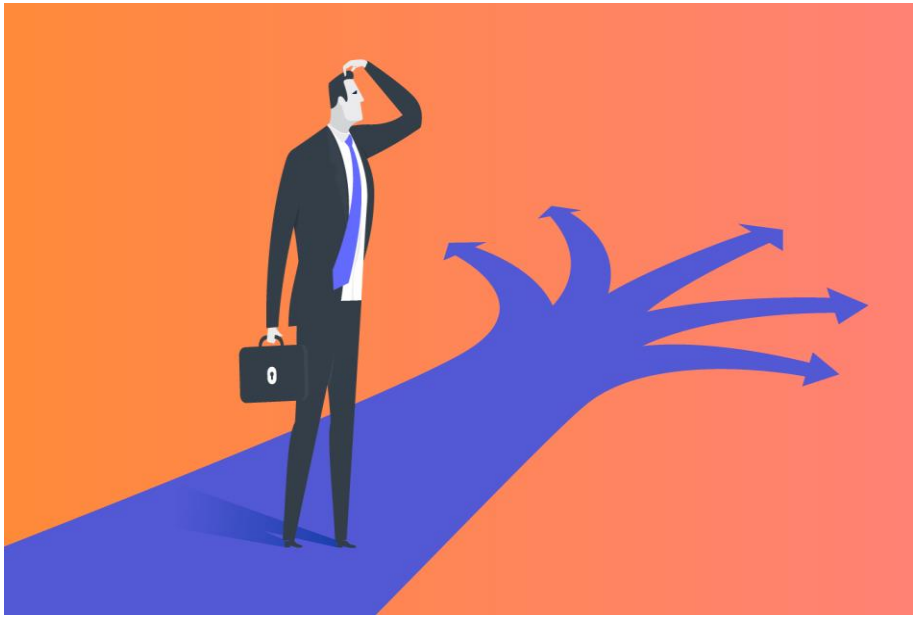
8. Decision Tree

1. Introduction

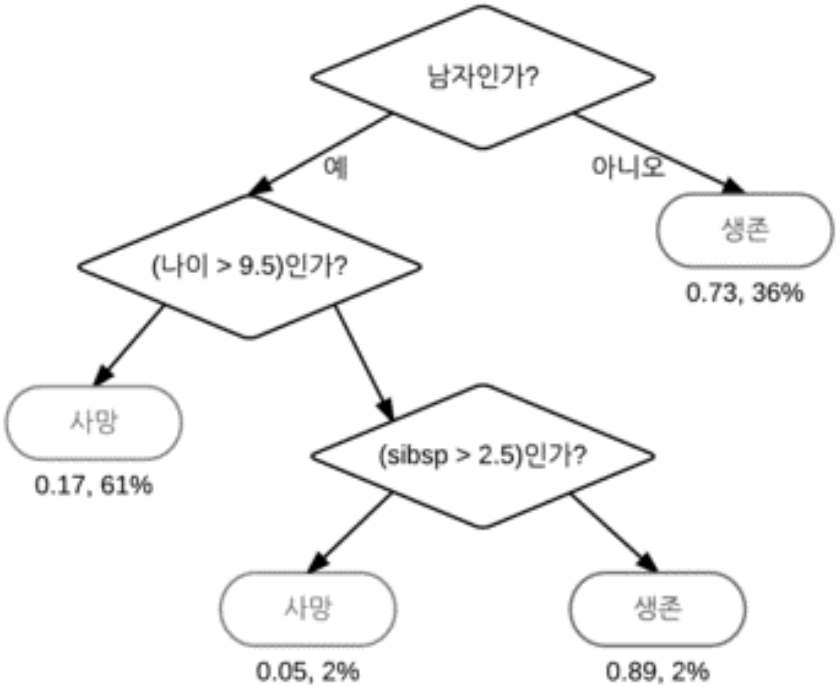
Decision Tree(의사 결정 나무)란?



Decision Tree(의사 결정 나무)란?

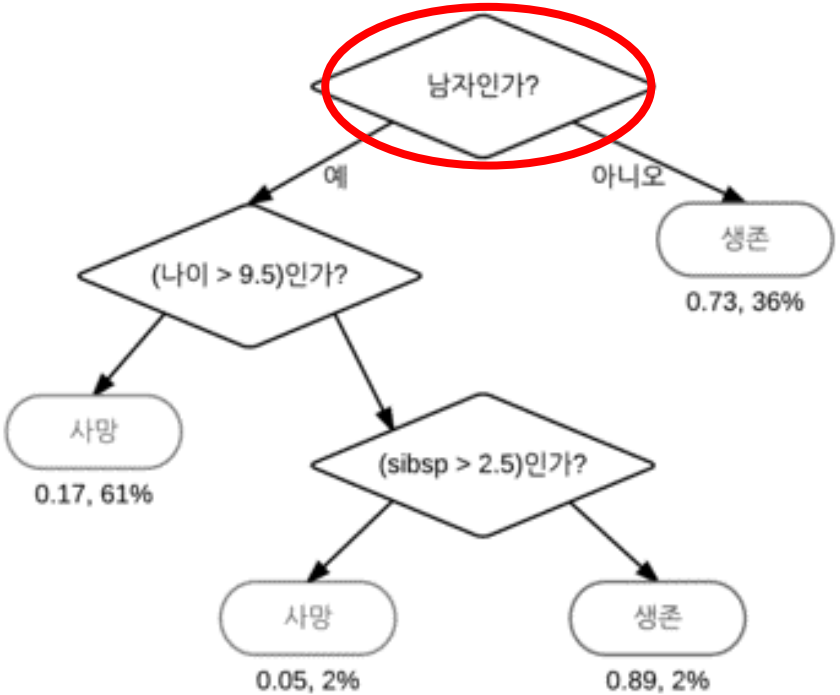
- 순차적으로 질문을 던져 이에 대한 답을 고르며 의사결정을 하는 모형

- 스무고개와 비슷함.



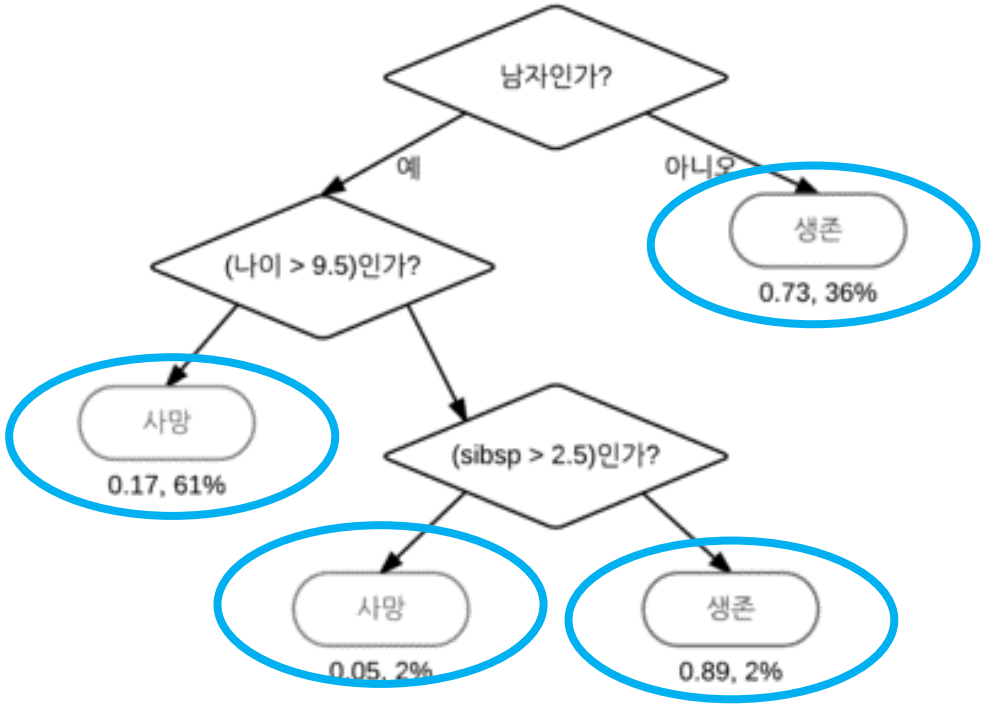
Decision Tree의 구조

- Root node
- Leaf node
- Parent node
- Child node



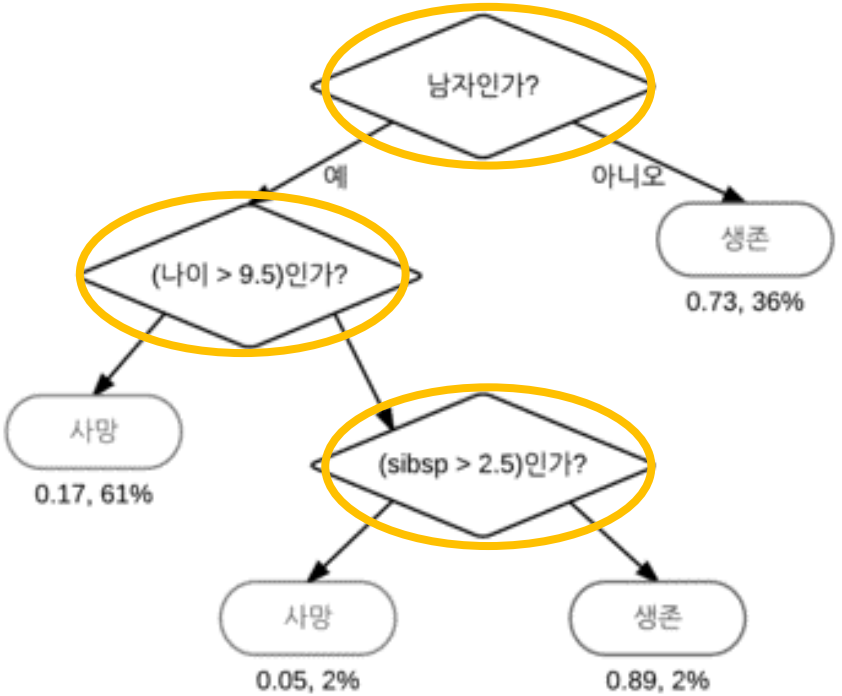
Decision Tree의 구조

- Root node
- Leaf node
- Parent node
- Child node



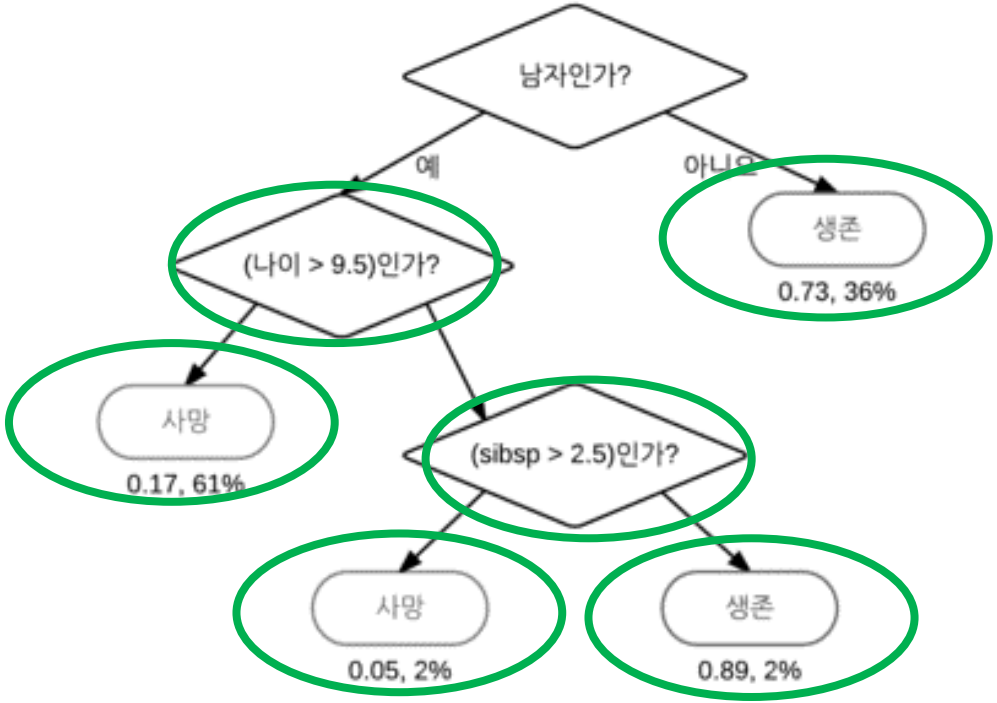
Decision Tree의 구조

- Root node
- Leaf node
- Parent node
- Child node

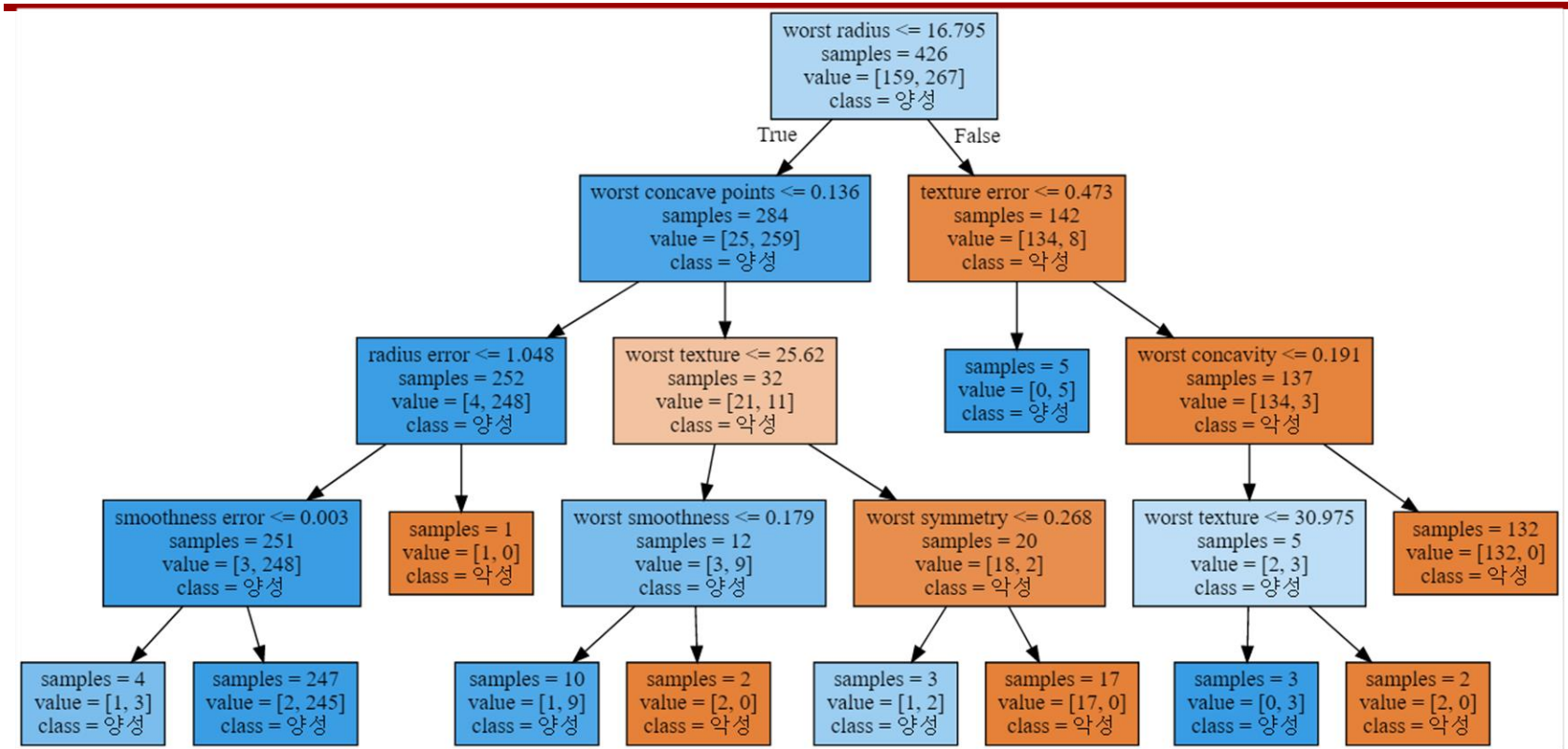


Decision Tree의 구조

- Root node
- Leaf node
- Parent node
- Child node



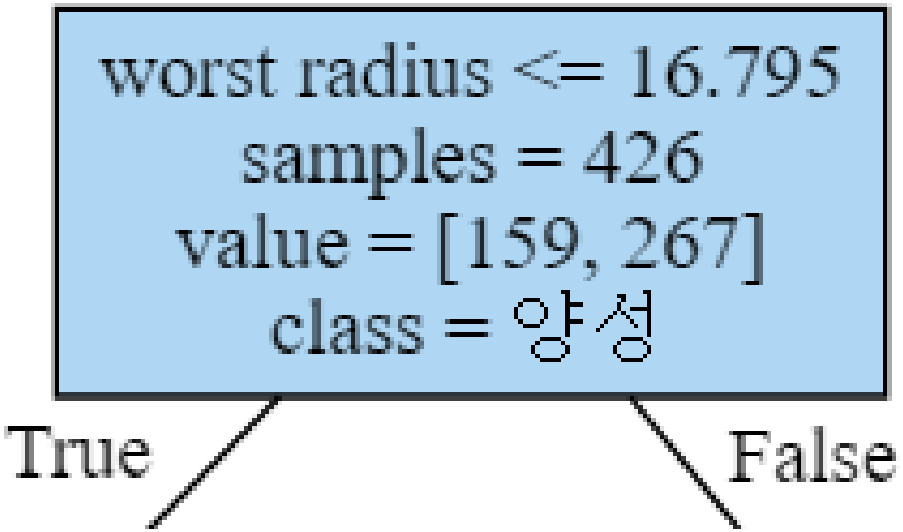
유방암 데이터


worst radius <= 16.795
samples = 426
value = [159, 267]
class = 양성
True
False
worst concave points <= 0.136
samples = 284
value = [25, 259]
class = 양성
texture error <= 0.473
samples = 142
value = [134, 8]
class = 악성
radius error <= 1.048
samples = 252
value = [4, 248]
class = 양성
worst texture <= 25.62
samples = 32
value = [21, 11]
class = 악성
samples = 5
value = [0, 5]
class = 양성
worst concavity <= 0.191
samples = 137
value = [134, 3]
class = 악성
smoothness error <= 0.003
samples = 251
value = [3, 248]
class = 양성
samples = 1
value = [1, 0]
class = 악성
worst smoothness <= 0.179
samples = 12
value = [3, 9]
class = 양성
worst symmetry <= 0.268
samples = 20
value = [18, 2]
class = 악성
worst texture <= 30.975
samples = 5
value = [2, 3]
class = 양성
samples = 132
value = [132, 0]
class = 악성
samples = 4
value = [1, 3]
class = 양성
samples = 247
value = [2, 245]
class = 양성
samples = 10
value = [1, 9]
class = 양성
samples = 2
value = [2, 0]
class = 악성
samples = 3
value = [1, 2]
class = 양성
samples = 17
value = [17, 0]
class = 악성
samples = 3
value = [0, 3]
class = 양성
samples = 2
value = [2, 0]
class = 악성

유방암 데이터 - Root node

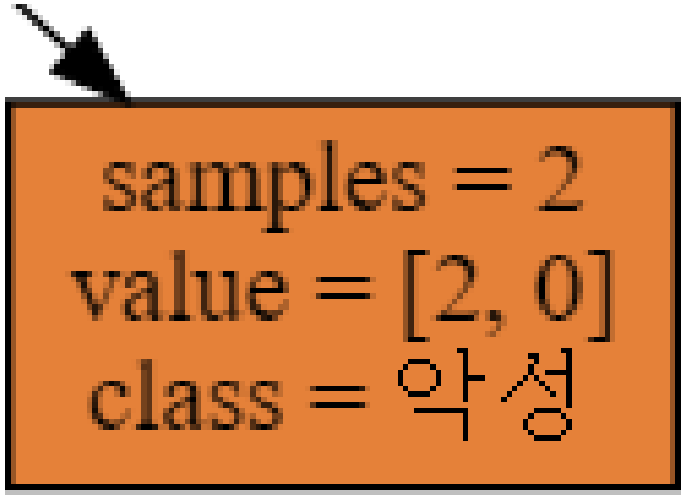
- 샘플 : 426개
- 악성 샘플 : 159개
- 양성 샘플 : 267개
- 분류 기준

  : worst radius가 16.795 이하인지



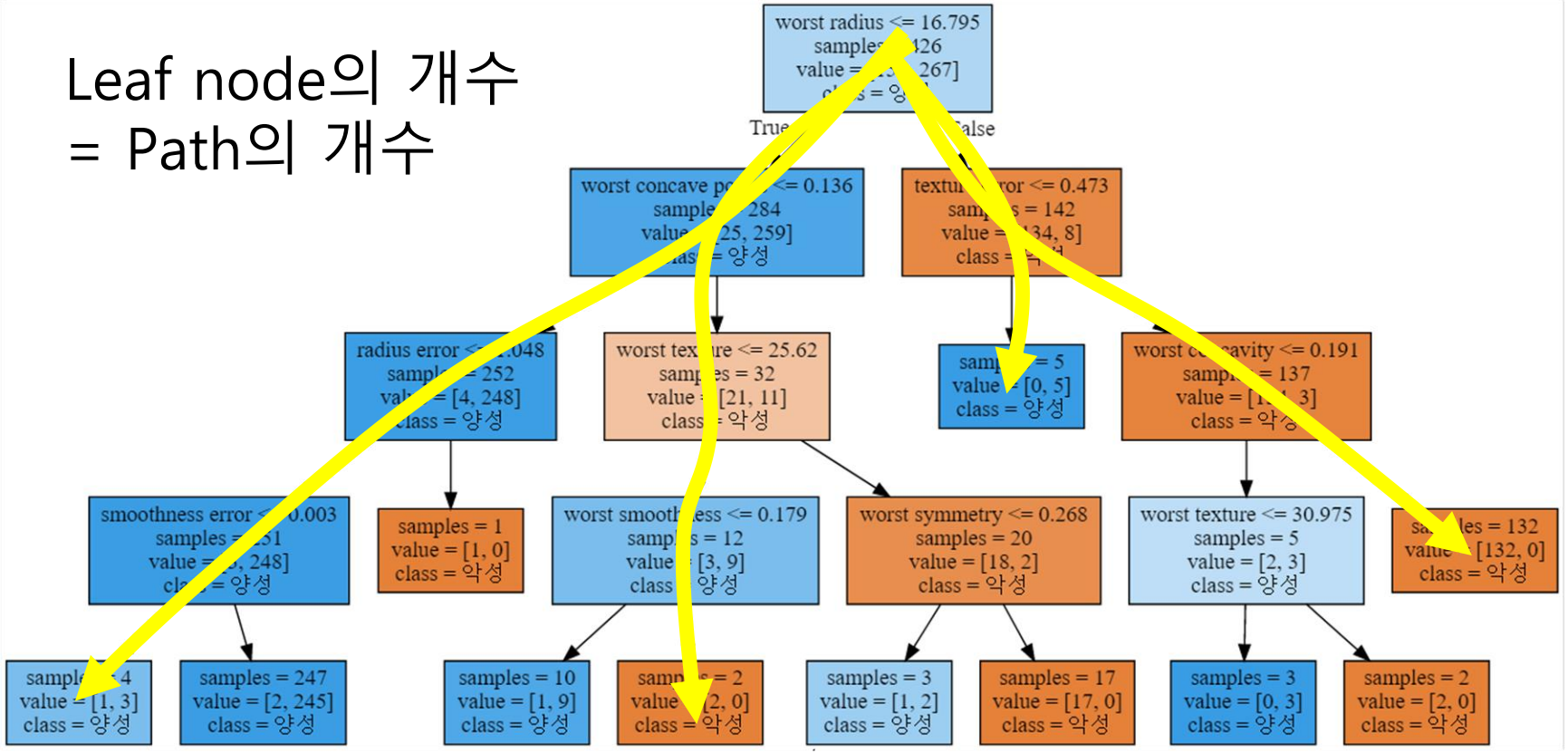
유방암 데이터 - Leaf node

- 샘플 : 2개
- 악성 샘플 : 2개
- 양성 샘플 : 0개
- 더 이상 분기가 일어나지 않음.



유방암 데이터 - Path

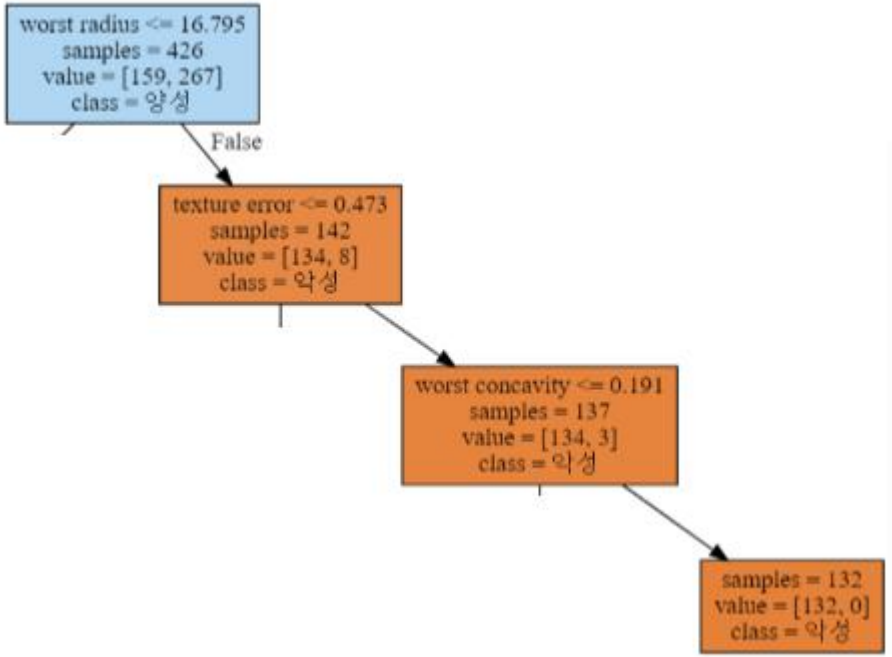
Leaf node의 개수
= Path의 개수



KU-BIG

유방암 데이터 - Path 수식화

(worst radius > 16.795 ^
texture error > 0.473 ^
worst concavity > 0.191)
→ 악성



2. 불순도 지표

불순도 지표

- 불순도(impurity)가 감소하고 순도(homogeneity)가 증가하는 방식으로 학습
- 매 노드마다 각각의 테스트의 결과들을 비교해서 가장 동질적인 집합을 갖는 child node들이 나오도록 테스트

동질적으로 질서정연하게 분류가 되었다는 것에 대한 척도

엔트로피(Entropy) - 데이터셋 전체(root)

$$H(S) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- S : 데이터셋
- $C_1, C_2, \ldots, C_n$: n개의 클래스
- p_i : 한 데이터 포인트가 클래스 C_i에 속할 확률

엔트로피(Entropy) - 분할된 데이터셋

$$H = \sum_{i=1}^{m} q_i \, H(S_i)$$

- S_i : 데이터셋의 파티션
- q_i : 파티션 S_i이 차지하는 비율
- $H(S_i)$: 파티션 S_i 각각의 엔트로피

엔트로피(Entropy) - Information gain

- 각 분기별 변수 선택 후 분류하는 방법.

$$GAIN_{entropy} = Entropy_{parent} - \sum_{i=1}^{m} q_i Entropy_{child}(k)$$

=> 과적합(overfitting) 발생!

엔트로피(Entropy) - Gain Ratio

$$GAINRATIO_{split} = \frac{GAIN_{entropy}}{SPlitINFO}, SPlitINFO = -\sum_{i=1}^{m} q_i \log q_i$$

- Information gain에 Split INFO를 penalty로 줌
  => 파티션을 많이 쪼갤수록 GAIN값이 감소

지니불순도(Gini Impurity)

- 가정 : 모집단이 완전히 동질적인 원소들로 구성 되어있을 때 랜덤하게 복원추출한 두 원소가 동일한 확률은 1이다.
- 정의 : 불순도를 나타내는 계수
- 특징 : 1. 잘못 분류될 확률을 최소화하는 기준
  2. 크기가 클수록 불순도가 높음.

지니불순도(Gini Impurity) - 데이터셋 전체(root)

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

- p_i : i번째 class에 속하는 원소들의 비율

지니불순도(Gini Impurity) - 분할된 데이터셋

$$G = \sum_{k=1}^{m} R_k \sum_{i=1}^{n} p_{ki}(1 - p_{ki}) = \sum_{k=1}^{m} R_k \, G(S_k)$$

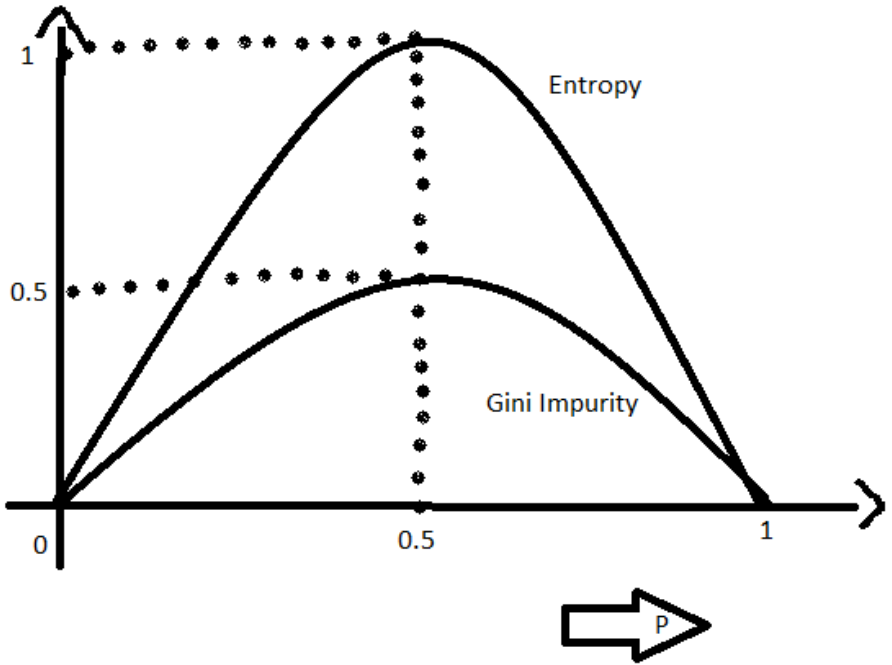
- R_k: k번째 파티션이 데이터셋에서 차지하는 비율
- p_{ki} : k번째 파티션 내에서 i번째 class가 차지하는 비율

지니불순도(Gini Impurity) - Gain

$$GAIN_{Gini} = Gini_{parent} - \sum_{k=1}^{m} R_k Gini_{child}(k)$$

엔트로피와 지니불순도 비교

엔트로피가 지니불순도 보다 차이가 더 뚜렷하게 나타남.
=> 최근에는 엔트로피를 이용함.



3. 가지치기(Pruning)

문제점

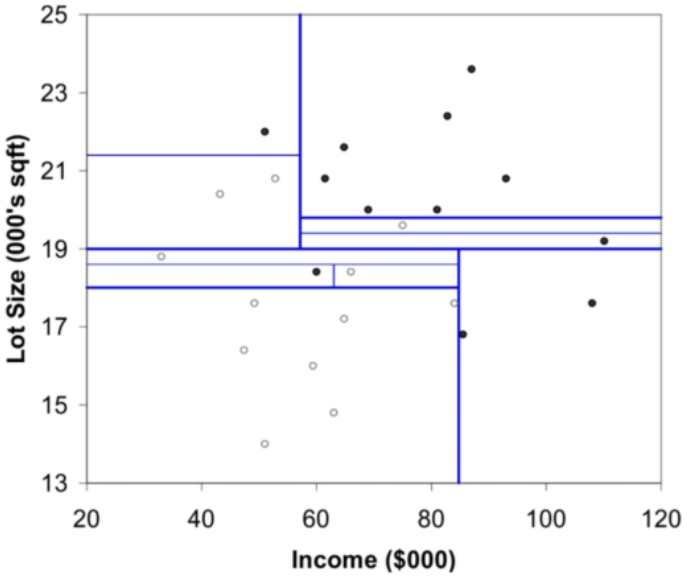
- 앞서 살펴본 불순도 지표를 통해 최적의 결정트리를 선택하고자 함.
- 그러나 불순도를 최소화하여도 과적합(Overfitting) 문제가 발생하게 됨.
- 즉, 새로운 데이터에서의 트리의 정확도가 떨어질 수 있음.

가지치기!!

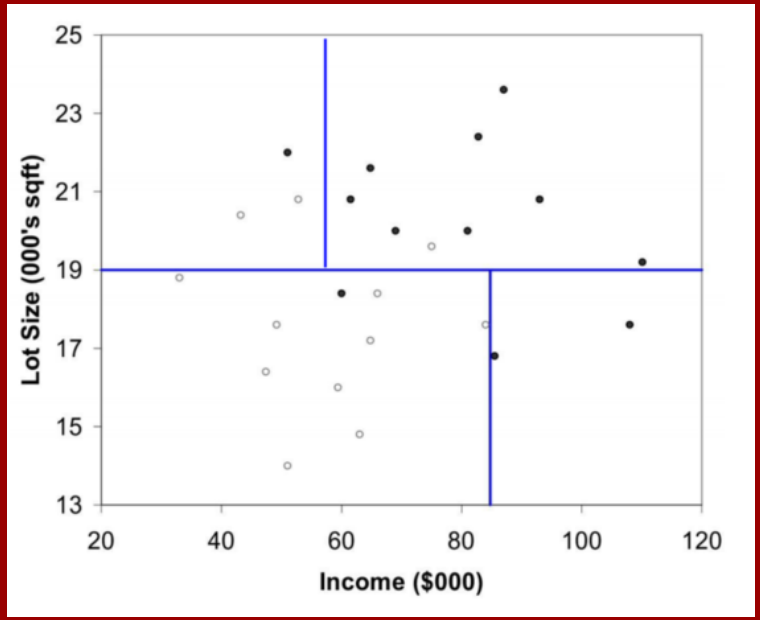
가지치기(Pruning)란?

- 정의 : 모델이 과적합 되지 않도록 tree의 노드 수를 의도적으로 줄이는 작업.
- **1. 사전 가지치기**
  : 트리의 최대 깊이나 리프의 최대 개수를 제한, 노드가 분할하기 위한 포인트의 최소 개수 지정

  2. 사후 가지치기
  : 트리를 만든 후 데이터 포인트가 적은 노드 삭제/병합



Overfitting, Pruning X



Overfitting, Pruning X

사전 가지치기(Pre-pruning)

- Fully grown tree를 만들기 전에 알고리즘을 STOP
- 정지 조건을 증가시킴.

=> 주로 현재 노드를 분기해도 impurity가 더이상 향상되지 않을 때 멈춘다는 조건을 사용한다.

사후 가지치기(Post-pruning)

- 결정나무를 완전히 자라게 한 후, 불필요한 가지를 CUT
- 진행 조건 : 1. sub tree를 leaf node로 합칠 때
  2. 이전보다 *error가 줄어들 때

사후 가지치기(Post-pruning)

* misclassification error를 변형한 generalized error(pessimistic error)

=> $error_{gen}(Train) = error(Train) + \Omega \times \frac{k}{N_{train}}$

- Ω : 분석자 임의로 선택
- K : leaf node의 개수
- N_{train} : train data의 개수
- 모델이 복잡 : error ↑ = overfitting 발생 확률 ↑

4. 결정트리의 장·단점

〈장점〉

- 모델 시각화가 간단함
- 데이터 분할 시 데이터 스케일의 영향X
- 표준화/정규화 같은 전처리 필요X
- 계산복잡성 대비 상대적으로 높은 성능

〈장점〉

- 특정 데이터에만 잘 작동할 가능성 高
- 가지치기를 해도 overfitting됨.
- 일반화 성능이 BAD


Thank You!