의사결정트리

- 권수태 교수

JEONJU UNIVERSITY
The Place for Superstars

1. 의사결정트리

❖ 의사결정트리

- 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 트리 기반의 분류 규칙을 만드는 것
  - ✓어떤 규칙을 하나의 트리 형태로 표현한 후 이를 바탕으로 분류나 회귀문제를 해결하는 것
  - ✓트리는 일종의 경로를 표현, If-else 구조
  - ✓트리의 마지막 노드는 분류문제에서는 클래스, 회귀문제에서는 예측치
- 설명력과 성능의 측면에서 딥러닝 모델들과 대등하게 경쟁하면서 여전히 실무에서 많이 사용

1. 의사결정트리

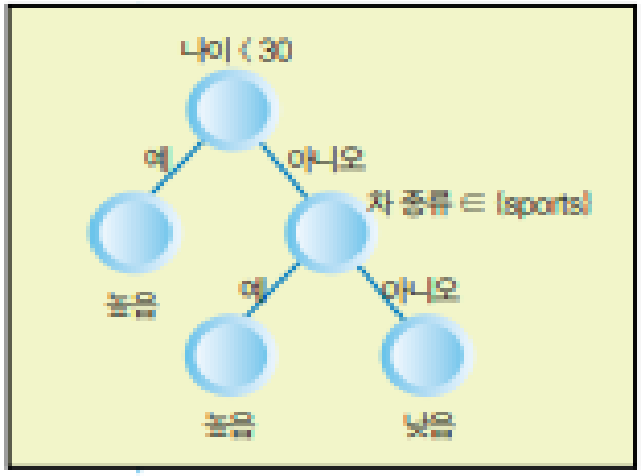
❖ 의사결정트리

➢ 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우함

✓ 분할 속성을 찾는 것이 중요

나이	차종류	위험도
21	Family	높음
25	Sports	높음
43	Sports	높음
68	Family	낮음
32	Truck	낮음
28	Family	높음

숫자 예측 게임



JEONJU UNIVERSITY

1. 의사결정트리

❖ 의사결정트리

- ➢ 탐색과 모형화라는 두 가지 특성을 모두 가지고 있음
- ➢ 장점
  - ✓ 결과에 대해 쉽게 이해하고 설명할 수 있음
  - ✓ 속성의 수가 불필요하게 많을 경우 영향을 미치지 않는 속성들을 자동으로 제외시키기 때문에 데이터 선정이 용이
  - ✓ 연속형이나 명목형 데이터 값을 그대로 사용하여 데이터변환 기간과 노력 단축
  - ✓ 즉, 각 피처의 스케일링과 정규화 같은 전처리 작업이 필요 없음
  - ✓ 두개 이상의 변수가 결합하여 목표변수에 어떻게 영향을 주는지를 쉽게 알 수 있음

1. 의사결정트리

❖ 의사결정트리

➢ 단점

✓ 연속형 변수일 경우 경계점 근방에서 예측 오류가 클 가능성이 있음

✓ 과적합으로 정확도가 떨어진다는 점

- 트리의 크기를 사전에 제한하는 것이 오히려 성능 튜닝에 더 도움이 됨

✓ 표본의 크기에 지나치게 민감하여 서로 상이한 값을 갖는 레코드들을 가능한 한 많이 포함하는 데이터가 필요

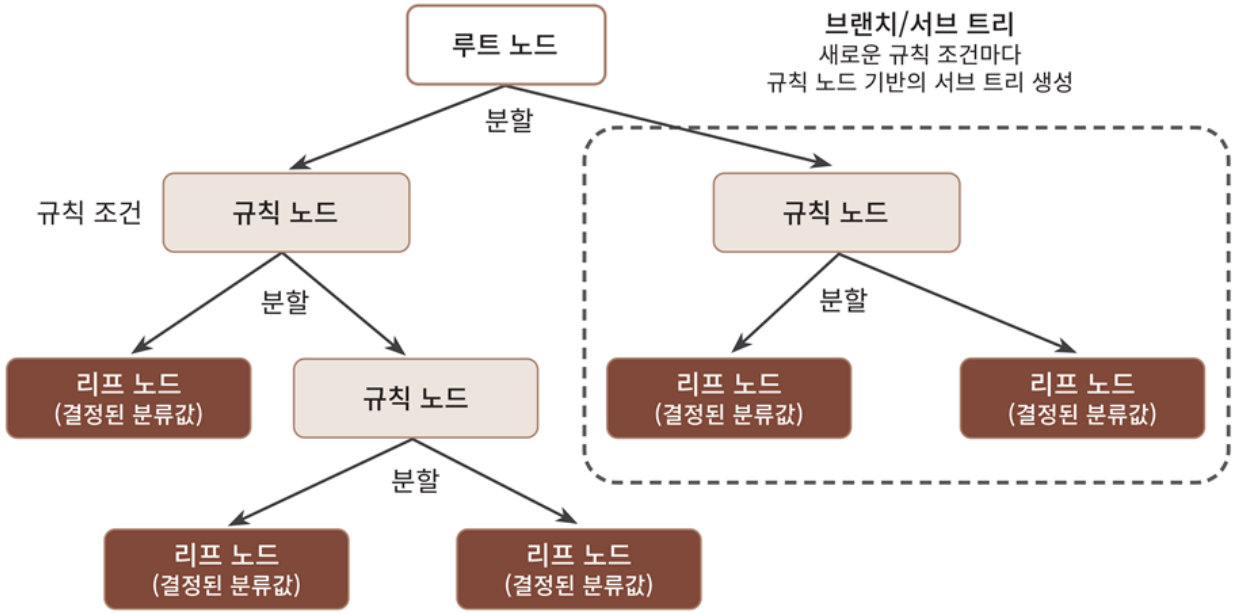
1. 의사결정트리

❖ 의사결정트리

➢ 구조

✓ 규칙노드(Decision Node), 리프노드(Leaf Node)로 구성

- 규칙노드: 하위트리를 분할하는 규칙
- 리프노드: 결정된 클래스 값



1. 의사결정트리

❖ 의사결정트리

- ➢ 순환적 분할방식을 이용하여 나무를 구축하는 기법
  - ✓ 뿌리마디 : 나무의 가장 상단에 위치
  - ✓ 내부마디 : 속성의 분리기준을 포함
  - ✓ 가지 : 마디와 마디를 연결
  - ✓ 잎 : 최종분류를 의미
- ➢ 각 속성들이 고객들을 분류하는데 영향을 미치는 정도를 측정한 후 그 중에서 가장 영향력이 있는 속성을 선정하여 나무의 뿌리마디에 지정
- ➢ 해당속성의 값에 따라 가지들로 분리

❖ 의사결정트리

➢ 데이터상의 오류값이나 결손값

- ✓ 불필요한 속성들이 나무의 마디에 포함되어 규칙자체가 엉뚱한 의미를 갖게 되며, 예측력 감소 초래
- ✓ 이러한 과잉맞춤(overfitting)으로 인해 불필요하게 복잡해진 나무의 의미없는 마디들을 제거하는 작업을 가지치기(pruning)라 함

1. 의사결정트리

❖ 의사결정트리

- 가능한 한 적은 결정노드로 높은 예측정확도를 가지려면 데이터를 분류할 때 최대한 많은 데이터 세트가 해당 분류에 속할 수 있도록 결정노드의 규칙이 정해져야 함
- 결정노드는 최대한 균일한 데이터세트를 구성할 수 있도록 분할
- 정보의 균일도를 측정하는 대표적인 방법
  - ✓ 정보이득(Information Gain) 지수 : 1-엔트로피
  - ✓ 지니계수(Gini coefficient) : 경제학자 코라도 지니가 경제학에서 불평등 지수를 나타날때 사용한 것으로 0이 가장 평등함(지니계수가 낮을 수록 데이터 균일도가 높은 것으로 해석)
  - ✓ 결정트리 알고리즘을 사이킷런에서 구현한 DecisionTreeClassifier 는 기본으로 지니계수를 이용

1. 의사결정트리

❖ 의사결정트리

➢ 엔트로피

S = Q(열)/T(온도)

✓ 독일의 이론 물리학자 Rudolph Julius Emanuel Clausius (1822~1888년)

- "entropy" 라는 개념을 고안(1865년 발표)
- 그리스어 energie + trope(turning) + y 의 합성으로부터 유래, "에너지 변화" 뜻함
- 자연계의 모든 변화는 반드시 "entropy"가 증대되는 방향으로 일어난다는 것

✓ 볼츠만 (L.Boltzman, 1844~1906) S = k log W

- 열역학의 통계적 해석으로 알려진 entropy 법칙을 주장
- entropy 법칙은 한 번 사용한 에너지는 두 번 다시 사용할 수 없음을 뜻함

✓ entropy는 쉽게 표현하면 무질서도

✓ 어떤 계의 '무질서한 정도', 혹은 '규칙적이지 않은 정도'를 측정하는 물리학적 양

1. 의사결정트리

❖ 의사결정트리

➢ 엔트로피

✓ 클로드 섀넌(Claude Elwood Shannon, 1916년~2001년)

- 1948년에 미국의 통신회사 벨 연구소 근무 시절에 통신의 수학적 이론(A Mathematical Theory of Communication)이라는 논문 발표
- 이 이론이 일약 정보를 수학적으로 다루는 시초가 되는 논문으로 인정받게 됨
- 통신회사가 어떤 정보를 전달할 때 얼마의 과금을 해야 하느냐에 대한 순수한 수학적인 정의

A: 00000.....0 (1억개의 0)

B: 100100111010111...01011011 (random으로 나열된 1과 0의 조합 100개)

1. 의사결정트리

❖ 의사결정트리

➢ 엔트로피

✓ 클로드 섀넌(Claude Elwood Shannon, 1916년~2001년)

- 어떤 정보를 보낼때 필요한 비트수가 얼마냐?
- 어떤 많은 정보들이 각기 출현 확률이 P(x)일때 각 값들을 전송하는데 필요한 비트 공식
- $H(P)=H(x)=-\sum P(x)\log P(x)$ (log는 밑이 2)
- 전달하고자 하는 데이터의 패턴을 보고 가장 효율적으로 압축했을 때의 bit수
- 어떤 데이터를 출현 빈도수 패턴에 맞게 가장 효율적으로 압축했을때 필요한 전송량
- 확률을 고려한 압축 가능 정도의 개념

1. 의사결정트리

❖ 의사결정트리

➢ 엔트로피

✓ 클로드 섀넌(Claude Elwood Shannon, 1916년~2001년)

전송하고자 하는 값이 2개(A,B), 동일하게 나타남(각각 50%)

2bit

$$H(P) = H(x) = -\sum P(x)\log P(x) = -(0.5\log 0.5 + 0.5\log 0.5)$$
$$= -(0.5 * -1 + 0.5 * -1) = 1$$

A,B,C,D,E,F,G,H라는 8가지 분류의 값 데이터를 전달하는데 각기 그 문건상에 등장 확률이 30%,20%,10%,10%,10%,10%,5%,5%

3bit

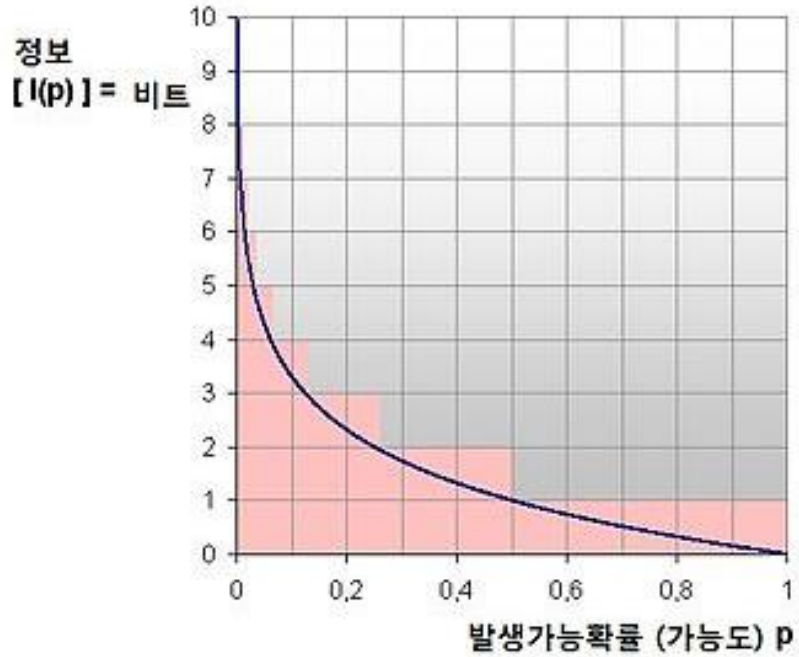
$$H(P) = H(x) = -\sum P(x)\log P(x)$$
$$= -(0.3\log 0.3 + 0.2\log 0.2 + 0.1\log 0.1 + 0.1\log 0.1 + 0.1\log 0.1 + 0.1\log 0.1 + 0.05\log 0.05 + 0.05\log 0.05)$$
$$= -(-2.74645) = 2.74645\ldots$$

1. 의사결정트리

❖ 의사결정트리

➢ 엔트로피

✓ 정보량 $$information = I(x) = \log_2 \frac{1}{p(x)}$$



1. 의사결정트리

❖ 의사결정트리

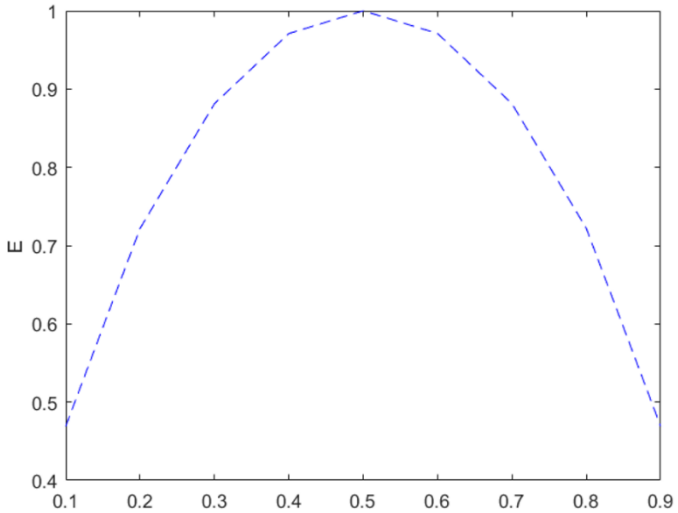
➢ 엔트로피

✓ 클로드 새넌(Shannon)

$$H(S) = -\sum_i p_i \, log_2(p_i)$$

p_i : label i 확률

- 2분류: $H(S) = -p^+ \log_2 p^+ - p^- \log_2 p^-$



✓ 값이 크면 많은 정보가 담겨있다는 것

- 모두 양성 : $-1 \log_2 1 - 0 \log_2 0 = 0$
- 균등하게 섞여있는 경우 $H(S) = -0.5 \log_2 2^{-1} - 0.5 \log_2 2^{-1} = 1$

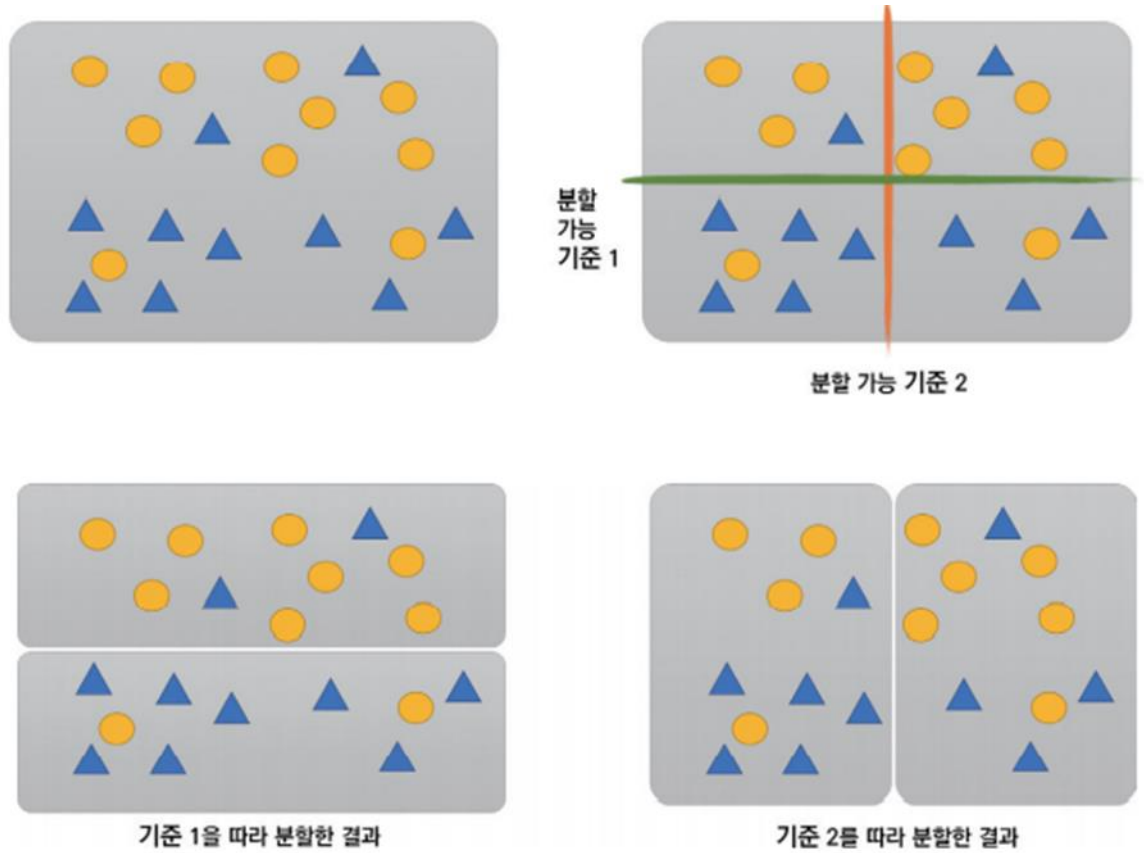
✓ 주어진 데이터 집합의 혼잡도를 의미(서로 다른 값이 섞여 있으면 엔트로피가 높고, 같은 값이 섞여 있으면 엔트로피가 낮음

JEONJU UNIVERSITY

1. 의사결정트리

❖ 의사결정트리

➢ 엔트로피

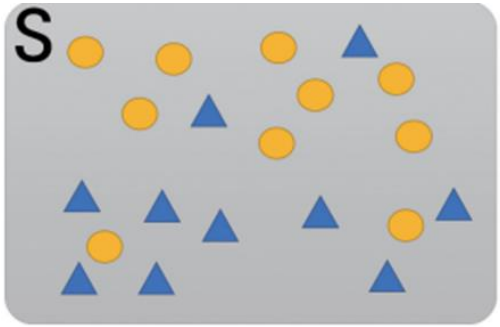


JEONJU UNIVERSITY

1. 의사결정트리

❖ 의사결정트리

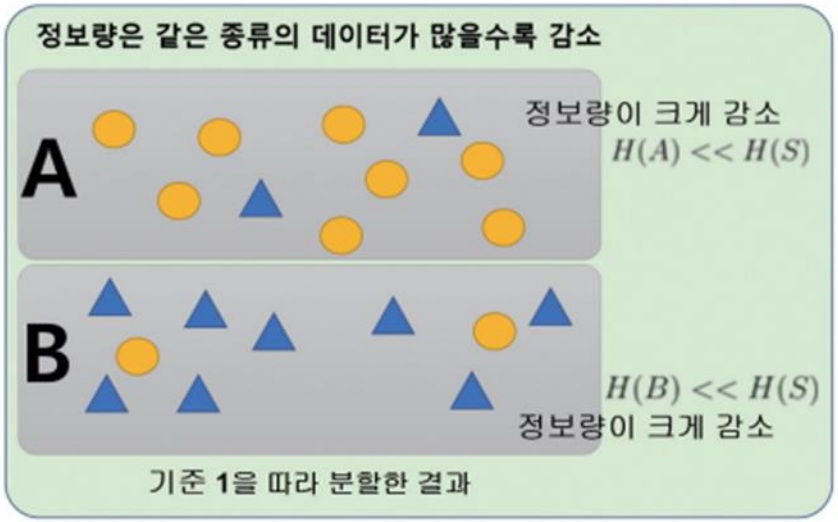
➢ 엔트로피

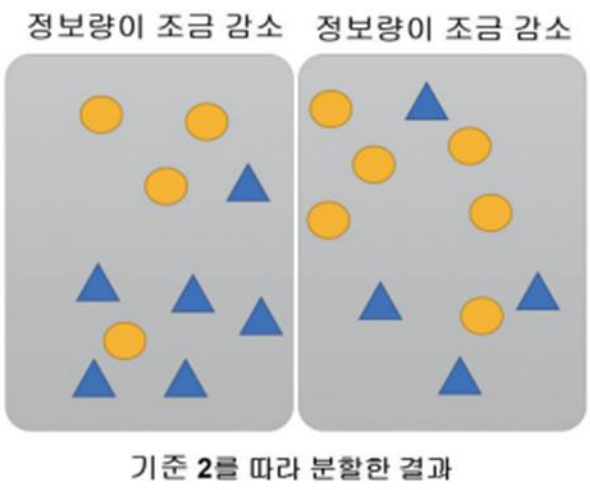


정보량 = 엔트로피

$$H(S) = -\left(\frac{1}{2}\lg\frac{1}{2} + \frac{1}{2}\lg\frac{1}{2}\right) = 1$$

정보량의 감소(정보이득)를 이용하여 좋은 분할을 찾는 방법: 정보이득 = 원래의 정보량 – 분할 후의 정보량





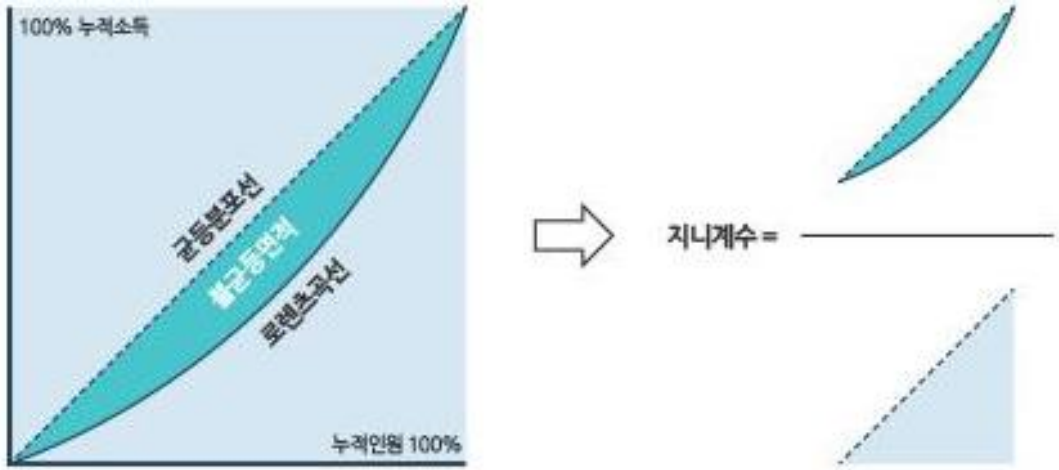
JEONJU UNIVERSITY

❖ 의사결정트리

➢ 지니계수

✓ 전체 인구의 소득불평등도를 나타내는 지표

- 로렌츠(Lorenz) 곡선: 소득순위에 따른 인구의 누적백분율과 가구의 소득을 균등화하여 개인화한 균등화소득의 누적백분율을 표시한 그래프
- 불평등면적: 로렌츠곡선과 완전균등선(대각선)이 이루는 면적

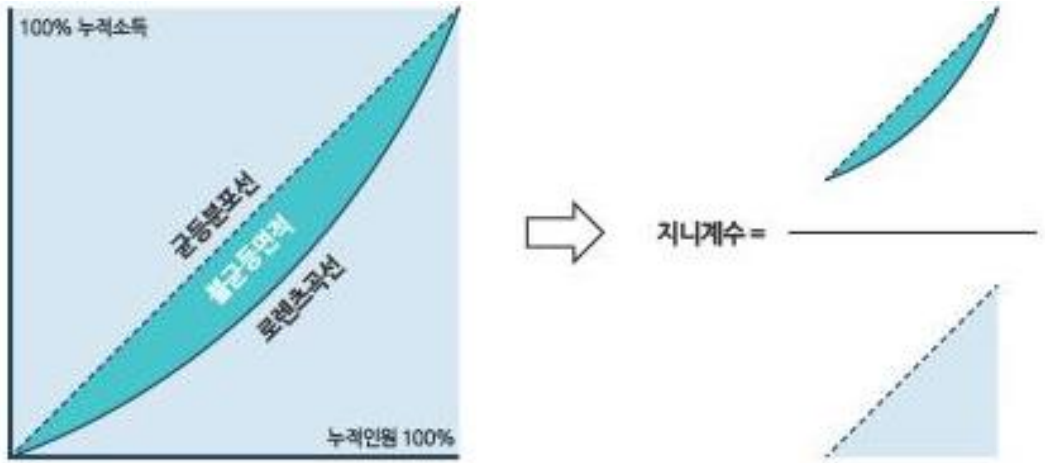


❖ 의사결정트리

➢ 지니계수

✓ **전**체 인구의 소득불평등도를 나타내는 지표

- 로렌츠(Lorenz) 곡선: 소득순위에 따른 인구의 누적백분율과 가구의 소득을 균등화하여 개인화한 균등화소득의 누적백분율을 표시한 그래프
- **불평등면적: 로렌츠곡선과** 완전균등선(대각선)이 이루는 면적



❖ 의사결정트리

➢ 지니계수

✓ **불확실성을 의미**

- 얼마나 불확실한가? **얼마나 많은 것들이 섞여 있는가? 를 보여줌**

✓ 지니지수가 0

- 불확실성이 0**이라는 것**
- **같은 특성을 가진 객체들끼리 잘 모여있다는 것을 의미**

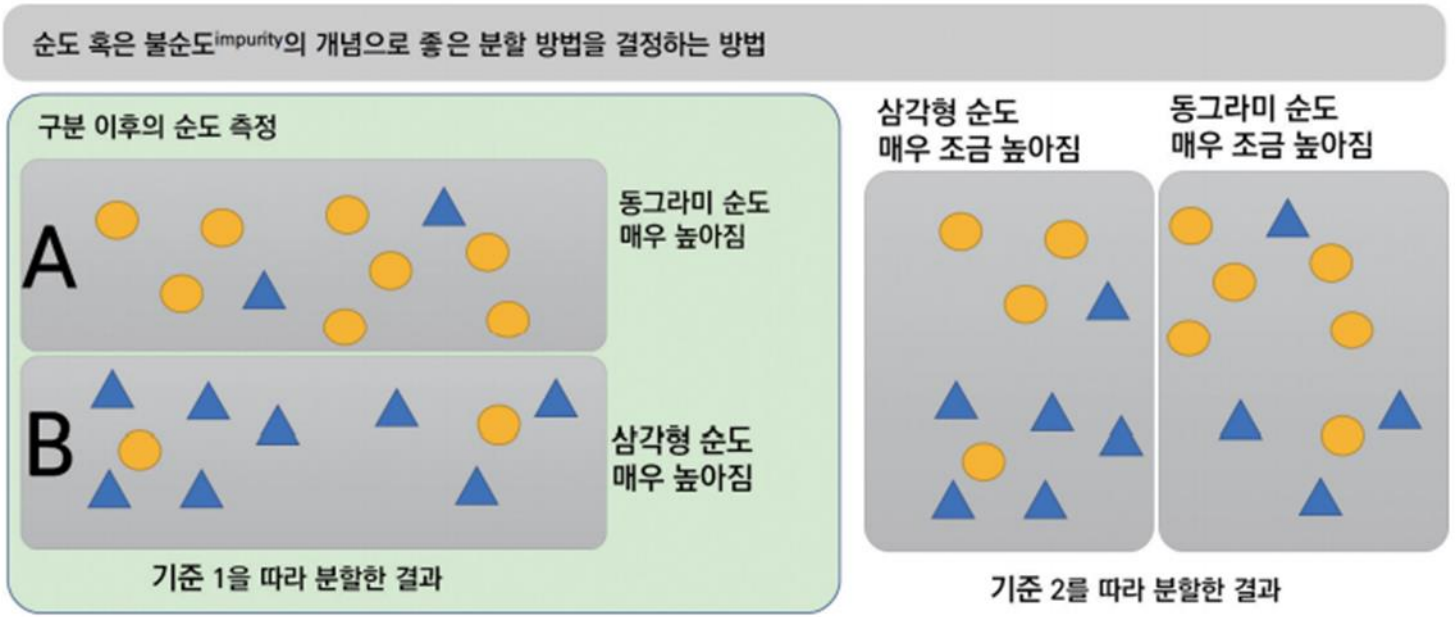
✓ **데이터의 통계적 분산정도를 정량화해서 표현한 값**

✓ $\sum_{i=1}^{c} p_i(1-p_i) = \sum_{i=1}^{c} p_i - \sum_{i=1}^{c} p_i^2 = 1 - \sum_{i=1}^{c} p_i^2$

1. 의사결정트리

❖ 의사결정트리

➢ 순도, 불순도(지니 불순도)



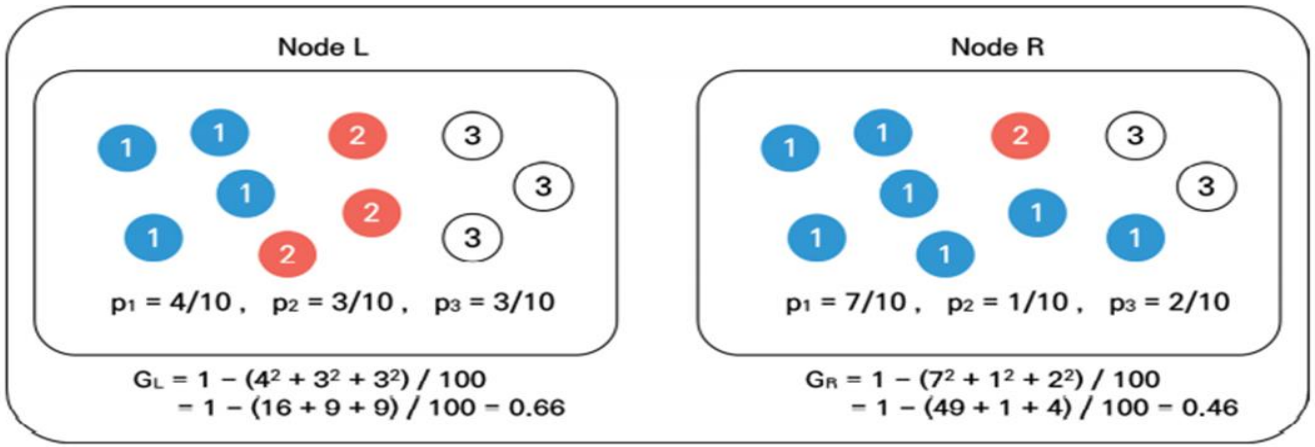
1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ 하나의 그룹 내에 섞여 있는 n개 종류의 레이블이 있고, 레이블이 i인 객체 비율이 p_i 라고 할 때 다음과 같은 값을 가짐

$$G = 1 - \sum_{i=1}^{n} p_i^2$$



JEONJU UNIVERSITY

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

- ✓ CART는 분류와 회귀 트리classification and regression tree의 약자
- ✓ 현재 데이터가 섞여 있는 노드를 두 개의 노드로 나눌 때 어떤 속성 A와 해당 속성에 어떤 값 a를 기준으로 쪼갤 것인지를 찾는 문제
- ✓ 쪼개어서 얻는 두 노드 L과 R이라고 하고, 각각의 원소 개수를 m_L, m_R이라고 하고 쪼개어지기 전 노드의 원소 개수는 $\boldsymbol{m}$
- ✓ 알고리즘이 하는 일은 불순도 G_L과 G_R로 결정되는 다음 비용함수 $\boldsymbol{J(A,a)}$가 최소가 되게 하는 $\boldsymbol{A}$와 $\boldsymbol{a}$를 선택하는 것

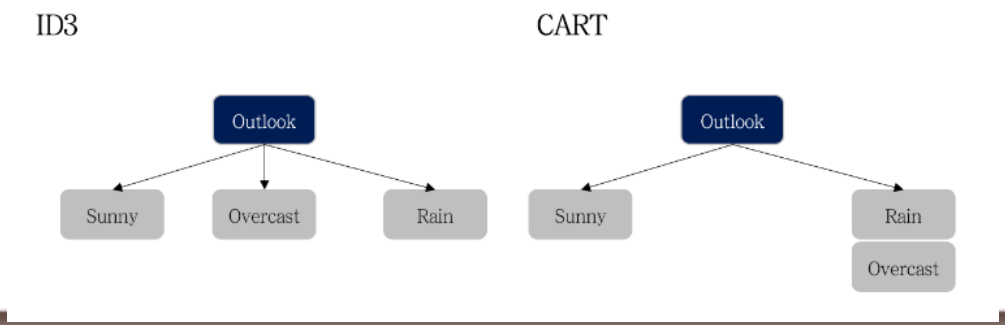
$$\underset{A,a}{\operatorname{argmin}} J(A,a) = \underset{A,a}{\operatorname{argmin}} \left(\frac{m_L}{m} G_L + \frac{m_R}{m} G_R \right)$$

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ Binary tree

- 가지 분기 시, 여러 개의 자식 노드가 아닌 단 두 개의 노드로 분기
- 데이터를 1개의 속성과 나머지로 분류
- $2^{k-1} - 1$ 개로 분할
- 지니$(D) = \frac{|D_1|}{|D|}\text{Gini}(D_1) + \frac{|D_2|}{|D|}\text{Gini}(D_2)$



1. 의사결정트리

❖ 의사결정트리

➢CART 알고리즘(지니 불순도)

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
3	middle_aged	high	no	fair	Yes
4	senior	medium	no	fair	Yes
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
7	middle_aged	low	yes	excellent	Yes
8	Youth	medium	no	fair	No
9	Youth	low	yes	fair	Yes
10	senior	medium	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes
12	middle_aged	medium	no	excellent	Yes
13	middle_aged	high	yes	fair	Yes
14	senior	medium	no	excellent	No

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ 4가지 속성에 대한 지니지수 계산

- $Gini_{age}(D)$,
- $Gini_{income}(D)$,
- $Gini_{student}(D)$,
- $Gini_{credit_{rating}}(D)$

1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ $Gini_{age}(D)$: 이진분할 경우 수 3개

- $Gini_{age\_youth}(D) = \frac{5}{14}Gini(D_1) + \frac{9}{14}Gini(D_2) = 0.3937$

$$Gini(D_1) = 1 - \left(\frac{3}{5}\right)^2 - \left(\frac{2}{5}\right)^2 = 0.48$$

$$Gini(D_2) = 1 - \left(\frac{2}{9}\right)^2 - \left(\frac{7}{9}\right)^2 = 0.3457$$

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
8	Youth	medium	no	fair	No
9	Youth	low	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes

	Age	Income	Student	Credit_rating	Class_buys_computer
3	middle_aged	high	no	fair	Yes
4	senior	medium	no	fair	Yes
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
7	middle_aged	low	yes	excellent	Yes
10	senior	medium	yes	fair	Yes
12	middle_aged	medium	no	excellent	Yes
13	middle_aged	high	yes	fair	Yes
14	senior	medium	no	excellent	No

1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ $Gini_{age}(D)$: 이진분할 경우 수 3개

- $Gini_{age\_middle\_aged}(D) = \frac{4}{14}Gini(D_1) + \frac{10}{14}Gini(D_2) = 0.3571$

$$Gini(D_1) = 1 - \left(\frac{0}{4}\right)^2 - \left(\frac{4}{4}\right)^2 = 0$$

$$Gini(D_2) = 1 - \left(\frac{5}{10}\right)^2 - \left(\frac{5}{10}\right)^2 = 0.5$$

	Age	Income	Student	Credit_rating	Class_buys_computer
3	middle_aged	high	no	fair	Yes
7	middle_aged	low	yes	excellent	Yes
12	middle_aged	medium	no	excellent	Yes
13	middle_aged	high	yes	fair	Yes

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
4	senior	medium	no	fair	Yes
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
8	Youth	medium	no	fair	No
9	Youth	low	yes	fair	Yes
10	senior	medium	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes
14	senior	medium	no	excellent	No

1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ $Gini_{age}(D)$: 이진분할 경우 수 3개

- $Gini_{age\_senior}(D) = \frac{5}{14} Gini(D_1) + \frac{9}{14} Gini(D_2) = 0.4571$

$$Gini(D_1) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

$$Gini(D_2) = 1 - \left(\frac{3}{9}\right)^2 - \left(\frac{6}{9}\right)^2 = 0.4444$$

	Age	Income	Student	Credit_rating	Class_buys_computer
4	senior	medium	no	fair	Yes
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
10	senior	medium	yes	fair	Yes
14	senior	medium	no	excellent	No

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
3	middle_aged	high	no	fair	Yes
7	middle_aged	low	yes	excellent	Yes
8	Youth	medium	no	fair	No
9	Youth	low	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes
12	middle_aged	medium	no	excellent	Yes
13	middle_aged	high	yes	fair	Yes

1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ $Gini_{income}(D)$: 이진분할 경우 수 3개

- $Gini_{income\_high}(D) = \frac{4}{14}Gini(D_1) + \frac{10}{14}Gini(D_2) = 0.4429$

$$Gini(D_1) = 1 - \left(\frac{2}{4}\right)^2 - \left(\frac{2}{4}\right)^2 = 0.5$$

$$Gini(D_2) = 1 - \left(\frac{3}{10}\right)^2 - \left(\frac{7}{10}\right)^2 = 0.42$$

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
3	middle_aged	high	no	fair	Yes
13	middle_aged	high	yes	fair	Yes

	Age	Income	Student	Credit_rating	Class_buys_computer
4	senior	medium	no	fair	Yes
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
7	middle_aged	low	yes	excellent	Yes
8	Youth	medium	no	fair	No
9	Youth	low	yes	fair	Yes
10	senior	medium	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes
12	middle_aged	medium	no	excellent	Yes
14	senior	medium	no	excellent	No

1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ $Gini_{income}(D)$: 이진분할 경우 수 3개

- $Gini_{income\_medium}(D) = \frac{6}{14}Gini(D_1) + \frac{8}{14}Gini(D_2) = 0.4583$

$$Gini(D_1) = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = 0.4444$$

$$Gini(D_2) = 1 - \left(\frac{3}{8}\right)^2 - \left(\frac{5}{8}\right)^2 = 0.4688$$

	Age	Income	Student	Credit_rating	Class_buys_computer
4	senior	medium	no	fair	Yes
8	Youth	medium	no	fair	No
10	senior	medium	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes
12	middle_aged	medium	no	excellent	Yes
14	senior	medium	no	excellent	No

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
3	middle_aged	high	no	fair	Yes
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
7	middle_aged	low	yes	excellent	Yes
9	Youth	low	yes	fair	Yes
13	middle_aged	high	yes	fair	Yes

1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ $Gini_{income}(D)$: 이진분할 경우 수 3개

- $Gini_{income\_low}(D) = \frac{4}{14} Gini(D_1) + \frac{10}{14} Gini(D_2) = 0.45$

$$Gini(D_1) = 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.375$$

$$Gini(D_2) = 1 - \left(\frac{4}{10}\right)^2 - \left(\frac{6}{10}\right)^2 = 0.48$$

	Age	Income	Student	Credit_rating	Class_buys_computer
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
7	middle_aged	low	yes	excellent	Yes
9	Youth	low	yes	fair	Yes

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
3	middle_aged	high	no	fair	Yes
4	senior	medium	no	fair	Yes
8	Youth	medium	no	fair	No
10	senior	medium	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes
12	middle_aged	medium	no	excellent	Yes
13	middle_aged	high	yes	fair	Yes
14	senior	medium	no	excellent	No

1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ $Gini_{student}(D)$: 이진분할 경우 수 1개

- $Gini_{student\_no}(D) = \frac{7}{14}Gini(D_1) + \frac{7}{14}Gini(D_2) = 0.3673$

$$Gini(D_1) = 1 - \left(\frac{4}{7}\right)^2 - \left(\frac{3}{7}\right)^2 = 0.4898$$

$$Gini(D_2) = 1 - \left(\frac{1}{7}\right)^2 - \left(\frac{6}{7}\right)^2 = 0.2449$$

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
3	middle_aged	high	no	fair	Yes
4	senior	medium	no	fair	Yes
8	Youth	medium	no	fair	No
12	middle_aged	medium	no	excellent	Yes
14	senior	medium	no	excellent	No

	Age	Income	Student	Credit_rating	Class_buys_computer
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
7	middle_aged	low	yes	excellent	Yes
9	Youth	low	yes	fair	Yes
10	senior	medium	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes
13	middle_aged	high	yes	fair	Yes

1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ $Gini_{credit\_rating}(D)$: 이진분할 경우 수 1개

- $Gini_{credit\_rating\_excellent}(D) = \frac{6}{14}Gini(D_1) + \frac{8}{14}Gini(D_2) = 0.4286$

$$Gini(D_1) = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$$

$$Gini(D_2) = 1 - \left(\frac{2}{8}\right)^2 - \left(\frac{6}{8}\right)^2 = 0.375$$

	Age	Income	Student	Credit_rating	Class_buys_computer
2	youth	high	no	excellent	No
6	senior	low	yes	excellent	No
7	middle_aged	low	yes	excellent	Yes
11	Youth	medium	yes	excellent	Yes
12	middle_aged	medium	no	excellent	Yes
14	senior	medium	no	excellent	No

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
3	middle_aged	high	no	fair	Yes
4	senior	medium	no	fair	Yes
5	senior	low	yes	fair	Yes
8	Youth	medium	no	fair	No
9	Youth	low	yes	fair	Yes
10	senior	medium	yes	fair	Yes
13	middle_aged	high	yes	fair	Yes

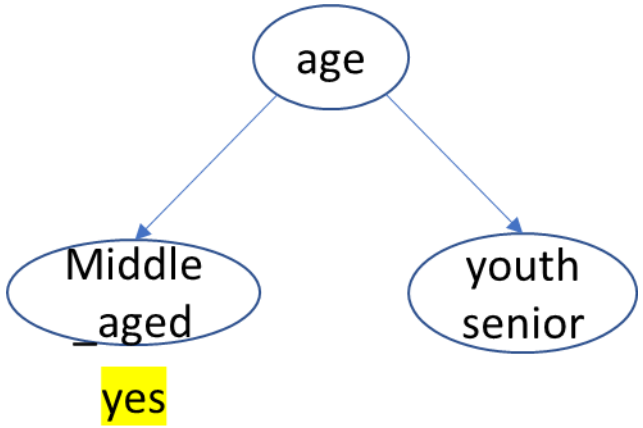
1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ 4가지 속성에 대한 지니지수 계산

- $Gini_{age}(D) = 0.3571$
- $Gini_{income}(D) = 0.4429$
- $Gini_{student}(D) = 0.3673$
- $Gini_{credit_{rating}}(D) = 0.4286$



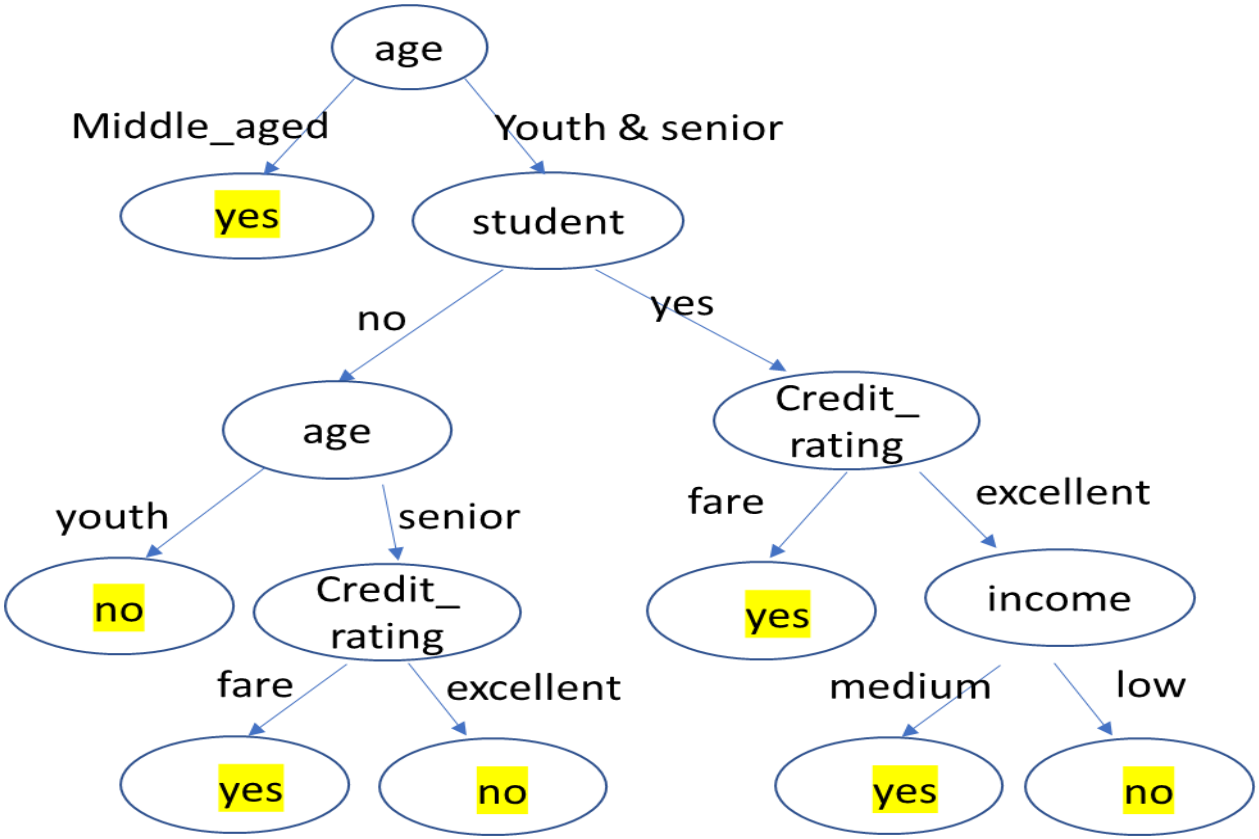
	Age	Income	Student	Credit_rating	Class_buys_computer
3	middle_aged	high	no	fair	Yes
7	middle_aged	low	yes	excellent	Yes
12	middle_aged	medium	no	excellent	Yes
13	middle_aged	high	yes	fair	Yes

	Age	Income	Student	Credit_rating	Class_buys_computer
1	youth	high	no	fair	No
2	youth	high	no	excellent	No
4	senior	medium	no	fair	Yes
5	senior	low	yes	fair	Yes
6	senior	low	yes	excellent	No
8	Youth	medium	no	fair	No
9	Youth	low	yes	fair	Yes
10	senior	medium	yes	fair	Yes
11	Youth	medium	yes	excellent	Yes
14	senior	medium	no	excellent	No

1. 의사결정트리

❖ 의사결정트리

➢CART 알고리즘(지니 불순도)



JEONJU UNIVERSITY

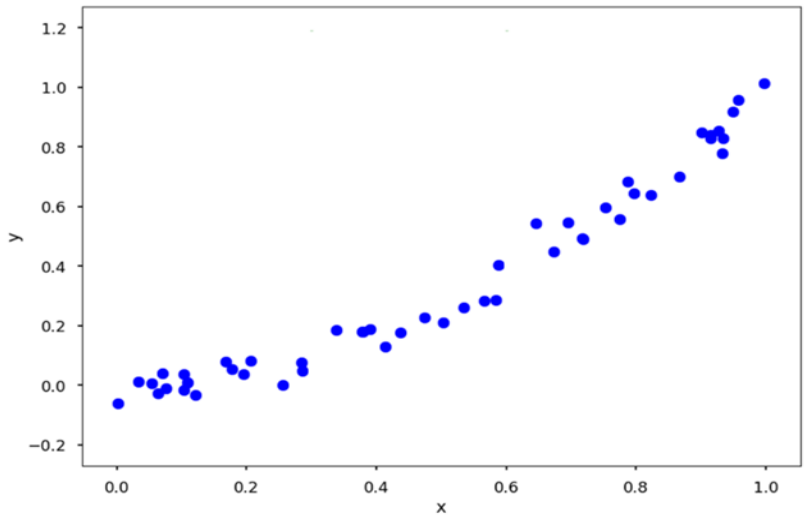
1. 의사결정트리

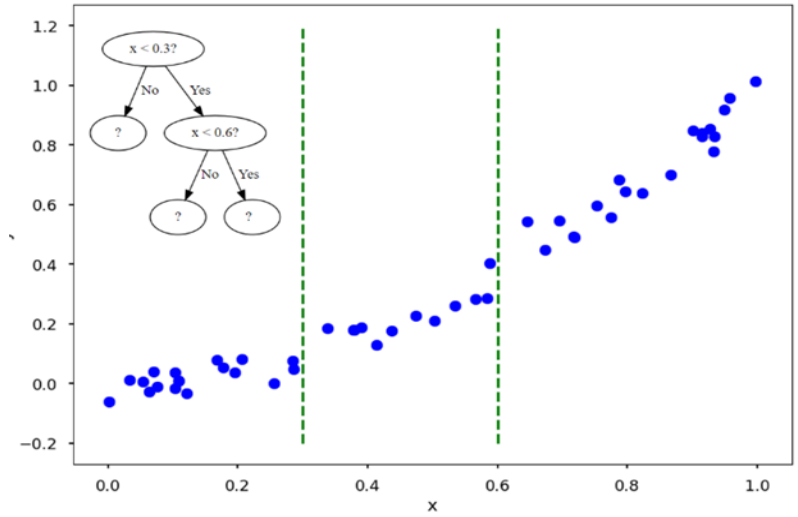
❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ Regression tree

- 분기 지표를 선택할 때 사용하는 index를 불순도 (Entropy, Gini index)가 아닌, 실제값과 예측값의 오차를 사용





참조: [tomaszgolan's blog]

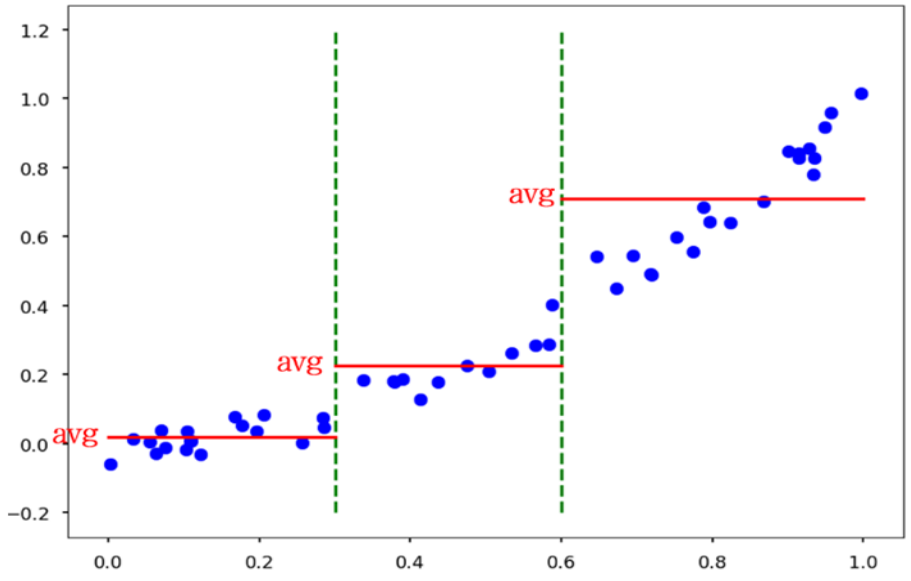
1. 의사결정트리

❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ Regression tree

- 각 구간의 x값이 들어올 경우, training data (파란 점)의 평균값 (빨간 실선)을 예측값으로 산출



❖ 의사결정트리

➢ CART 알고리즘(지니 불순도)

✓ Regression tree

- 실제값과 예측값의 오차

$$RSS\ (Residual\ Sum\ of\ Squares) = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

- 전체**오차값**을 계산하여 지표간 비교를 통해 최적의 분기를 결정

$$J(k, t_k) = \frac{m_{left}}{m} MSE_{left} + \frac{m_{right}}{m} MSE_{right}$$

1. 의사결정트리

❖ 결정트리

➢ 결정트리 파라미터

- ✓ 사이킷런은 결정 트리 알고리즘을 구현한 DecisionTreeClassifier와 DecisionTreeRegressor 클래스를 제공
- ✓ 사이킷런의 결정트리 구현은 CART(Classification and Regression Trees) 알고리즘 기반

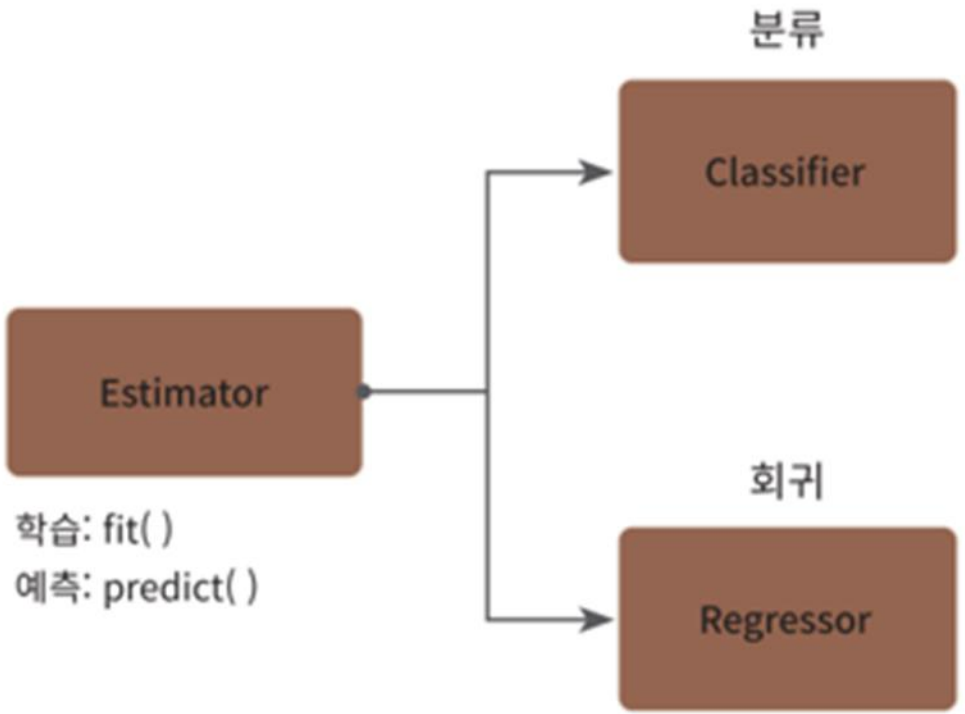
1. 의사결정트리

파라미터 명	설명
min_samples_split	• 노드를 분할하기 위한 최소한의 샘플 데이터 수로 과적합을 제어하는 데 사용됨. • 디폴트는 2이고 작게 설정할수록 분할되는 노드가 많아져서 과적합 가능성 증가 • 과적합을 제어. 1로 설정할 경우 분할되는 노드가 많아져서 과적합 가능성 증가
min_samples_leaf	• 말단 노드(Leaf)가 되기 위한 최소한의 샘플 데이터 수 • Min_samples_split와 유사하게 과적합 제어 용도. 그러나 비대칭적(imbalanced) 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 이 경우는 작게 설정 필요.
max_features	• 최적의 분할을 위해 고려할 최대 피처 개수. 디폴트는 None으로 데이터 세트의 모든 피처를 사용해 분할 수행. • int 형으로 지정하면 대상 피처의 개수, float 형으로 지정하면 전체 피처 중 대상 피처의 퍼센트임 • 'sqrt'는 전체 피처 중 sqrt(전체 피처 개수), 즉 $\sqrt{\text{전체 피처 개수}}$만큼 선정 • 'auto'로 지정하면 sqrt와 동일 • 'log'는 전체 피처 중 log2(전체 피처 개수) 선정 • 'None'은 전체 피처 선정
max_depth	• 트리의 최대 깊이를 규정. • 디폴트는 None. None으로 설정하면 완벽하게 클래스 결정 값이 될 때까지 깊이를 계속 키우며 분할하거나 노드가 가지는 데이터 개수가 min_samples_split보다 작아질 때까지 계속 깊이를 증가시킴. • 깊이가 깊어지면 min_samples_split 설정대로 최대 분할하여 과적합할 수 있으므로 적절한 값으로 제어 필요.
max_leaf_nodes	• 말단 노드(Leaf)의 최대 개수

1. 의사결정트리

❖ 사이킷런의 기반 프레임워크 익히기

➢ Estimator 이해 및 fit(), predict() 메서드



분류 구현 클래스

DecisionTreeClassifier
RandomForestClassifier
GradientBoostingClassifer
GaussianNB
SVC

회귀 구현 클래스

LinearRegression
Ridge
Lasso
RandomForestRegressor
GradientBoostingRegressor

1. 의사결정트리

❖ 사이킷런의 기반 프레임워크 익히기

➢ Estimator 이해 및 fit(), predict() 메서드

- ✓ fit() : ML 모델 학습
- ✓ predict() : 학습된 모델의 예측
- ✓ Estimator : Classifier(분류 알고리즘을 구현한 클래스) 와 Regressor(회귀 알고리즘을 구현한 클래스) 를 합친 것. 지도학습의 모든 알고리즘을 구현한 클래스

cross_val_score( )와 같은 evaluation 함수, GridSearchCV와 같은 하이퍼 파라미터 튜닝을 지원하는 클래스의 경우 이 Estimator를 인자로 받음
인자로 받은 Estimator에 대해서 cross_val_score( ), GridSearchCV.fit( ) 함수 내에서 이 Estimator의 fit( )과 predict( )를 호출해서 평가를 하거나 하이퍼 파라미터 튜닝을 수행하는 것

1. 의사결정트리

❖붓꽃 품종 예측

➢붓꽃 데이터 세트로 붓꽃 품종 분류

✓꽃잎의 길이와 너비, 꽃받침의 길이와 너비 피처를 기반으로 꽃의 품종 예측

1. 의사결정트리

❖ 붓꽃 품종 예측

1. 데이터 세트 분리: 데이터를 학습 데이터와 테스트 데이터로 분리

2. 모델 학습: 학습 데이터를 기반으로 ML 알고리즘을 적용해 모델을 학습

3. 예측 수행: 학습된 ML 모델을 이용해 테스트 데이터의 분류를 예측

4. 평가: 이렇게 예측된 결괏값과 테스트 데이터의 실제 결괏값을 비교해 ML 모델 성능을 평가

1. 의사결정트리

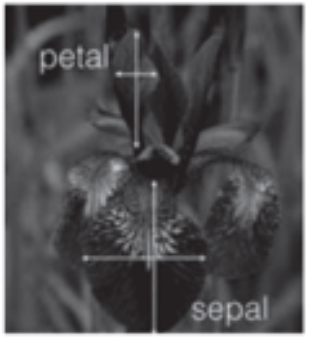
❖ 붓꽃 품종 예측

학습 데이터

번호	꽃받침 길이	꽃받침 너비	꽃잎 길이	꽃잎 너비	Iris 꽃 종류
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
.....					.....
5.0	6.4	3.5	4.5	1.2	Versicolor
.....					.....
150	5.9	3.0	5.0	1.8	Virginica

학습 데이터로 모델 학습

모델학습



학습 모델 통해 테스트 데이터의 레이블 값 예측

예측된 레이블 값과 실제 레이블 값 예측 정확도 평가

테스트 데이터

번호	꽃받침 길이	꽃받침 너비	꽃잎 길이	꽃잎 너비	Iris 꽃 종류는?
1	5.1	3.5	1.4	0.2	?
2	4.9	3.0	1.4	0.2	Setosa
.....					.....

학습된 모델로 테스트 데이터 예측

평가

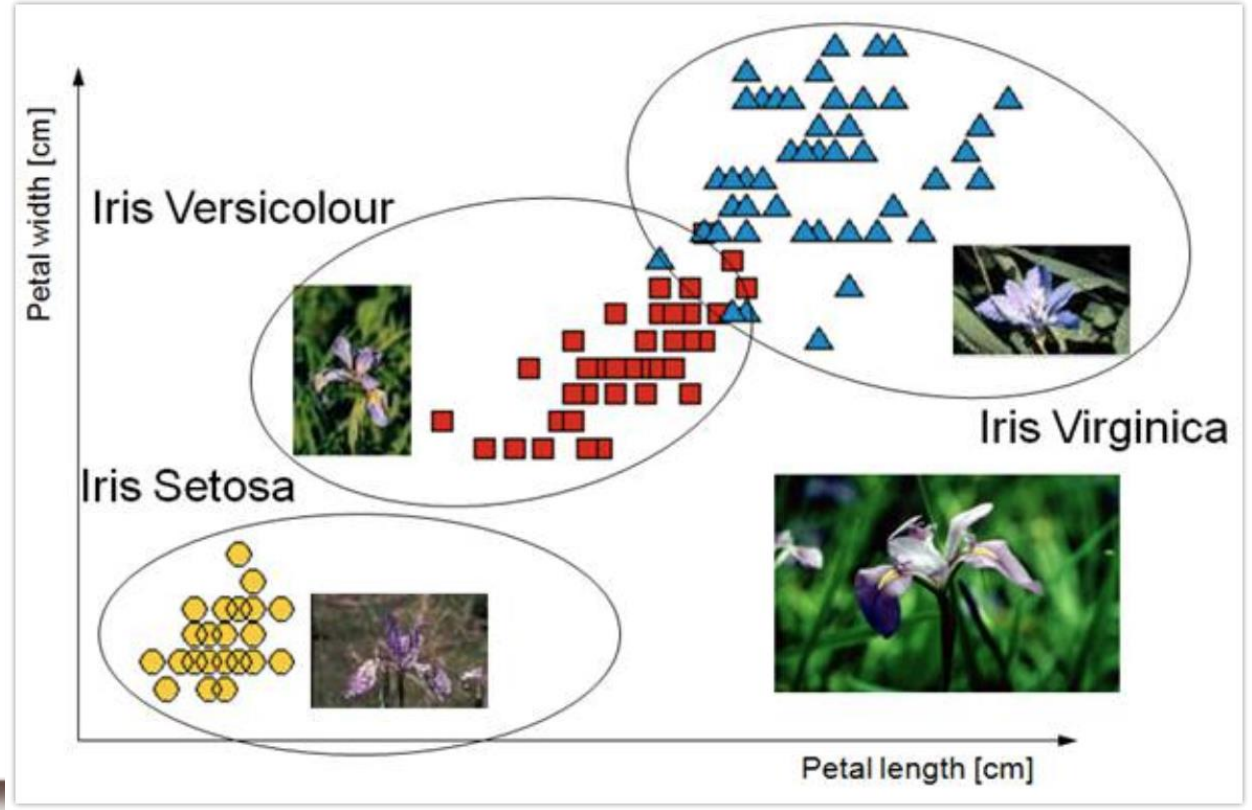
〈붓꽃 데이터 세트 기반의 ML 분류 예측 수행 프로세스〉

1. 의사결정트리

❖붓꽃 품종 예측

➢붓꽃 데이터 세트로 붓꽃 품종 분류

✓꽃잎의 길이와 너비, 꽃받침의 길이와 너비 피처를 기반으로 꽃의 품종 예측



JEONJU UNIVERSITY

1. 의사결정트리

❖ 붓꽃 품종 예측

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

import만 사용하면 모듈 안의 함수를 사용할 때, **모듈명.함수명( )**으로 하고,

from을 사용하면 바로 **함수명( )**으로 사용

모듈: 누군가 만들어 놓은 파이썬 파일(.py)이며, 이를 모아둔 폴더를 패키지

대부분의 패키지는 그 안에 하위 패키지나 모듈을 가지고 있음
이러한 하위 패키지 중에는 상위 패키지를 임포트할 때 자동으로 임포트되는 것도 있지만 자동으로 임포트되지 않는 것도 있음

자동으로 임포트되지 않는 하위 패키지는 수동으로 임포트

```
import 패키지명.모듈명
import 패키지명.하위 패키지명
```


1. 의사결정트리

❖ 붓꽃 품종 예측

```
import pandas as pd

# 붓꽃 데이터 세트를 로딩
iris = load_iris()

# iris.data는 Iris 데이터 세트에서 피처만으로 된 데이터를 numpy로 가지고 있음
iris_data = iris.data

# iris.target은 붓꽃 데이터 세트에서 레이블(결정 값) 데이터를 numpy로 가지고 있음
iris_label = iris.target
print('iris target값:', iris_label)
print('iris target명:', iris.target_names)
```

```
iris target값: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
iris target명: ['setosa' 'versicolor' 'virginica']
```

1. 의사결정트리

❖ 붓꽃 품종 예측

```
# 붓꽃 데이터 세트를 자세히 보기 위해 DataFrame으로 변환

iris_df = pd.DataFrame(data=iris_data, columns=iris.feature_names)
iris_df['label'] = iris.target
iris_df
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

1. 의사결정트리

❖ 붓꽃 품종 예측

```
iris_df.describe()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

1. 의사결정트리

❖ 붓꽃 품종 예측

```
iris_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   label              150 non-null    int32
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
```


1. 의사결정트리

❖ 붓꽃 품종 예측

iris

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
```


1. 의사결정트리

❖ 붓꽃 품종 예측

```
X_train, X_test, y_train, y_test
= train_test_split(iris_data, iris_label, test_size=0.2, random_state=11)


# DecisionTreeClassifier 객체 생성
dt_clf = DecisionTreeClassifier(random_state=11)


# 학습 수행
dt_clf.fit(X_train, y_train)
```

피처데이터세트, 레이블 데이터세트, 테스트 데이터 세트 비율, 호출할 때마다 같은 학습/테스트 용 데이터 세트를 생성하기 위해 주어지는 난수 발생 값

1. 의사결정트리

❖ 붓꽃 품종 예측

```
# 학습이 완료된 DecisionTreeClassifier 객체에서 테스트 데이터 세트로 예측
pred = dt_clf.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print('예측 정확도: {0:.4f}'.format(accuracy_score(y_test,pred)))
```

예측 정확도: 0.9333

1. 의사결정트리

❖ 붓꽃 품종 예측

pred

array([2, 2, 1, 1, 2, 0, 1, 0, 0, 1, 1, 1, 1, 2, 2, 0, 2, 1, 2, 2, 1, 0, 0, 1, 0, 0, 2, 1, 0, 1])

y_test

array([2, 2, 2, 1, 2, 0, 1, 0, 0, 1, 2, 1, 1, 2, 2, 0, 2, 1, 2, 2, 1, 0, 0, 1, 0, 0, 2, 1, 0, 1])

2/30=6.7% 틀림

1. 의사결정트리

❖ 결정트리

➢ 결정트리 모델의 시각화

- ✓ Graphviz 패키지 사용(원래 그래프 기반의 dot 파일로 기술된 다양한 이미지를 쉽게 시각화할 수 있는 패키지 www.graphviz.org)
- ✓ export_graphviz() API 를 제공 : 함수인자로 학습이 완료된 Estimator, 피처의 이름 리스트, 레이블 이름 리스트
- ✓ 윈도우 버전의 Graphviz 를 설치(https://graphviz.org/download/ graphviz-6.0.1(64-bit) EXE installer [sha256])
- ✓ Graphviz 의 파이썬 래퍼 모듈을 PIP 를 이용해 관리자 권한으로 설치 (C:₩Windows₩system32>pip install grahpviz)
- ✓ 윈도우 사용자 변수의 경로에 C:₩Program Files₩Graphviz₩bin 추가
- ✓ 시스템 변수의 경로에 C:₩Program Files₩Graphviz₩bin₩dot.exe 추가

1. 의사결정트리

❖ 결정트리

➢ 결정트리 모델의 시각화

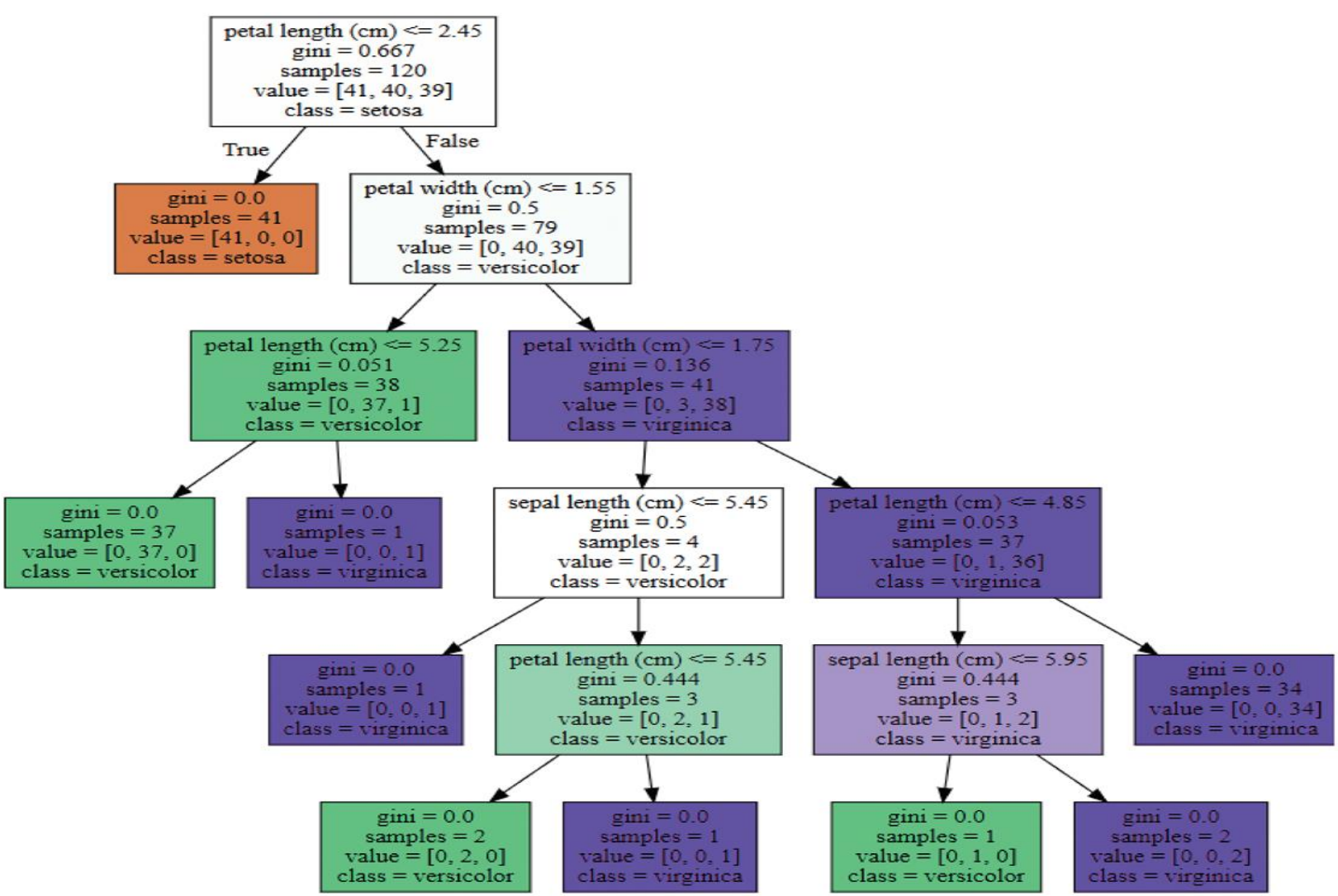
```
from sklearn.tree import export_graphviz

# export_graphviz()의 호출 결과로 out_file로 지정된 tree.dot 파일을 생성함.
export_graphviz(dt_clf, out_file="tree.dot",
class_names=iris_data.target_names , feature_names =
iris_data.feature_names, impurity=True, filled=True)
```

```
import graphviz

# 위에서 생성된 tree.dot 파일을 Graphviz 읽어서 Jupyter Notebook상에서 시각화
with open("tree.dot") as f:
    dot_graph = f.read()
graphviz.Source(dot_graph)
```


1. 의사결정트리


petal length (cm) <= 2.45
gini = 0.667
samples = 120
value = [41, 40, 39]
class = setosa
True
False
gini = 0.0
samples = 41
value = [41, 0, 0]
class = setosa
petal width (cm) <= 1.55
gini = 0.5
samples = 79
value = [0, 40, 39]
class = versicolor
petal length (cm) <= 5.25
gini = 0.051
samples = 38
value = [0, 37, 1]
class = versicolor
petal width (cm) <= 1.75
gini = 0.136
samples = 41
value = [0, 3, 38]
class = virginica
gini = 0.0
samples = 37
value = [0, 37, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
sepal length (cm) <= 5.45
gini = 0.5
samples = 4
value = [0, 2, 2]
class = versicolor
petal length (cm) <= 4.85
gini = 0.053
samples = 37
value = [0, 1, 36]
class = virginica
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
petal length (cm) <= 5.45
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
sepal length (cm) <= 5.95
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 34
value = [0, 0, 34]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica

1. 의사결정트리

❖ 의사결정트리

➢ 결정트리 모델의 시각화

✓ Feature Importance 시각화

- 학습을 통해 규칙을 정하는 데 있어 피처의 중요도를 DecisionTreeClassifier 객체의 feature_importances 속성으로 확인

  →기본적으로 ndarray형태로 값을 반환하며 피처 순서대로 값이 할당

```
# feature importance 추출
print("Feature Importances:\n{0}\n".format(np.round(dt_clf.feature_importances_, 3)))

# feature 별 feature importance 매핑
for name, value in zip(iris_data.feature_names, dt_clf.feature_importances_):
    print('{0}: {1:.3f}'.format(name, value))

# feature importance 시각화
sns.barplot(x=dt_clf.feature_importances_, y=iris_data.feature_names)
```

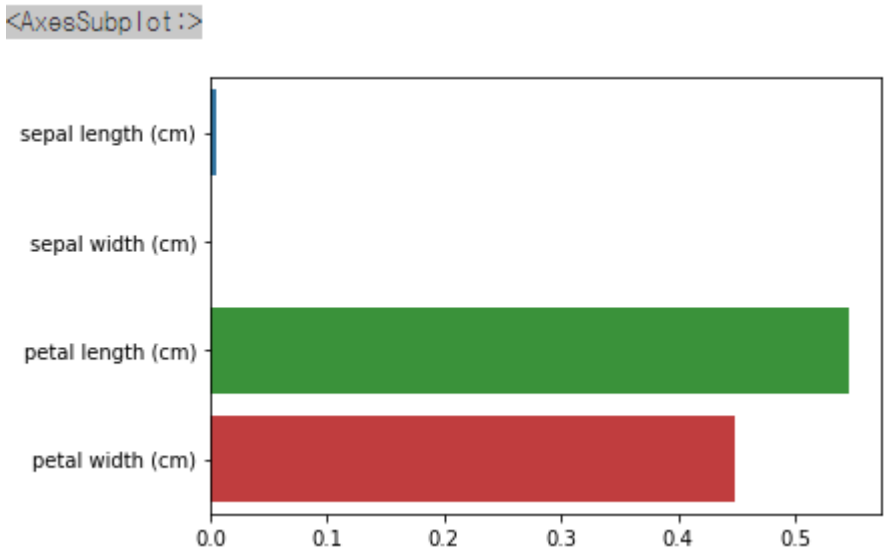

❖ 의사결정트리

➢ 결정트리 모델의 시각화

✓ Feature Importance 시각화

Feature Importances:
[0.006 0. 0.546 0.448]

sepal length (cm): 0.006
sepal width (cm): 0.000
petal length (cm): 0.546
petal width (cm): 0.448
<AxesSubplot:>



1. 의사결정트리

❖ 결정트리

➢ 파라미터(Parameter)

- ✓ 머신러닝에서 사용되는 파라미터는 모델 파라미터라고도 하며, 모델에 적용할 하나 이상의 파라미터를 사용하여 새로운 샘플에 대한 예측을 하기 위해 사용
- ✓ 즉, 머신러닝 훈련 모델에 의해 요구되는 변수

➢ 파라미터의 특징

- ✓ 예측 모델은 새로운 샘플을 주어지면 무엇을 예측할지 결정할 수 있도록 파라미터를 필요로 함
- ✓ 머신러닝 훈련 모델의 성능은 파라미터에 의해 결정
- ✓ 파라미터는 데이터로부터 추정 또는 학습
- ✓ 파라미터는 개발자에 의해 수동으로 설정하지 않음(임의로 조정이 불가능하다)
- ✓ 학습된 모델의 일부로 저장된다.

➢ 모델 파라미터의 예

- ✓ 선형 회귀 또는 로지스틱 회귀에서의 결정계수, 인공신경망의 가중치, SVM(Support Vector Machine)의 서포트 벡터,

1. 의사결정트리

❖ 결정트리

➢ 하이퍼파라미터(Hyperparameter)

- ✓ 머신러닝에서 하이퍼파라미터는 최적의 훈련 모델을 구현하기 위해 모델에 설정하는 변수로 학습률(Learning Rate), 에포크 수(훈련 반복 횟수), 가중치 초기화 등을 결정할 수 있습니다. 또한 하이퍼파라미터 튜닝 기법을 적용하여 훈련 모델의 최적값들을 찾을 수 있습니다.

➢ 하이퍼파라미터의 특징

- ✓ 모델의 매개 변수를 추정하는 데 도움이 되는 프로세스에서 사용된다.
- ✓ 하이퍼파라미터는 개발자에 의해 수동으로 설정할 수 있다.(임의 조정 가능)
- ✓ 학습 알고리즘의 샘플에 대한 일반화를 위해 조절된다.

➢ 하이퍼파라미터의 예

- ✓ 학습률, 손실 함수, 일반화 파라미터, 미니배치 크기, 에포크 수
- ✓ 가중치 초기화, 은닉층의 개수, k-NN의 k값

1. 의사결정트리

❖ 결정트리

- ➢ 하이퍼파라미터의 튜닝 기법
  - ✓ 그리드 탐색
  - ✓ 랜덤 탐색
  - ✓ 베이지안 최적화
  - ✓ 휴리스틱 탐색
- ➢ 모델 파라미터는 새로운 샘플이 주어지면 무엇을 예측할지 결정하기 위해 사용하는 것이며 학습 모델에 의해 결정
- ➢ 하이퍼파라미터는 학습 알고리즘 자체의 파라미터로 모델이 새로운 샘플에 잘 일반화 되도록 하이퍼파라미터들의 최적값을 찾으나, 데이터 분석 결과로 얻어지는 값이 아니므로 절대적인 최적값은 존재하지 않고, 사용자가 직접 설정

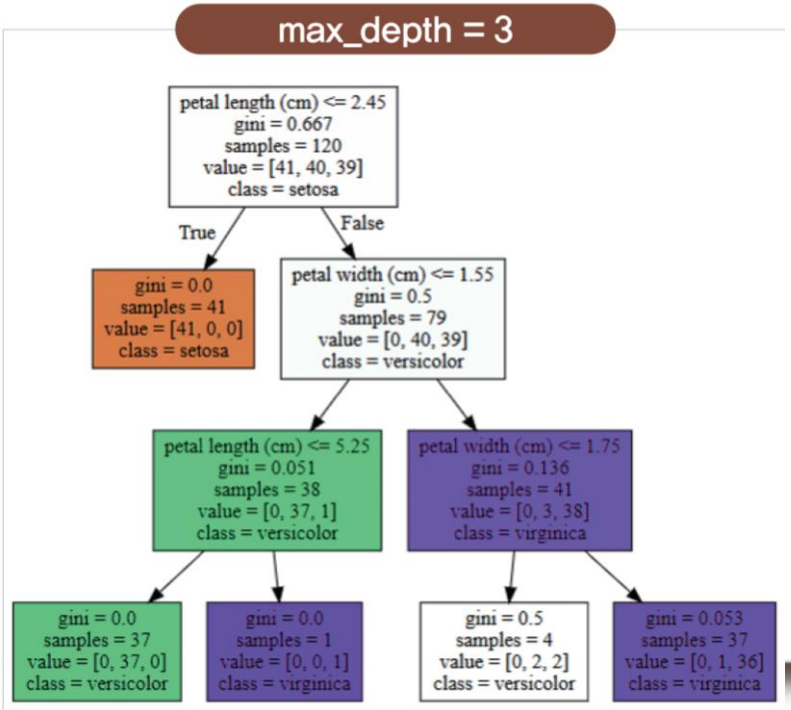
1. 의사결정트리

결정트리 알고리즘을 제어하는 대부분 하이퍼 파라미터는 복잡한 트리가 생성되는 것을 막기 위한 용도

```
# dt_clf = DecisionTreeClassifier(max_depth=3 ,random_state=156)
dt_clf.fit(X_train, y_train)

export_graphviz(dt_clf, out_file="tree.dot", class_names = iris_data.target_names,
                feature_names = iris_data.feature_names, impurity=True, filled=True)

import graphviz
with open("tree.dot") as f:
    dot_graph = f.read()
graphviz.Source(dot_graph)
```



1. 의사결정트리

❖ 의사결정트리

➢ Decision Tree의 과적합(Overfitting)

✓ 임의의 데이터 세트를 통한 과적합 문제 시각화

✓ make_classification 함수는 설정에 따른 분류용 가상 데이터를 생성하는 명령

✓ 인수:

- n_samples : 표본 데이터의 수, 디폴트 100
- n_features : 독립 변수의 수, 디폴트 20
- n_informative : 독립 변수 중 종속 변수와 상관 관계가 있는 성분의 수, 디폴트 2
- n_redundant : 독립 변수 중 다른 독립 변수의 선형 조합으로 나타나는 성분의 수, 디폴트 2

❖ 의사결정트리

➢ Decision Tree의 과적합(Overfitting)

✓ 인수:

- n_repeated : 독립 변수 중 단순 중복된 성분의 수, 디폴트 0
- n_classes : 종속 변수의 클래스 수, 디폴트 2
- n_clusters_per_class : 클래스 당 클러스터의 수, 디폴트 2
- weights : 각 클래스에 할당된 표본 수
- random_state : 난수 발생 시드

✓ 반환값:

- X : [n_samples, n_features] 크기의 배열
- y : [n_samples] 크기의 배열

❖ 의사결정트리

➢ Decision Tree의 과적합(Overfitting)

✓ Decision Tree의 과적합을 줄이기 위한 파라미터 튜닝

(1) max_depth 를 줄여서 트리의 깊이 제한

(2) min_samples_split 를 높여서 데이터가 분할하는데 필요한 샘플 데이터의 수를 높이기

(3) min_samples_leaf 를 높여서 말단 노드가 되는데 필요한 샘플 데이터의 수를 높이기

(4) max_features를 높여서 분할을 하는데 고려하는 feature의 수 제한

1. 의사결정트리

❖ 결정트리

➢ GridSearchCV 란?

- ✓ 사이킷런에서는 분류 알고리즘이나 회귀 알고리즘에 사용되는 하이퍼파라미터를 순차적으로 입력해 학습을 하고 측정을 하면서 가장 좋은 파라미터를 알려줌
- ✓ GridSearchCV가 없다면 max_depth 가 3일때 가장 최적의 스코어를 뽑아내는지 1일때 가장 최적인 스코어를 뽑아내는지 일일이 학습을 해야 하지만 grid 파라미터 안에서 집합을 만들고 적용하면 최적화된 파라미터를 뽑아낼 수 있음

➢ GridSearchCV 클래스의 생성자 정리

- ✓ estimator : classifier, regressor, pipeline 등 가능
- ✓ param_grid : 튜닝을 위해 파라미터, 사용될 파라미터를 dictionary 형태로 입력
- ✓ scoring : 예측 성능을 측정할 평가 방법. 보통 accuracy 로 지정하여 정확도로 성능 평가
- ✓ cv : 교차 검증에서 몇개로 분할되는지 지정
- ✓ refit : True가 디폴트로 True로 하면 최적의 하이퍼 파라미터를 찾아서 재학습 시킴

1. 의사결정트리

❖ 의사결정트리

➢ GridSearchCV 란?

```
params = {
    'max_depth' : [ 8 , 12, 16 ,20],
    'min_samples_split' : [16, 24],
}


grid_cv = GridSearchCV(dt_clf, param_grid=params, scoring='accuracy', cv=5, verbose=1 )
grid_cv.fit(X_train , y_train)
print('GridSearchCV 최고 평균 정확도 수치: {0:.4f}'.format(grid_cv.best_score_))
print('GridSearchCV 최적 하이퍼 파라미터:', grid_cv.best_params_)
```


Thank You !

JEONJU UNIVERSITY
The Place for Superstars