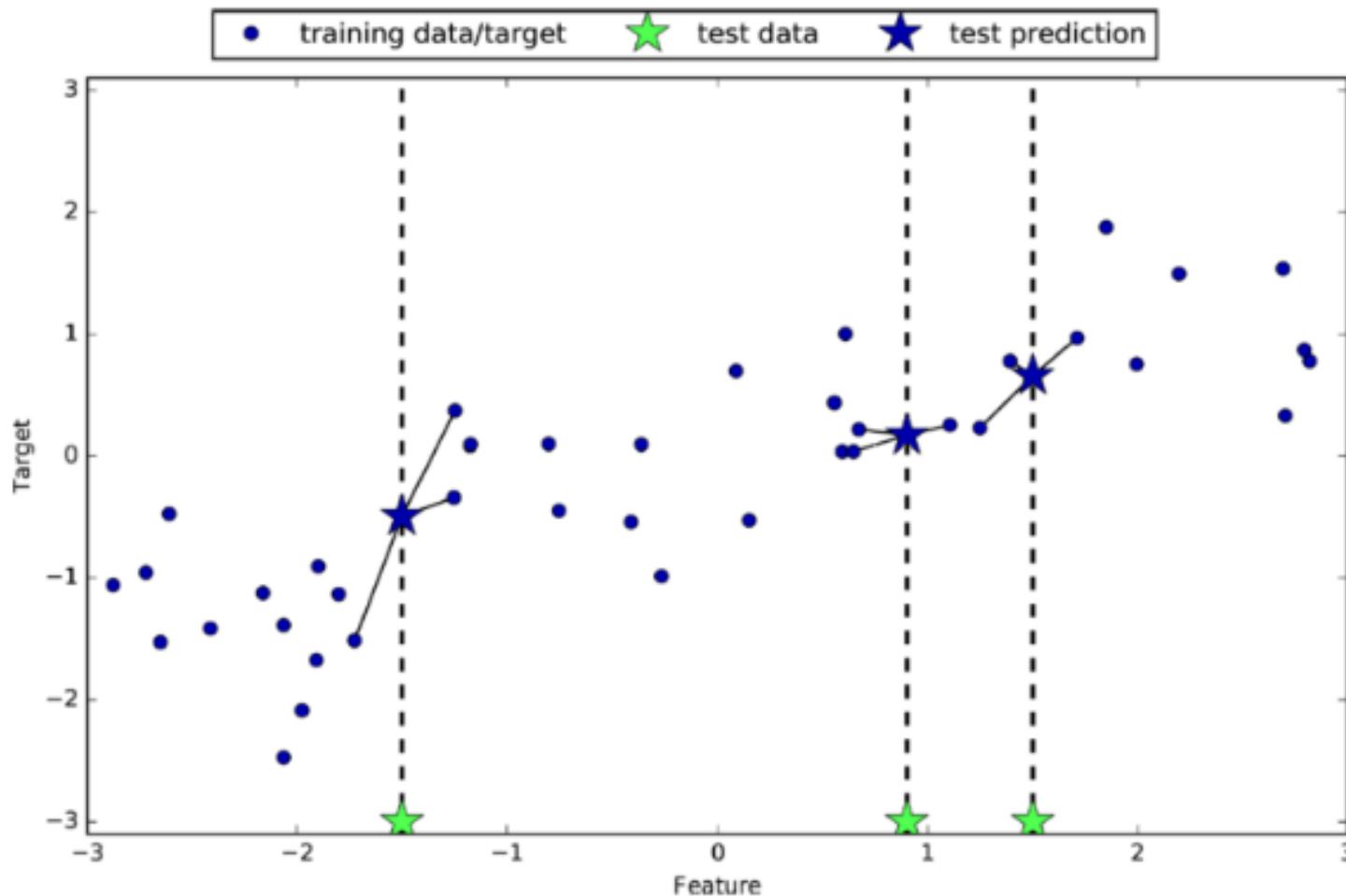
- 경계선이 근처 100개의 점에 의해 결정
- 경계선이 주변 점에 매우 **둔감** (Less Flexible)

# KNN - Hyper Parameter Tuning

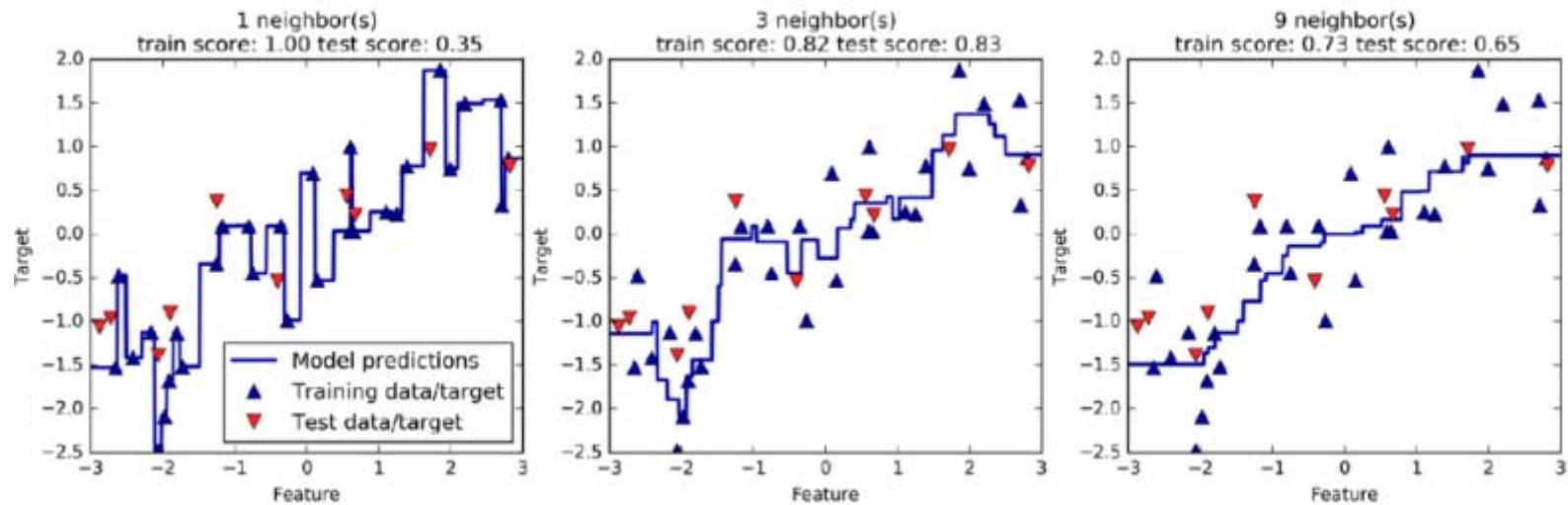


# KNN Regression

- K=3의 경우



# KNN Regression의 Hyperparameter 효과



- 회귀선이 주변 1개 선에 의해 결정
- Very Flexible (Bumpy line)
- 회귀선이 주변 9개 선에 의해 결정
- Less Flexible (Smoother line)

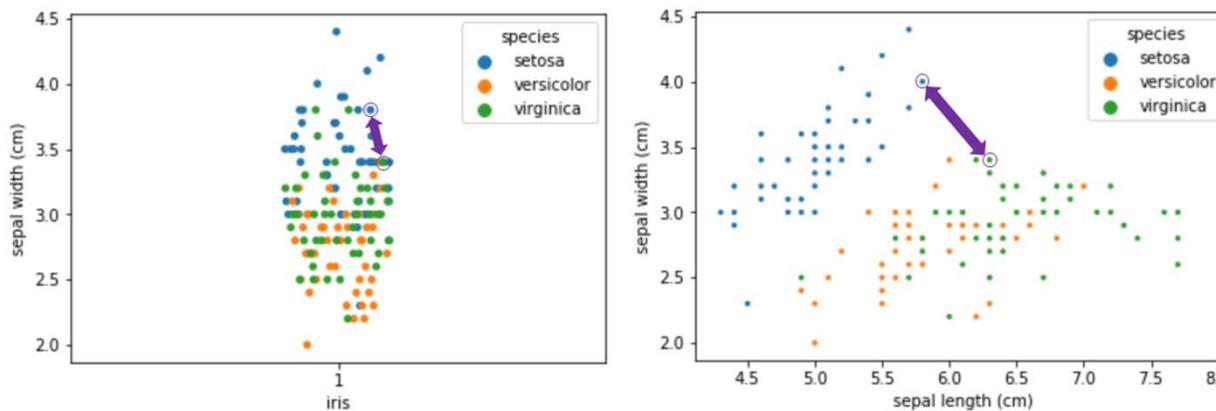
# KNN의 장점과 단점

- 장점

- 예측이 주변 데이터에 의해 결정되므로 모형의 학습이 필요 없다.  
(Non Parametric and Instance-based method)
- 데이터의 구조에 대한 사전 지식이 필요 없음
- 하이퍼 패러미터가 K 한 개로 단순하며, 생각보다 성능이 좋은 경우 존재

- 단점

- 데이터의 수가 많다면, 예측에 걸리는 시간이 길어짐  
(왜일까요?)
- 데이터 특성의 수가 커지면 성능이 저하됨 → 차원의 저주



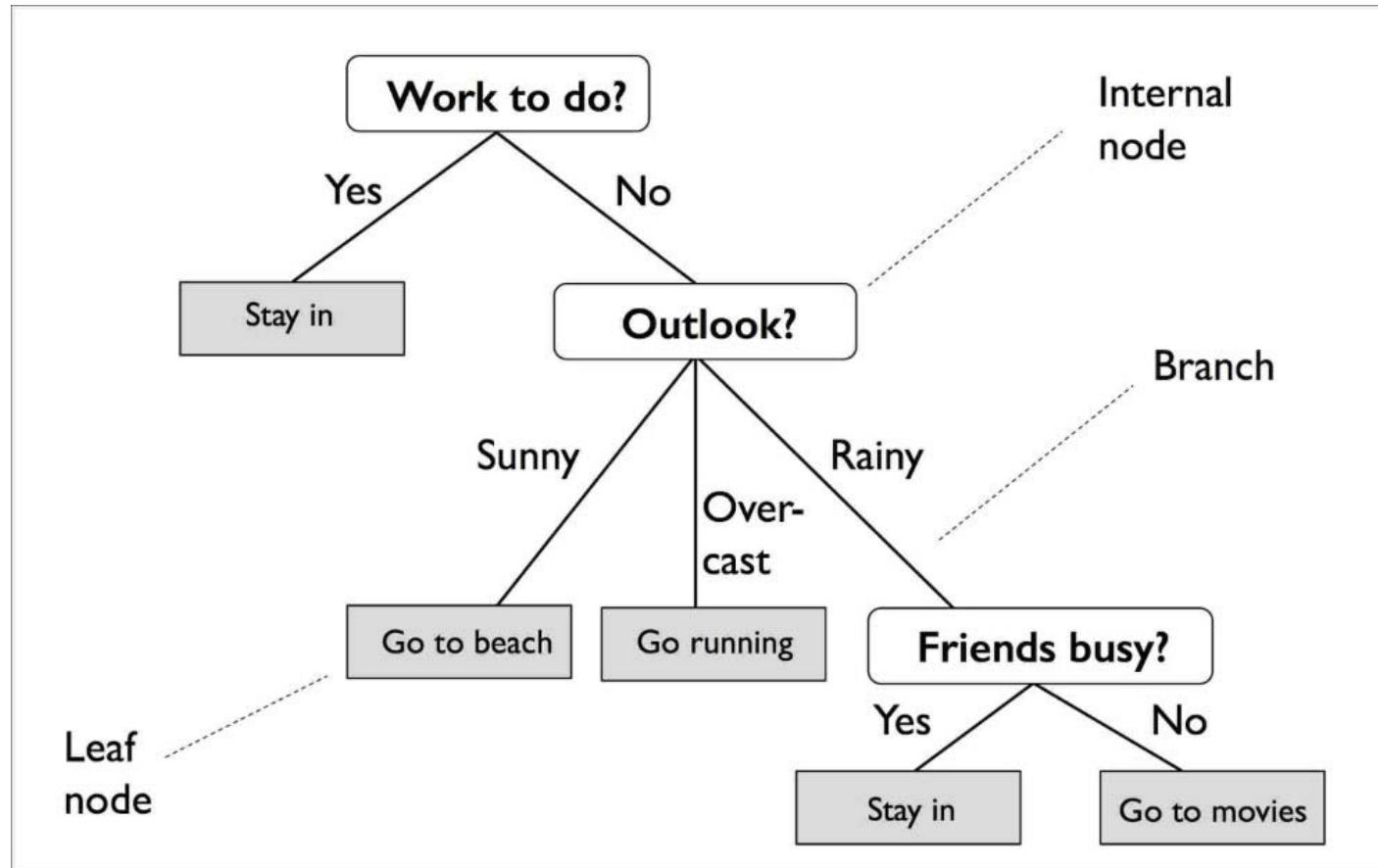
# 선형회귀모형을 넘어 (Decision Tree)

---

시스템경영공학부  
이지환 교수

# Decision Tree

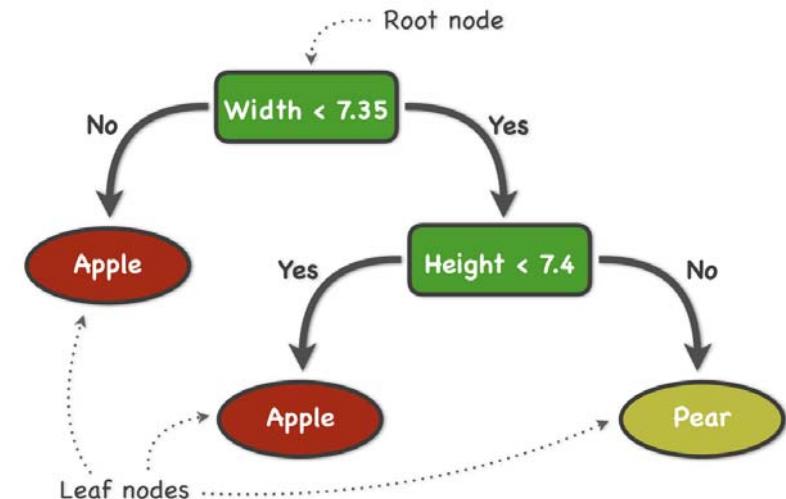
- Making a decision based on asking a series of questions!
- Strength: Easy to explain



# 데이터로부터 Decision Tree 만들기

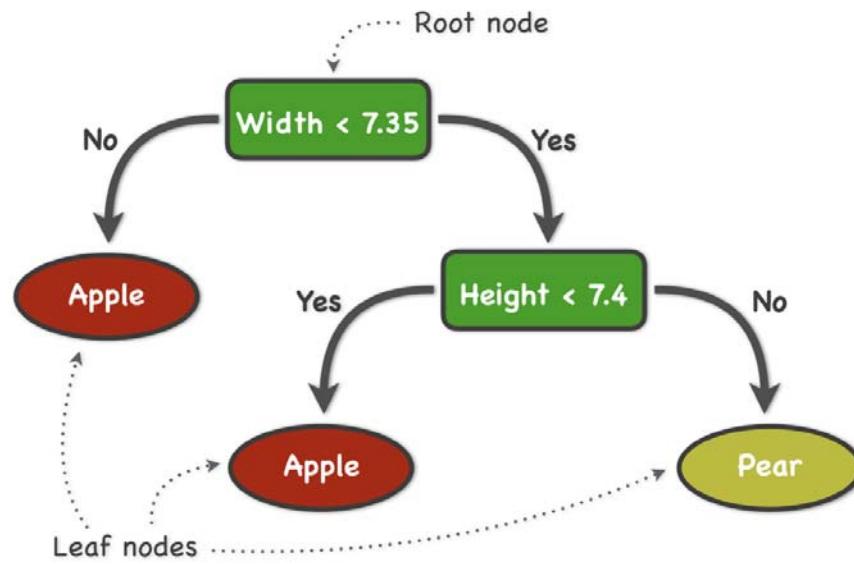
- 너비와 높이를 통해 사과인지 배인지 예측하는 모형을 Decision Tree로 표현하면 다음과 같다.

Width	Height	Fruit
7.1	7.3	Apple
7.9	7.5	Apple
7.4	7.0	Apple
8.2	7.3	Apple
7.6	6.9	Apple
7.8	8.0	Apple
7.0	7.5	Pear
7.1	7.9	Pear
6.8	8.0	Pear
6.6	7.7	Pear
7.3	8.2	Pear
7.2	7.9	Pear

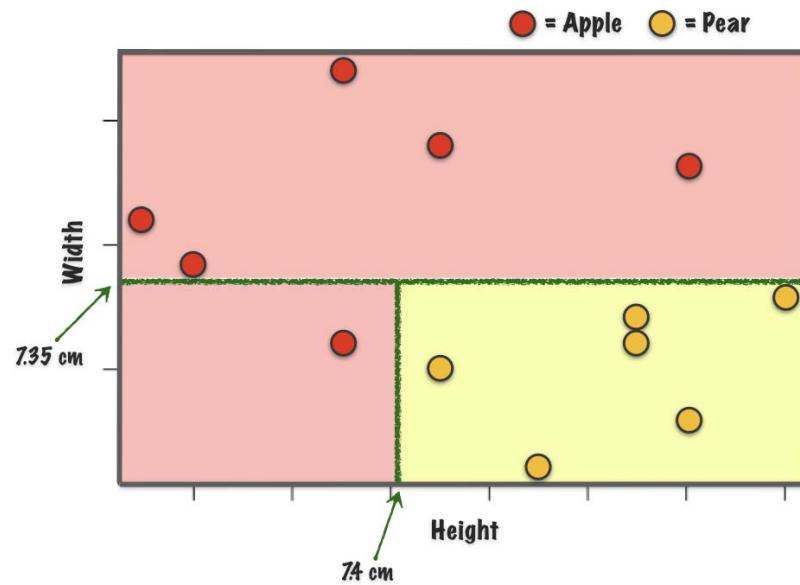


# Decision Tree의 원리

- 사과/배를 분류할 수 있는 Decision Tree



- Decision Tree에 의해 분할된 공간

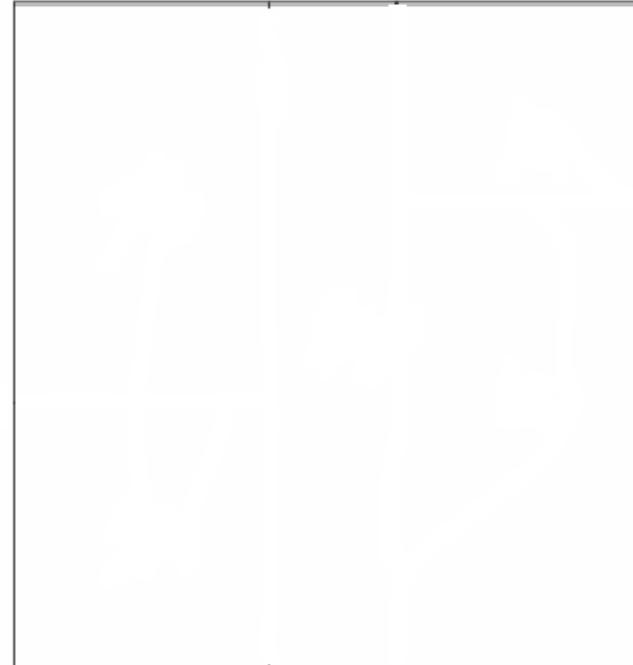


- 차원에 따라 공간을 쪼개나는 것 → Decision Tree에 질문을 추가하는 것
- 두 Label이 잘 분리되도록 공간을 쪼개면 → 좋은 Decision Tree

# Decision Tree와 공간의 분할

- First split on  $X_1 = t_1$

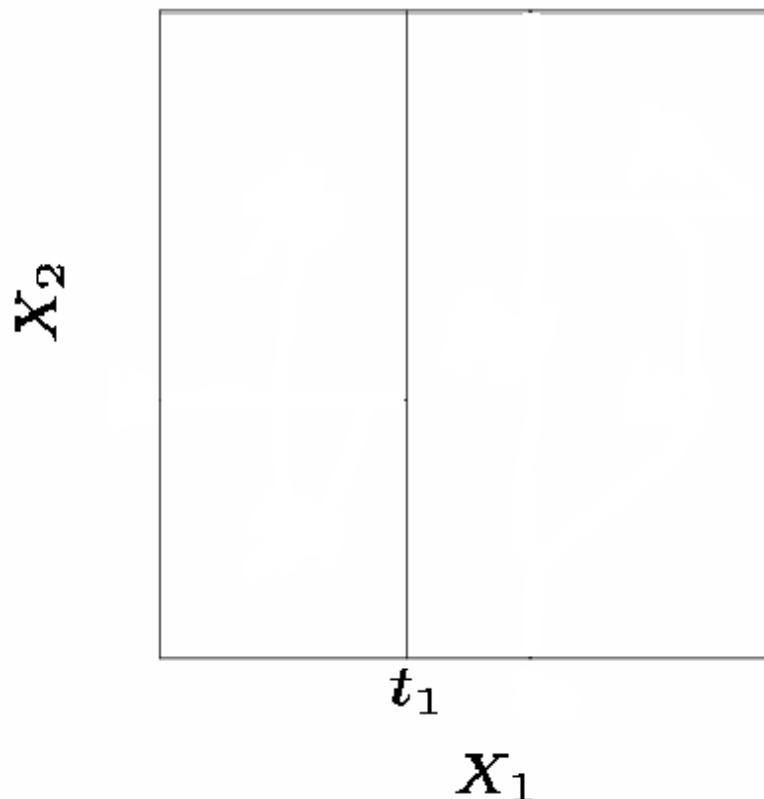
$X_2$



$X_1$

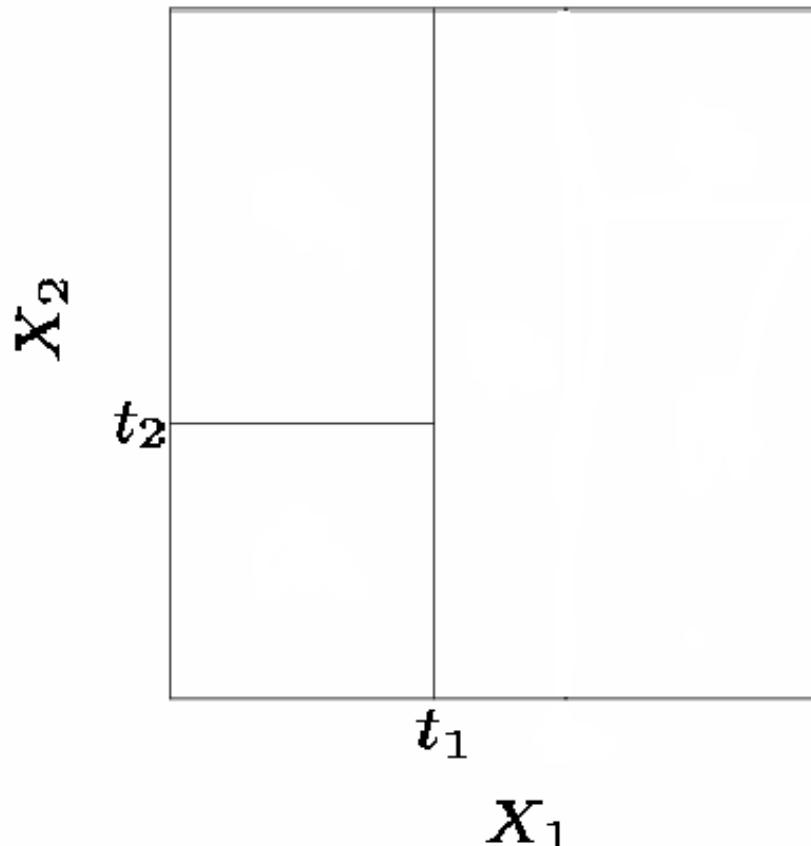
# Decision Tree와 공간의 분할

- First split on  $X_1 = t_1$



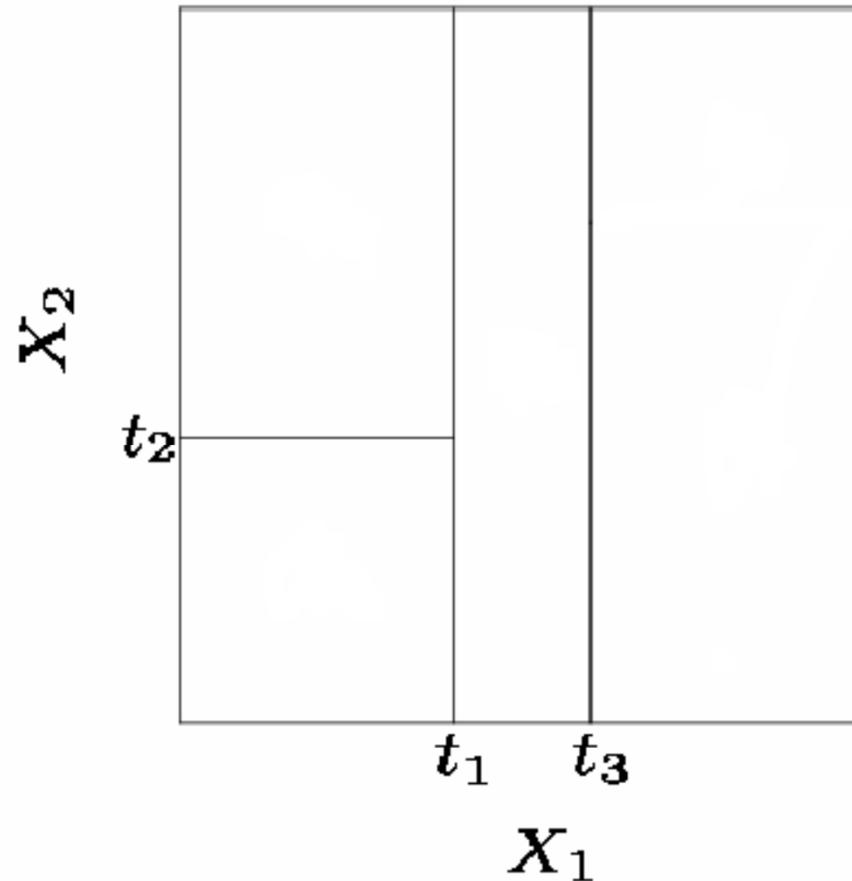
# Decision Tree와 공간의 분할

- First split on  $X_1=t_1$
- If  $X_1 < t_1$ , split on  $X_2=t_2$



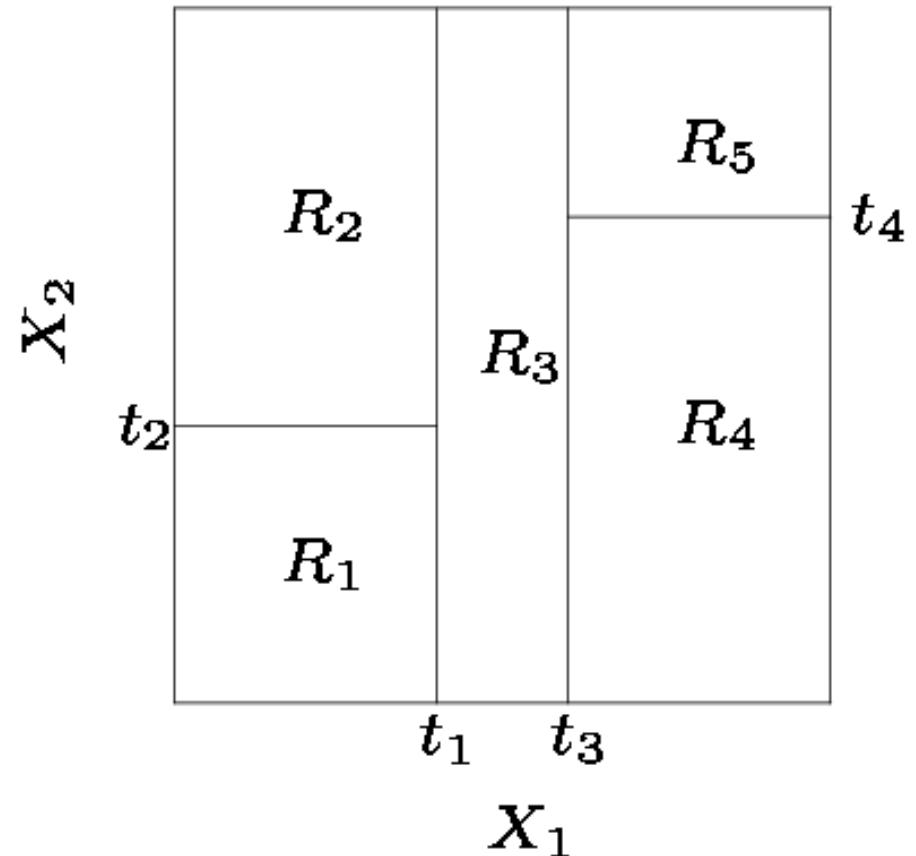
# Decision Tree와 공간의 분할

1. First split on  $X_1 = t_1$
2. If  $X_1 < t_1$ , split on  $X_2 = t_2$
3. If  $X_1 > t_1$ , split on  $X_1 = t_3$

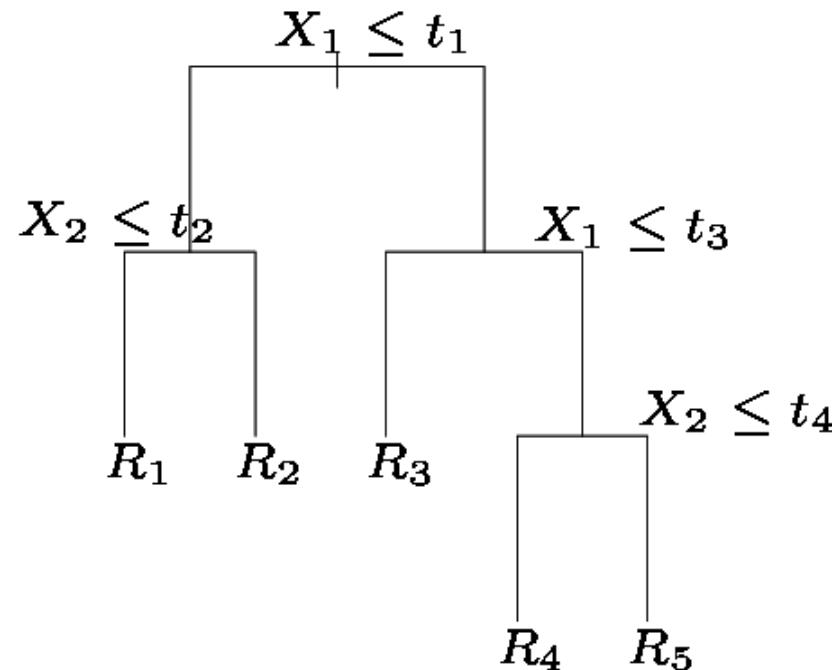


# Decision Tree와 공간의 분할

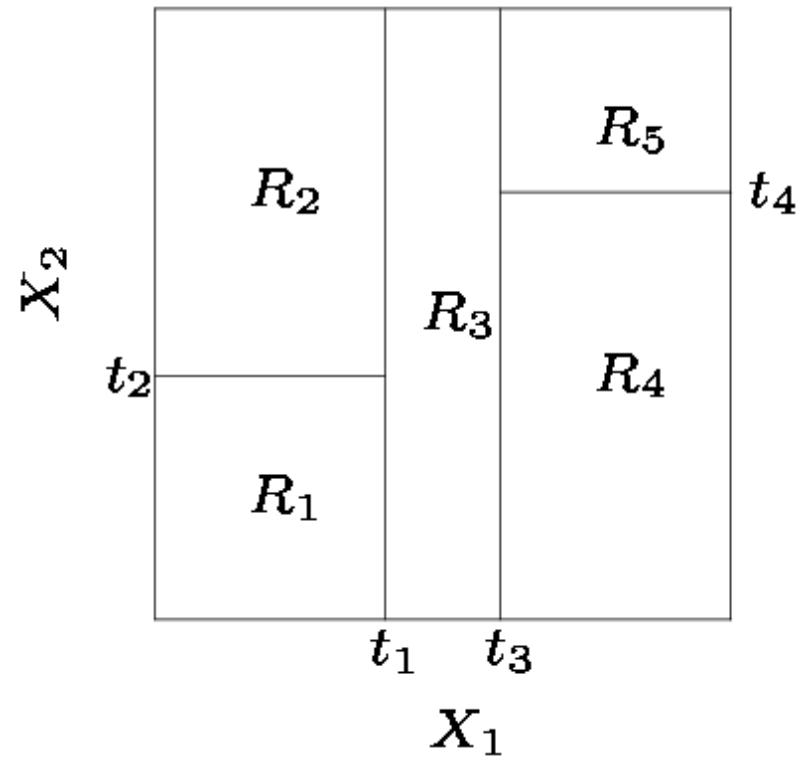
1. First split on  $X_1 = t_1$
2. If  $X_1 < t_1$ , split on  $X_2 = t_2$
3. If  $X_1 > t_1$ , split on  $X_1 = t_3$
4. If  $X_1 > t_3$ , split on  $X_2 = t_4$



# Decision Tree와 공간의 분할



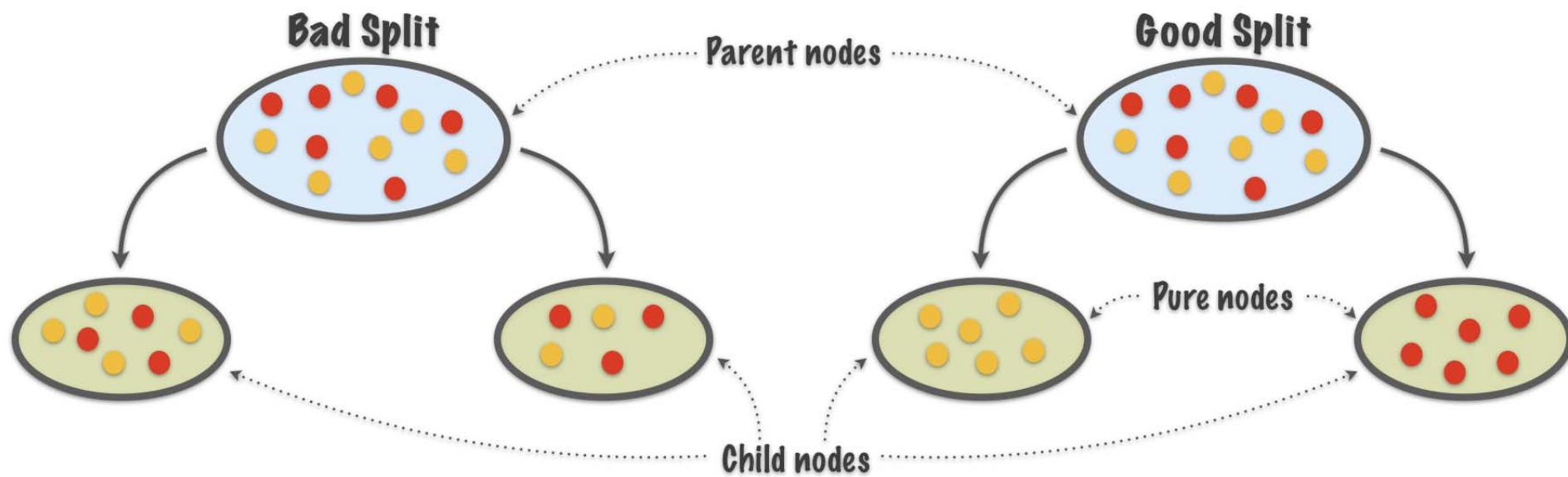
Decision Tree



분할된 공간

# 어떤 분할이 좋은 분할인가?

- Split의 결과로 두 Class가 깨끗하게 나뉘어질 수록 좋은 분할 일 것이다.

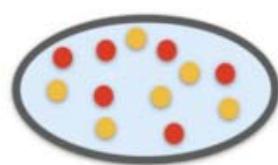


# 엔트로피

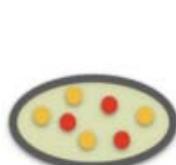
- 무질서 함의 정도를 측정하는 지표 → 1에 가까울수록 무질서
- 무질서: 두 Label이 동등히 섞였을때 무질서함이 가장 크다

$$H(X) = - \sum_{i=1}^n P(x_i) \log_2 P(x_i)$$

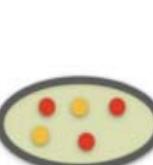
## Calculate Entropy



$$\begin{aligned} H(X) &= \sum_{i=1}^n P(x_i) \log_2 P(x_i) \\ &= -\frac{6}{12} \cdot \log_2(\frac{6}{12}) - \frac{6}{12} \cdot \log_2(\frac{6}{12}) \\ &= \frac{1}{2} + \frac{1}{2} = 1 \end{aligned}$$



$$\begin{aligned} H(X) &= -\frac{4}{7} \cdot \log_2(\frac{4}{7}) - \frac{3}{7} \cdot \log_2(\frac{3}{7}) \\ &= 0.4613 + 0.5239 = 0.9852 \end{aligned}$$



$$\begin{aligned} H(X) &= -\frac{3}{5} \cdot \log_2(\frac{3}{5}) - \frac{2}{5} \cdot \log_2(\frac{2}{5}) \\ &= 0.4422 + 0.5288 = 0.9710 \end{aligned}$$



$$H(X) = -\frac{6}{6} \cdot \log_2(\frac{6}{6}) = 0$$

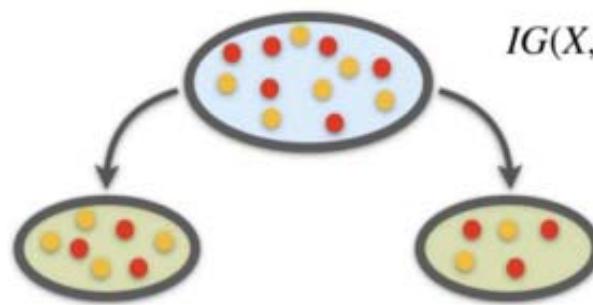


$$H(X) = -\frac{6}{6} \cdot \log_2(\frac{6}{6}) = 0$$

# Information Gain

- 공간을 쪼갤으로 인해서 얻어지게는 무질서함의 감소정도

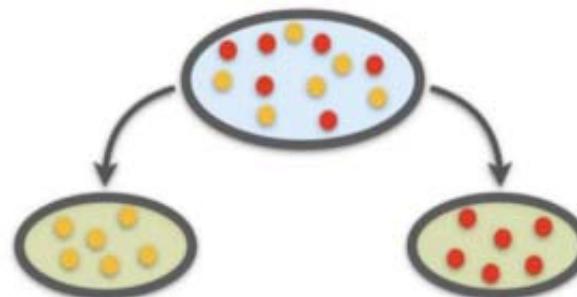
## Calculate Information Gain



$$\begin{aligned}IG(X, F) &= (\text{original entropy}) - (\text{entropy after split}) \\&= H(X) - \sum_{i=1}^n \frac{|x_i|}{X} H(x_i) \\&= 1 - \frac{7}{12} \cdot (0.9852) - \frac{5}{12} \cdot (0.9710) = 0.0213\end{aligned}$$

---

$$IG(X, F) = 1 - \frac{6}{12} \cdot (0) - \frac{6}{12} \cdot (0) = 1$$

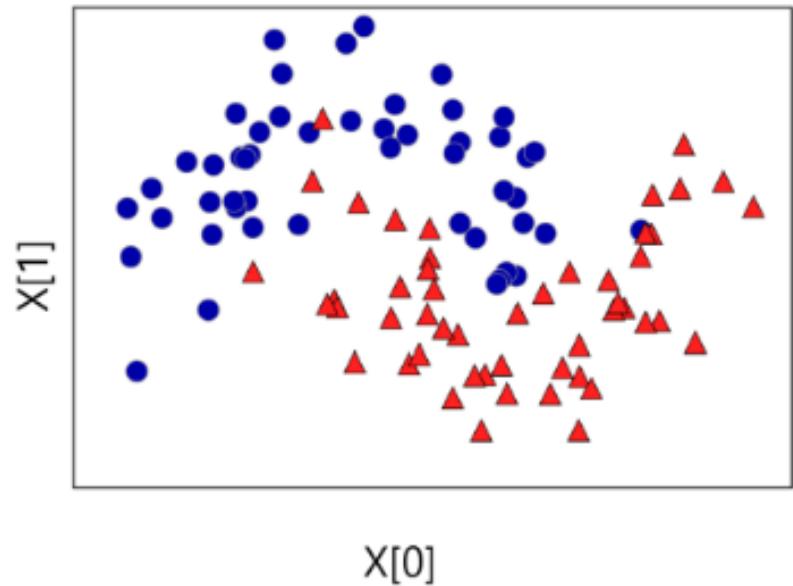


# Decision Tree Algorithm

## ID3 algorithm

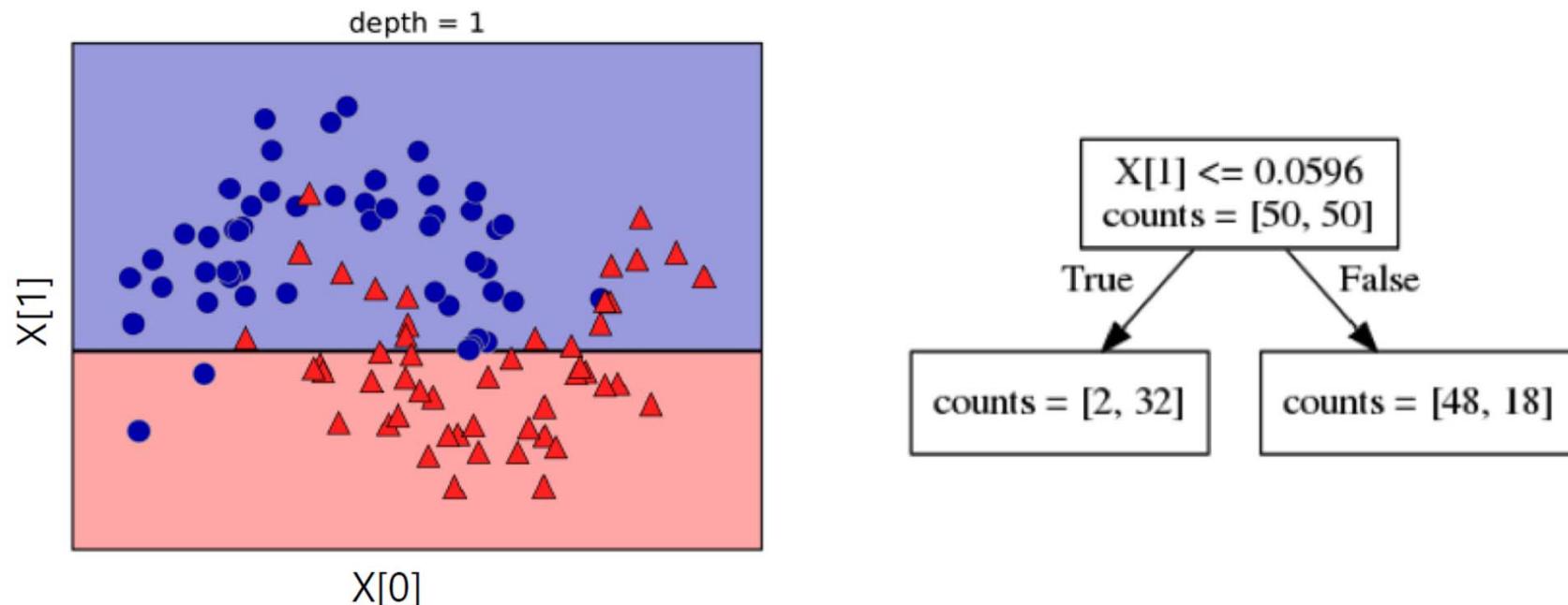
- Split (node, {examples} ):
    1.  $A \leftarrow$  the best attribute for splitting the {examples}
    2. Decision attribute for this node  $\leftarrow A$
    3. For each value of A, create new child node
    4. Split training {examples} to child nodes
    5. For each child node / subset:  
if subset is pure: STOP  
else: Split (child\_node, {subset} )
- 재귀 알고리즘!!

# Decision Tree Algorithm



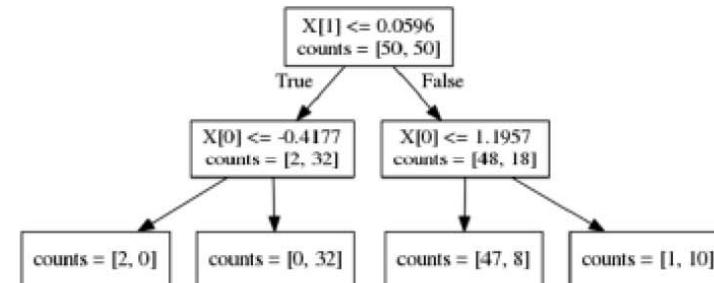
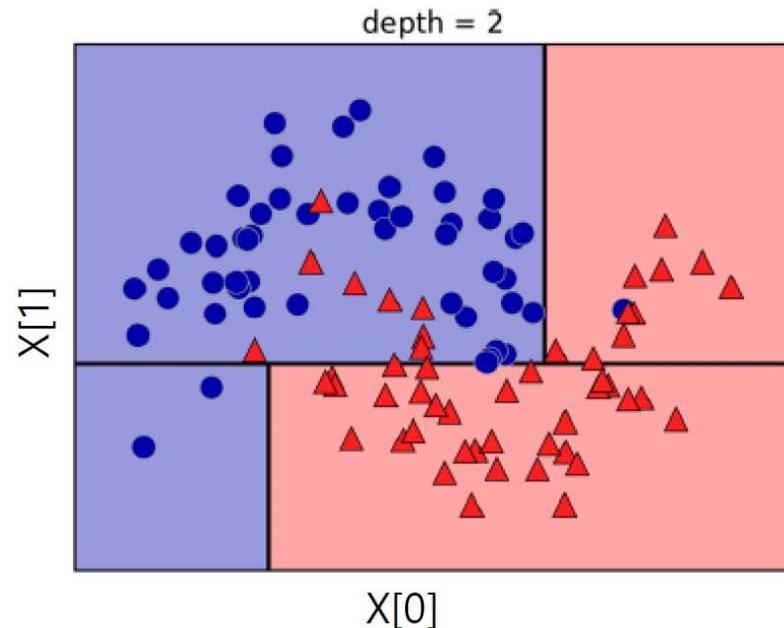
# Decision Tree Algorithm

- Information Gain을 높이는 가장 좋은 Feature 고르기 →  
 $X_1 \leq 0.0596$
- 이 값에 따라 두개의 Child Node 생성



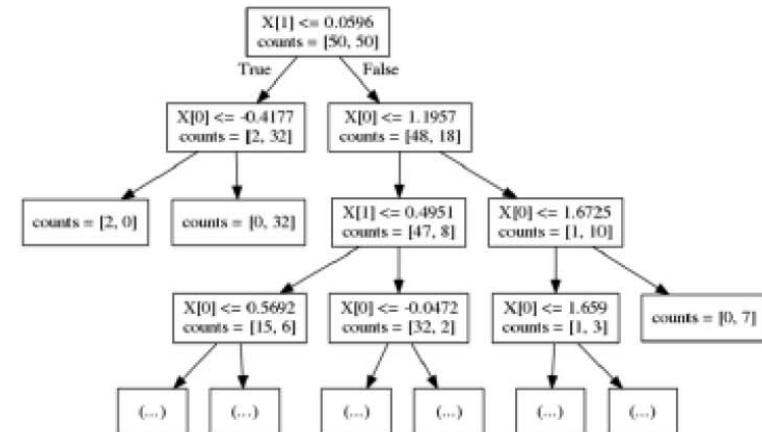
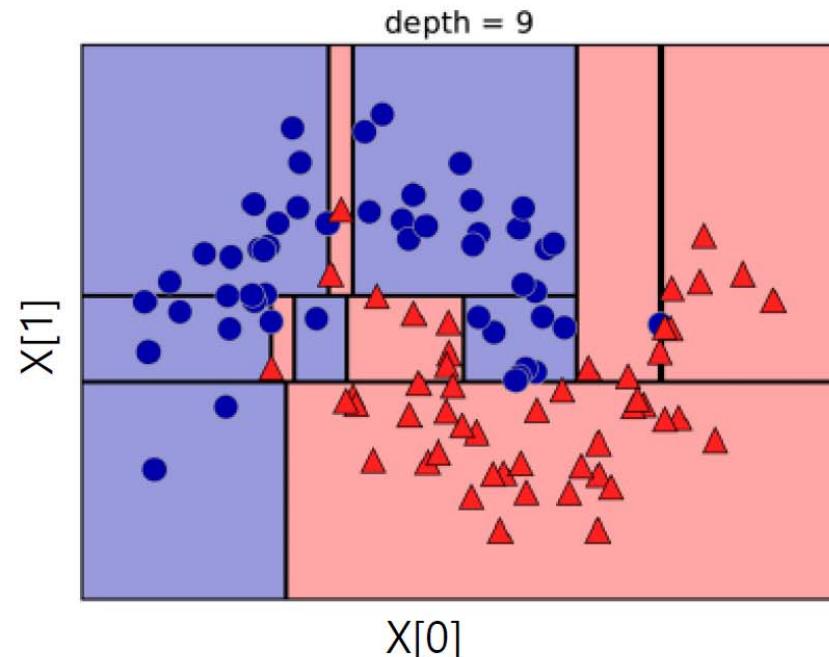
# Decision Tree Algorithm

- 각각의 Child node에 대하여 똑같은 과정을 수행 (recursive partitioning)

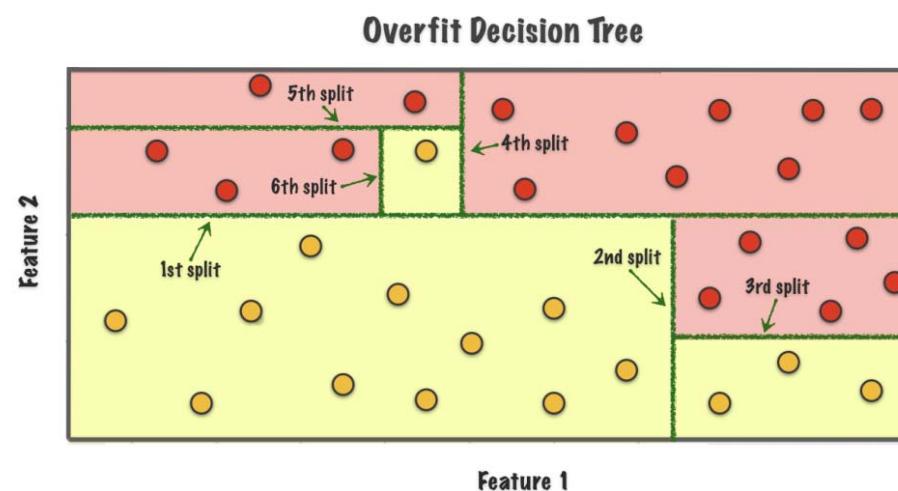
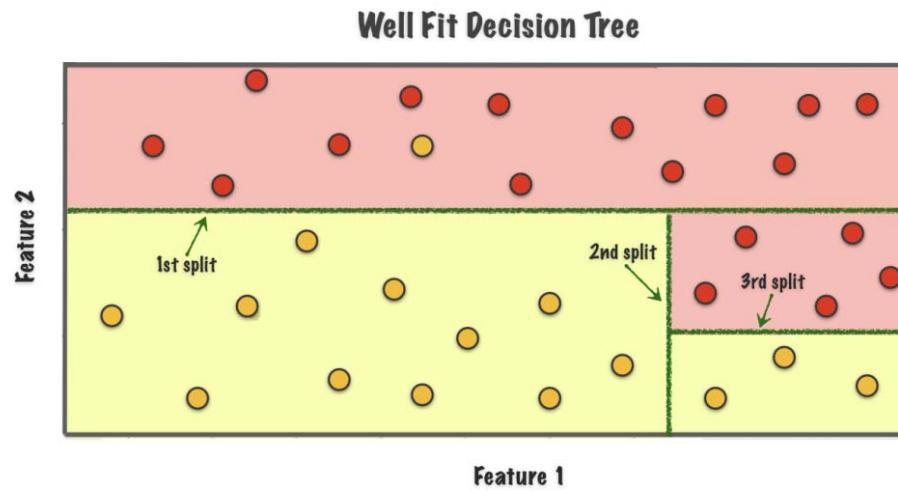


# Decision Tree Algorithm

- 모든 Child Node가 Pure 할 때까지 같은 과정을 반복



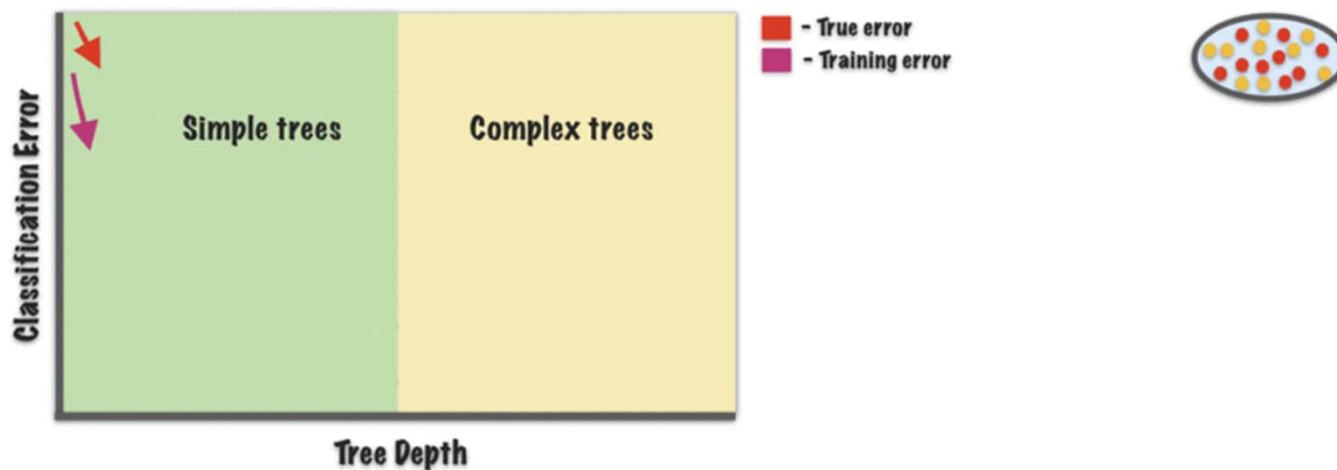
# Decision Tree의 오버피팅



- Tree의 깊이가 깊어질수록 개별적인 데이터를 잘 분류할 수 있게 됨 (Flexible)
- 하지만 지나치게 Tree의 깊이가 깊어지면 Overfitting이 발생할 수 있음

# Decision Tree의 오버피팅

- Tree의 깊이가 깊어질수록 개별적인 데이터를 잘 분류할 수 있게 됨 (Flexible)
- 하지만 지나치게 Tree의 깊이가 깊어지면 Overfitting이 발생할 수 있음

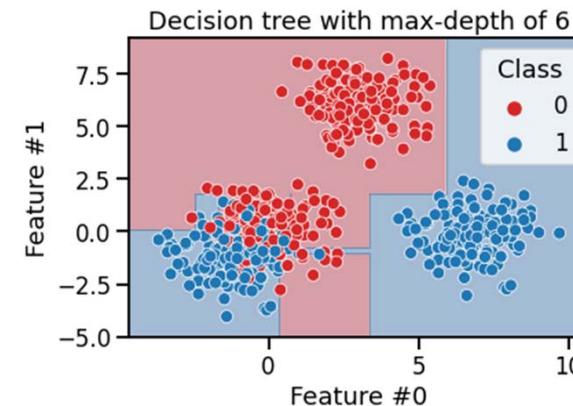
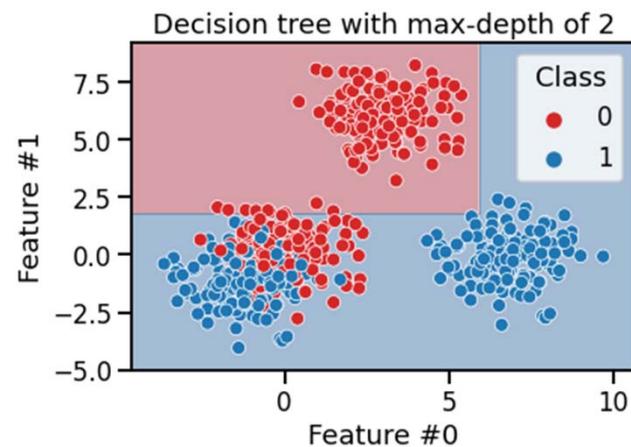


# Decision Tree의 하이퍼 패러미터

- 트리의 최대 깊이 (Max Depth)

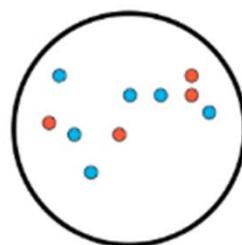


Maximum depth of a decision tree

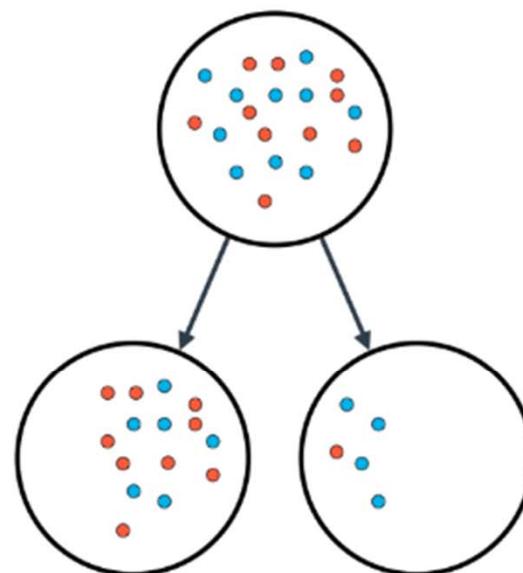


# Decision Tree의 하이퍼 패러미터

- Minimum number of samples to split
- Split을 멈추는 노드안의 최소 데이터 개수
  - 작을수록: 유연한 모델이 될 것이다.



No split!

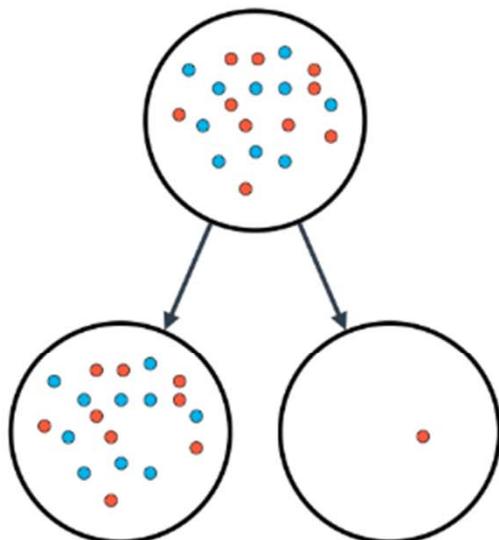


Minimum number of samples to split = 11

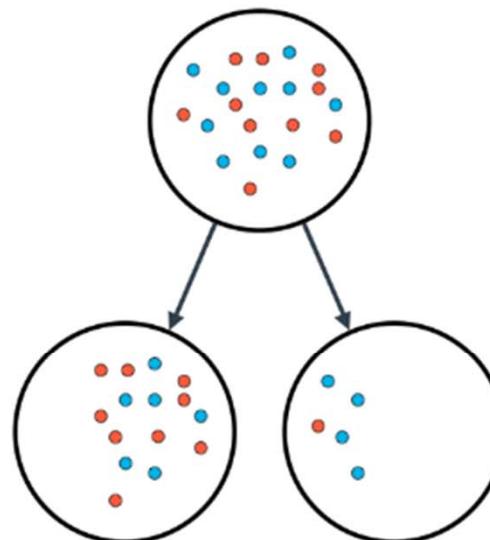
Minimum number of samples to split

# Decision Tree의 하이퍼 패러미터

- Minimum number of samples per leaf
  - 작을수록: 유연한 모델이 될것이다.
  - 또한 Decision Tree의 비대칭성이 강화된다. (왜?)



Minimum samples per leaf = 1



Minimum samples per leaf = 5

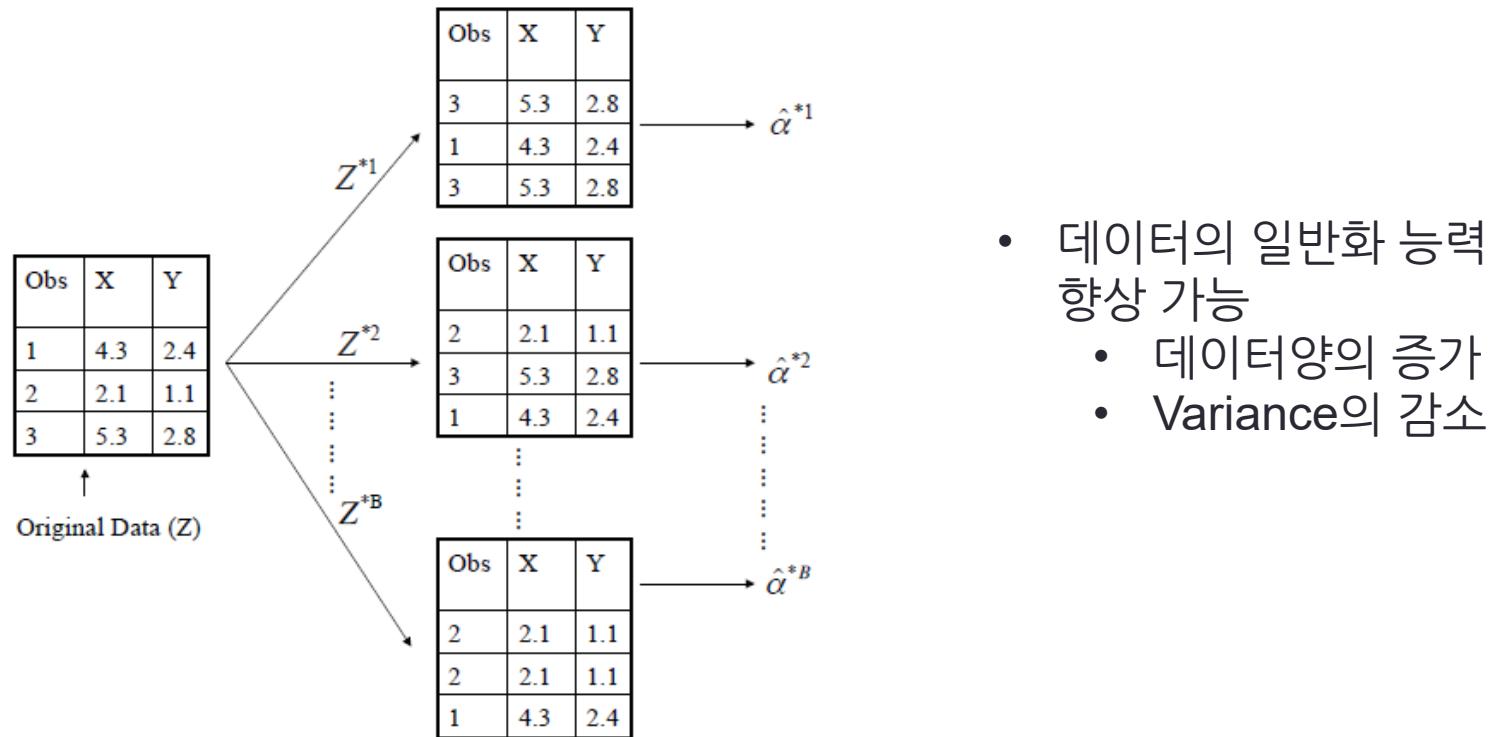
Minimum number of samples per leaf

# Decision Tree의 장단점

- 장점
  - 이해하기 쉬운 규칙 : If ~ Then 형식
  - 분류예측에 유용
  - 연속형, 범주형 모두 취급 가능
  - 변수의 중요성 비교 가능
  - 비교적 빠른 속도
- 단점
  - 연속형 변수값을 예측할 때 적당하지 않음
  - 회귀모형에서 예측력이 떨어짐
  - 트리 모형이 복잡하면 예측력 저하, 해석 어려움
  - 데이터 변형에 민감하여 안정적이지 않음 (오버피팅이 발생하기 쉬움)

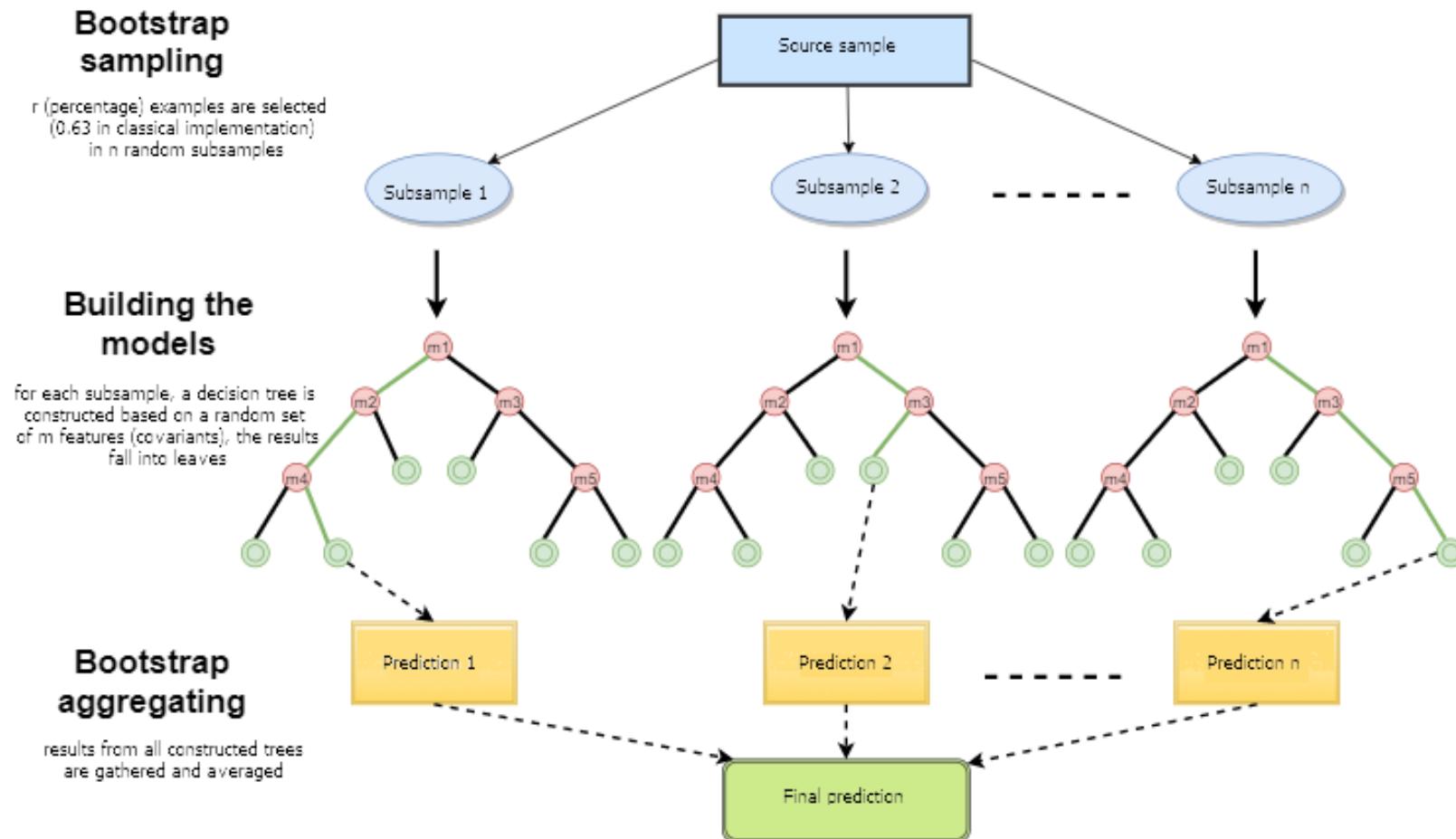
# 부트스트래핑 (Bootstrapping)

- 특정 통계량을 얻기 위해 다수의 중복된 데이터셋을 다수 만들고, 이들의 결과를 평균내는 기법
  - 각 데이터셋은 전체 데이터셋의 일부가 무작위로 추출됨
  - 일부 중복된 데이터를 가지고 있음



# 랜덤 포레스트(Random Forest)

- 부트스트래핑을 통해 여러 개의 데이터셋을 만들고
- 각자 Decision Tree를 학습 --> 학습된 결과를 취합하여 예측



# 랜덤 포레스트 (random forest) 장단점

- 장점
  - 일반화 능력이 좋으면서도 예측력이 좋음
  - 중간 사이즈의 데이터에서 매우 많이 사용되는 실용적인 기법
- 단점
  - 여러 개의 Decision Tree가 중첩되어 있으므로 설명하기가 어려움
  - 단일 트리에 비해 계산이 많이 소요됨 (컴퓨팅 파워로 극복 가능)

# 다루지 않은 방법론

- SVM (Support Vector Machine)
- ANN (Artificial Neural Network)
  - Deep Neural Network
  - 곧 다룰 예정
- Random Forest의 개량버전
  - XGBoost (높은 정확도로 유명)
  - LGBM (병렬 계산 가능)
  - ..
- ...

# Model Trade-off

- 무조건 예측력이 좋다고 해서 좋은 모델은 아님.
- 설명력-정확도의 Trade-off를 잘 따져서 문제에 맞는 문제를 풀어야 함

