데이터마이닝팀

4팀
성준혁
노정아
이상혁
진재언
한준호

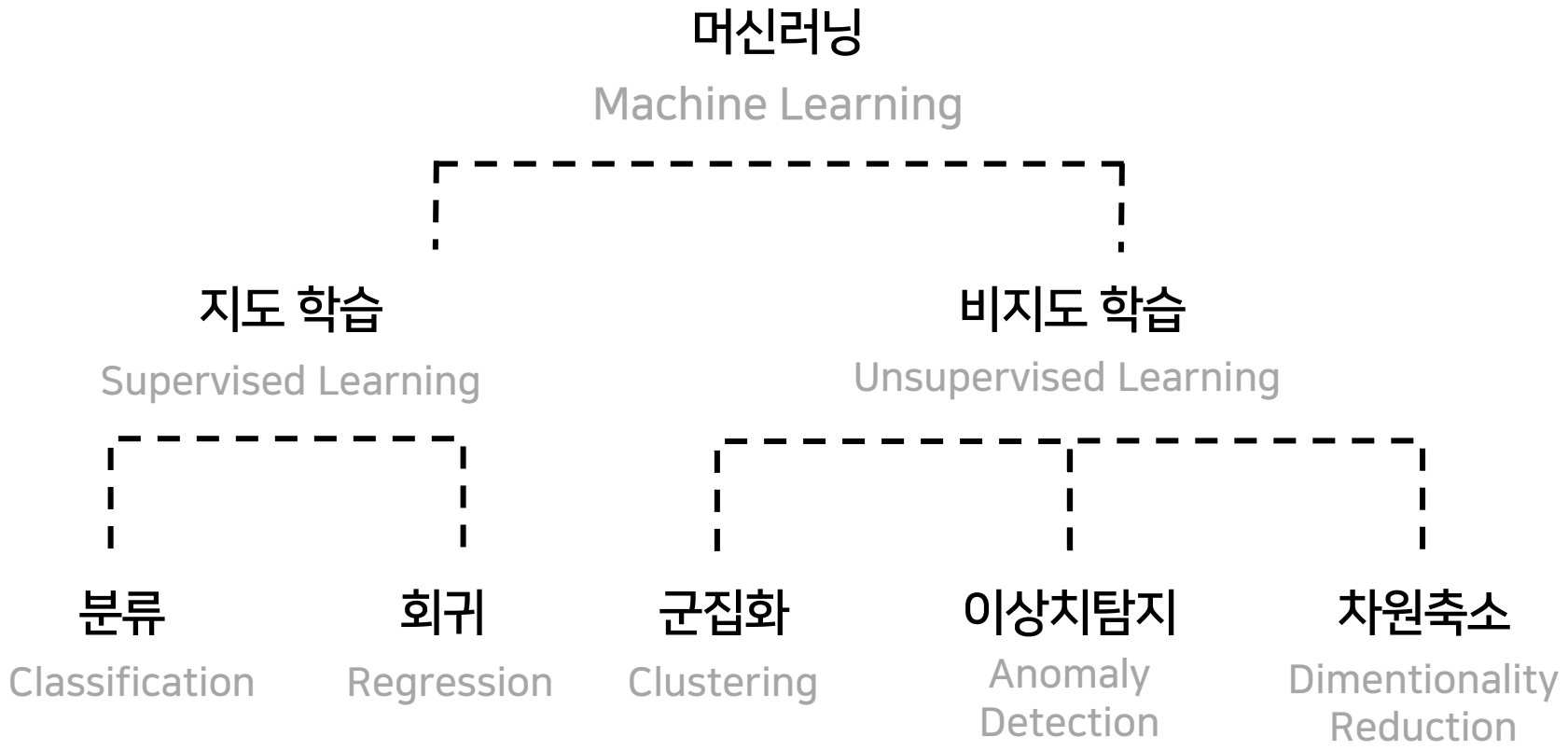
INDEX

1

트리 기반 모델

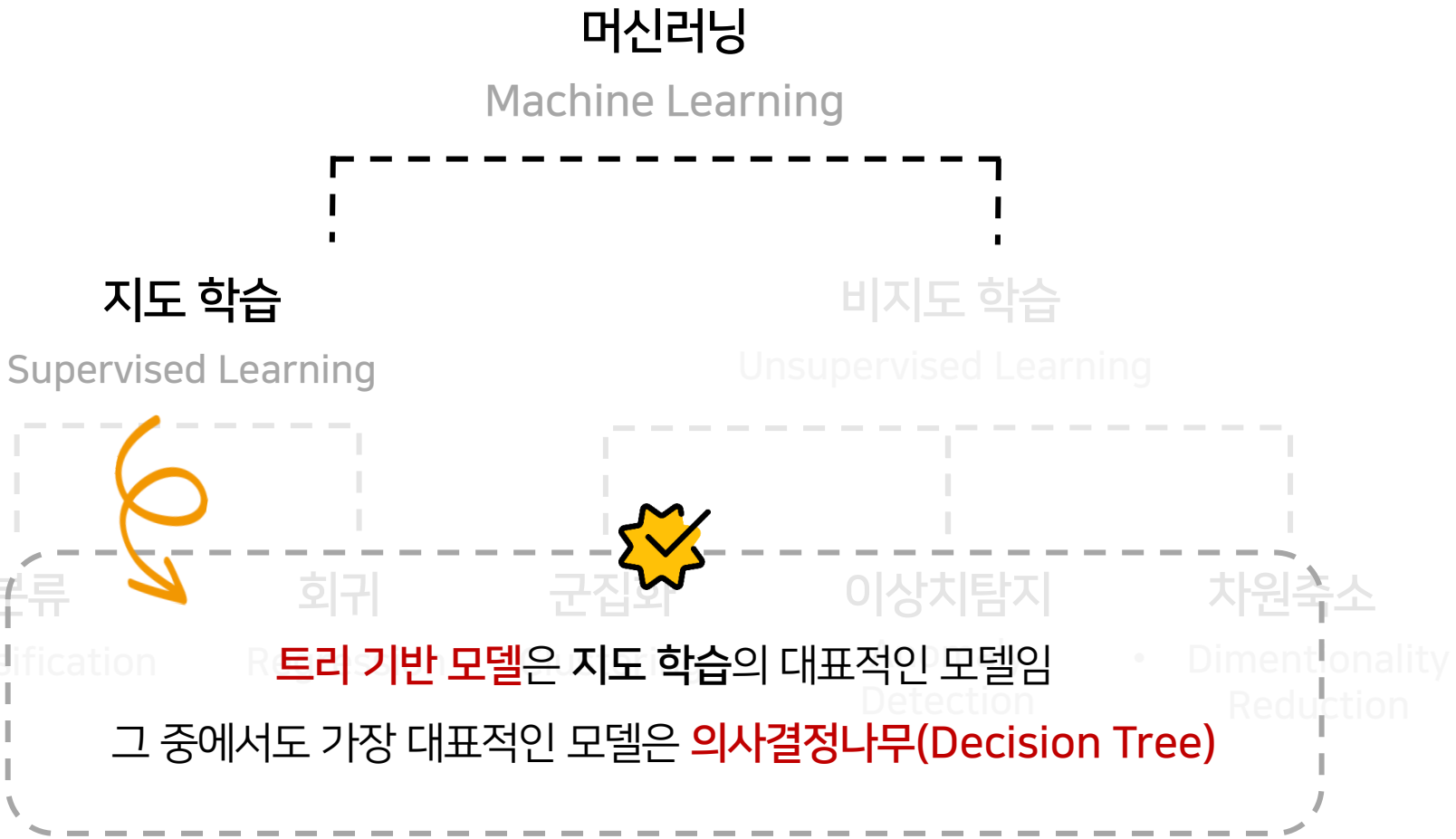
1 트리 기반 모델

머신러닝 Remind



1 트리 기반 모델

머신러닝 Remind



의사결정나무(Decision Tree)

의사결정나무(Decision Tree)

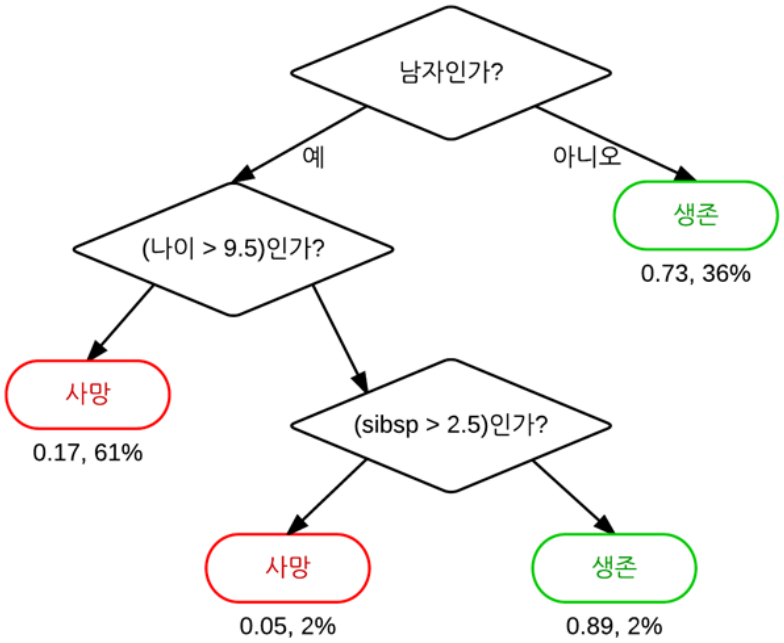
간단한 규칙들을 논리적으로 결합한 **순차적 모델**(sequential model)

흔히 아는 스무고개와 비슷한 매커니즘으로 작동함

1 트리 기반 모델

의사결정나무(Decision Tree)

Ex) 타이타닉 프로젝트



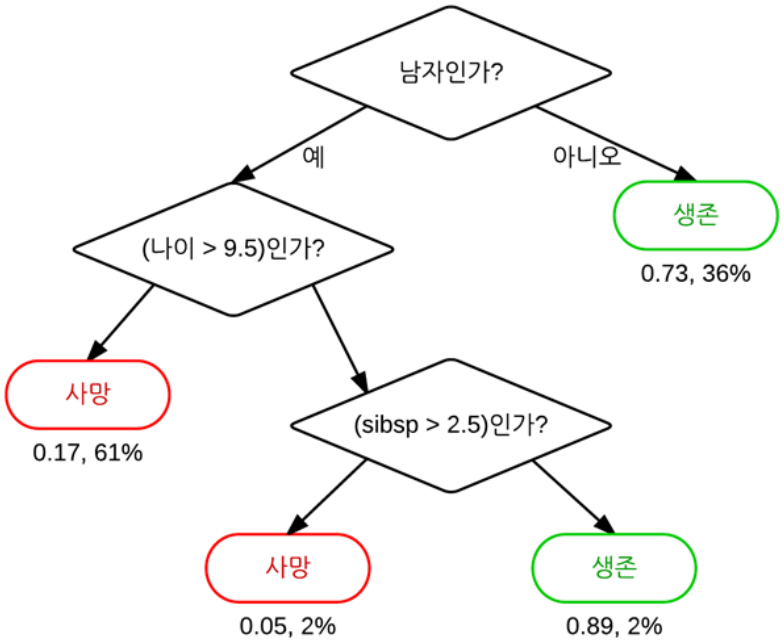
타이타닉 탑승자들의 정보(나이, 성별, 동행자 수 등)를 바탕으로 **생존 여부를 예측**하는 프로젝트

스무고개처럼 각 질문에 대해서 **'예'**와 **'아니오'** 로 **구분**

1 트리 기반 모델

의사결정나무(Decision Tree)

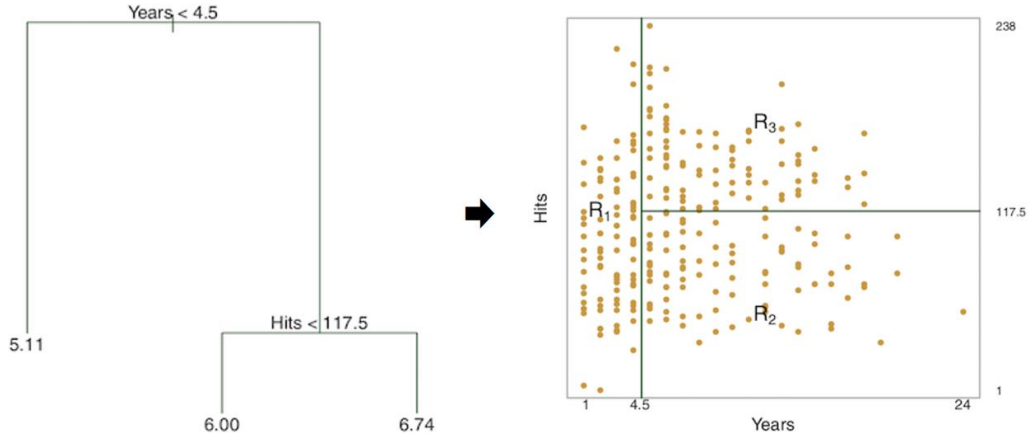
Ex) 타이타닉 프로젝트



'남자인가?'라는 질문에 대해
'예'와 '아니오' 에 따라서
영역이 분할됨

'나이 > 9.5?'라는 질문에 대해
'예'와 '아니오' 에 따라 다시
영역이 분할됨

의사결정나무(Decision Tree)

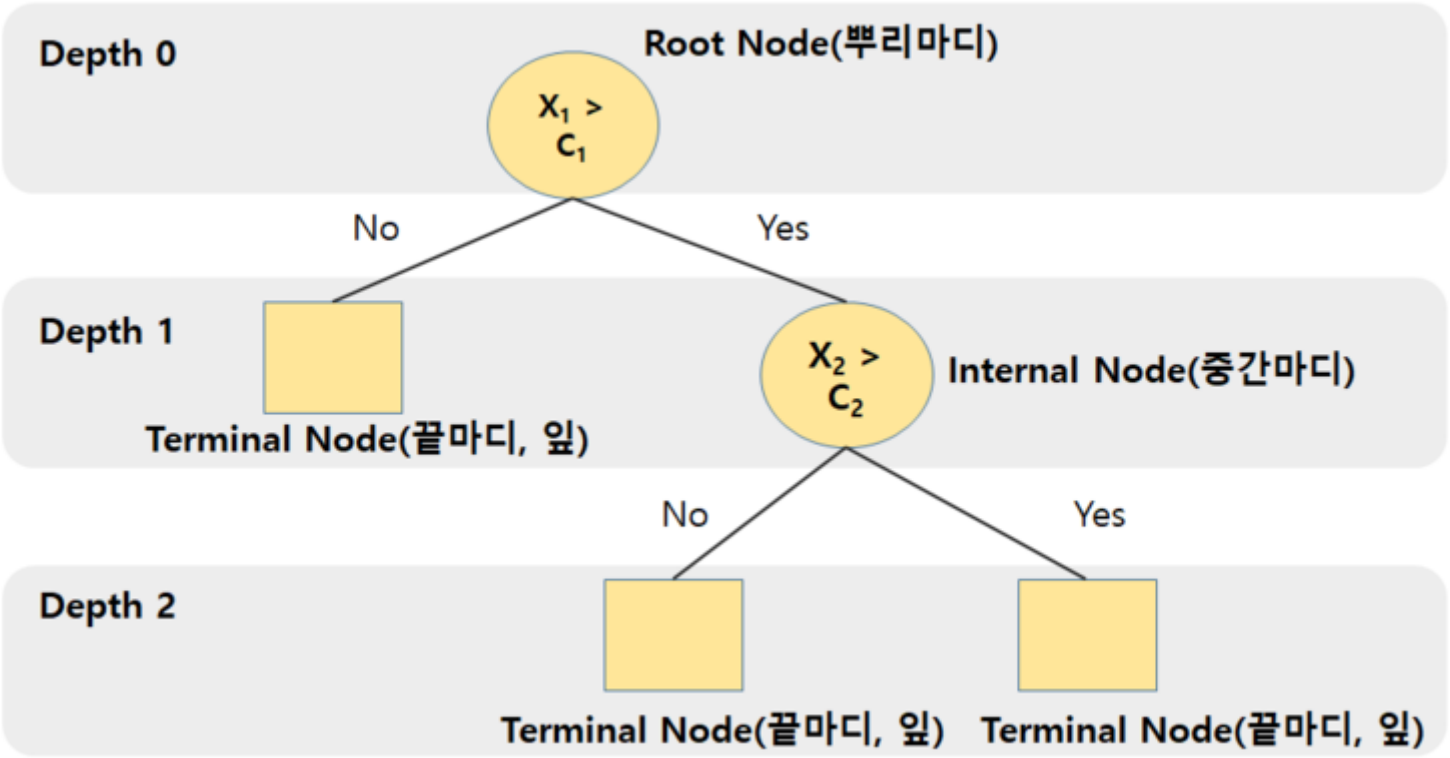


이처럼 의사결정나무는 질문과 그에 따른 대답을 바탕으로

순차적으로 독립변수의 공간을 분할함

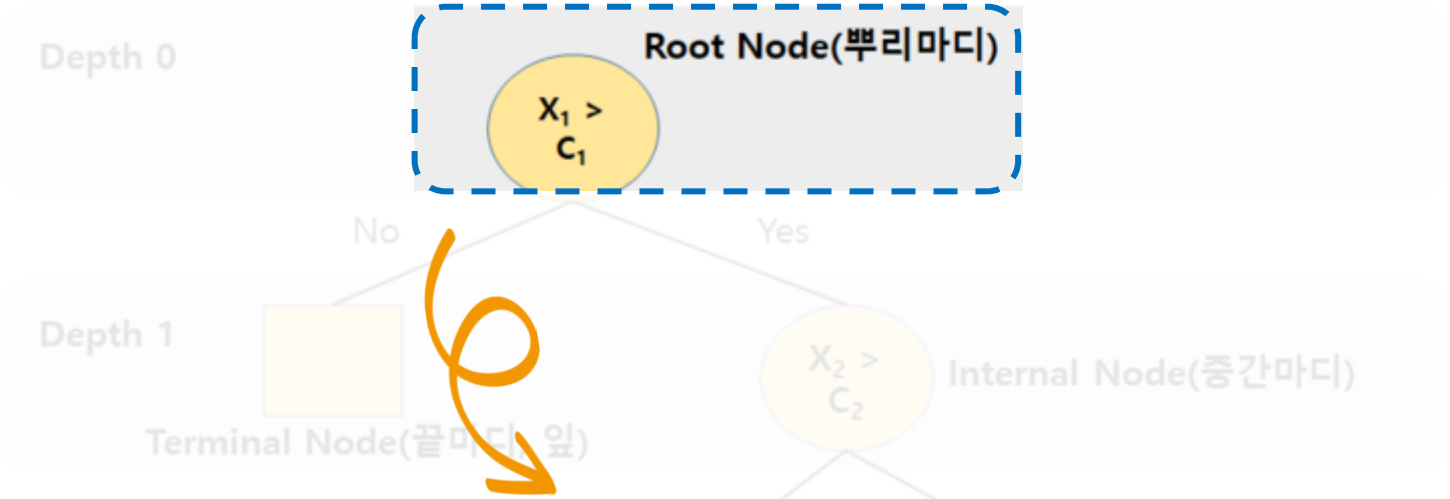
1 트리 기반 모델

용어 정리



1 트리 기반 모델

용어 정리

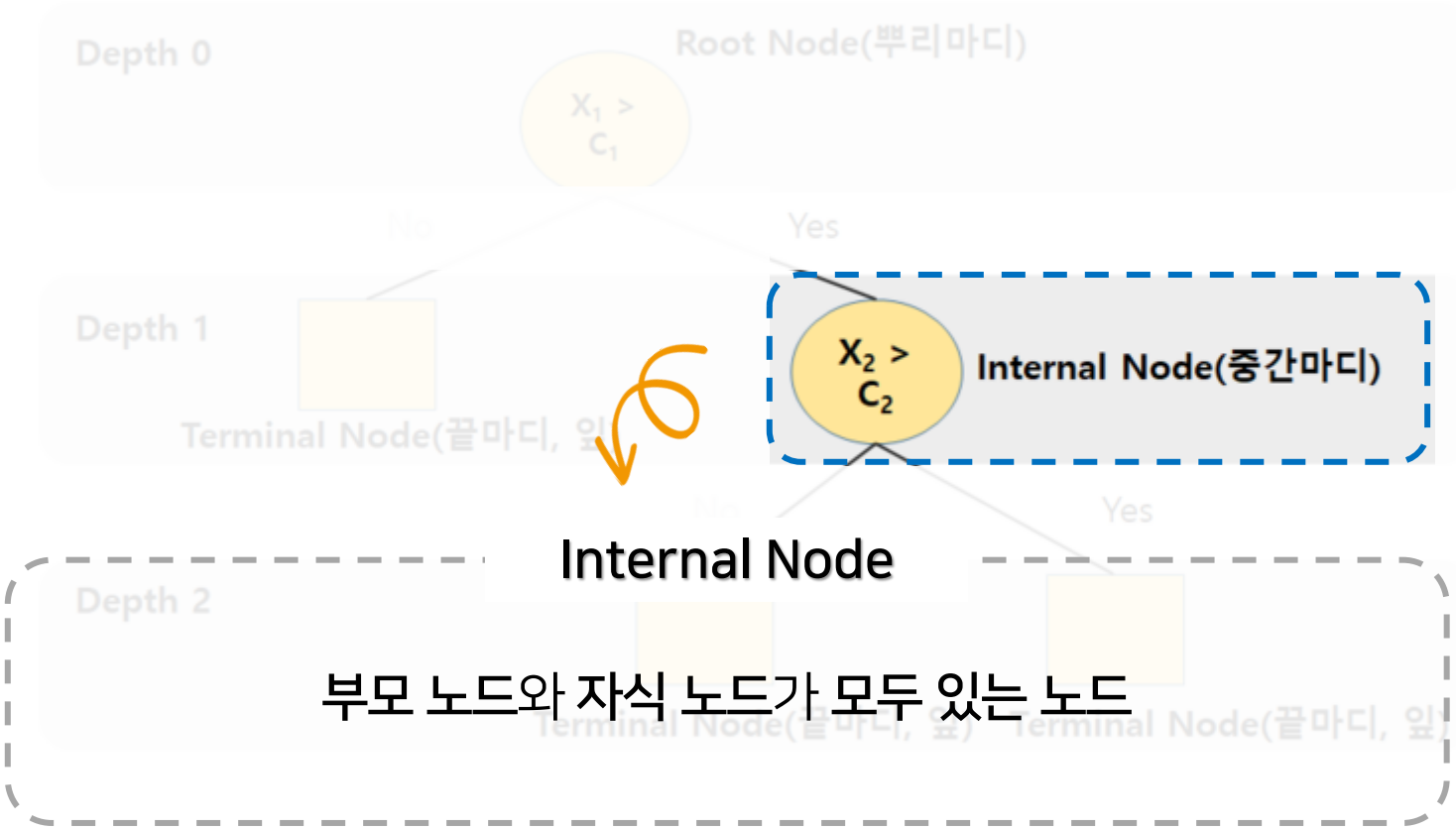


Root Node

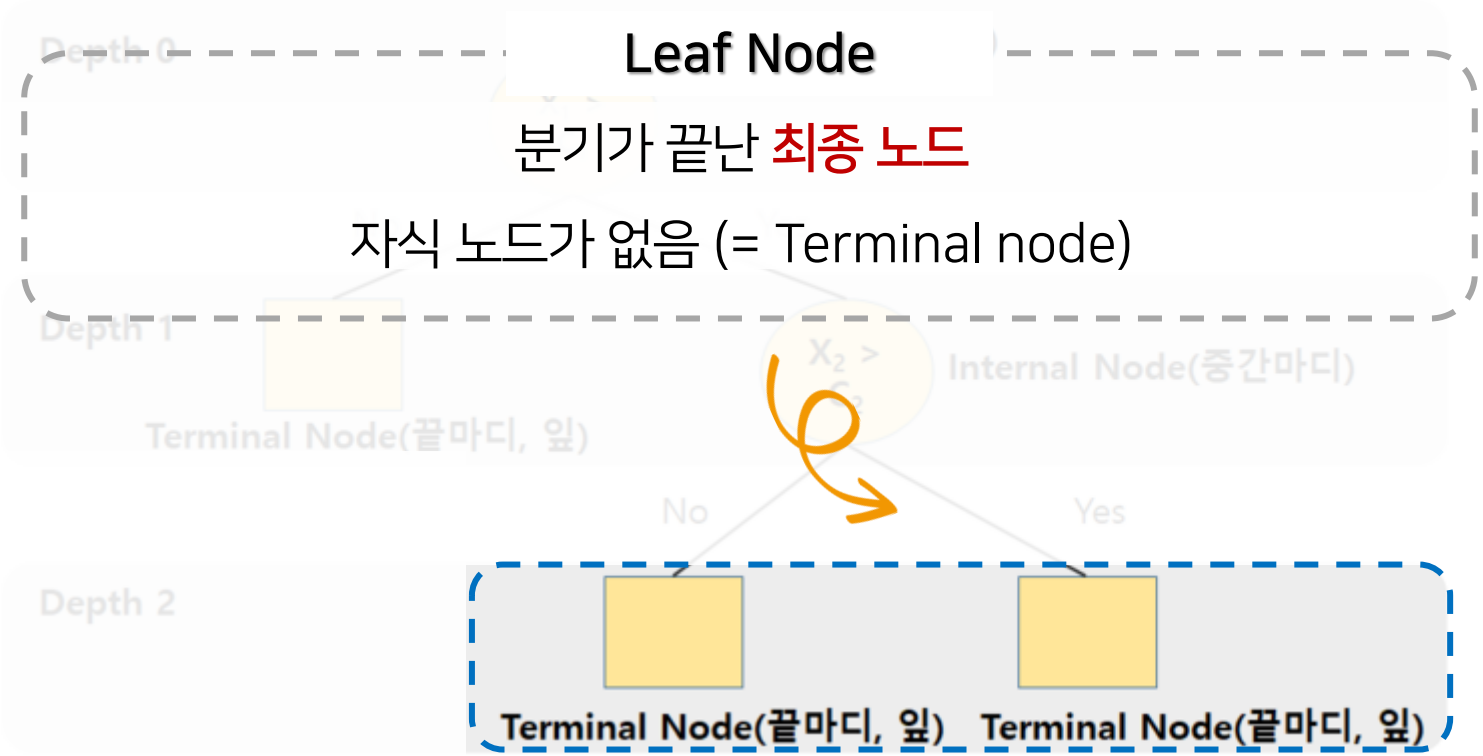
전체 관측값을 포함하는 **최상위 노드**

부모 노드가 없음

용어 정리


Depth 0
Root Node(뿌리마디)
$X_1 > C_1$
No
Yes
Depth 1
Terminal Node(끝마디, 잎)
$X_2 > C_2$
Internal Node(중간마디)
Internal Node
Depth 2
부모 노드와 자식 노드가 모두 있는 노드

용어 정리


Leaf Node
분기가 끝난 최종 노드
자식 노드가 없음 (= Terminal node)
Depth 0
Depth 1
Terminal Node(끝마디, 잎)
Internal Node(중간마디)
No
Yes
Depth 2
Terminal Node(끝마디, 잎)
Terminal Node(끝마디, 잎)

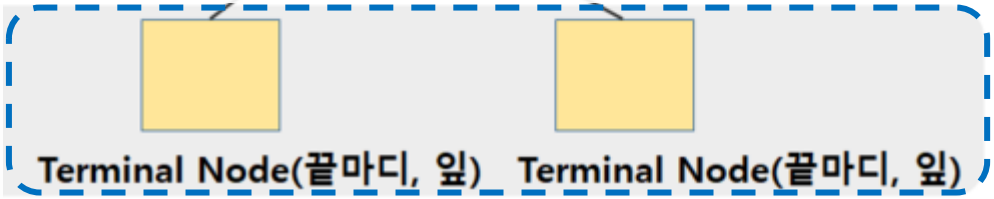
1 트리 기반 모델

용어 정리

모델 학습이 끝나면 Leaf node에 결과값이 들어감

회귀 문제인 경우, 노드 내 관측값의 **평균**

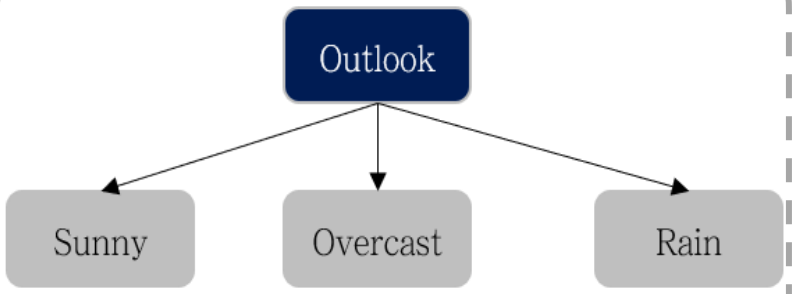
분류 문제인 경우, 노드 내 관측값 중 **최다 클래스**



1 트리 기반 모델

트리 알고리즘의 종류

ID3

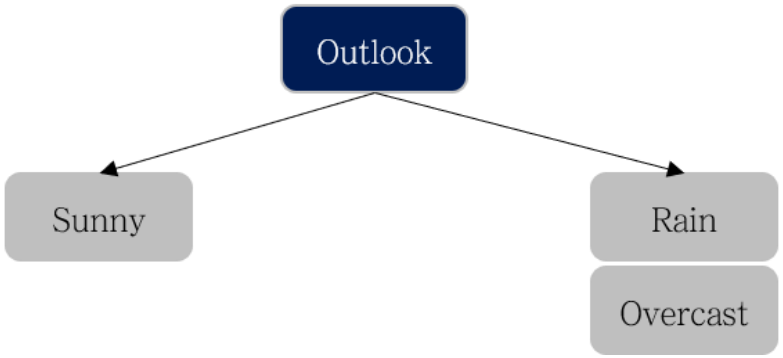


독립변수가 **범주형**인 경우에만 사용

한 노드를 **여러 개로 분기** 가능

분기 기준은 **엔트로피(Entropy)**

CART



독립변수는 **연속형**과 **범주형** 모두 가능

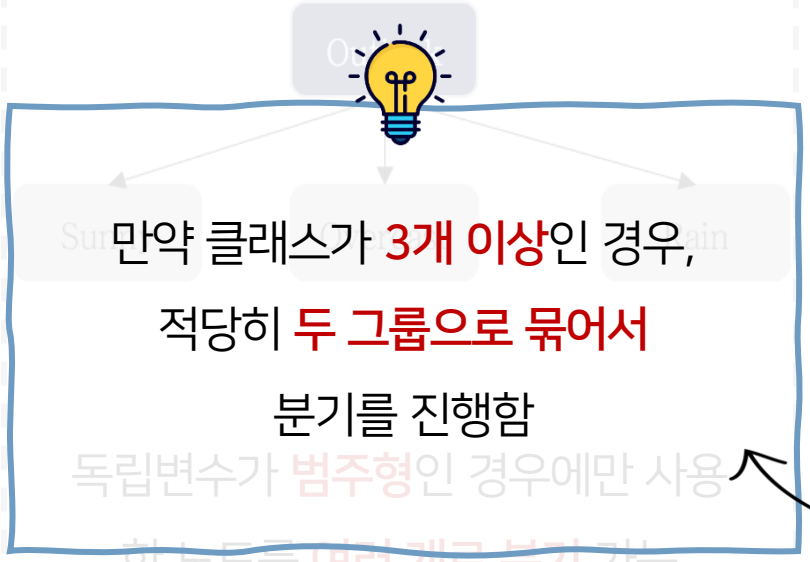
한 노드를 **두 가지로만 분기** 가능

분기 기준은 **지니 계수(Gini Index)**

1 트리 기반 모델

트리 알고리즘의 종류

ID3



만약 클래스가 **3개 이상**인 경우,

적당히 **두 그룹으로 묶어서**

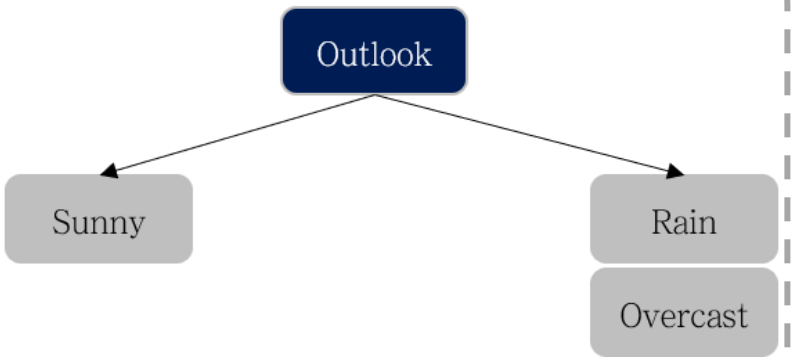
분기를 진행함

독립변수가 **범주형**인 경우에만 사용

한 노드를 **여러 개로 분기** 가능

분기 기준은 **엔트로피(Entropy)**

CART



독립변수는 **연속형**과 **범주형** 모두 가능

한 노드를 **두 가지로만 분기** 가능

분기 기준은 **지니 계수(Gini Index)**

DecisionTreeClassifier

분기 기준

DecisionTreeClassifier는 영역들의 **불순도(impurity)**를 기준으로 분기

영역 내에서 **서로 다른 범주의 개체**들이 얼마나 포함되어 있는가를 의미

불순도가 높을수록 분류가 잘 되지 않았다고 판단함

1 트리 기반 모델

DecisionTreeClassifier

불순도를 측정하는 방법으로는 Misclassification Rate,

Gini Index, Entropy 등이 여러가지가 존재함

그 중에서 Entropy에 대해서 알아보자!

DecisionTreeClassifier

자세한 내용은 딥러닝 1주차 클린업 참고!

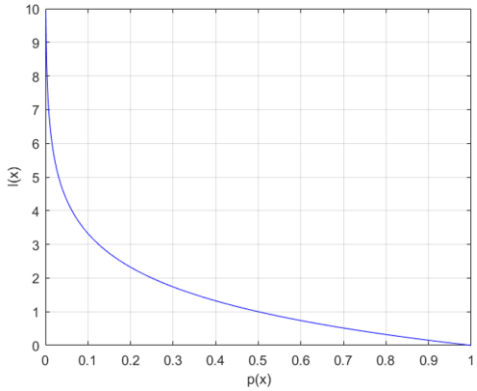
Entropy

확률분포가 가지는 **정보량**을 수치로 표현한 것

특정 사건이 발생할 확률이 작아질수록 정보량이 높음

$$I(x) = -\log_b P(X)$$

$P(X)$: 사건 X가 발생할 확률



DecisionTreeClassifier

자세한 내용은 딥러닝 1주차 클린업 참고!

Entropy

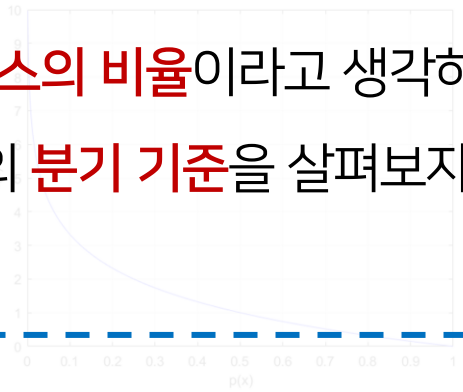
확률분포가 가지는 **정보량**을 수치로 표현한 것

특정 사건이 발생할 확률이 작아질수록 정보량이 높음

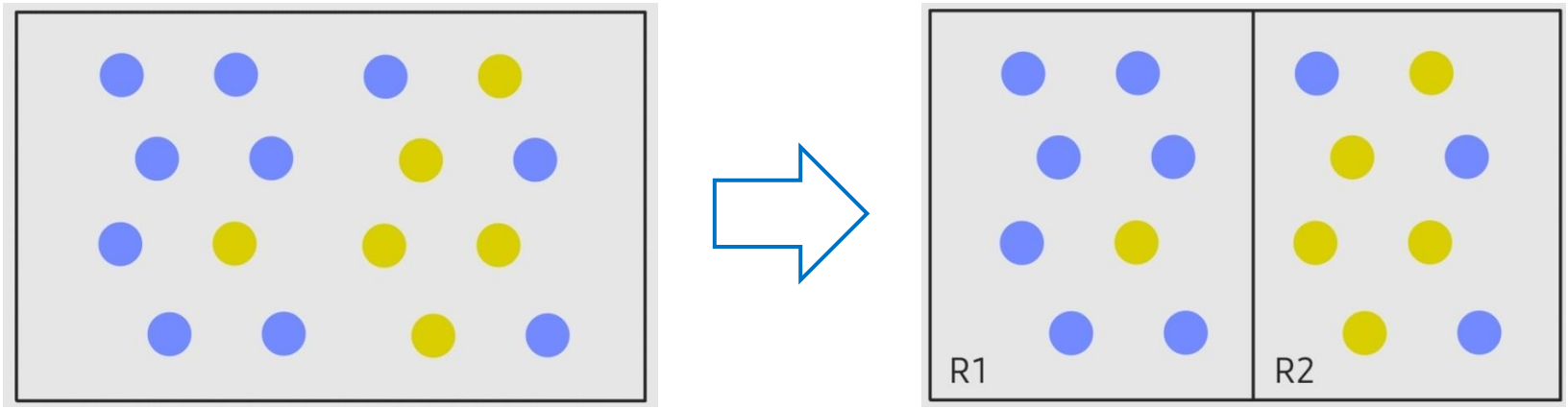
발생확률을 영역 내 특정 **클래스의 비율**이라고 생각하고

DecisionTreeClassifier의 **분기 기준**을 살펴보자

$P(X)$: 사건 X가 발생할 확률



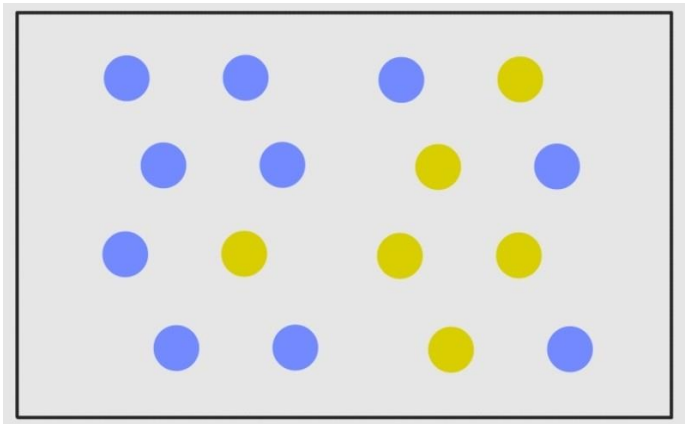
엔트로피 계산 과정



DecisionTreeClassifier는 **불순도를 낮추는 방향**으로 학습을 진행

Entropy를 사용해서 불순도를 계산

엔트로피 계산 과정



분기 이전 엔트로피

전체 16개의 공

파란 공 10개, 노란 공 6개

R1 R2

$$-\frac{10}{16}\log_2\left(\frac{10}{16}\right) - \frac{6}{16}\log_2\left(\frac{6}{16}\right) \approx 0.95$$

DecisionTreeClassifier의 핵심은 불순도를 낮추는 방향으로 학습을 진행

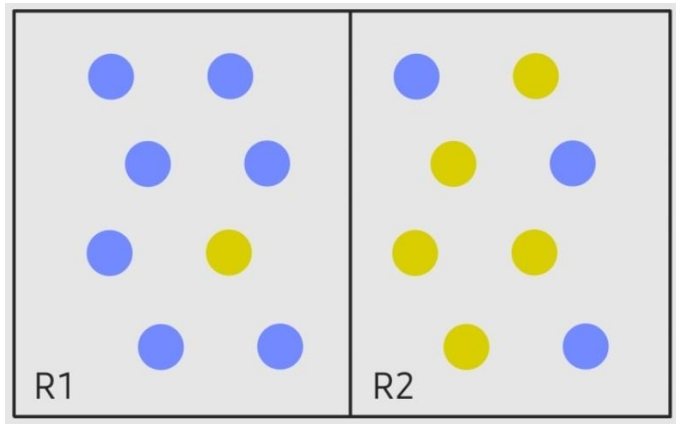
엔트로피 계산 과정

분기 이후 엔트로피

$$R1 : -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right)$$

$$R2 : -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right)$$

엔트로피 = $R1 + R2 \approx 0.75$



DecisionTreeClassifier의 핵심은 불순도를 낮추는 방향으로 학습을 진행

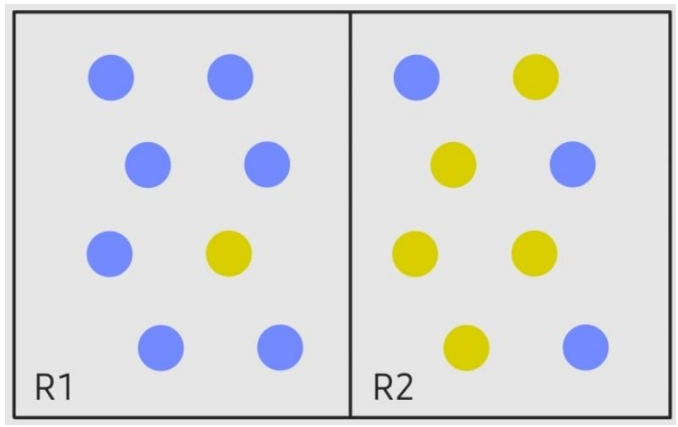
엔트로피 계산 과정

분기 이후 엔트로피

$$R1: -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right)$$

$$R2: -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right)$$

엔트로피 = $R1 + R2 \approx 0.75$



불순도가 분기 이전에 비해 0.2만큼 감소

이러한 불순도의 감소량을 정보이득이라고 함

1 트리 기반 모델

엔트로피 계산 과정

분기 이후 엔트로피

$$R1: -\frac{7}{8}\log_2\left(\frac{7}{8}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right)$$

$$R2: -\frac{3}{8}\log_2\left(\frac{3}{8}\right) - \frac{5}{8}\log_2\left(\frac{5}{8}\right)$$

엔트로피 = $R1 + R2 \approx 0.75$

DecisionTreeClassifier는

불순도의 합은 줄이는 방향으로,

정보이득은 늘리는 방향으로 학습을 진행함!

불순도가 분기 이전에 비해 0.2만큼 감소

분기 기준을 살펴봤으니 이제 학습 과정을 살펴보자!

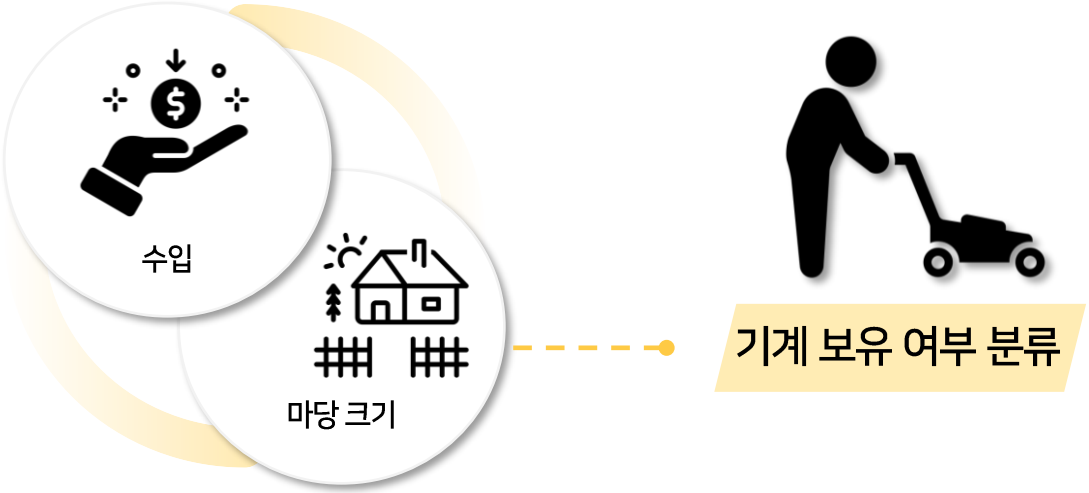
1 트리 기반 모델

Decision Tree Classifier | 학습 프로세스

Ex)

24개 가구에 대해서 **잔디깎이 기계가 있는지 여부를 분류**하려고 한다.

독립변수로는 **수입(Income)**과 **마당 크기(Lot size)**를 사용했다.



Decision Tree Classifier | 학습 프로세스

Income	Lot size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
85.5	16.8	Owner
64.8	17.2	Non-owner
⋮	⋮	⋮
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner

① 하나의 변수 기준으로 **오름차순으로 정렬**함 Ex) Lot size

Decision Tree Classifier | 학습 프로세스

Income	Lot size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
85.5	16.8	Owner
64.8	17.2	Non-owner
⋮	⋮	⋮
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner

분기 이전의 엔트로피

$$1-\left(\frac{12}{24}\right)^2-\left(\frac{12}{24}\right)^2=0.5$$

분기 이후의 엔트로피

$$\frac{1}{24}\left(1-\left(\frac{1}{1}\right)^2\right)+\frac{23}{24}\left(1-\left(\frac{12}{23}\right)^2-\left(\frac{11}{23}\right)^2\right)$$

$$\approx 0.48$$

② Split 가능한 모든 경우의 **불순도**를 탐색

Decision Tree Classifier | 학습 프로세스

Income	Lot size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
85.5	16.8	Owner
64.8	17.2	Non-owner
⋮	⋮	⋮
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner

분기 이전의 엔트로피

불순도가 분기 이전에 비해

0.02만큼 감소 = 0.02만큼의 **정보 이득**

(Information Gain)

분기 이후의 엔트로피

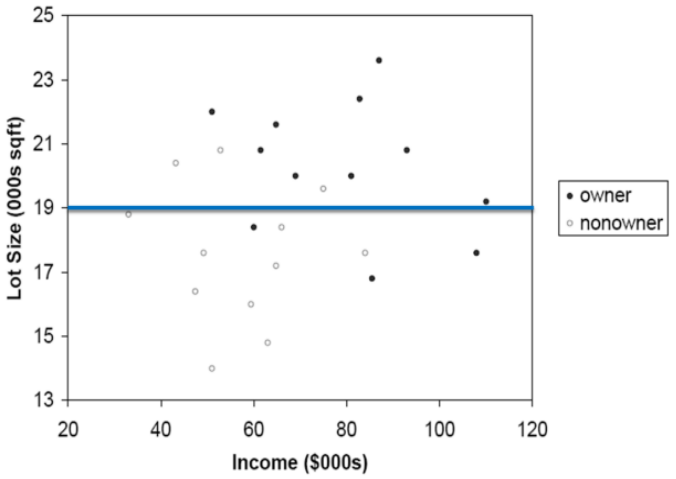
변수가 여러 개이기 때문에

모든 경우를 탐색하여 최선의 결과를 찾음

② Split 가능한 모든 경우의 **불순도**를 탐색

Decision Tree Classifier | 학습 프로세스

Income	Lot size	Ownership
51.0	14.0	Non-owner
63.0	14.8	Non-owner
59.4	16.0	Non-owner
85.5	16.8	Owner
64.8	17.2	Non-owner
⋮	⋮	⋮
33.0	18.8	Non-owner
110.1	19.2	Owner
75.0	19.6	Non-owner

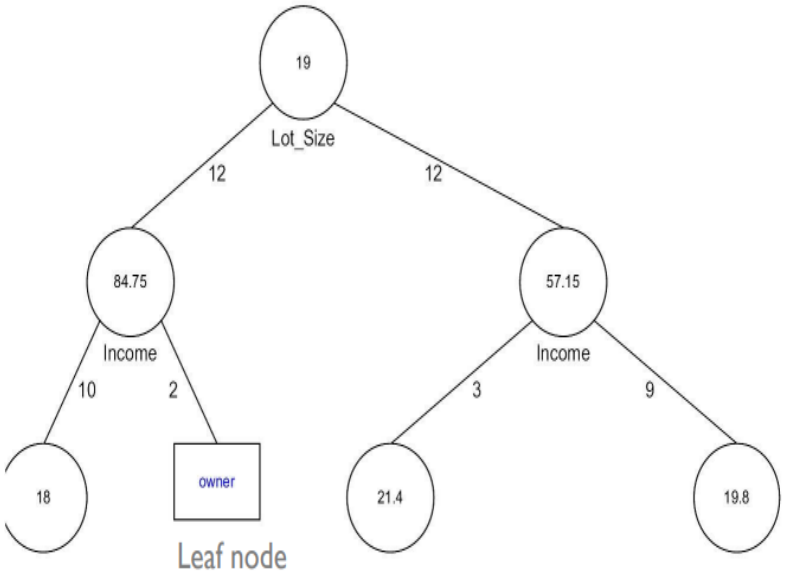


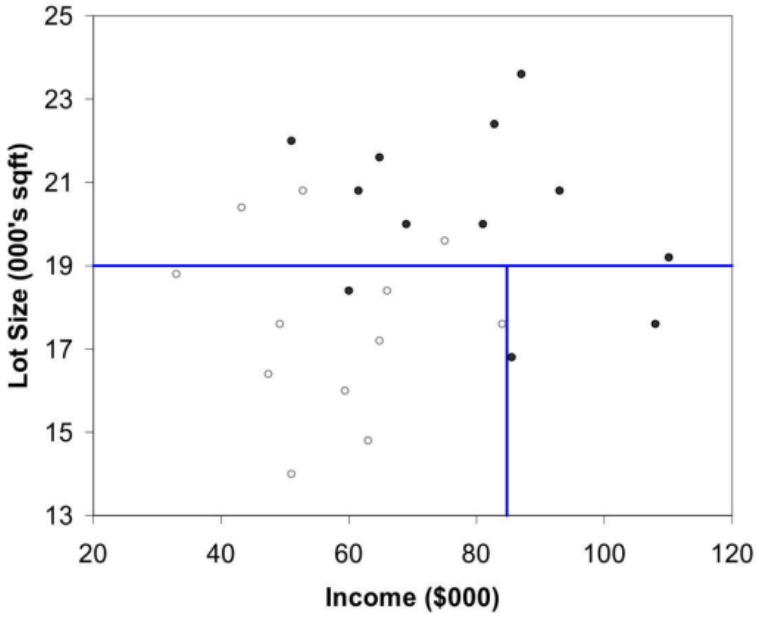
→ Lot size라는 **독립변수**

영역을 19를 기준으로 분할

③ Information Gain이 **가장 큰 경우**에 따라 분기

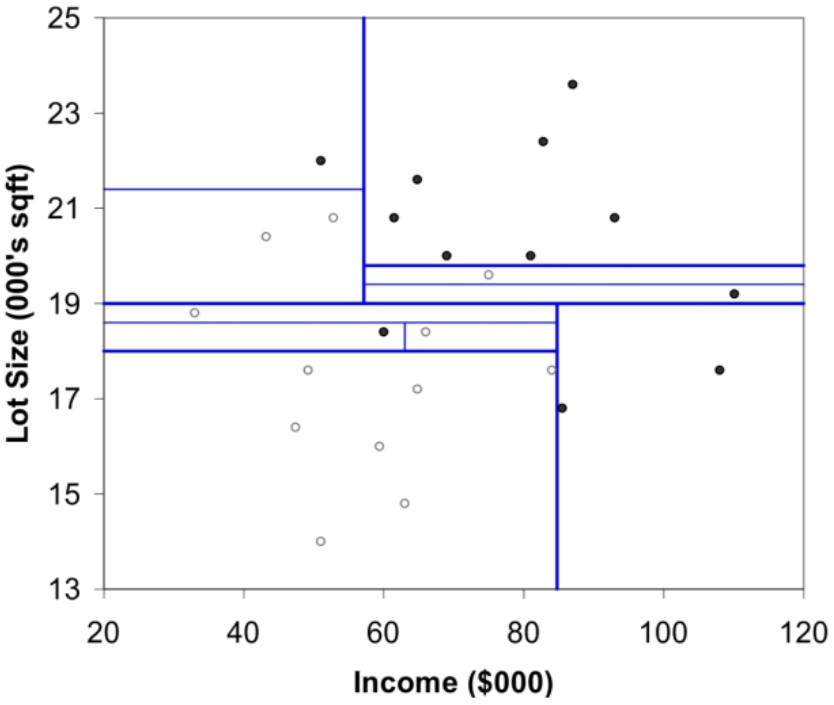
Decision Tree Classifier | 학습 프로세스





④ 분기되는 노드를 따라가면서, 위 과정 반복

Decision Tree Classifier | 학습 프로세스



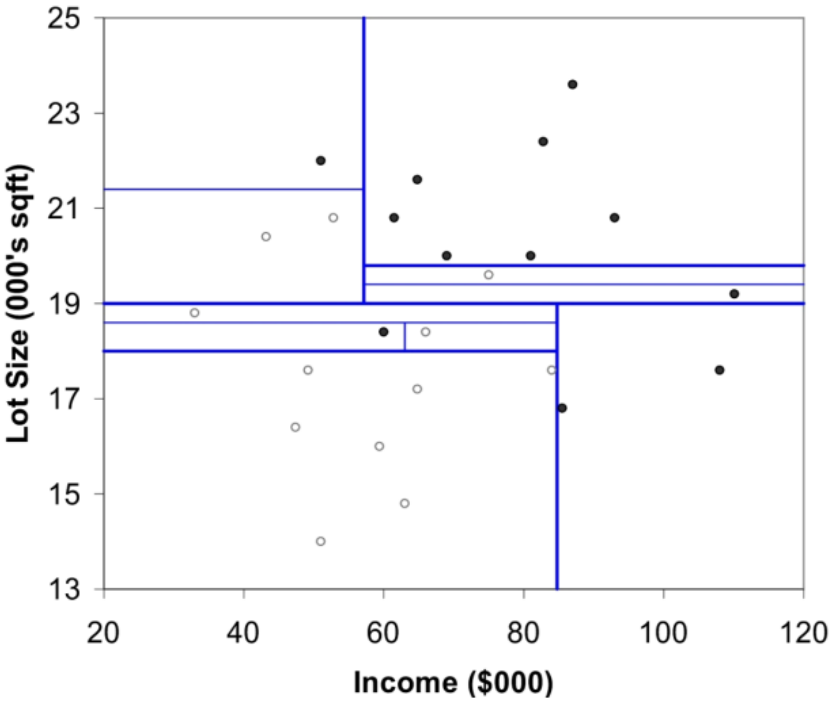
서로 다른 클래스의 관측값이
겹쳐 있지 않다면,
학습 데이터를 **100% 구분할 수 있음**
→ 이를 Full-Tree라고 함

이후 새로운 관측값은 트리를 따라 내려가며
Leaf node의 비율대로 판단함

⑤ Information Gain=0 이 되면 **분기를 종료**

1 트리 기반 모델

Decision Tree Classifier | 학습 프로세스



서로 다른 클래스의 관측값이

겹쳐 있지 않다면,

학습 데이터를 **100% 구분할 수 있음**

→ 이를 Full-Tree라고 함

이후 새로운 관측값은 트리를 따라 내려가며

Leaf node의 비율대로 판단함

⑤ Information Gain=0 이 되면 **분기를 종료**

Decision Tree Regressor

Decision Tree를 회귀 문제에 적용할 수 있게 변형시킴

분기하는 기준이 달라짐

RSS (Residual Sum of Squares, 실제값과 추정값의 차이)

$$MSE(Mean\ Square\ error) = \frac{\mathrm{RSS(Residual\ sum\ of\ square)}}{degree\ of\ freedom}$$

Decision Tree의 모델 학습은 **RSS 값을 줄이는 방향**으로 작동함

Decision Tree Regressor

Decision Tree를 회귀 문제에 적용할 수 있게 변형시킴

분기하는 기준이 달라짐

RSS (Residual Sum of Squares, 실제값과 추정값의 차이)

$$MSE(Mean\ Square\ error) = \frac{\text{RSS(Residual sum of square)}}{degree\ of\ freedom}$$

→ Decision Tree의 모델 학습은 **RSS 값을 줄이는 방향**으로 작동함

Decision Tree Regressor

목적함수

$$min_{cm} \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 = min_{cm} \sum_{i=1}^{N} \left\{ y_i - \sum_{m=1}^{M} c_m I(X \in R_m) \right\}^2$$

c_m: m번째 노드의 예측값 (=해당 노드 관측값의 평균)

R_m: m번째 노드

M: 전체 노드 개수

N: 전체 관측치 개수

Decision Tree Regressor

목적함수

$$min_{cm} \sum_{i=1}^{N} \{y_i - f(x_i)\}^2 = min_{cm} \sum_{i=1}^{N} \left\{ y_i - \sum_{m=1}^{M} c_m I(X \in R_m) \right\}^2$$

c_m : m번째 노드의 예측값 (=해당 노드 관측값의 평균)

N : 전체 관측치 개수

i번째 관측치의 실제값과 예측값의 차이가 있고,

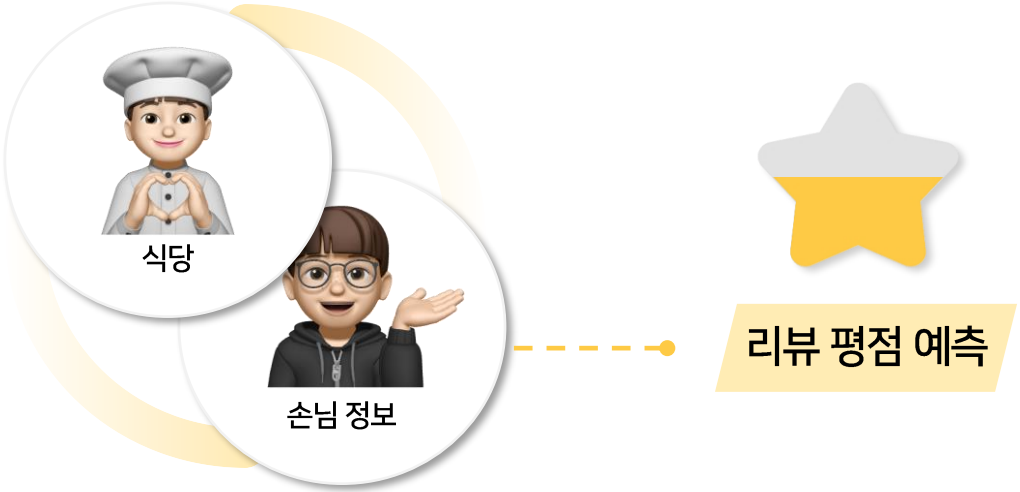
그 차이의 제곱합을 최소화시켜야 한다는 의미

Decision Tree Regressor | 학습 프로세스

Ex)

어느 식당에서 손님들의 정보를 바탕으로 **리뷰 평점을 예측**하려고 한다.

독립변수로는 **지불 금액(Price), 주차 비용(Parking fee), 외국인 여부(Foreign)**를 사용했다.



Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
12.0	2.0	1	10
13.0	3.0	0	10
14.0	4.0	1	13
15.0	5.0	0	20
17.5	7.5	1	35
18.0	8.0	1	44
18.5	8.5	1	52
19.0	9.0	0	55
21.0	11.0	1	80
22.0	12.0	0	83
23.0	13.0	1	80
24.0	14.0	1	83

① 한 변수를 기준으로 관측값을 정렬 Ex) Price

Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
12.0	2.0	1	10
13.0	3.0	0	10
14.0	4.0	1	13
15.0	5.0	0	20
17.5	7.5	1	35
18.0	8.0	1	44
18.5	8.5	1	52
19.0	9.0	0	55
21.0	11.0	1	80
22.0	12.0	0	83
23.0	13.0	1	80
24.0	14.0	1	83

$R1$의 예측값 $= \frac{10+10+10+10}{4} = 10$

$R2$의 예측값 $= \frac{13+20+\cdots+83}{10} = 54.5$

$$RSS = (10-10)^2 + (10-10)^2 + \cdots + (83-54.5)^2$$

$$= 10114.75$$

⋮

만약 'Price ≤ 13.0'을 기준으로 Split한 경우,

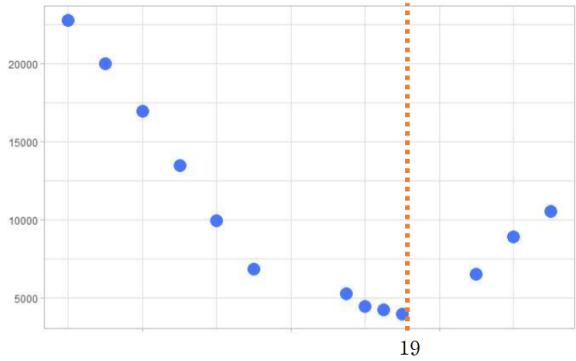
RSS 값은 위와 같이 계산됨

② Split 가능한 모든 경우의 RSS를 탐색

분기 조건에 따라 RSS 값은 달라지고, 그때마다 모든 Split Point를 탐색함

Decision Tree Regressor | 학습 프로세스

Price	Parking fee	Foreign	Review Rating
10.0	0.0	0	10
11.0	1.0	0	10
12.0	2.0	1	10
13.0	3.0	0	10
14.0	4.0	1	13
15.0	5.0	0	20
17.5	7.5	1	35
18.0	8.0	1	44
18.5	8.5	1	52
19.0	9.0	0	55
21.0	11.0	1	80
22.0	12.0	0	83
23.0	13.0	1	80
24.0	14.0	1	83



트리의 첫 번째 분기 기준을 'Price ≤ 19.0'으로 채택

(ex. 3961.8 vs 6412.4 vs 11685.0)

③ 가장 RSS가 작은 경우에 따라 분기

변수가 여러 개라면, 각 변수별로 최소 RSS가 나오는 지점을 찾고, 그 값들 중 가장 작은 경우를 채택

Decision Tree Regressor | 학습 프로세스

④ 분기되는 노드를 따라 내려가면서, 위 과정을 반복

⑤ RSS 값이 더이상 작아지지 않으면 분기를 종료

이렇게 트리가 완성되고 나면, 새로운 객체는 만들어둔 트리를 따라 내려가서 Leaf node의 예측값(평균값)대로 판단하면 됨

Decision Tree Regressor | 학습 프로세스

④ 분기되는 노드를 따라 내려가면서, 위 과정을 반복

⑤ RSS 값이 더이상 작아지지 않으면 분기를 종료

이렇게 트리가 완성되고 나면, **새로운 객체는 만들어둔 트리를 따라 내려가서 Leaf node의 예측값(평균값)대로 판단**하면 됨

Decision Tree Regressor | 학습 프로세스

어디까지 분기할 것인가?

④ 분기되는 노드를 따라 내려가면서, 위 과정을 반복

Decision Tree는 불순도나 RSS가 더 줄어들지 않을 때까지 분기

하지만 이 Full-Tree는 노이즈의 정보까지 학습

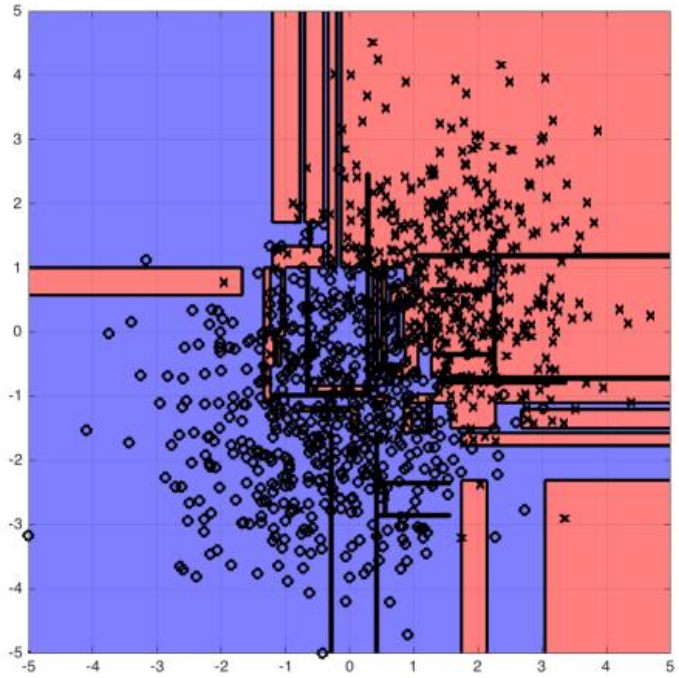
⑤ RSS 값이 더이상 작아지지 않으면 분기를 종료

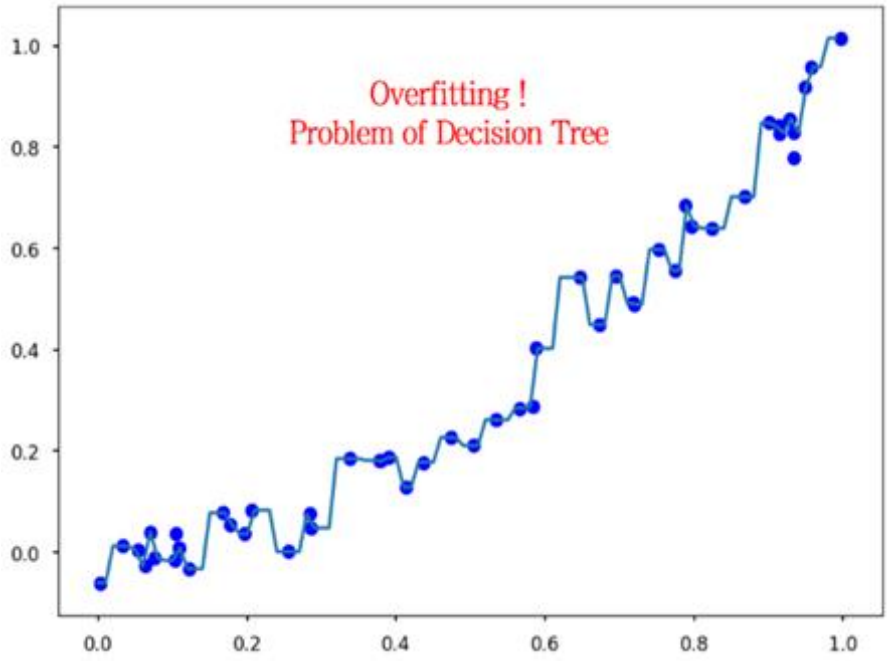
과대적합(overfitting) 문제 야기

이렇게 트리가 완성되고 나면, 새로운 객체는 만들어둔 트리를 따라 내려가서 Leaf node의 예측값(평균값)대로 판단하면 됨

1 트리 기반 모델

트리 기반 모델의 과적합 (feat. Pruning)





과대적합(overfitting) = Bias ↑ Variance ↓

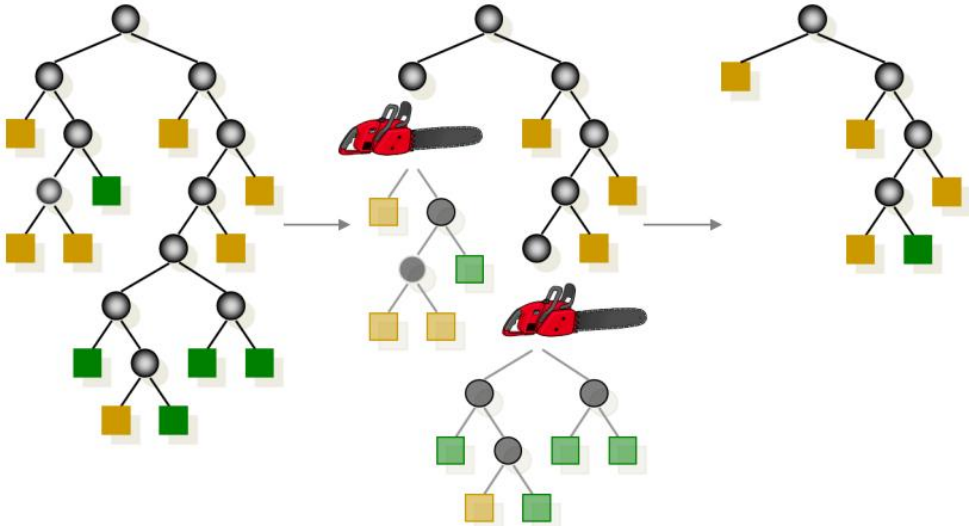
= 새로운 데이터에 강건하지 못함(non-robust)

트리 기반 모델의 과적합 (feat. Pruning)

가지치기 (Pruning)

과적합을 방지하고자 Decision Tree의 **분기 깊이를 적정 수준에서 제한**하는 방법

= Full-Tree까지 다 만들지 않겠다는 뜻



트리 기반 모델의 과적합 (feat. Pruning)

가지치기 (Pruning)

과적합을 방지하고자 Decision Tree의 **분기 깊이를 적정 수준에서 제한**하는 방법

= Full-Tree까지 다 만들지 않겠다는 뜻

훈련 성능이 완벽하지 않더라도, 적당히 학습량을 조절하여

일반화 성능을 확보하자는 것이 핵심

트리 기반 모델의 과적합 (feat. Pruning)

가지치기 (Pruning)

과적합을 방지하고자 Decision Tree의 **분기 깊이를 적정 수준에서 제한**하는 방법

= Full-Tree까지 다 만들지 않겠다는 뜻

가지치기

사후 가지치기

Full-Tree를 만든 후,
적절한 수준에서
Leaf node를 결합함

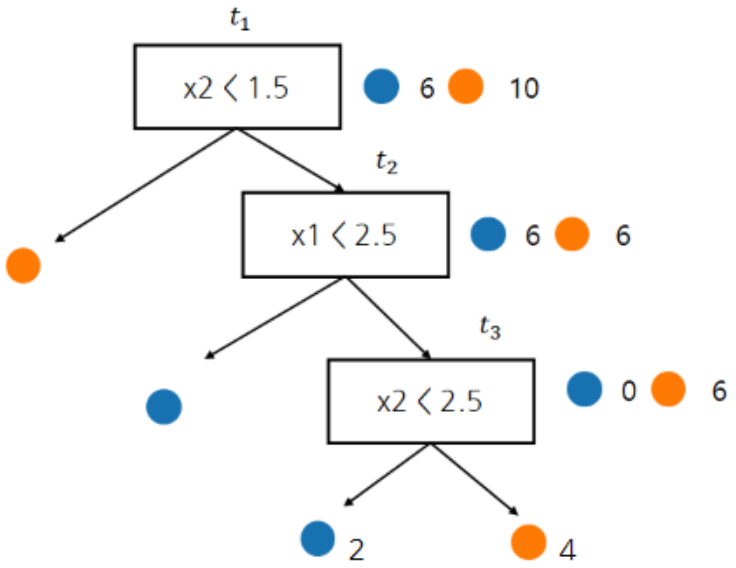
사전 가지치기

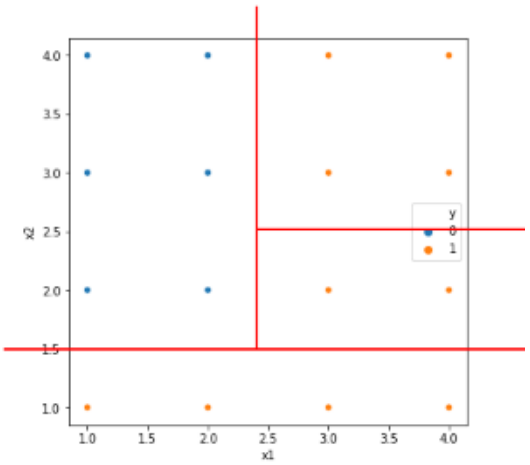
트리의 깊이나 Leaf node 내 최소
관측값 수를 미리 지정해, Tree가
일정 수준 이상 자라지 못하게 함

사후 가지치기(post-pruning)

REP (Reduce Error Pruning)

Full-Tree의 밑에서부터 각 분할 지점을 돌아보면서 분할 이전과 이후의 불순도를 비교하고, **분할 이전의 불순도가 더 낮다면 해당 자식노드는 쳐내는 방식**

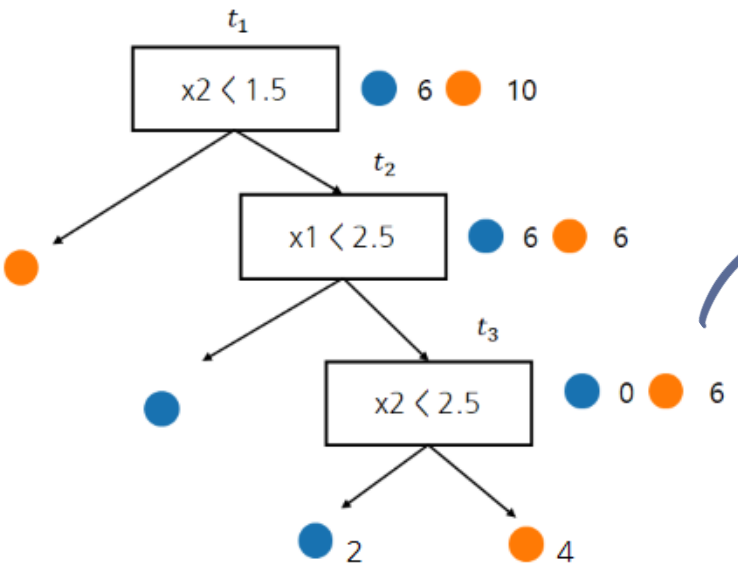




사후 가지치기(post-pruning)

REP (Reduce Error Pruning)

Full-Tree의 밑에서부터 각 분할 지점을 돌아보면서 분할 이전과 이후의 불순도를 비교하고, **분할 이전의 불순도가 더 낮다면 해당 자식노드는 쳐내는 방식**



t3에서 분할 이후의 불순도가
분할 이전의 부모 노드보다 더 커짐

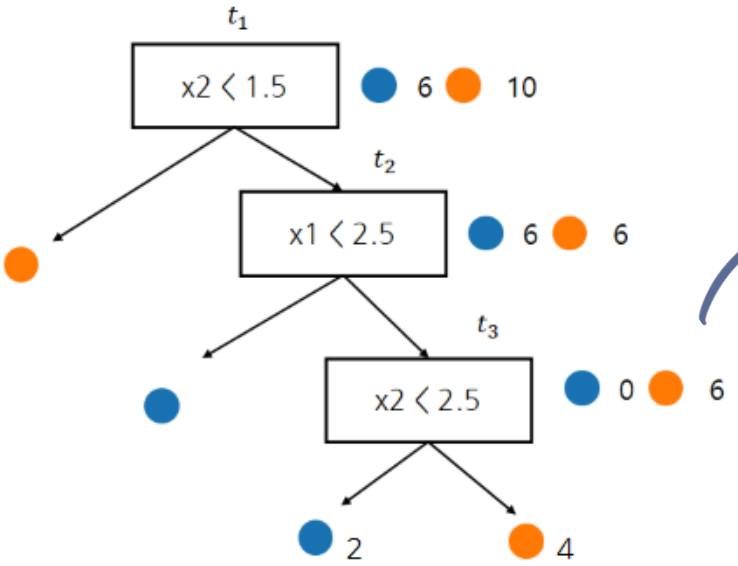
⋮

t3의 자식노드를 쳐내고, 주황색 공이
6개 든 노드를 리프노드로 바꿔주면
가지치기 완료!

사후 가지치기(post-pruning)

REP (Reduce Error Pruning)

Full-Tree의 밑에서부터 각 분할 지점을 돌아보면서 분할 이전과 이후의 불순도를 비교하고, 분할 이전의 불순도가 더 낮다면 해당 자식노드는 쳐내는 방식



불순도가 감소하지 않은

분할을 다시 결합

사후 가지치기(post-pruning)

REP (Reduce Error Pruning)

Full-Tree의 밑에서부터 각 분할 지점을 돌아보면서 분할 이전과 이후의 불순도를 비교하고, **분할 이전의 불순도가 더 낮다면 해당 자식노드는 쳐내는 방식**

원리가 단순해 구현은 쉽지만, 단순히 불순도만으로 pruning 여부를 판단해 **일반화 성능 향상에는 제한적**임

사후 가지치기(post-pruning)

CCP (Cost Complexity Pruning)

트리의 복잡도를 함께 고려하여 더욱 안정적인 Tree를 얻을 수 있음

$$CC(T) = Err(T) + \alpha \times L(T)$$

$Error\ Term$: 불순도(gini, entropy, misclassification)

$L(T)$: 리프노드의 개수

→ **원래 불순도 지표에서 트리의 복잡도에 따라 적절한 수준(α)의 페널티를 부여**

사후 가지치기(post-pruning)

CCP (Cost Complexity Pruning)

트리의 복잡도를 함께 고려하여 더욱 안정적인 Tree를 얻을 수 있음

$$CC(T) = Err(T) + \alpha \times L(T)$$

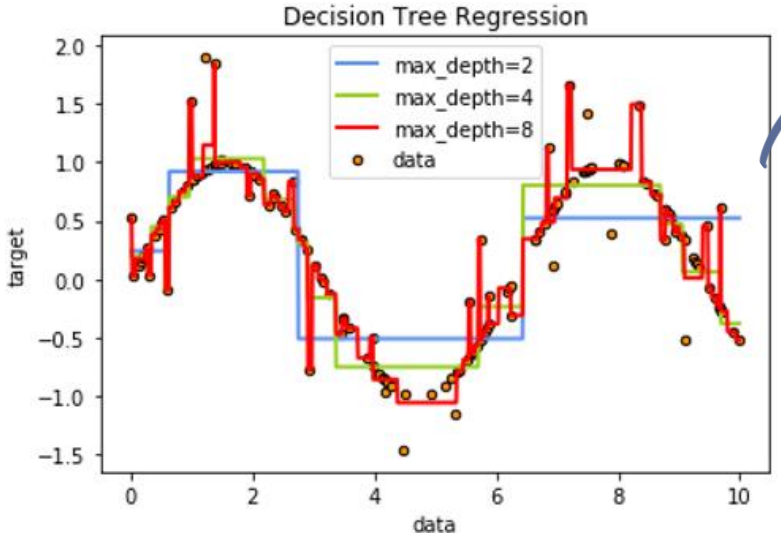
최적의 α 는 `.cost_complexity_pruning_path` 를 통해 구할 수 있음

→ α의 후보들을 대상으로 교차검증을 실시해 가장 valid score가 좋은 것을 택하면 됨

사전 가지치기(pre-pruning)

Max_depth

트리가 분기하는 최대 깊이를 제한하여 사전 가지치기를 수행하는 하이퍼 파라미터



max_depth=2일때 가장 단순,

max_depth=8일때 가장 복잡한 모델

⋮

복잡도가 높을수록 훈련 데이터

하나하나를 정확하게 예측하지만,

새로운 데이터 예측 시 성능 저하 초래

사전 가지치기(pre-pruning)

Max_depth

트리가 분기하는 **최대 깊이를 제한**하여 사전 가지치기를 수행하는 하이퍼 파라미터

max_depth를 포함한 **여러 파라미터 값을 교차검증으로 비교**하며
가장 좋은 성능을 도출하는 모델을 만드는 것이 핵심

1 트리 기반 모델

사전 가지치기(pre-pruning)

min_samples_split	- 노드를 분할하기 위한 최소 샘플 데이터 수 - *"최소한 이 개수보다는 많이 들어있어야 Split 하겠다"* - 작게 설정할수록 과적합 가능성 증가 (default=2)
min_impurity_decrease	- 노드를 분할하기 위한 최소 불순도 감소량 - *"최소한 이 정도는 불순도 줄어들어야 Split 하겠다"* - 작게 설정할수록 과적합 가능성 증가
min_samples_leaf	- 리프노드가 되기 위한 최소 샘플 데이터 수 - *"리프노드에도 최소한 이 개수의 샘플은 들어있어야 한다"* - 작게 설정할수록 과적합 가능성 증가 (default=1) - 불균형 데이터의 경우에는 작게 설정해야 할 수도 있음
max_leaf_nodes	- 리프노드의 최대 개수 - *"리프노드 아무리 많아도 이 정도까지만 있어야 한다"* - 크게 설정할수록 과적합 가능성 증가

외에도 위의 파라미터들을 사용하면 과적합 미리 억제 가능

1 트리 기반 모델

사전 가지치기(pre-pruning)

min_samples_split	- 노드를 분할하기 위한 최소 샘플 데이터 수 - *"최소한 이 개수보다는 많이 들어있어야 Split 하겠다"* - 작게 설정할수록 과적합 가능성 증가 (default=2)
min_impurity_decrease	- 노드를 분할하기 위한 최소 불순도 감소량 - 작게 설정할수록 과적합 가능성 증가
min_samples_leaf	- 리프노드가 되기 위한 최소 샘플 데이터 수 - *"리프노드에도 최소한 이 개수의 샘플은 들어있어야 한다"* - 작게 설정할수록 과적합 가능성 증가 (default=1) - 불균형 데이터의 경우에는 작게 설정해야 할 수도 있음
max_leaf_nodes	- 리프노드의 최대 개수 - 크게 설정할수록 과적합 가능성 증가

하지만 **단일 트리 알고리즘**은 가지치기(Pruning)로도

Overfitting의 문제를 완전히 극복할 수 없음

단일 트리들을 모아 강력한 모델을 만드는

Ensemble method가 있는데,

이는 3주차 클린업에서 다룰 예정!

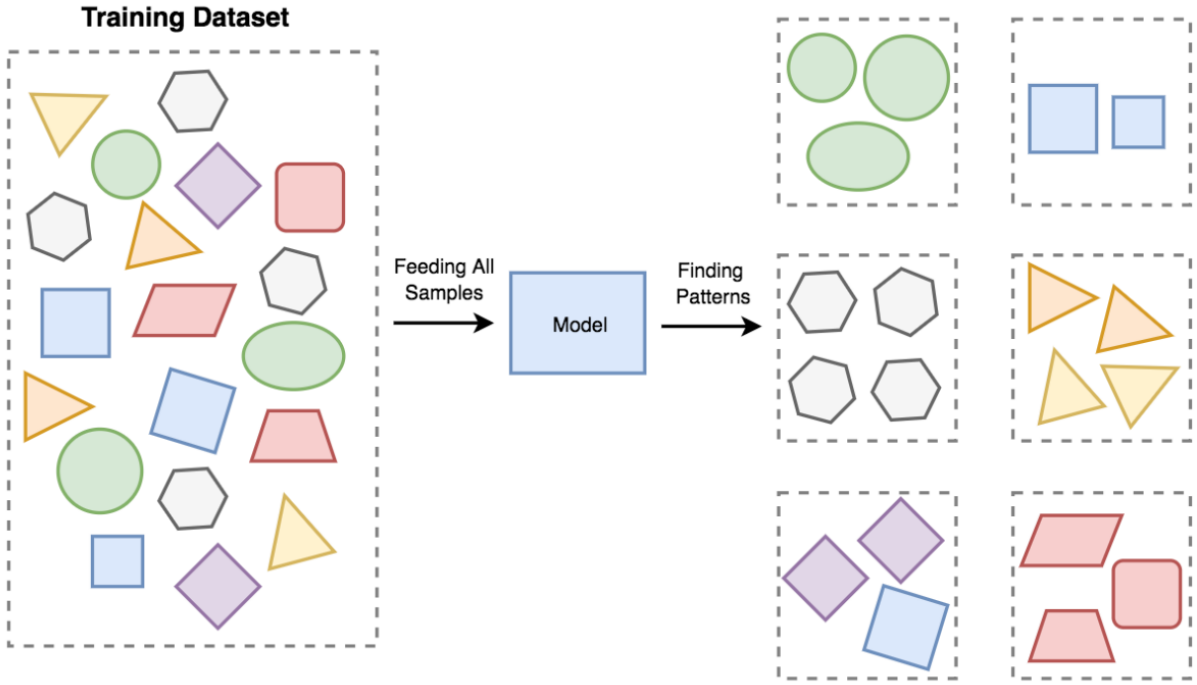
외에도 위의 파라미터들을 사용하면 과적합 미리 억제 가능

2

클러스터링

머신러닝 Remind

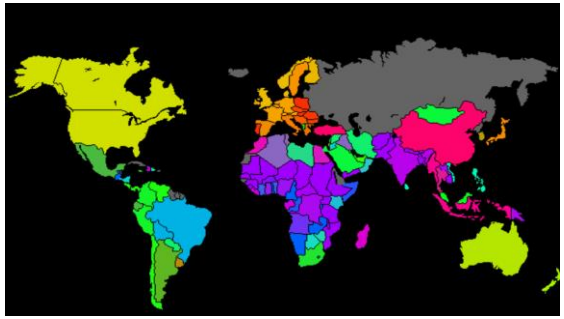
클러스터링

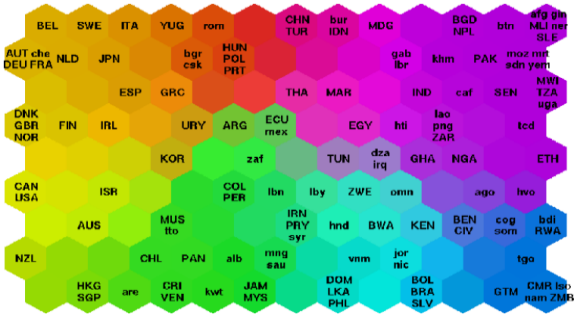


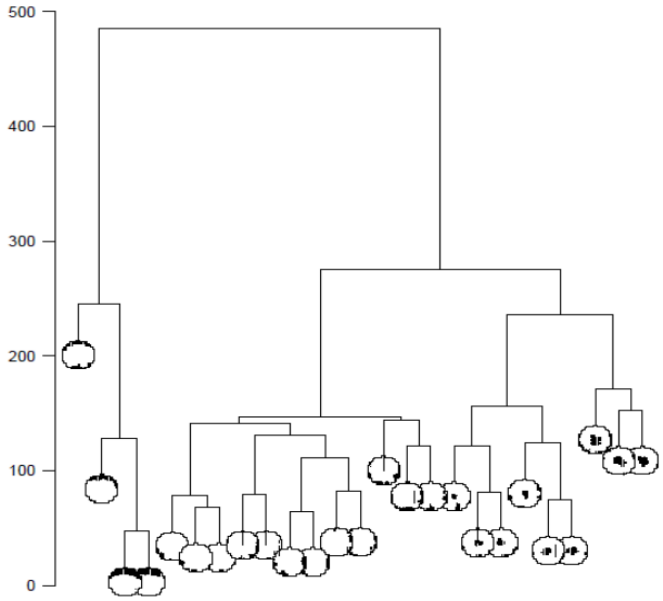
공통된 특징들을 가진 객체끼리 분류하여 그룹으로 구성함

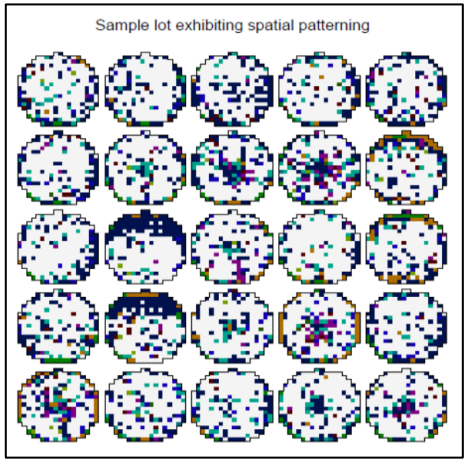
2 클러스터링

클러스터링









데이터에 대한 이해를 높이고, 군집별로 전략을 수립하는 등

다양한 의사결정 과정에 활용됨

클러스터링

정답(Y)이 없다면 좋은 클러스터링을 어떻게 판단할 수 있을까?

같은 그룹끼리는 비슷하게, 다른 그룹끼리는 차이가 나게 만들자!

군집 내 분산 (Intra-cluster variance)은 최소화하고,

군집 간 분산 (Inter-cluster variance)은 최대화함

클러스터링

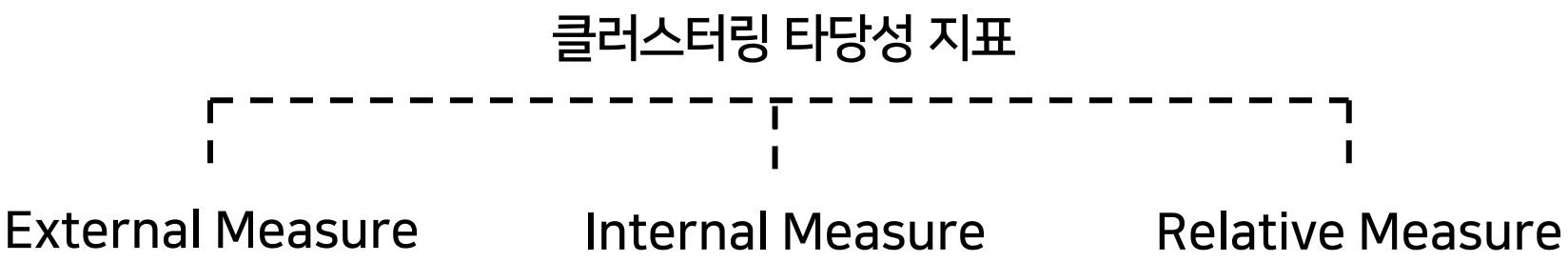
정답(Y)이 없다면 좋은 클러스터링을 어떻게 판단할 수 있을까?

같은 그룹끼리는 비슷하게, 다른 그룹끼리는 차이가 나게 만들자!

군집 내 분산 (Intra-cluster variance)은 **최소화**하고,

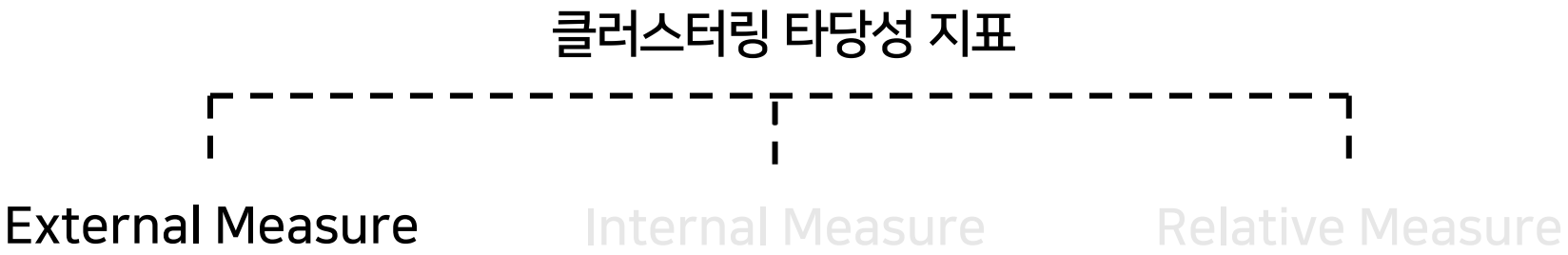
군집 간 분산 (Inter-cluster variance)은 **최대화**함

클러스터링 타당성 지표



클러스터링 타당성 지표에는 위와 같은 3가지가 존재함

클러스터링 타당성 지표

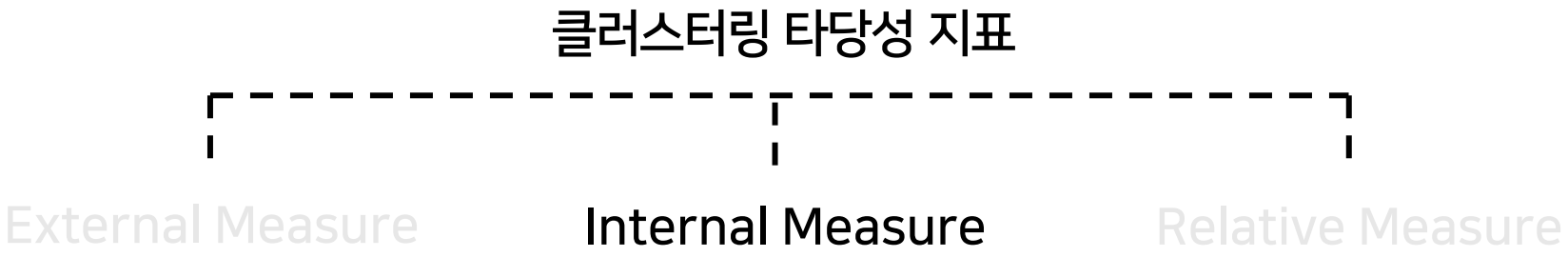


External Measure

클러스터의 개수와 멤버가 정해져있다고 가정하고 평가하는 방식

새로운 알고리즘 개발 및 연구과정에서 확인용으로 사용됨

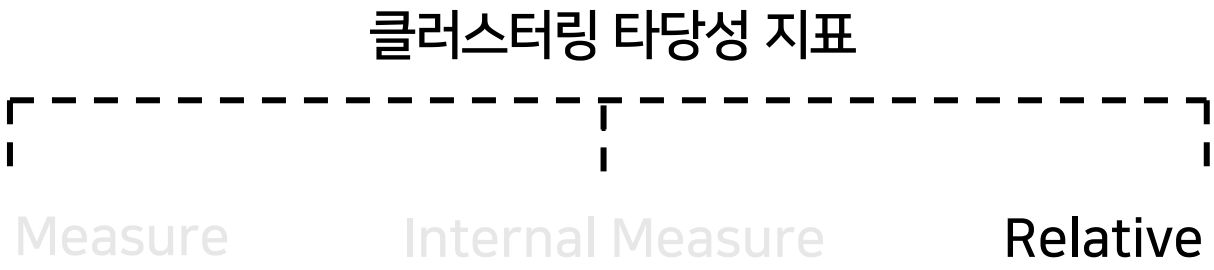
클러스터링 타당성 지표



Internal Measure

군집 내 분산(Intra-cluster variance)에 집중한 방식

클러스터링 타당성 지표



Relative Measure

군집 내 분산과 군집 간 분산에 집중한 방식

가장 일반적으로 많이 사용됨

Internal Measure

WSS(Within Sum of Squares)

군집 내의 중심과 객체들 간의 거리를 제곱합한 값으로

작을수록 군집화가 잘 된 것으로 판단 가능

$$WSS = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|X_j - C_i\|^2$$

클러스터를 많이 생성할수록 줄어들게 되는데

이를 클러스터 개수 선택에 활용할 수 있음

Internal Measure

WSS(Within Sum of Squares)

군집 내의 중심과 객체들 간의 거리를 제곱합한 값으로

작을수록 군집화가 잘 된 것으로 판단 가능

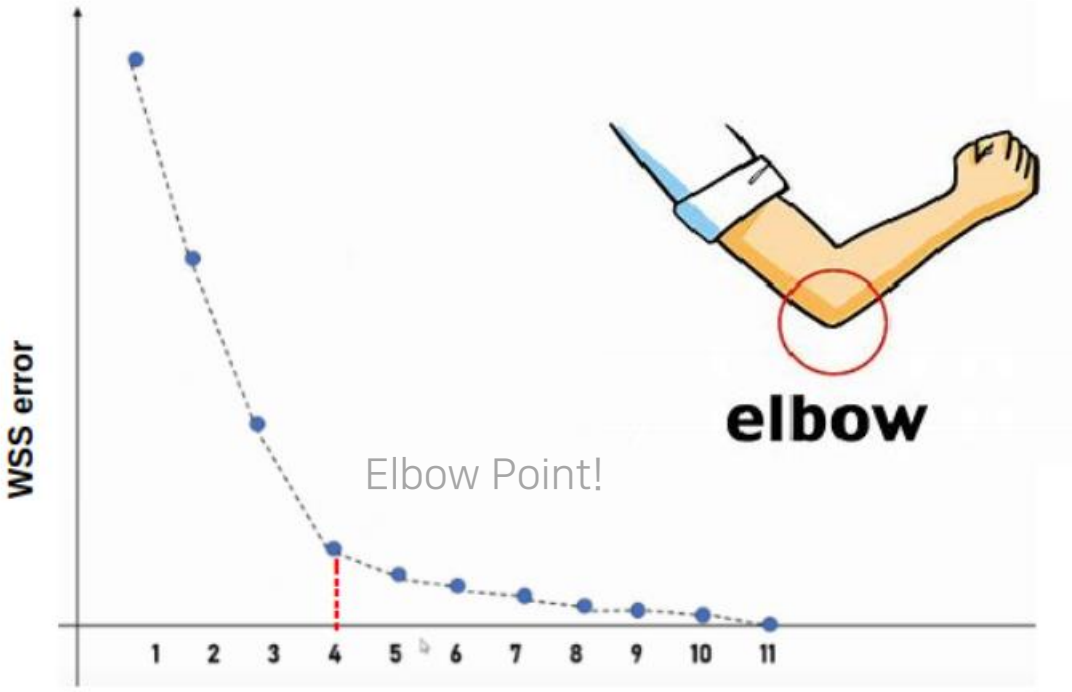
$$WSS = \sum_{i=1}^{K} \sum_{x_j \in C_i} \|X_j - C_i\|^2$$

클러스터를 많이 생성할수록 줄어들게 되는데,

이를 클러스터 개수 선택에 활용할 수 있음 

2 클러스터링

Internal Measure



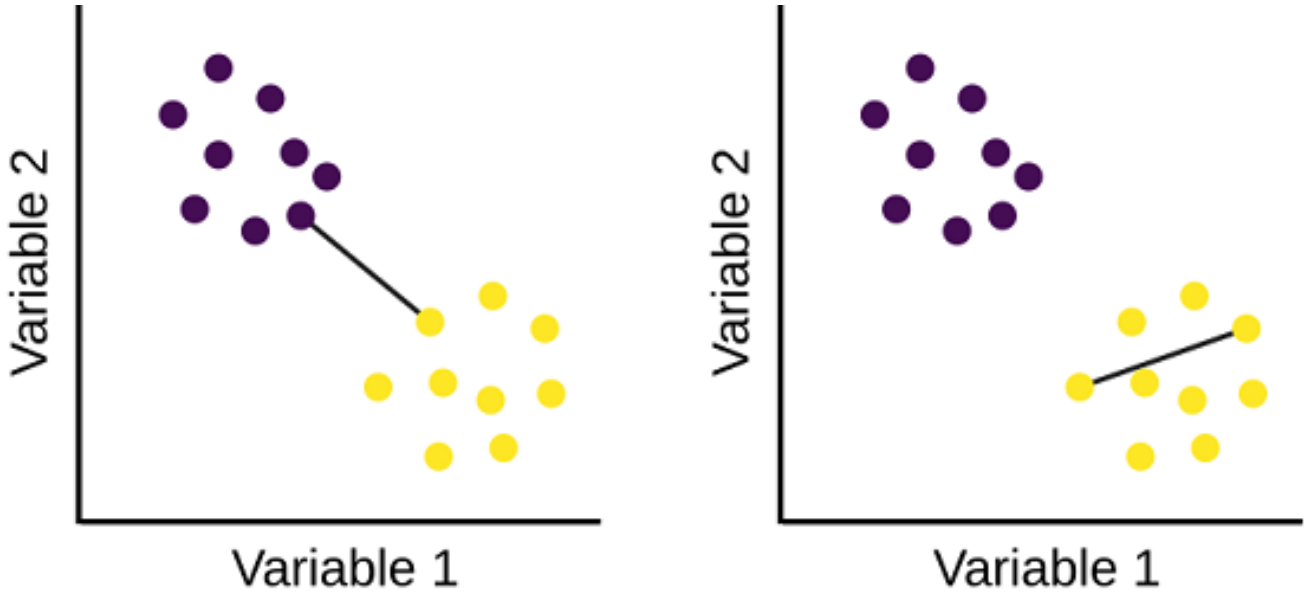
WSS의 감소 폭이 급격하게 줄어드는 지점을

최적의 클러스터 개수로 판단하는 **Elbow Method**

Relative Measure

Dunn Index

클러스터 내 최대거리에 대한 클러스터 간 최소거리의 비율



Relative Measure

Dunn Index

클러스터 내 최대거리에 대한 클러스터 간 최소거리의 비율

군집 간 거리 최소화

$$I(C) = \frac{min_{i \neq j}\{dc(C_i, C_j)\}}{max_{1 \leq l \leq k}\{\Delta(C_l)\}}$$

군집 내 거리 최대화

2 클러스터링

Relative Measure

Dunn Index

클러스터 내 최대거리에 대한 클러스터 간 최소거리의 비율

이상치의 영향을 상쇄시키기 위해 min, max

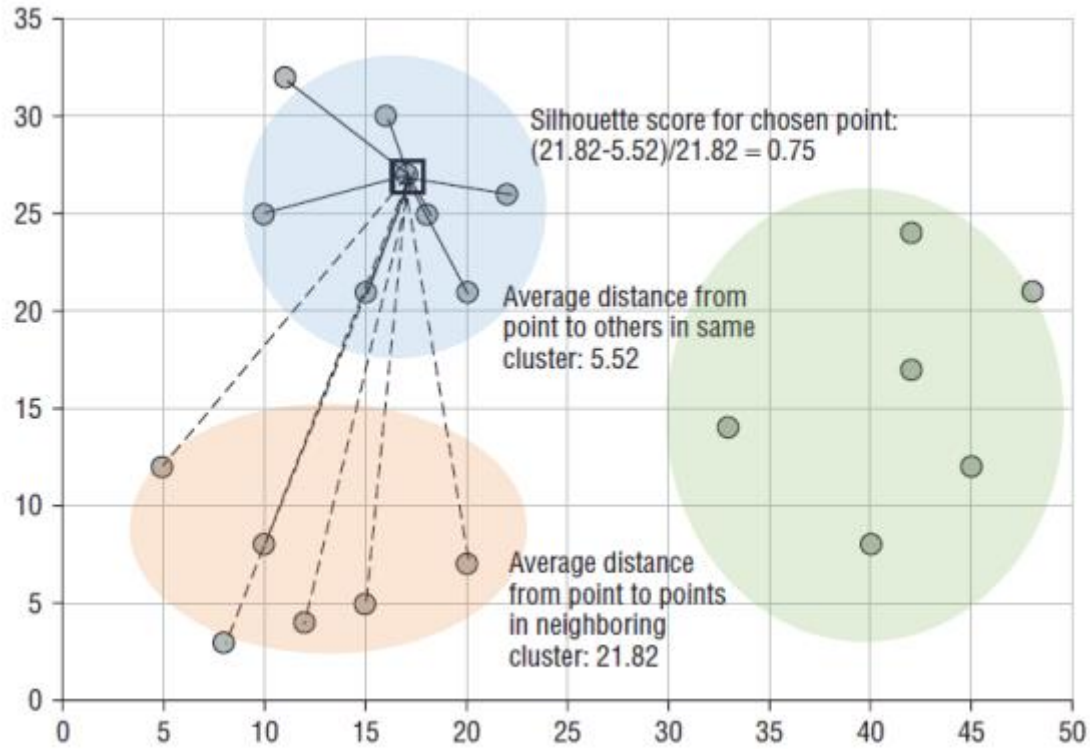
대신에 $average$를 사용하기도 함

Dunn Index 값이 클수록 클러스터링이 잘 된 것으로 판단함

Relative Measure

Silhouette Index

각 데이터 포인트마다 군집 내 거리와 군집 외 거리를 비교한 지표



Relative Measure

Silhouette Index

각 데이터 포인트마다 군집 내 거리와 군집 외 거리를 비교한 지표

$$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

$a(i)$: 객체 i와 **같은 군집** 안에 속하는 나머지 객체들 간의 거리의 평균

$b(i)$: 객체 i와 **다른 군집** 안에 속하는 나머지 객체들 간의 거리의 평균의 **최솟값**

Relative Measure

Silhouette Index

각 데이터 포인트마다 군집 내 거리와 군집 외 거리를 비교한 지표

모든 데이터 포인트에 대해서 실루엣 계수를 계산하고,

최종적으로 실루엣 계수들의 **평균값** 사용

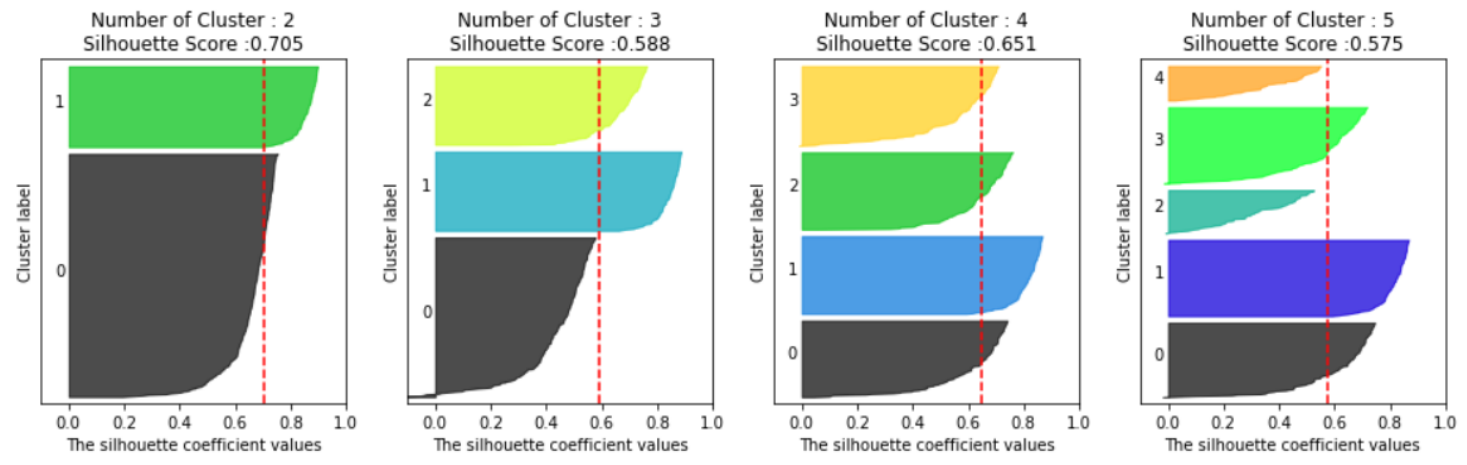
이론적인 실루엣 계수 범위는 $-1 \leq s(i) \leq 1$

1에 가까울수록 잘 묶인 군집이라고 판단함

2 클러스터링

Relative Measure

실루엣 계수가 높으면 항상 좋을까?

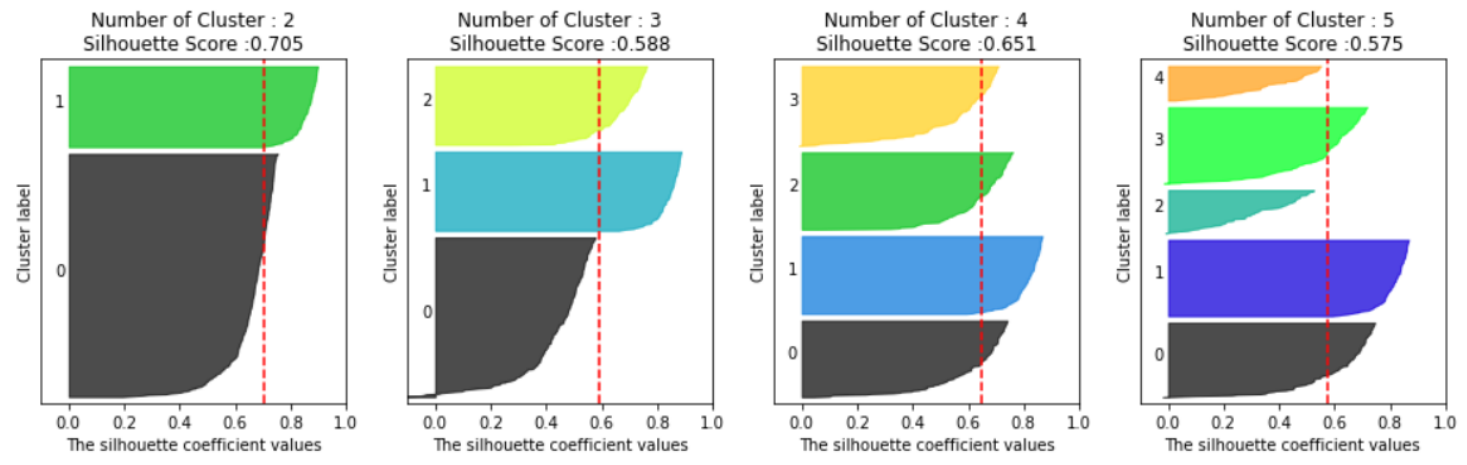


Cluster가 2개인 경우, 전체 실루엣 score는 높지만
이론적인 실루엣 계수 범위는 $-1 \leq s(i) \leq 1$ 이나
0번 군집의 대부분이 전체 score보다 낮음
경험적으로 0.5나 0.7을 넘으면 잘 묶인 군집이라고 판단함

2 클러스터링

Relative Measure

실루엣 계수가 높으면 항상 좋을까?

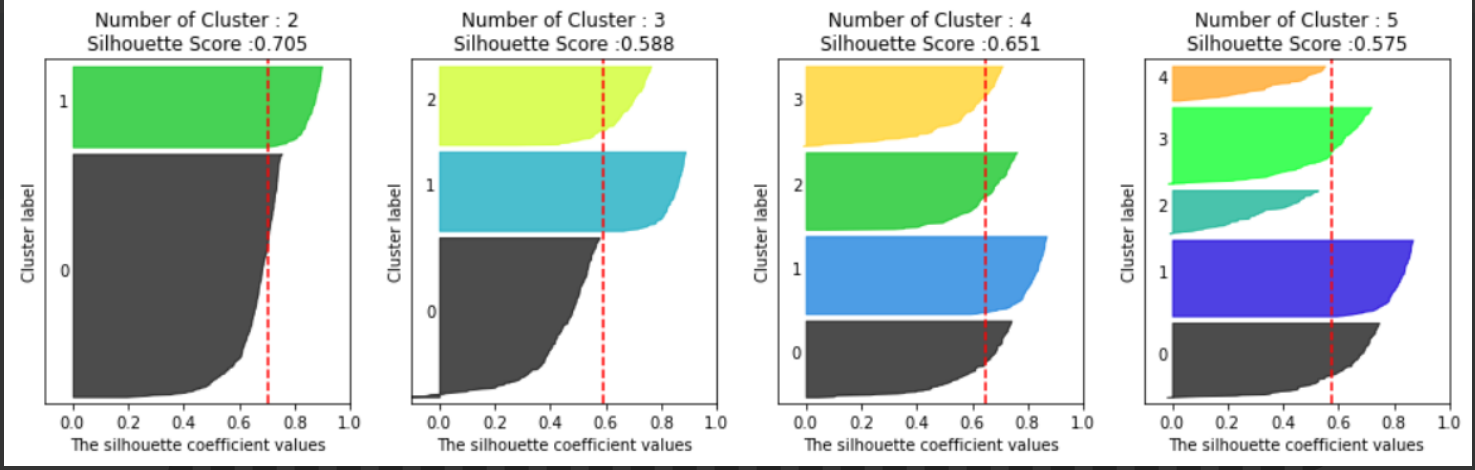


Cluster 4개인 경우, 전체 실루엣 score가 더 낮지만
이론적인 실루엣 계수 범위는 $-1 \leq s(i) \leq 1$ 이나
개별 score가 균일하게 전체 score를 웃돌고 있음
경험적으로 0.5나 0.7을 넘으면 잘 묶인 군집이라고 판단함

2 클러스터링

Relative Measure

실루엣 계수가 높으면 항상 좋을까?



실루엣 계수가 높다고 해서 가장 좋은 클러스터링이라고 단정지을 수 없음

이론적인 실루엣 계수 범위는 $-1 \leq s(i) \leq 1$ 이나

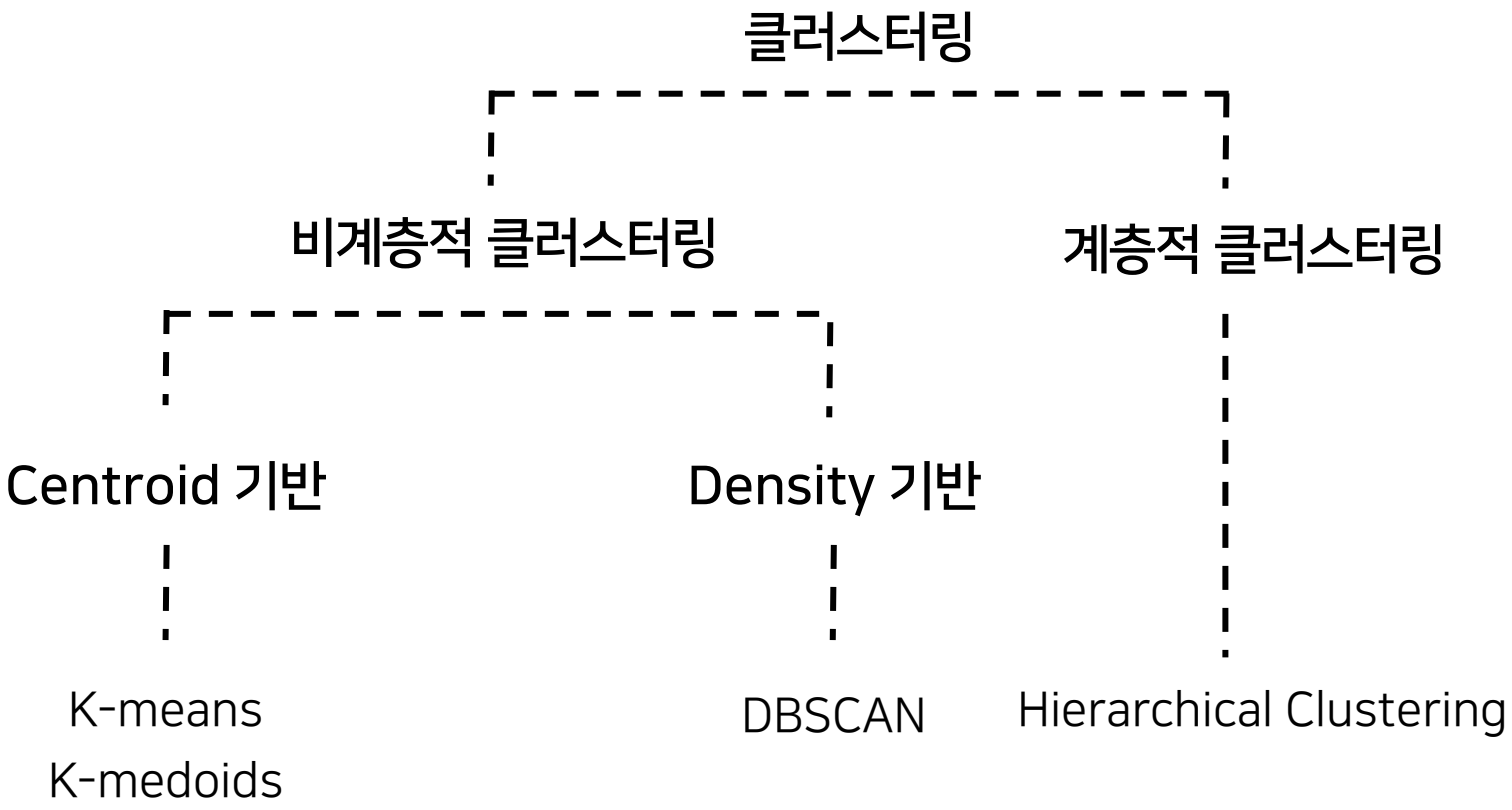
데이터가 고르게 분포하고,

경험적으로 0.5나 0.7을 넘으면 잘 묶인 군집이라고 판단함

특정 군집의 실루엣 계수만 높지 않아야 함

2 클러스터링

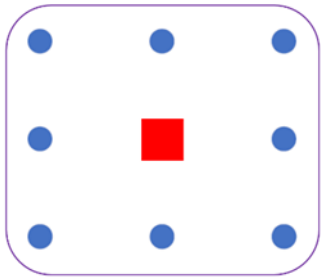
클러스터링 방법

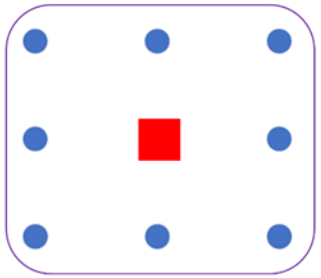


K-means Clustering

K-means Clustering

데이터들의 **평균 지점**을 Centroid로 활용하는 클러스터링



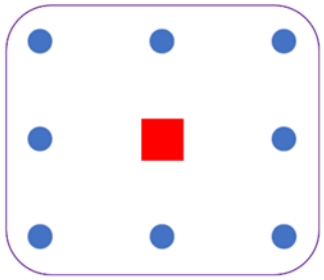


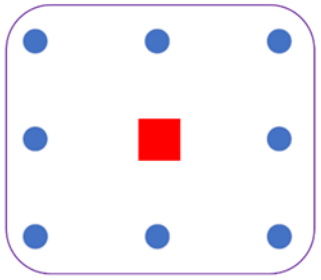
$$\arg\,min \sum_{i=1}^{K} \sum_{x_j \in C_i} \left\| X_j - C_i \right\|^2$$

K-means Clustering

K-means Clustering

데이터들의 **평균 지점**을 Centroid로 활용하는 클러스터링





Hyperparameter

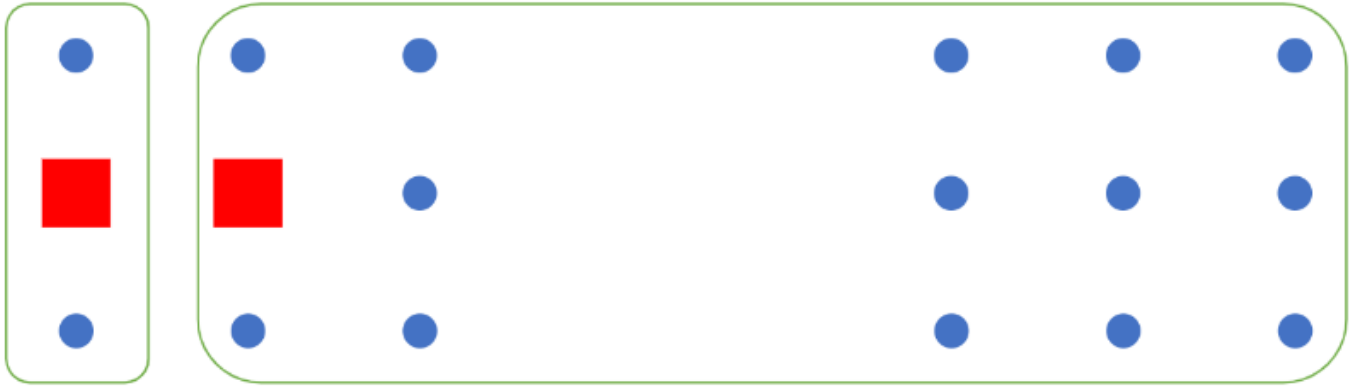
사전에 클러스터 개수 K를 지정해주어야 함

데이터들의 평균 지점이기 때문에 실제 좌표 값이 아닐 수도 있음

K-means Clustering 학습 과정

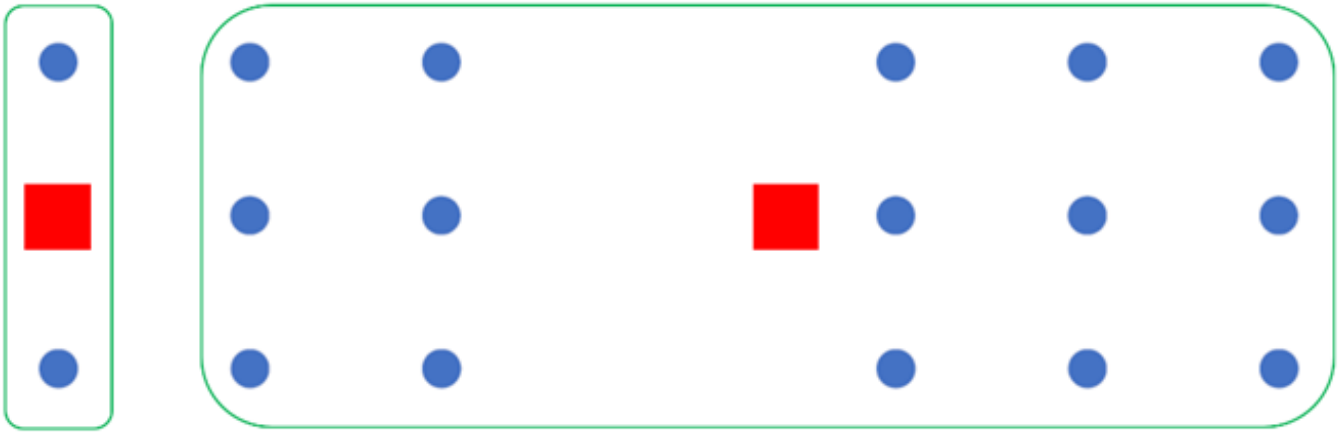
① K개의 Centroid 랜덤하게 생성함

K-means Clustering 학습 과정



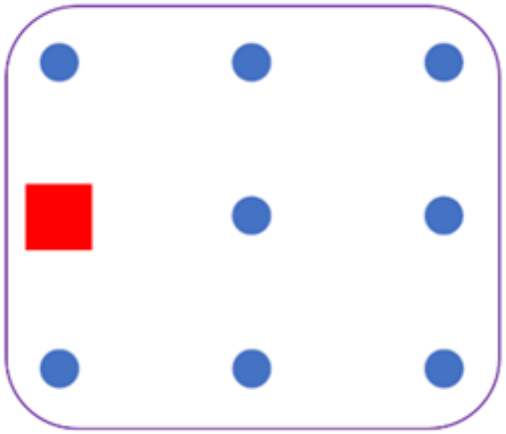
② 관측치들(파란 점)을 가장 가까운 Centroid에 맞게
1차 군집(초록 박스)으로 할당함

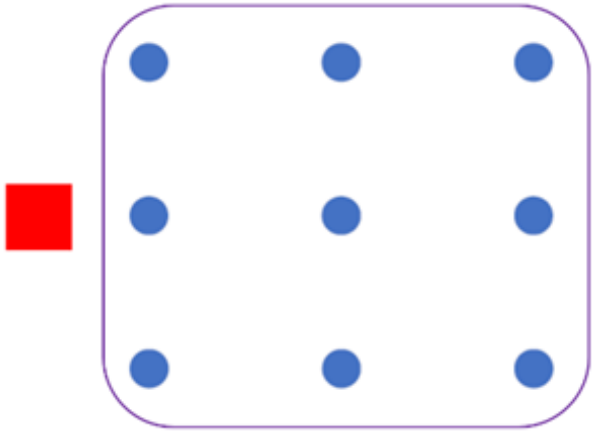
K-means Clustering 학습 과정



③ 1차 군집의 관측치들을 기준으로 Centroid를 업데이트함

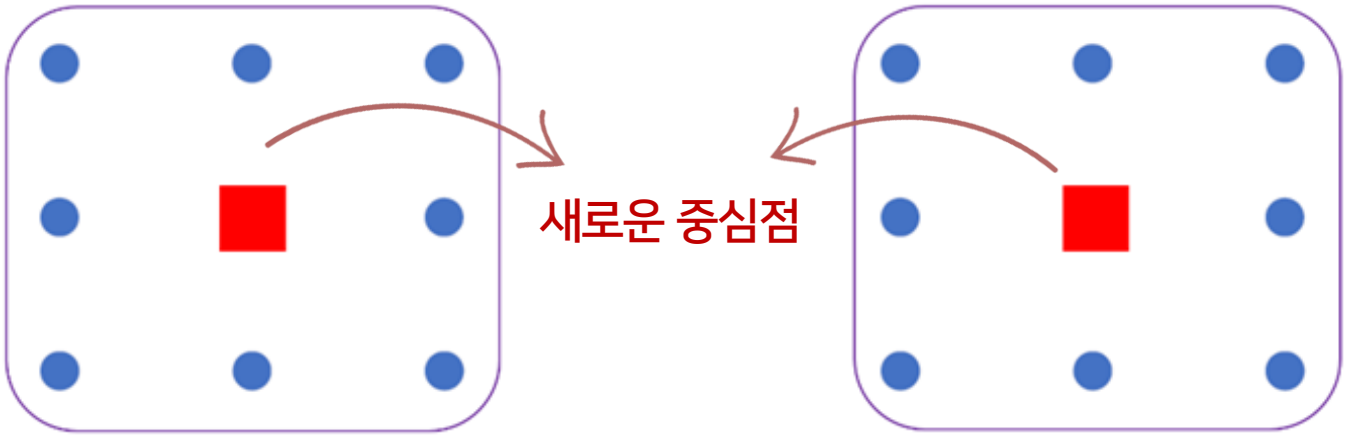
K-means Clustering 학습 과정





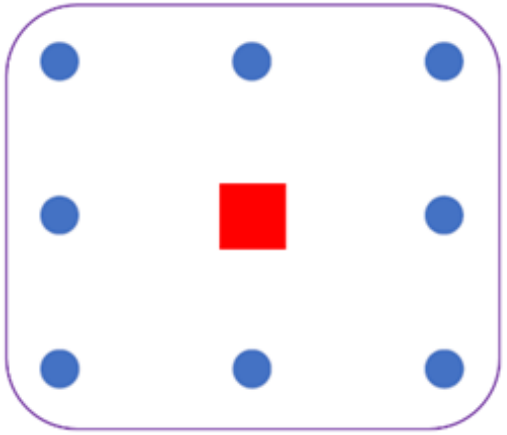
④ 관측치들을 수정된 Centroid에 맞게 2차 군집으로 할당함

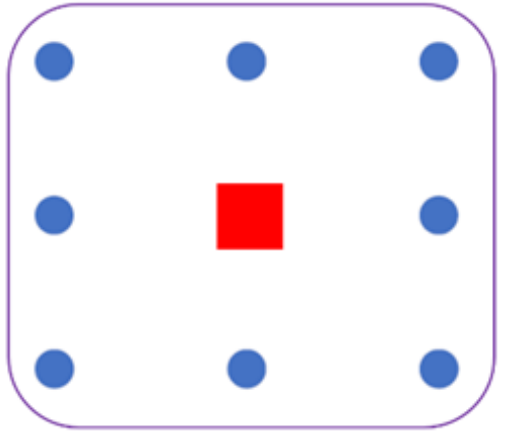
K-means Clustering 학습 과정



⑤ 2차 군집의 관측치들을 기준으로 Centroid를 다시 업데이트함

K-means Clustering 학습 과정





⑥ 앞선 과정을 반복하면서 Centroid와 Membership에
더 이상 변화가 없으면 학습을 종료함

K-means Clustering 학습 과정

K-means Clustering은 직관적이고 구현이 쉬움

그러나 **초기값에 민감**하고, **이상치의 영향**을 많이 받으며

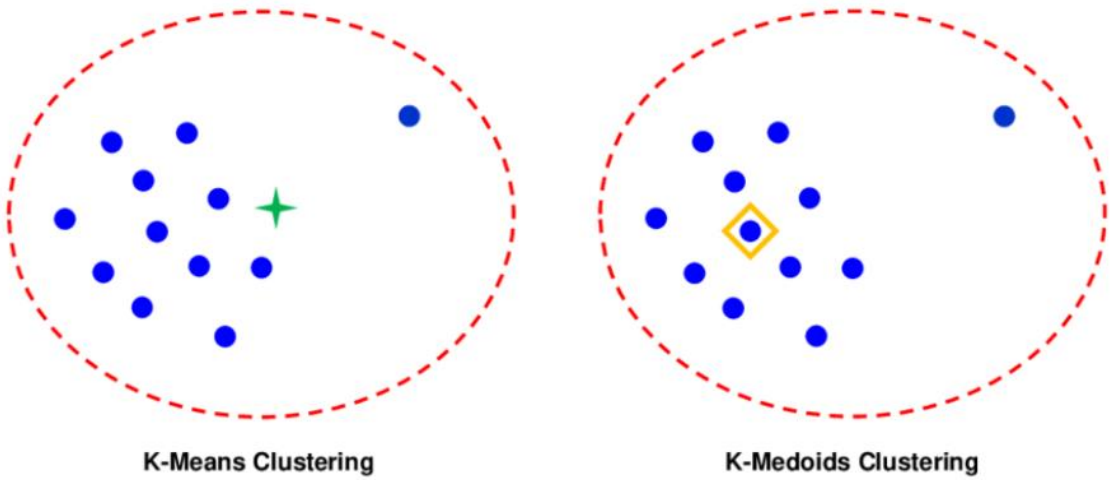
범주형 데이터에는 적용할 수 없는 단점 존재

⑥ ①~⑤을 반복하면서 Centroid와 Membership에
더이상 변화가 없으면 학습을 종료함

K-medoids Clustering

K-medoids Clustering

데이터들의 **중앙값**을 Centroid로 활용하는 클러스터링



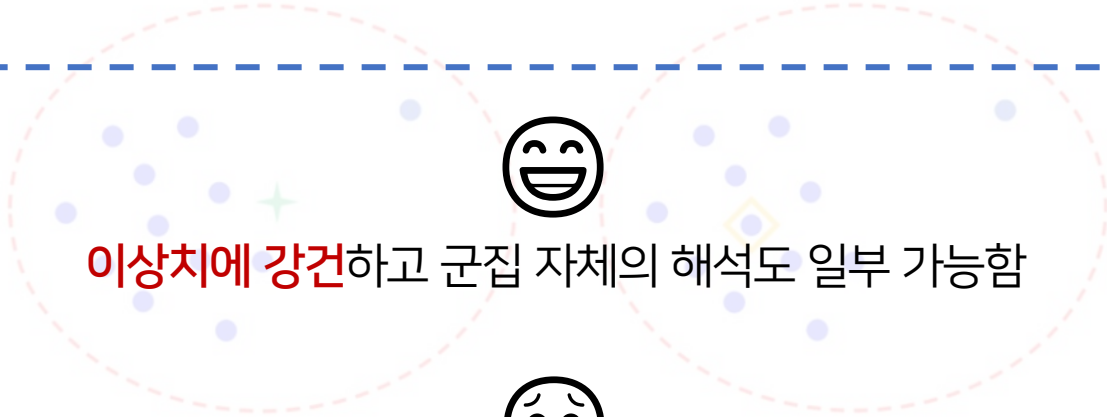
K-means와 전체적인 단계가 매우 유사하지만,

K-medoids는 **실제 포인트**를 centroid로 활용함

K-medoids Clustering

K-medoids Clustering

데이터들의 **중앙값**을 Centroid로 활용하는 클러스터링



😄

이상치에 강건하고 군집 자체의 해석도 일부 가능함

😟

그러나 실제 포인트 간 거리를 전부 계산해야 하므로 **연산량이 늘어나는 단점**

K-means와 전체적인 관계가 매우 유사하지만,
K-medoids는 **실제 포인트**를 centroid로 활용함

2 클러스터링

거리 기반 클러스터링의 한계

K-Means Clustering

데이터의 평균 지점을

Centroid로 활용

K-Medoids Clustering

데이터의 중앙값을

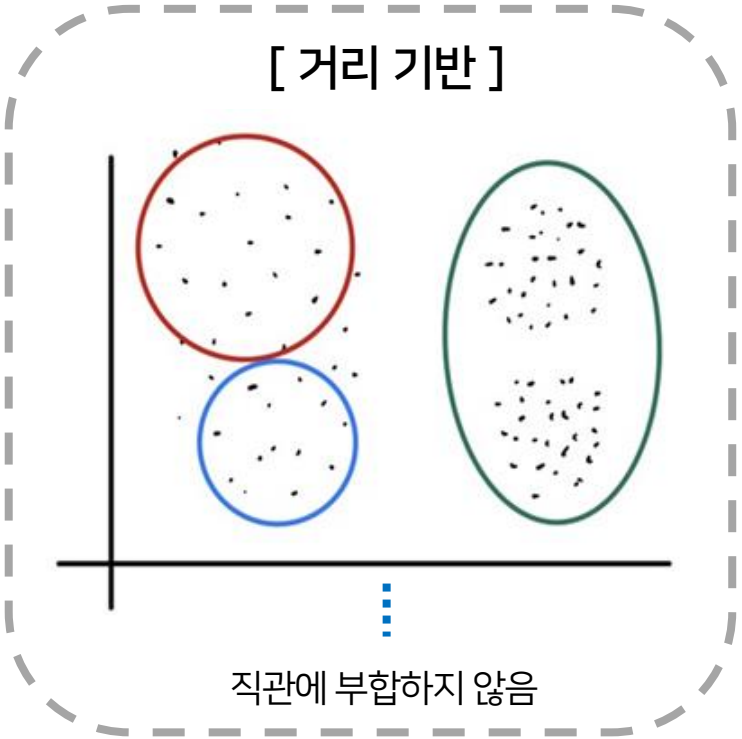
Centroid로 활용

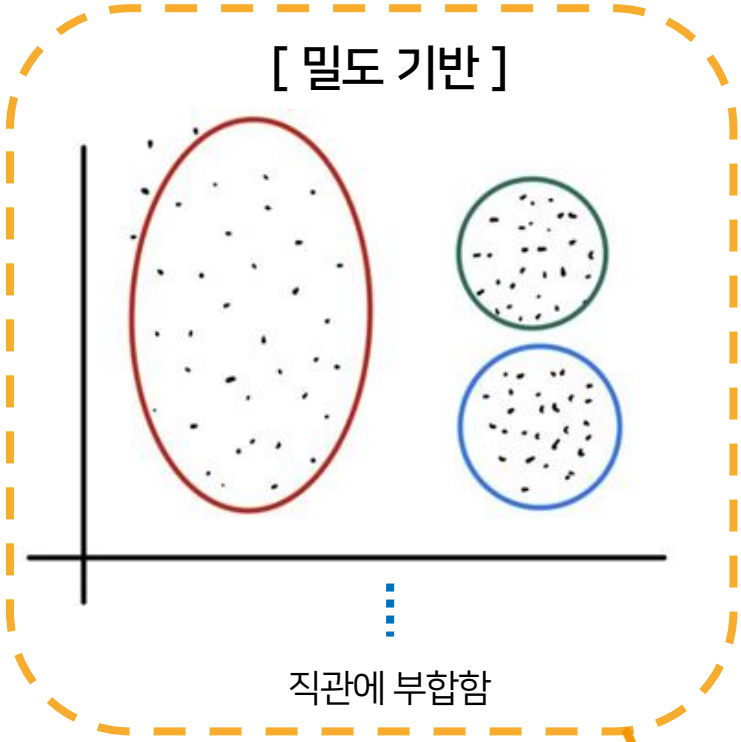
거리 기반 클러스터링은

밀도의 차이나 특수한 형태를 반영하지 못함

2 클러스터링

거리 기반 클러스터링의 한계



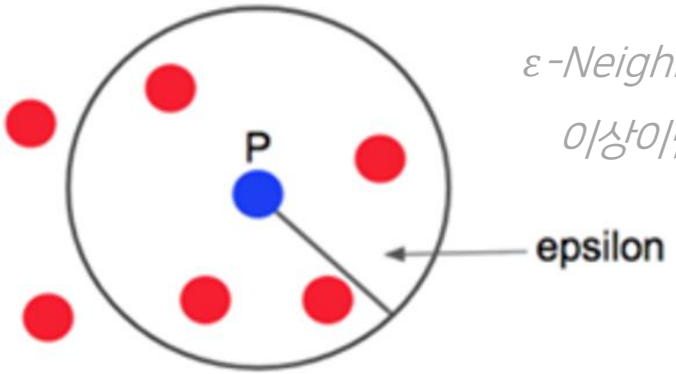


밀도 기반 클러스터링은 데이터의 **분포(뭉쳐있는 정도)를 고려**하므로 직관에 부합하는 결과를 보임

DBSCAN 용어 정리

ε-Neighborhood (of a point p)

$$N_\epsilon(p) = \{q \in D \mid dist(p,q) \leq \epsilon\}$$



ε-Neighborhood가 최소 minPts개 이상이면 그것들을 군집으로 묶음

p와의 거리가 ε보다 작은 점들

즉, p를 기준으로 반경 ε내에 있는 모든 점들

DBSCAN 용어 정리

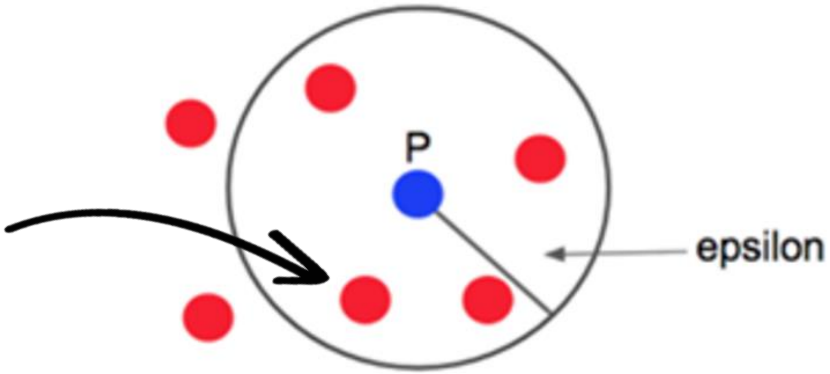
ε-Neighborhood (of a point p)

$$N_\epsilon(p) = \{q \in D \mid dist(p,q) \leq \epsilon\}$$

minPts

반경 내

점의 최소 개수



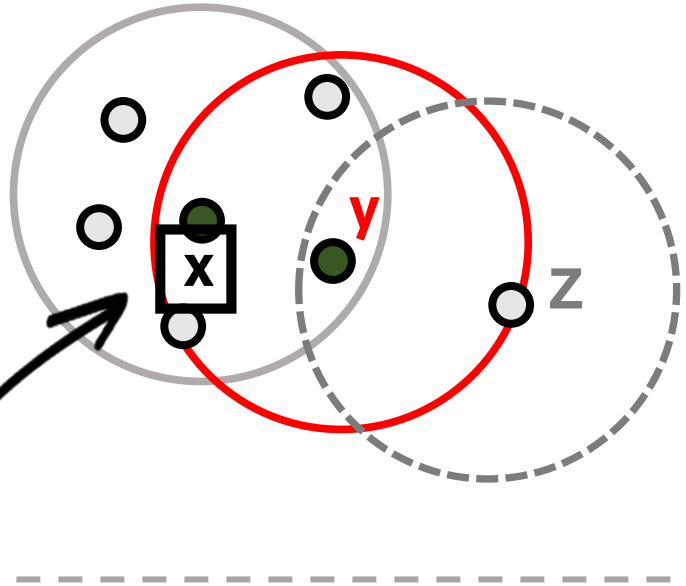
엡실론(ε)

점 p와의 거리

이때, minPts와 엡실론(ε)은 사용자가 지정

DBSCAN 용어 정리

Core point

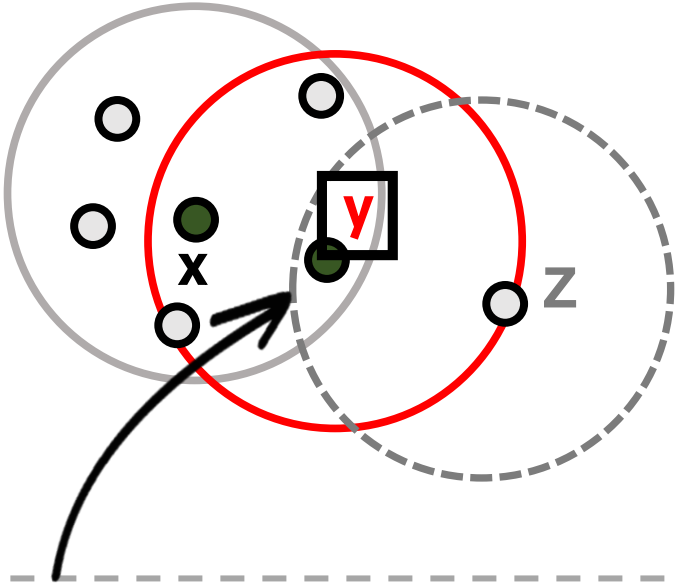


엡실론(ε) 반경 내에 minPts개 이상의 점이 존재하는 점

나중에 군집을 이루는 중심이 됨

DBSCAN 용어 정리

Border point

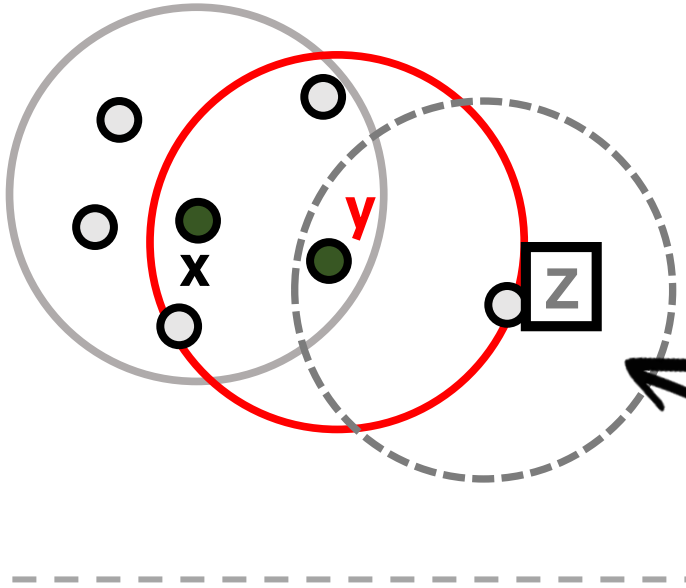


엡실론(ε) 반경 내에 minPts개 미만의 점을 갖는 점들 중,

Core point의 엡실론(ε) 반경 내에 포함되는 점

DBSCAN 용어 정리

Noise point

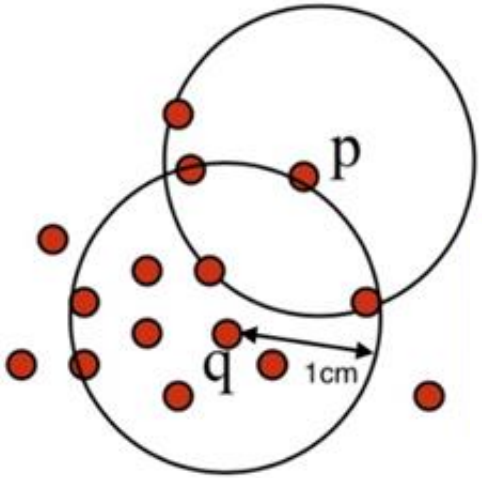


Core도 아니고 Border도 아닌 점

즉, 주변에 minPts개 이하의 점을 가지고 주변에 Core point도 없는 경우

DBSCAN 용어 정리

Directly Density-reachable



MinPts = 5

Eps = 1cm

점 p가 q로부터 밀도 관점에서 직접적으로 접근 가능

⋮

2가지 조건을 만족해야 함

DBSCAN 용어 정리

Directly Density-reachable

Reachability

$$p \in N_\varepsilon(q)$$

p가 q의 ε-Neighborhood에 속해야 함

Core point condition

$$|N_\varepsilon(q)| \geq MinPts$$

q의 ε-Neighborhood 개수는 $MinPts$ 이상이어야 함

점 p가 q로부터 밀도 관점에서 직접적으로 접근 가능

2가지 조건을 만족해야 함

DBSCAN 용어 정리

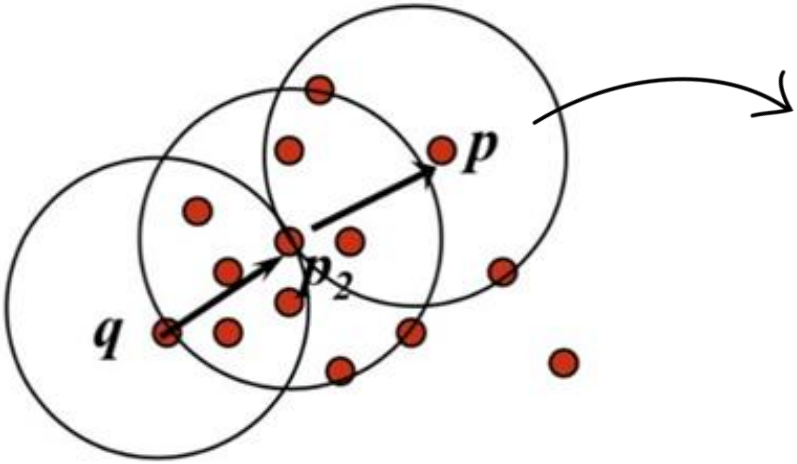
Directly Density-reachable

만약 D.D.R.(직접밀도접근가능)한 점들만 한 군집으로 묶으면,

비교적 바깥쪽에 있는 점들을 놓칠 수 있음

DBSCAN 용어 정리

Density-reachable



p가 q의 반경 밖에 있어도,

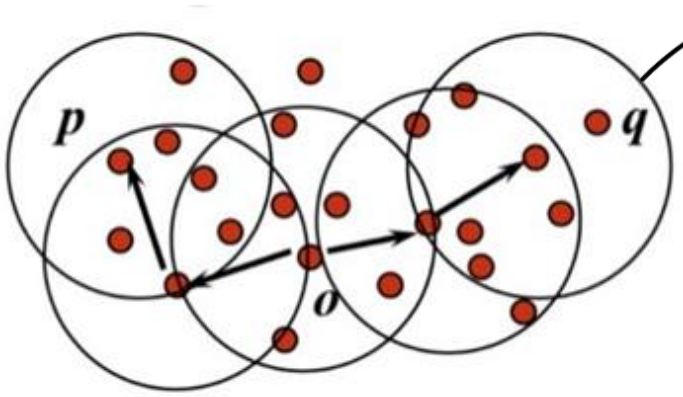
그 사이의 점들을 D.D.R.로 연결 가능

= p가 q로부터 D.R.함

점 p가 q로부터 밀도 관점에서 접근 가능

DBSCAN 용어 정리

Density-connected



p가 O로부터 D.R.하고

q도 O로부터 D.R.함

= p가 q로부터 D.C.함

⋮

즉, 같은 군집에 할당된 점들은

모두 Density-connected된 것

점 p가 q로부터 밀도 관점에서 연결되어 있음

2 클러스터링

DBSCAN 학습 과정

① 임의의 데이터 포인트 선택

② p로부터 Density-reachable한 포인트 탐색

- p가 Core point면 클러스터에 할당
- p가 Border point면 다른 point 선택

③ 모든 데이터 포인트가 탐색될 때까지 반복

DBSCAN

DBSCAN의 장점

특정한 모양의 클러스터를 찾아낼 수 있음 (형태의 한정이 없음)

군집에 할당되지 않는 객체도 허용함
즉, 이상치(Noise)를 군집에 할당하지 않아도 됨

클러스터링의 랜덤성이 작음

클러스터 개수(K)를 지정해주지 않아도 됨

2 클러스터링

DBSCAN

DBSCAN의 한계

DBSCAN의 장점

엡실론(ε)과 minPts를 설정하기가 어려움

특히, 데이터셋 내에서 밀도가 제각각이라면 더욱 힘들어짐

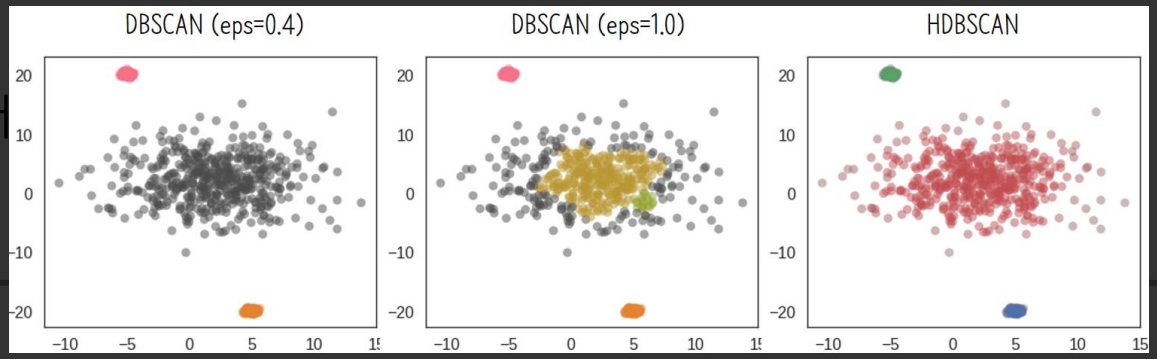
군집에 할당되지 않는 객체도 허용함

즉, 이상치(Noise)를 군집에 할당하지 않아도 됨

클러스터링의 랜덤성이 적음

클러

HDBSCAN (Hierarchical DBSCAN)

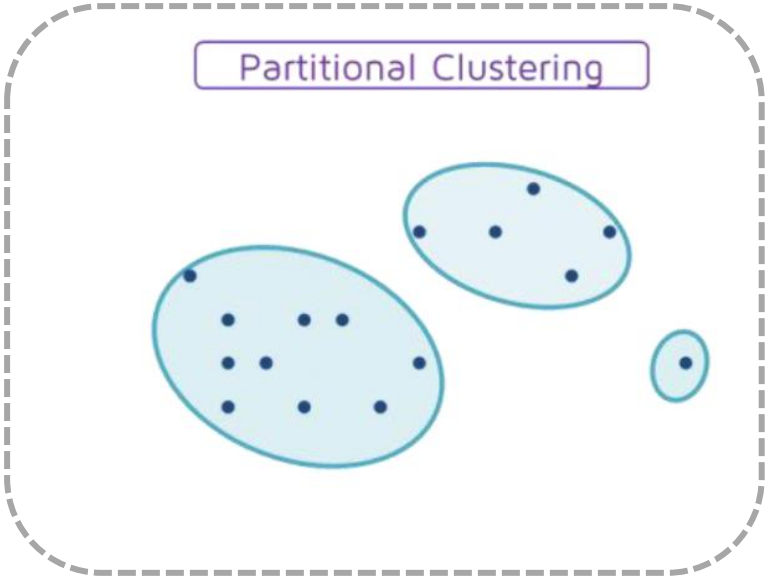


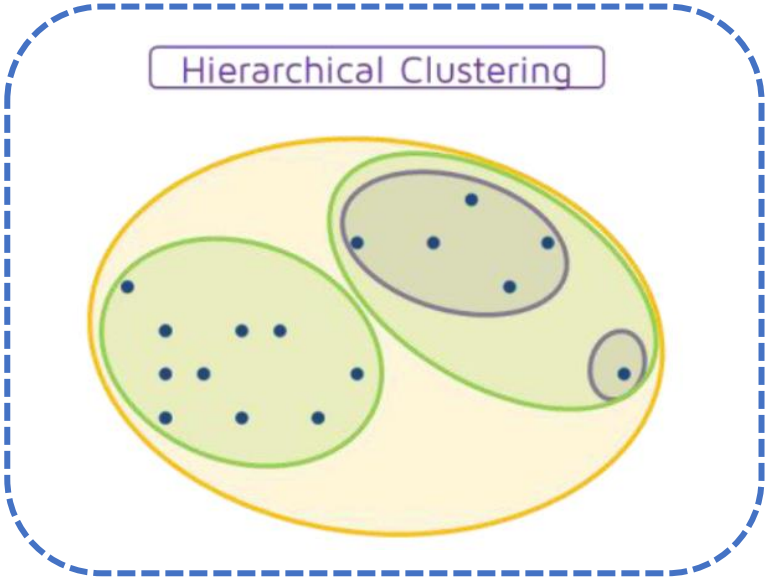
계층적 클러스터링

계층적 클러스터링 (Hierarchical clustering)

계층이 존재하는 트리를 이용하여 개별 개체들을

Bottom-up 방식으로 묶어나가는 클러스터링 방식





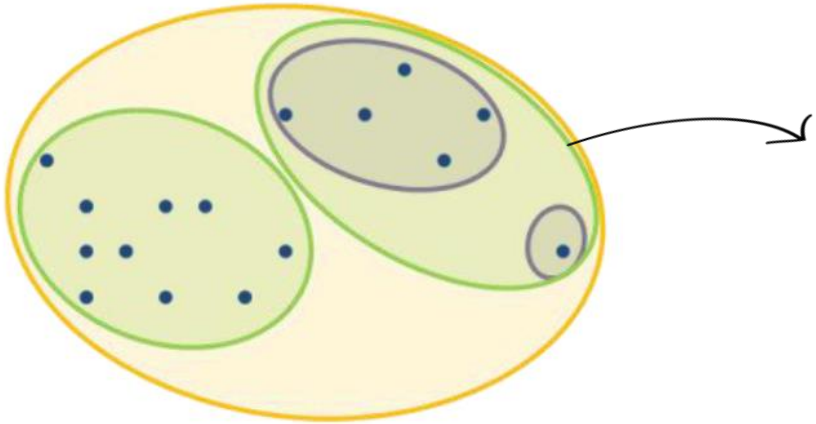
2 클러스터링

계층적 클러스터링

계층적 클러스터링의 장점

클러스터의 개수를 **지정해주지 않아도** 학습이 가능

Ex)



초록색 군집 2개를 선택해도 되고,

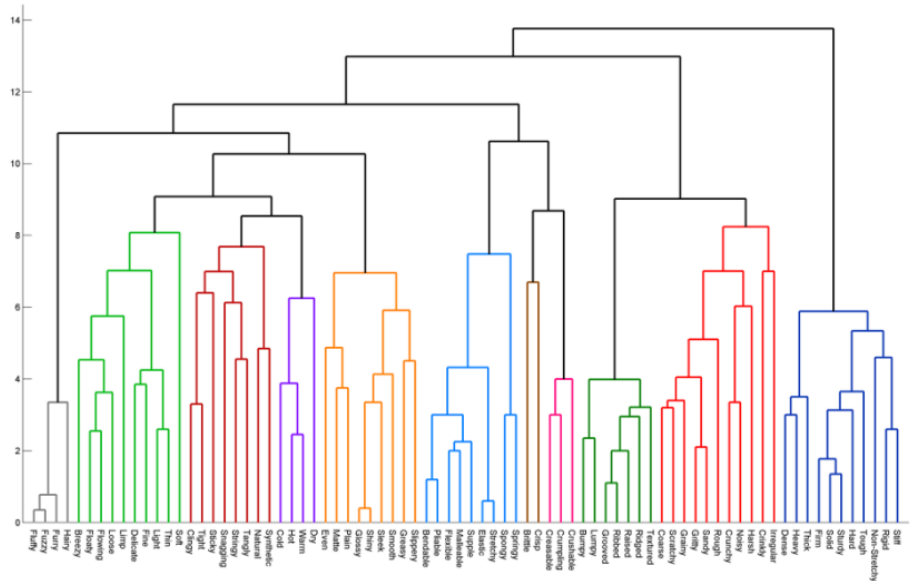
보라색 군집 3개를 선택해도 됨

2 클러스터링

계층적 클러스터링

계층적 클러스터링의 장점

클러스터링 과정을 **덴드로그램**으로 시각화 가능



2 클러스터링

계층적 클러스터링

계층적 클러스터링은 **가장 가까운 두 객체**를 순차적으로 병합해나가는 방식

즉, **거리**가 계산되어 있어야 함

Affinity

개체 간 거리 측정 방식

Linkage

군집 간 거리 측정 방식

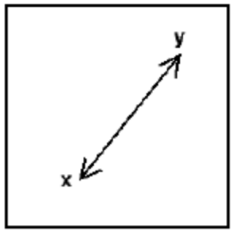
2 클러스터링

Affinity (개체 간 거리 측정)

유클리드 거리

두 관측치 사이의
직선 최단거리

Ex)

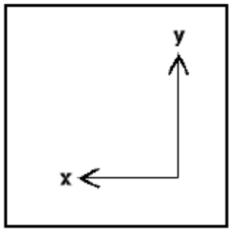


Euclidean

맨하탄 거리

각 좌표축의
방향으로만 이동

Ex)

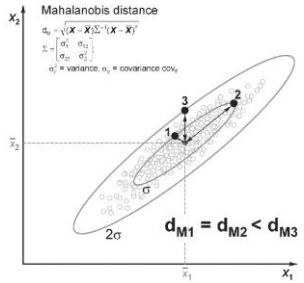


Manhattan

마할라노비스 거리

주변 데이터의
맥락을 반영한 거리

Ex)



2 클러스터링

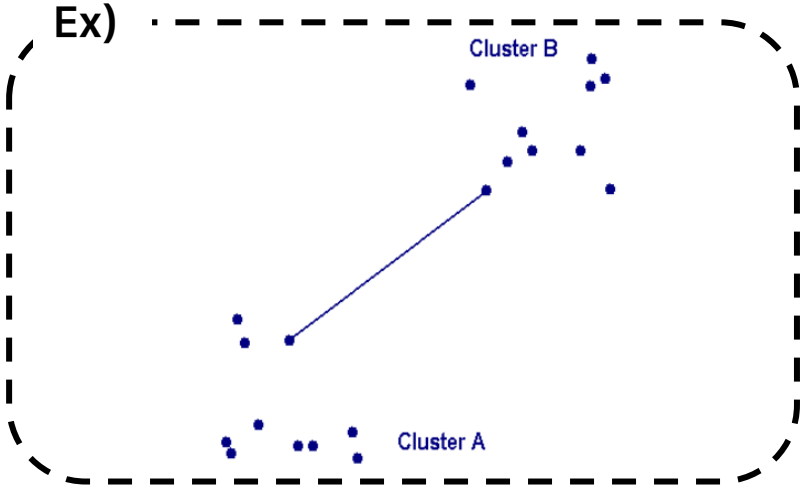
Linkage (군집 간 거리 측정)

Single linkage (min)

서로 다른 군집의

모든 데이터 간 거리 중 **최소값**

Ex)

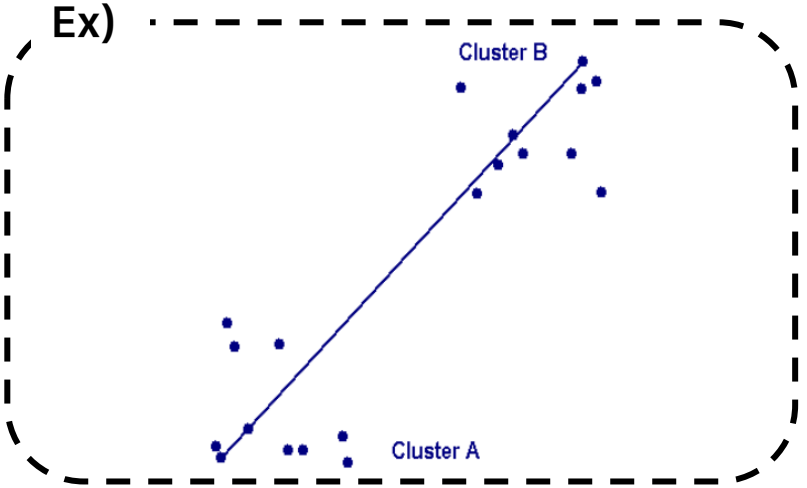


Complete linkage (max)

서로 다른 군집의

모든 데이터 간 거리 중 **최대값**

Ex)



2 클러스터링

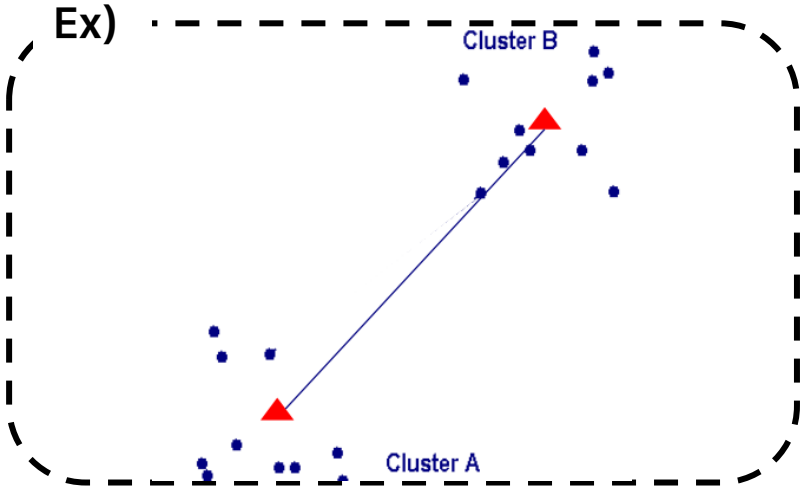
Linkage (군집 간 거리 측정)

Centroid linkage

서로 다른 군집의

중심 점(centroid) 간 거리

Ex)

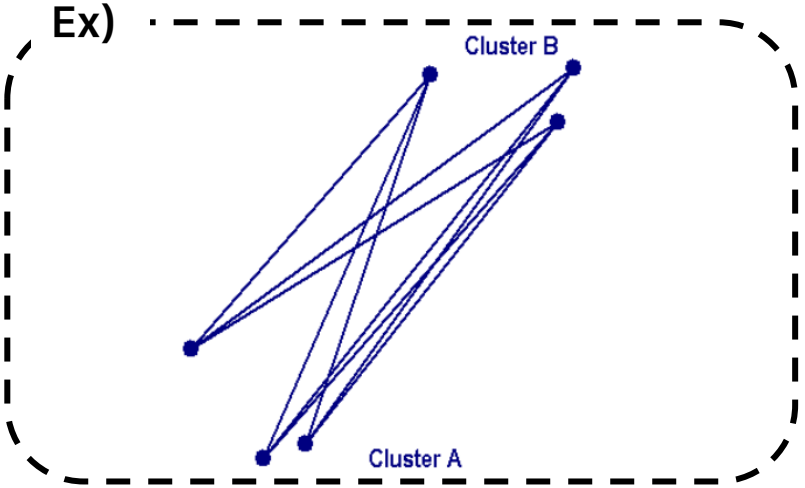


Average linkage

서로 다른 군집의

모든 데이터 간 **거리들의 평균**

Ex)



2 클러스터링

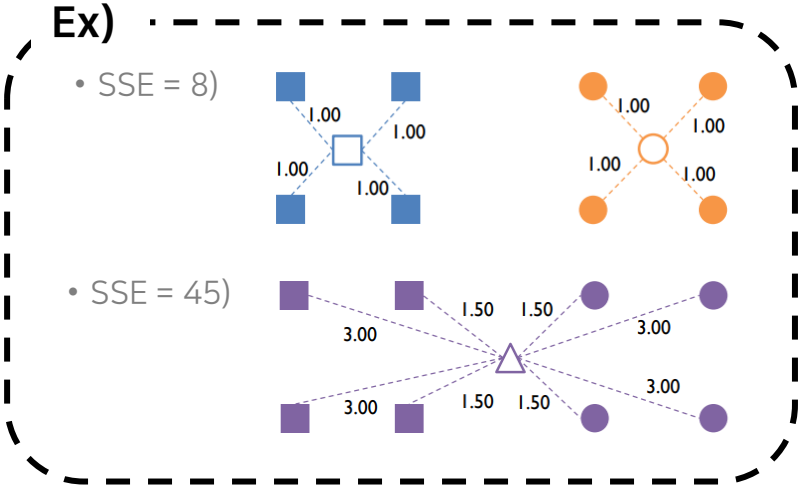
Linkage (군집 간 거리 측정)

Ward method

서로 다른 군집의

병합 전 후 분산 차이

Ex)



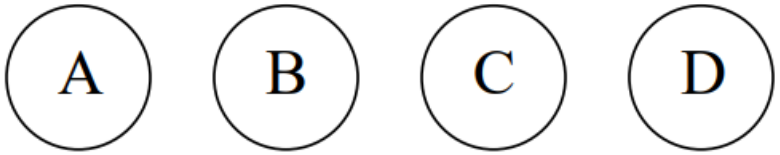
계층적 클러스터링의 프로세스를 알아보고,

덴드로그램도 함께 그려보자!

2 클러스터링

계층적 클러스터링

	A	B	C	D
A		20	7	2
B	20		10	25
C	7	10		3
D	2	25	3	



① 각 관측치(또는 군집) 간의 거리를 모두 계산

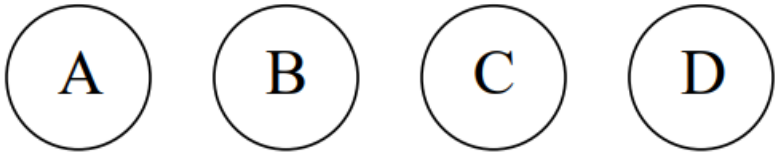
주로 거리 행렬(Distance Matrix) 사용

2 클러스터링

계층적 클러스터링

	A	B	C	D
A		20	7	2
B	20		10	25
C	7	10		3
D	2	25	3	

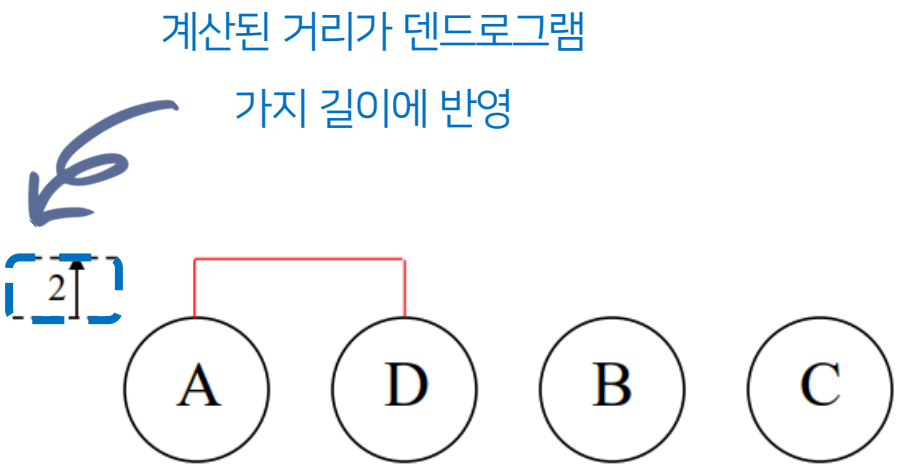
A와 D 병합



② 계산된 거리가 가장 가까운 두 관측치(또는 군집) 병합

계층적 클러스터링

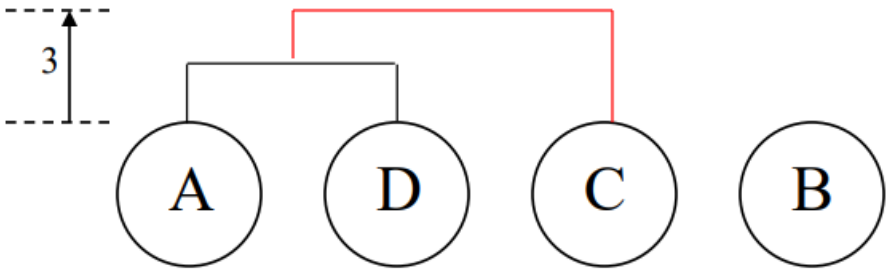
	AD	B	C
AD			
B			
C			



② 계산된 거리가 가장 가까운 두 관측치(또는 군집) 병합

계층적 클러스터링

	AD	B	C
AD		20	3
B	20		10
C	3	10	

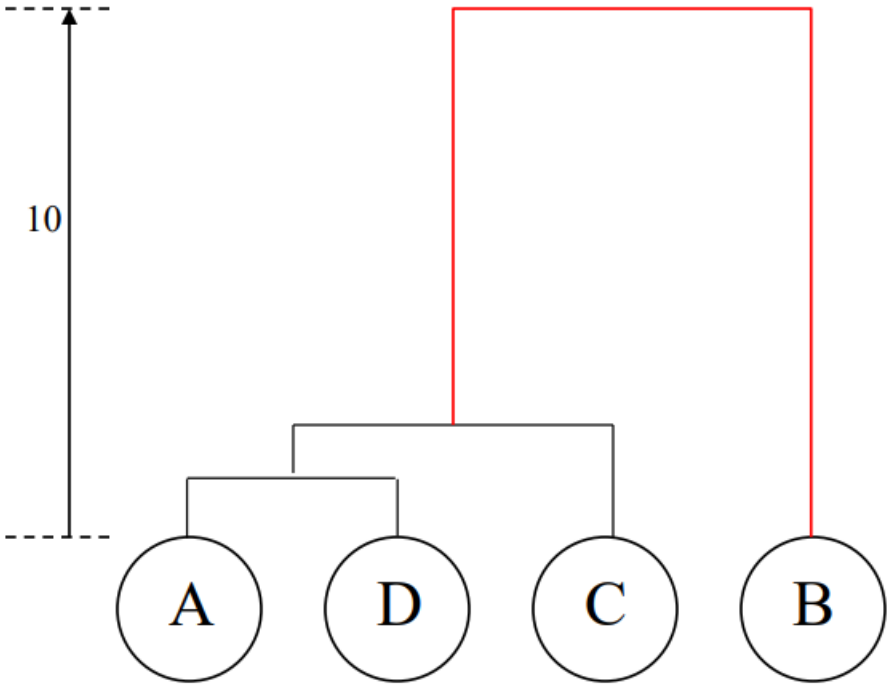


③ 병합 이후 Distance Matrix를 업데이트

2 클러스터링

계층적 클러스터링

	ADC	B
ADC		10
B	10	



④ 클러스터가 1개만 남을 때까지 ①~③ 반복

2 클러스터링

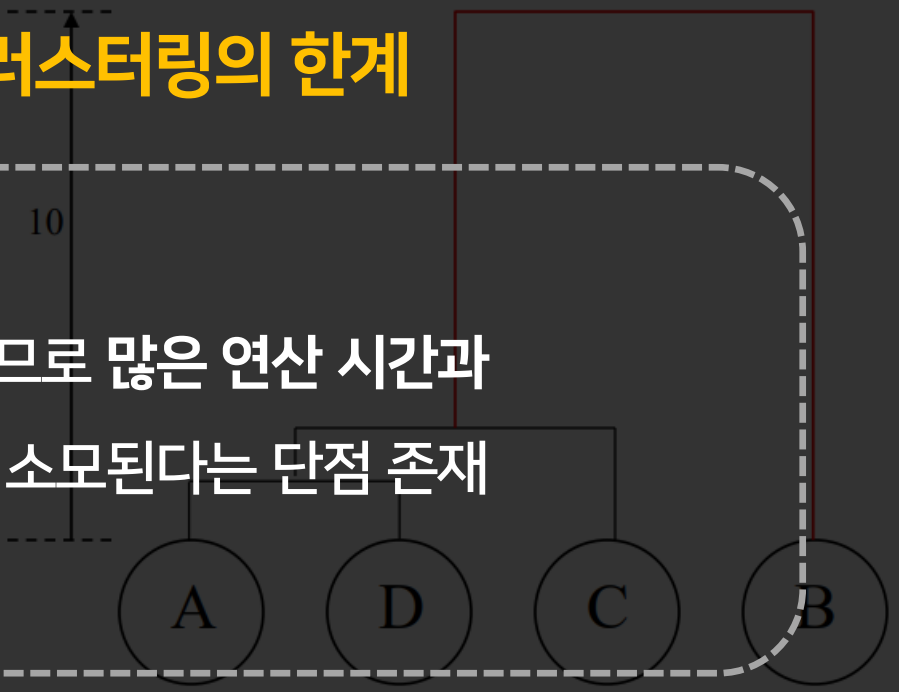
계층적 클러스터링

계층적 클러스터링의 한계

	ADC	B
ADC		10
B	10	

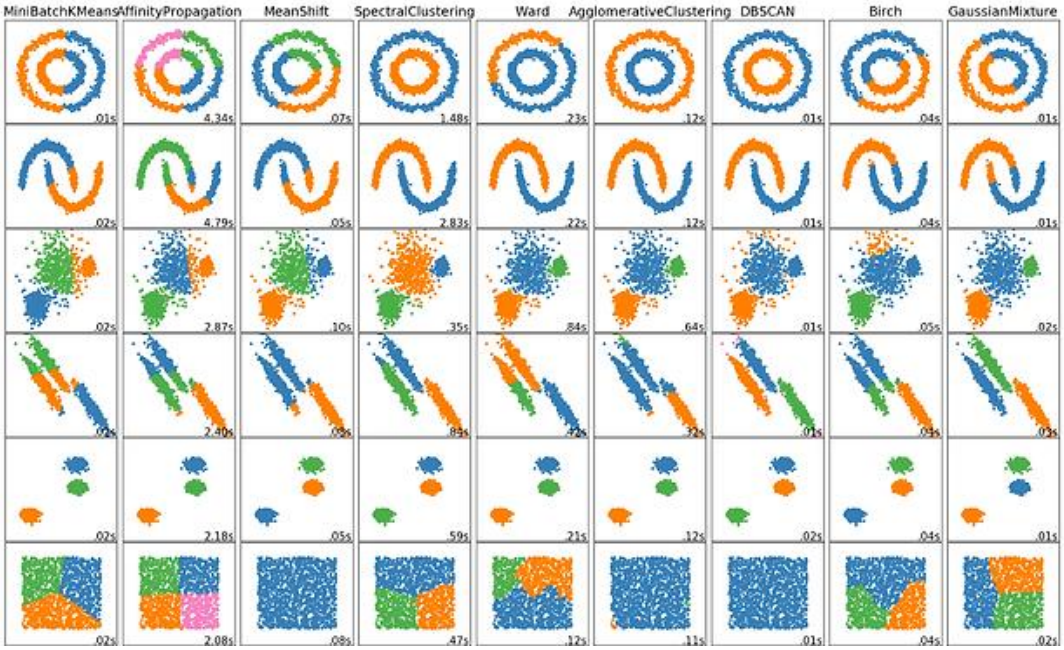
계산량이 많으므로 **많은 연산 시간과**

컴퓨팅 파워가 소모된다는 단점 존재



④ 클러스터가 1개만 남을 때까지 ①~③ 반복

그 외 클러스터링



그 외에도 Mean shift, GMM, Spectral 등

다양한 클러스터링 방법이 존재함

3

비선형 모델

3 모델링 전략

비선형 모델 (Non-Linear Model)

선형 모델의 한계

흔히 접하는 선형모델은 해석과 추론 측면에서 강점이 있지만
현실 데이터와 잘 맞지 않아 예측 성능이 제한적임

비선형 모델

선형성을 완화하여 예측 성능을 높이면서
해석력은 최대한 유지하는 것을 목표로 함

비선형 모델 (Non-Linear Model)

비선형 모델의 Key Idea

Input matrix에 해당되는 X를 비선형적으로 **변환(transform)함**

$$X \Rightarrow X^*$$

변환된 Input matrix X*와 Y를 **Linear**한 모델로 학습!

$$Y = \beta_0 + \beta_1 * {X_1}^* + \beta_2 * {X_2}^* + \cdots + \beta_n * {X_n}^*$$

비선형 모델 (Non-Linear Model)

비선형 모델의 Key Idea

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X)$$

Input matrix에 해당하는 X를 비선형적으로 변환(transform)함

이때, 선형결합이 가능한 형태로 transform 해주는 것이

기저함수 (Basis Function)

Linear Basis Expansion of X

변환된 Input matrix X*와 Y를 Linear한 모델로 학습!

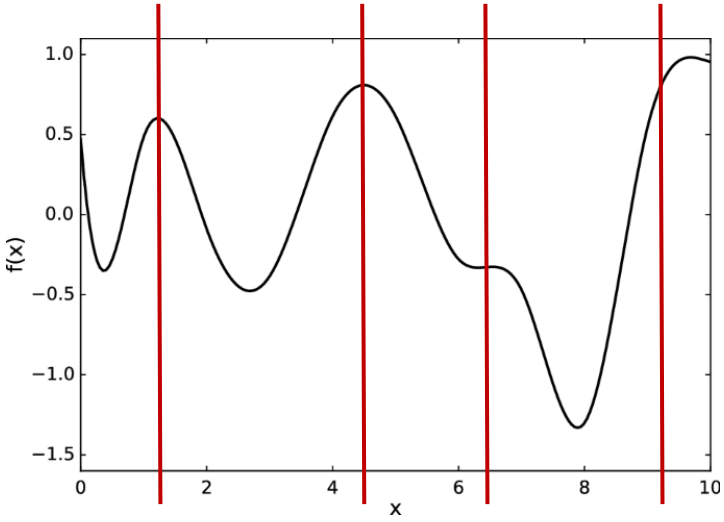
$$Y = \beta_0 + \beta_1 * X_1^* + \beta_2 * X_2^* + \cdots + \beta_n * X_n^*$$

기저함수(Basis Function) 기반 모델

Piecewise Polynomials

설명변수의 공간을 매듭점(knot)으로 분할하고,

각 공간에 다항 함수를 추정하는 회귀 방법

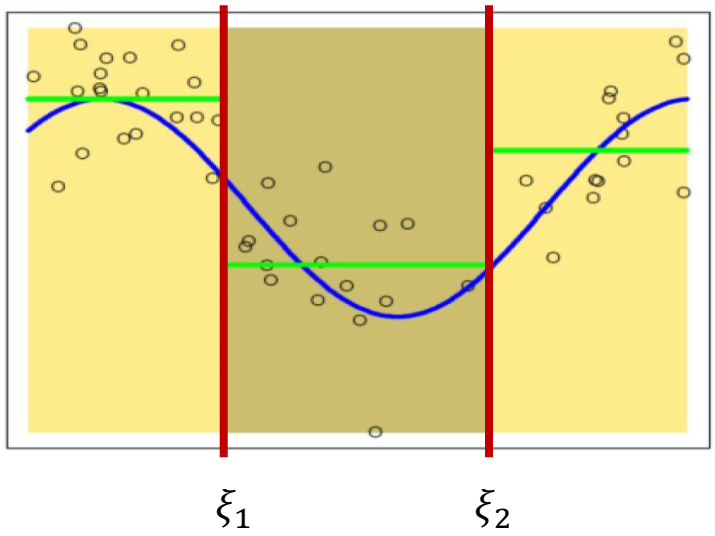


기저함수(Basis Function) 기반 모델

Piecewise Constant Model

Knot으로 분할된 각 공간에 Constant 함수를 적합한 경우

$$h_1(x) = I(X < \xi_1), h_2(x) = I(\xi_1 < X < \xi_2), h_3(x) = I(X \geq \xi_2)$$



$$f(X) = \sum_{m=1}^{3} \beta_m h_m(X)$$

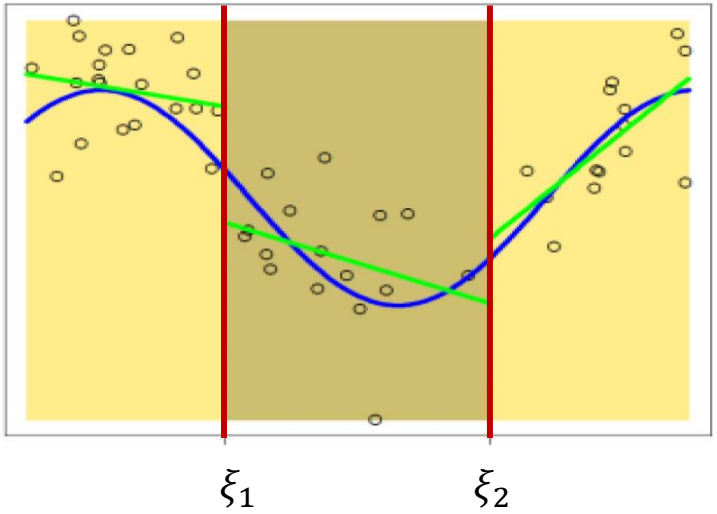
기저함수(Basis Function) 기반 모델

Piecewise Linear Model

Knot으로 분할된 각 공간에 Linear 함수를 적합한 경우

$$\beta_1 I(x < \xi_1) \quad \beta_2 I(\xi_1 \le \mathrm{x} < \xi_2) \quad \beta_3 I(\xi_2 \le \mathrm{x})$$

$$\beta_4 XI(x < \xi_1) \quad \beta_5 XI(\xi_1 \le \mathrm{x} < \xi_2) \quad \beta_6 XI(\xi_2 \le \mathrm{x})$$

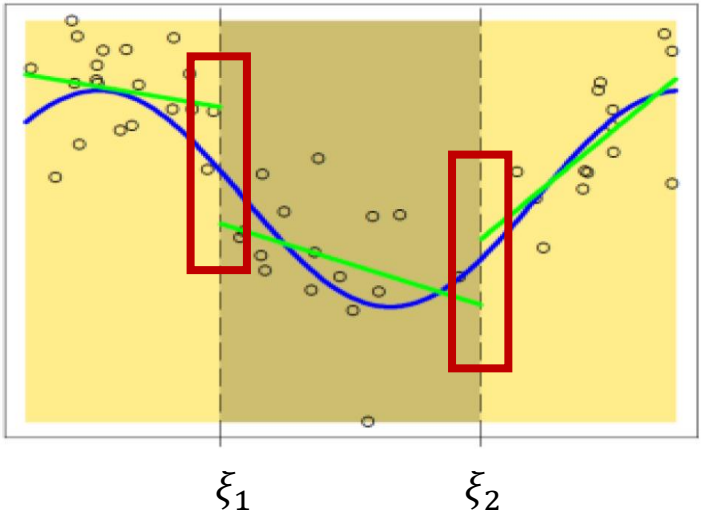


$$f(X) = \sum_{m=1}^{6} \beta_m h_m(X)$$

기저함수(Basis Function) 기반 모델

Knot 지점에서 **불연속성(Discontinuity)** 문제 발생

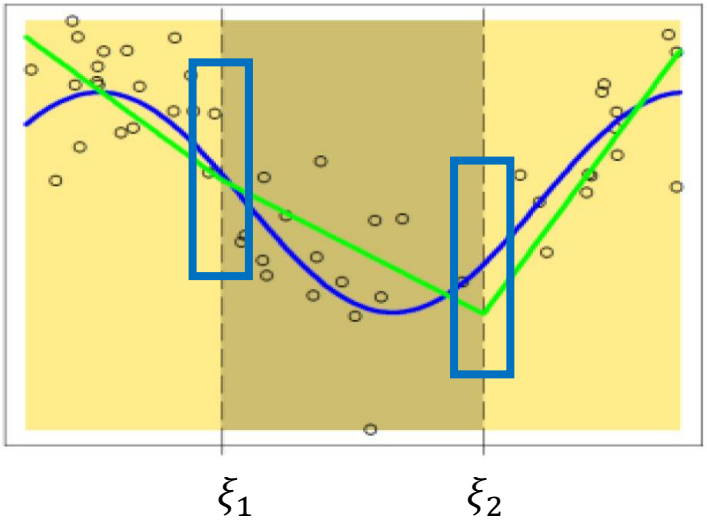
합리적인 예측이 이루어질 수 없음



기저함수(Basis Function) 기반 모델

Knot 기준으로 좌극한과 우극한이 같도록

만들어주는 제약식 설정



$$f(\xi_1^-) = f(\xi_1^+)$$
$$f(\xi_2^-) = f(\xi_2^+)$$

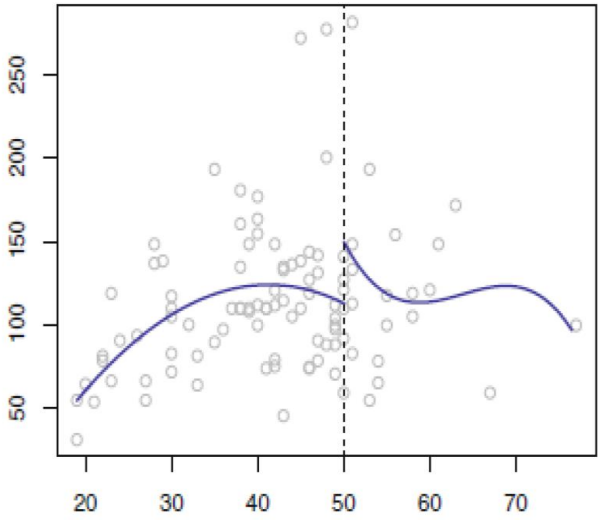
제약식에 의해 연속성 확보!

Spline Regression

Cubic Spline

Piecewise 모델의 일종으로, 각 공간에 **3차 다항함수**를 적합한 경우

$$y_i = \begin{cases} \beta_{01}+\beta_{11}x_i+\beta_{21}x_i^2+\beta_{31}x_i^3+\epsilon_i & if\ x_i < c; \\ \beta_{02}+\beta_{12}x_i+\beta_{22}x_i^2+\beta_{32}x_i^3+\epsilon_i & if\ x_i \geq c \end{cases}$$



3 모델링 전략

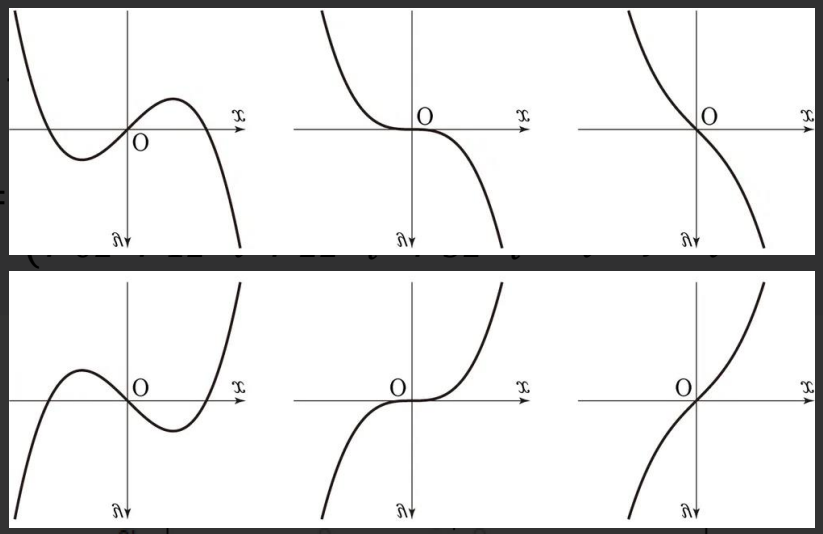
Non-linear Spline

왜 하필 3차 다항식을 사용할까?

Cubic Spline

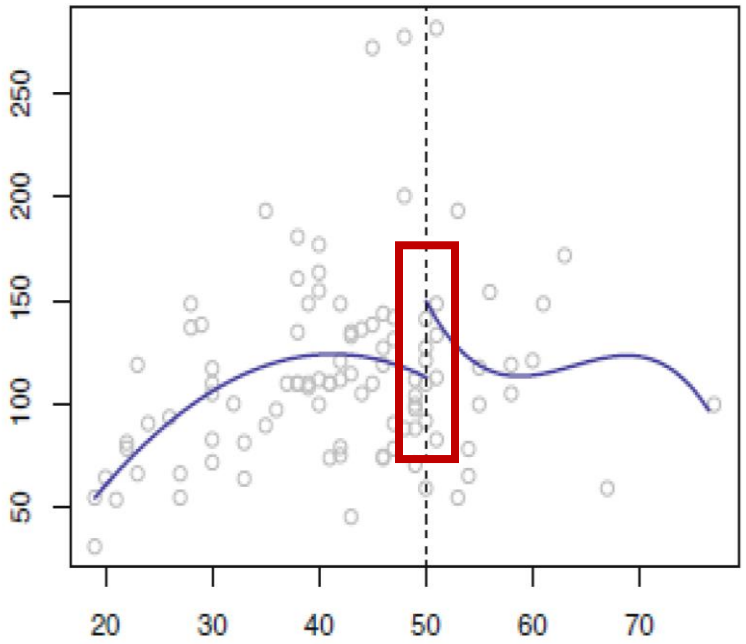
Piecewise 적합한 경우

$y_i =$



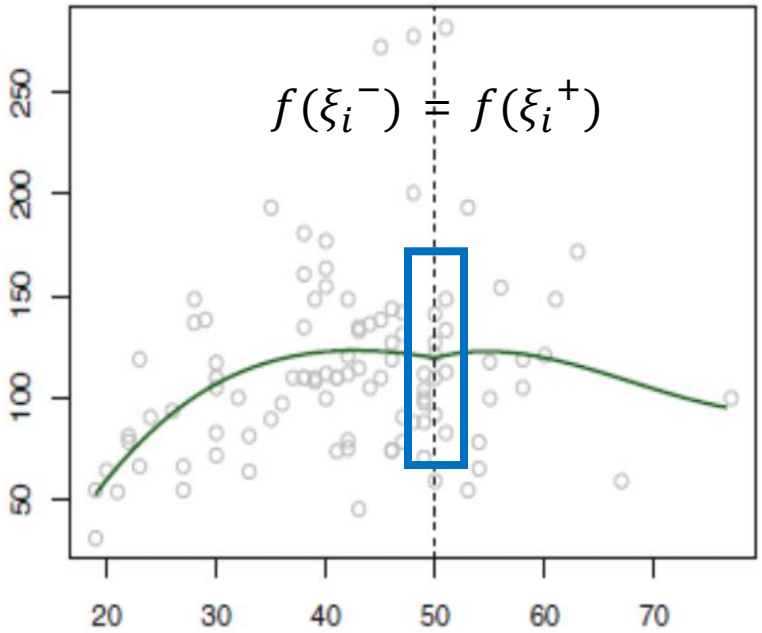
현실에 존재하는 X와 Y의 비선형 관계를 표현하기에
최대 3차항까지 고려하는 것만으로도 충분하기 때문

Spline Regression



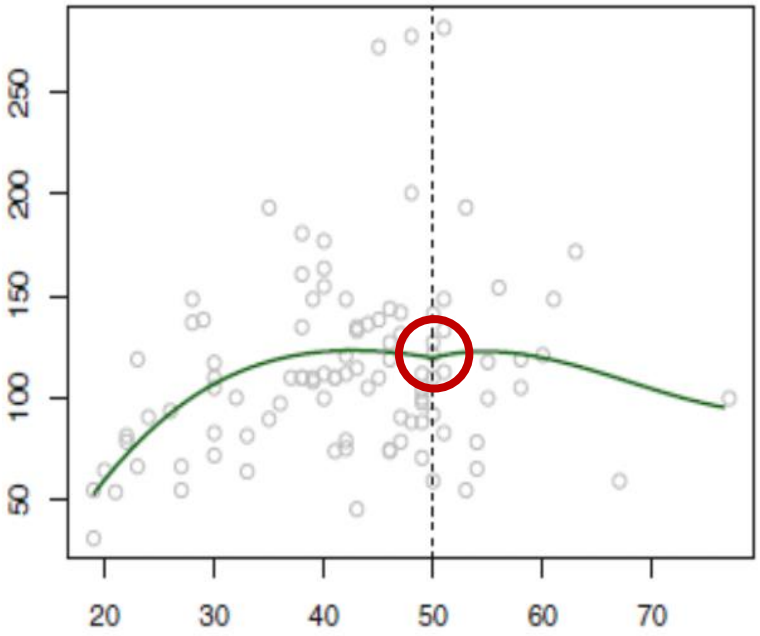
Cubic Spline 역시 불연속성(Discontinuity) 문제가 발생함!

Spline Regression



마찬가지로 극한값 제약식을 추가하여 연속성 부여 가능!

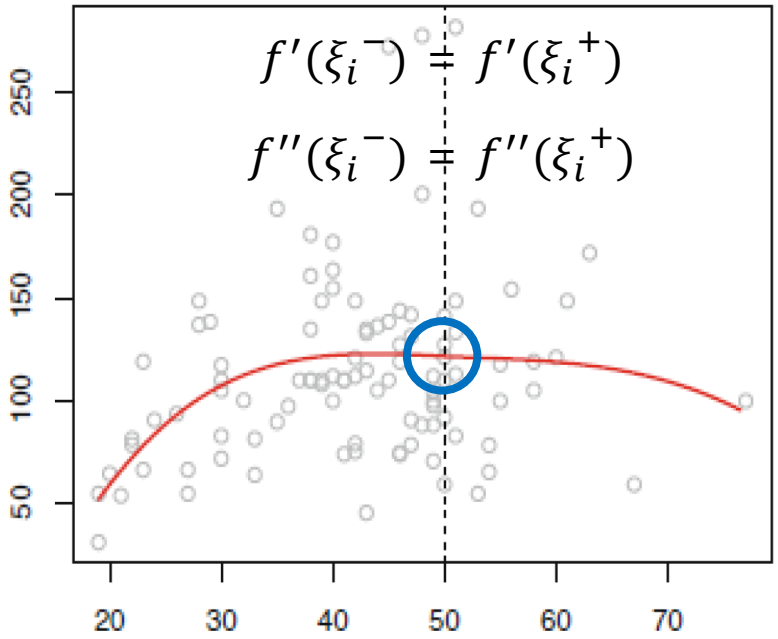
Spline Regression



"미분 불가능"

하지만, 여전히 Knot을 기준으로 부자연스럽게 꺾인 지점이 존재

Spline Regression



"미분 가능"

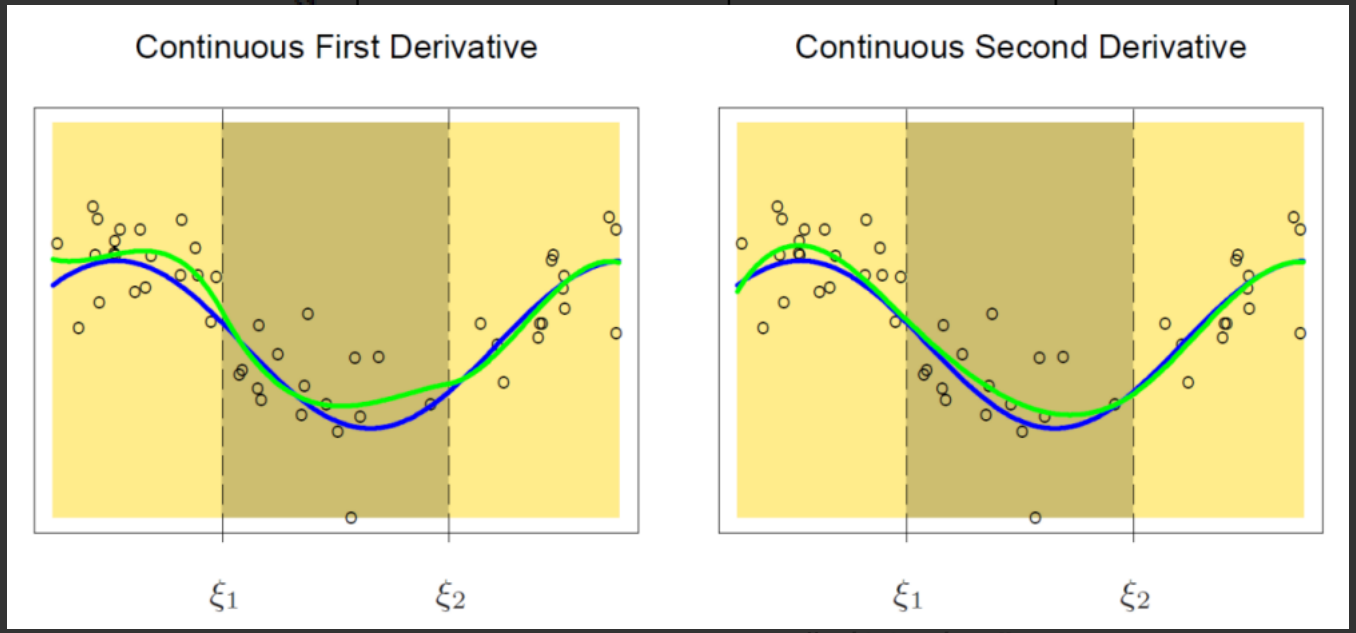
Knot을 기준으로 좌미분계수와 우미분계수가 같다는 제약식 추가

3 모델링 전략

Spline Regression

왜 2계 미분이 연속이라는 조건까지 추가할까?

$$f'(\xi_i^-) = f'(\xi_i^+)$$



"미분 가능"

곡선의 곡률까지 연속으로 만들어 줌으로써

Knot 기준으로 잔미분계수와 우미분계수가 같다는 제약식 추가

함수를 더욱 Smooth하게 적합하고자 함

3 모델링 전략

Spline Regression

장점

- 복잡한 데이터에 적합하기 위해 차수를 높이는 것이 아니라,
  Knot을 늘려 **단순한 다항식으로도 유연하게** 표현할 수 있음
- 데이터의 **보간법(interpolation)**으로도 사용할 수 있음

단점

- **Knot의 개수와 위치**를 어떻게 선정할지 정하기 쉽지 않음
- **설명변수가 3개 이상**으로 늘어날 경우,
  Model Variance가 급격히 증가해 사용하기 어려움

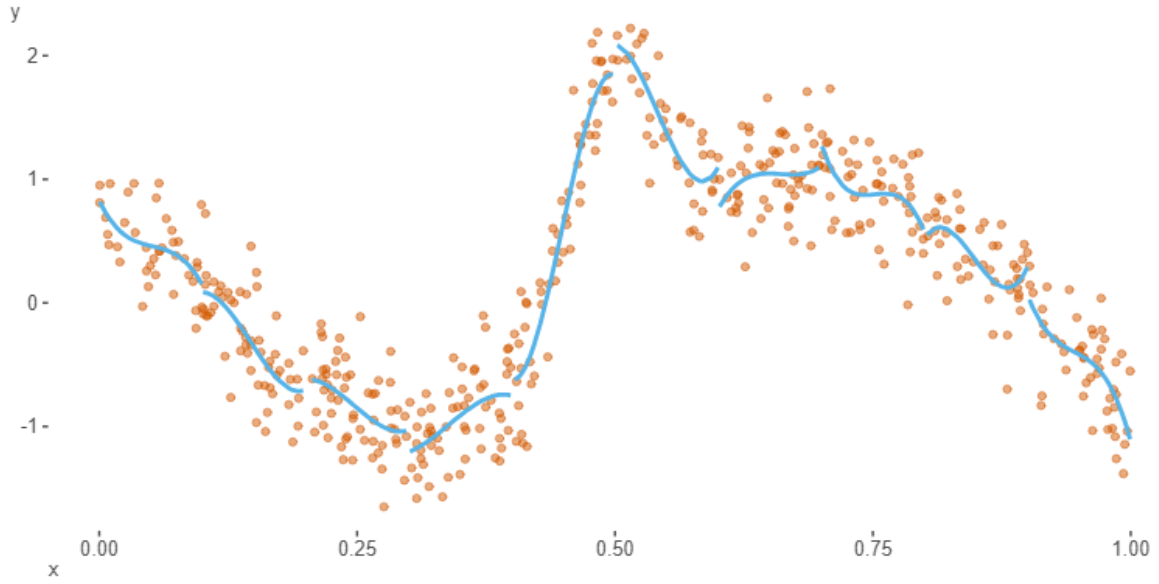
단점을 보완하고자 한 Natural Spline, Smoothing Spline 등이 있음

GAM(Generalized Additive Model)

GAM (일반화 가법 모델)

기존의 선형 모델에서 **가법성(additivity)은 유지**하면서도

각 변수에 Non-linear한 적합을 가능하게 한 방법



GAM(Generalized Additive Model)

GAM (일반화 가법 모델)

$$y = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon$$

$$y = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \epsilon_i$$

$$y = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon$$

다중선형회귀 모형을 확장한 것으로 볼 수 있음!

$\beta_j x_{ij}$ 의 선형결합을 smooth한 **비선형함수 $f_j(x_{ij})$의 선형결합**으로 대체

GAM(Generalized Additive Model)

GAM (일반화 가법 모델)

각각의 Y에 대한 각각의 예측변수들의 기여를 '더하여' 표현함

$$y = \beta_0 + f_1(x_{i1}) + f_2(x_{i2}) + \cdots + f_p(x_{ip}) + \epsilon$$

⋮

여전히 각 변수 x_{ij}을 각각의 함수 f_j에 부과하고 있기에 가법성이 유지됨

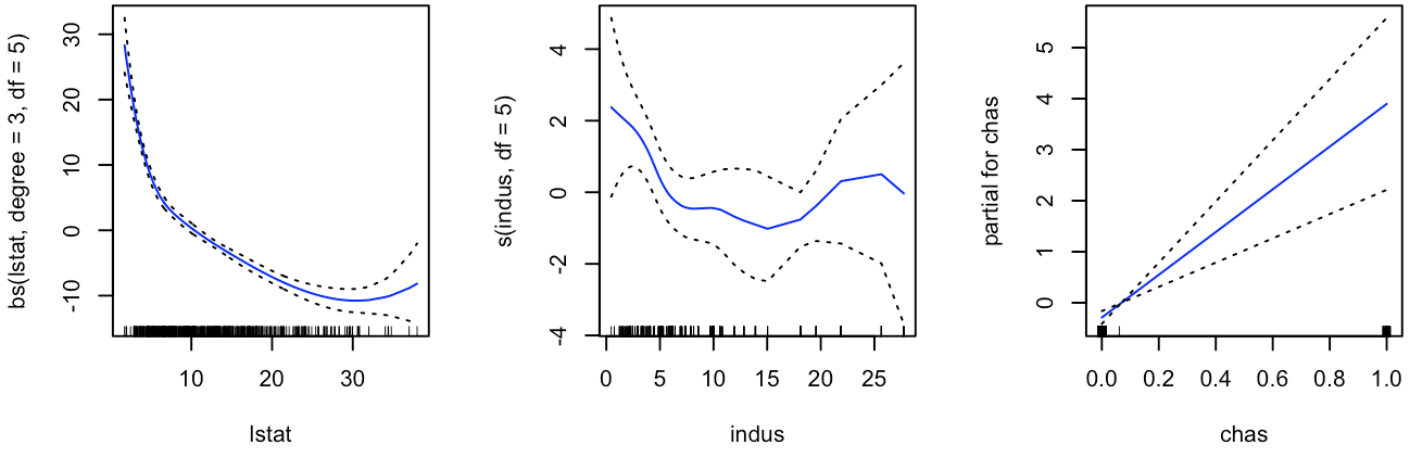
⋮

각 변수별로 다양한 선형/비선형 함수를 적합하여

보다 유연한 관계에 fitting 가능!

GAM(Generalized Additive Model)

GAM (일반화 가법 모델)



각 변수별로 다양한 선형/비선형 함수를 적합하여

보다 유연한 관계에 fitting 가능!

3 모델링 전략

GAM(Generalized Additive Model)

장점

- 일반적인 선형모델이 놓칠 수 있는 비선형 관계를 파악할 수 있음
- 변수 각각의 영향력을 확인할 수 있어 설명력을 어느 정도 확보함

단점

- 모델이 가산적이어야 한다는 제한이 있음
- 변수가 많은 경우 중요한 상호작용을 놓칠 수 있음

다음 주 예고

1. 앙상블
2. eXplainable AI
3. 추천 시스템

감사합니다