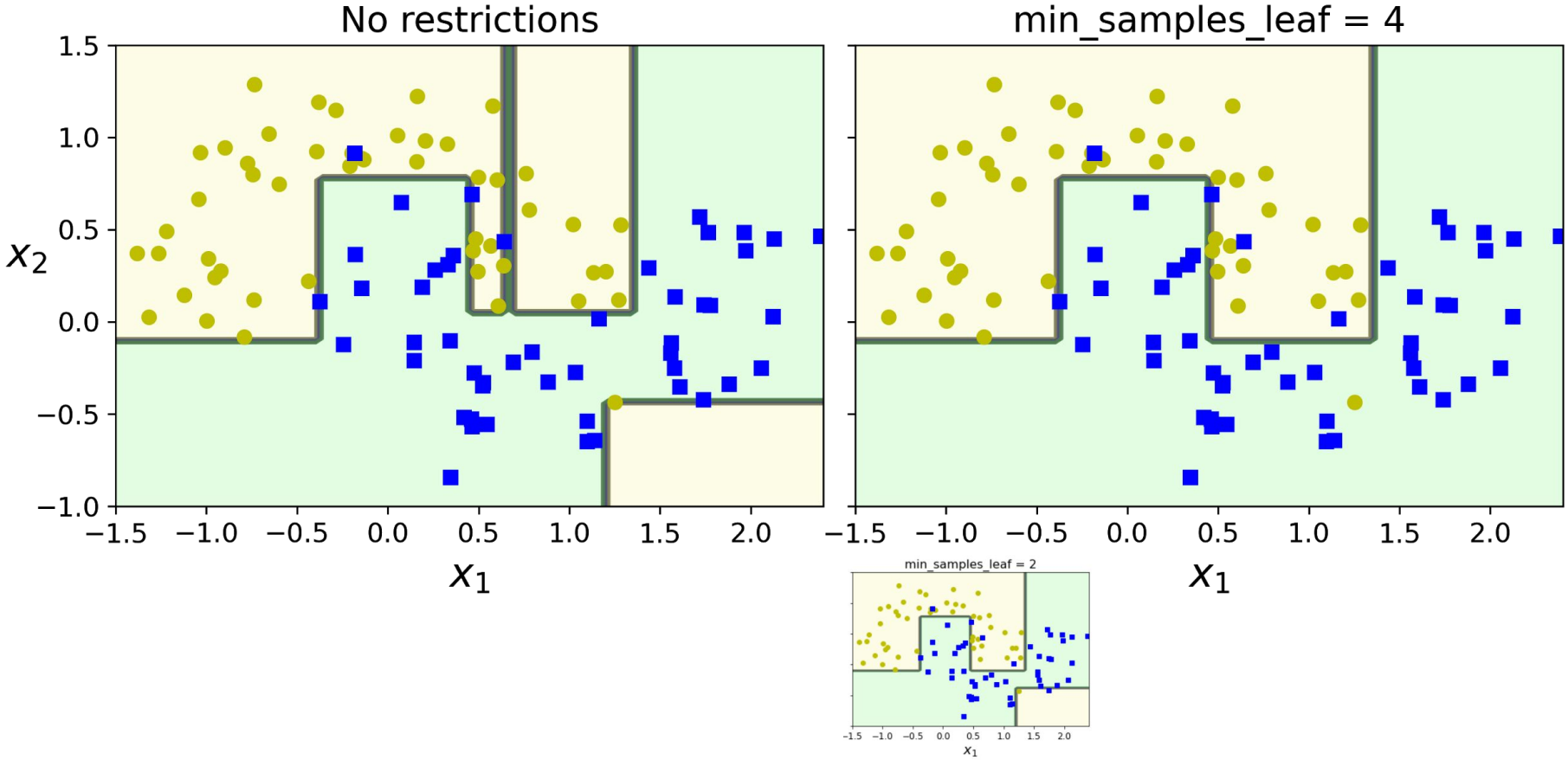
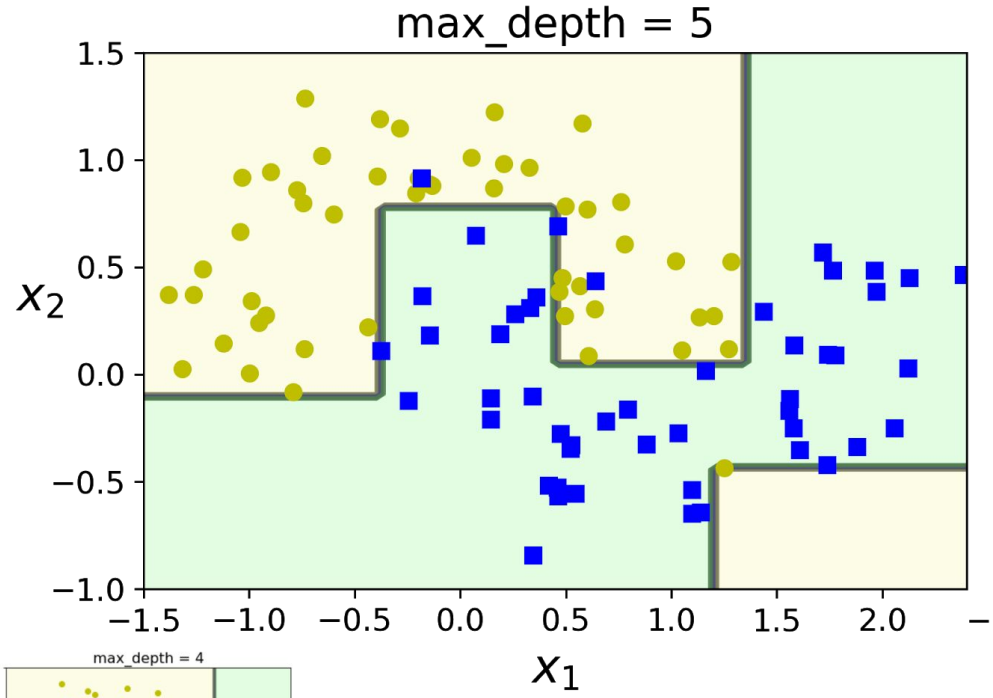
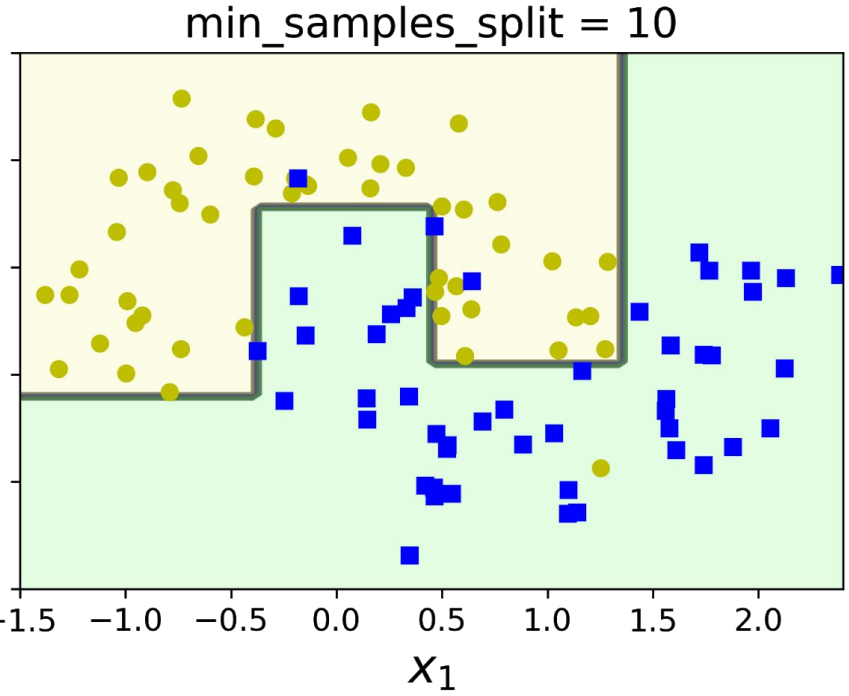
DecisionTreeClassifier

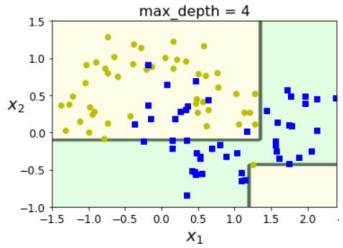
조성진



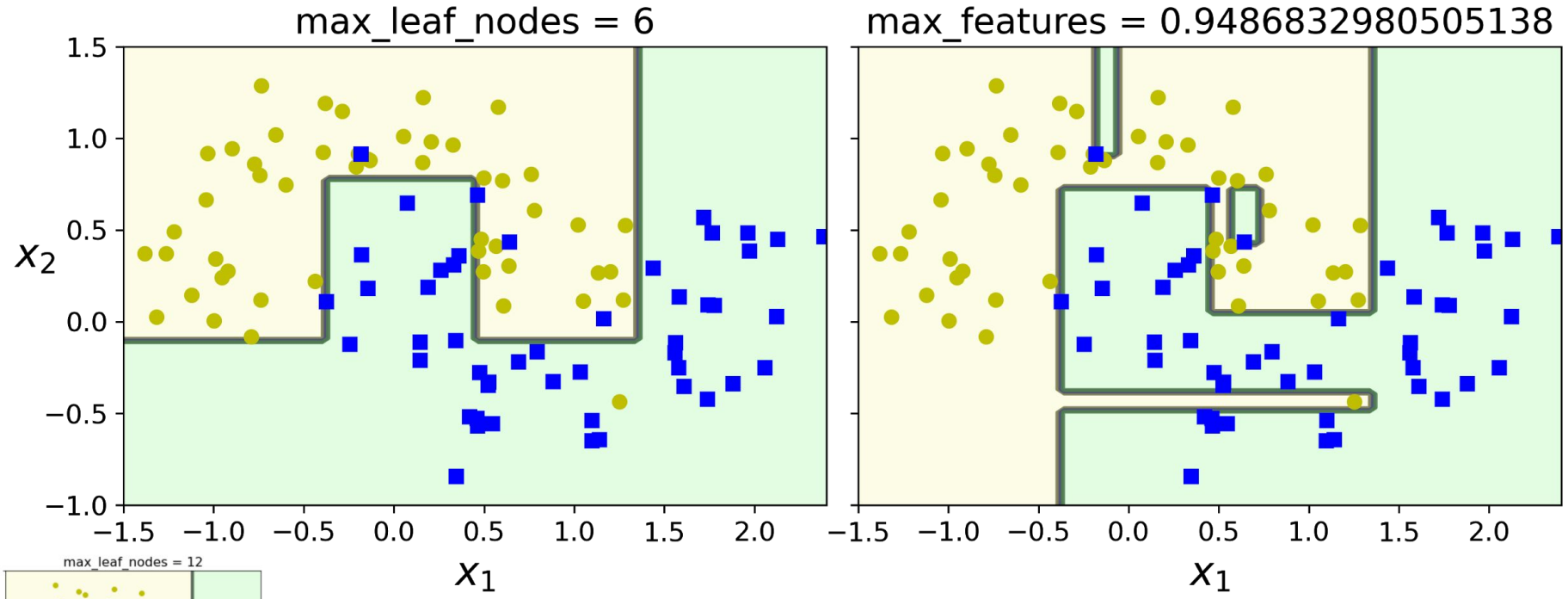
조성진

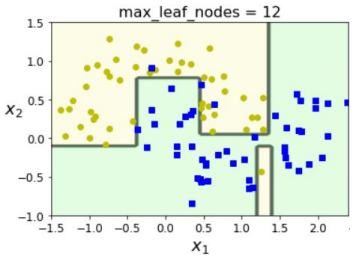






조성진


max_leaf_nodes = 6
max_features = 0.9486832980505138
x_1
x_2


max_leaf_nodes = 12
x_1
x_2

min_samples_split = 10
최소 나눠지는 집합에 포함되는 원소 수 10개 되도록

max_features: 각 노드에서 분할에 사용할 특성의 최대 수
If float, then max_features is a fraction and int(max_features * n_features) features are considered at each split.

CART 알고리즘이란

식 6-1: 지니 불순도

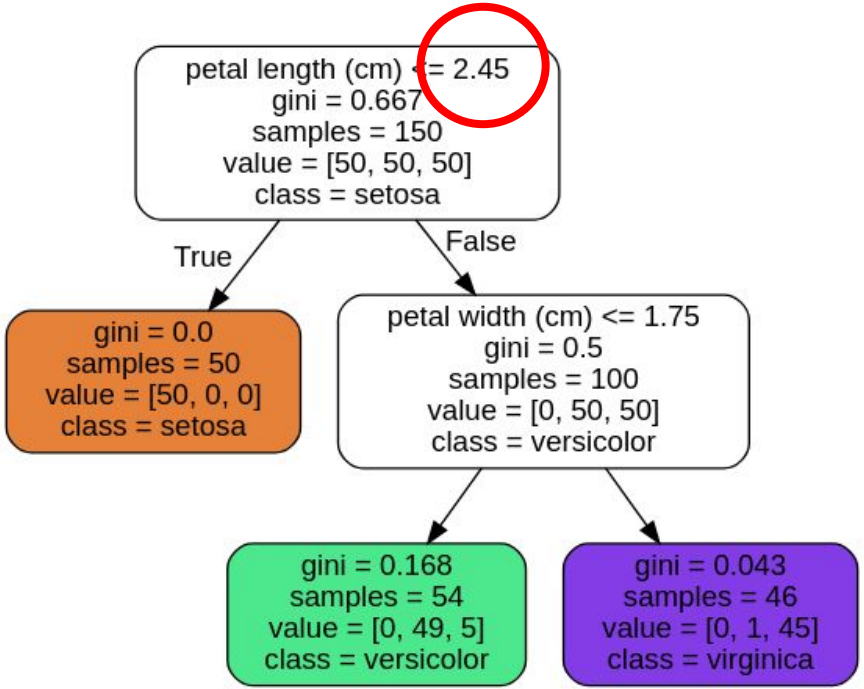
$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

CART 알고리즘은

CART 알고리즘은 지니 지수(Gini Index) 또는 분산의 감소량을 사용하여 나무의 가지를 이진(Binary) 분리한다. (범주형 변수에 대해서는 지니 지수를 사용하고, 연속형 변수에 대해서는 분산의 감소량을 사용한다.)

의사결정나무는 Gini Index가 작아지는 방향으로 움직이며 Gini Index값을 가장 많이 감소시켜 주는 변수가 영향을 가장 많이 끼치는 변수가 된다.

2.45라는 값을 결정하는 기준이 궁금해서 찾아봤는데 모든 조합의 수를 계산한 후 지니 불순도가 가장 낮은 값을 선택한다.



1) https://tyami.github.io/machine%20learning/decision-tree-4-CART/
2)https://leedakyeong.tistory.com/entry/%EC%9D%98%EC%82%AC%EA%B2%B0%EC%A0%95%EB%82%98%EB%AC%B4Decision-Tree-CART-%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98-%EC%A7%80%EB%8B%88%EA%B3%84%EC%88%98Gini-Index%EB%9E%80

CART 알고리즘이란

모든 조합의 수를 보는 것이다

petal length 2.45, 2.3, 2.5, … -> 모두 계산해서 최소값 구해서 leaf node로 나아간다

백서윤

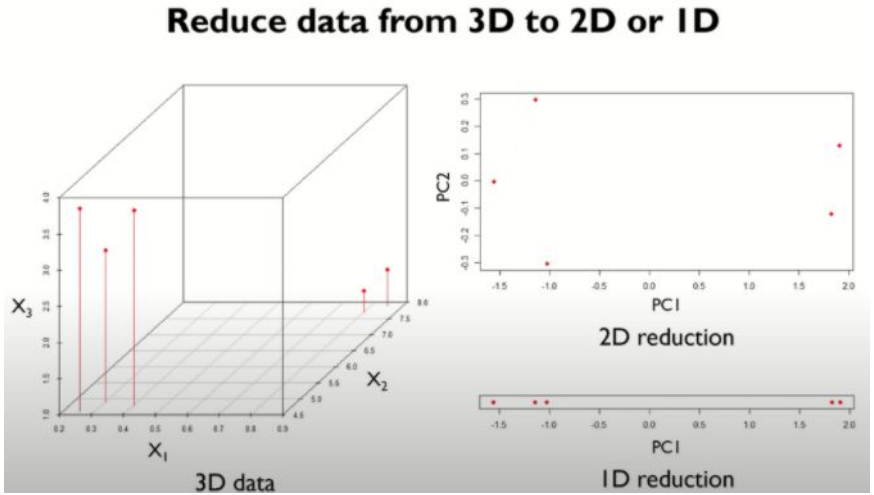
PCA (Principle Component Analysis) - 주성분 분석

차원 축소
: 고차원 데이터 -> 저차원 데이터로 환원

변수 간의 관계(공분산, 상관계수) 기반으로 정보의 손실을 최소화
공분산, 상관계수 기반으로 주요 성분(주성분)을 선택
=> 차원을 축소

사영의 개념
: 갯수와 분포가 유지되는 것 확인 가능
-> 기준 축이 중요 !

축 - 데이터 분산이 커지는 방향으로



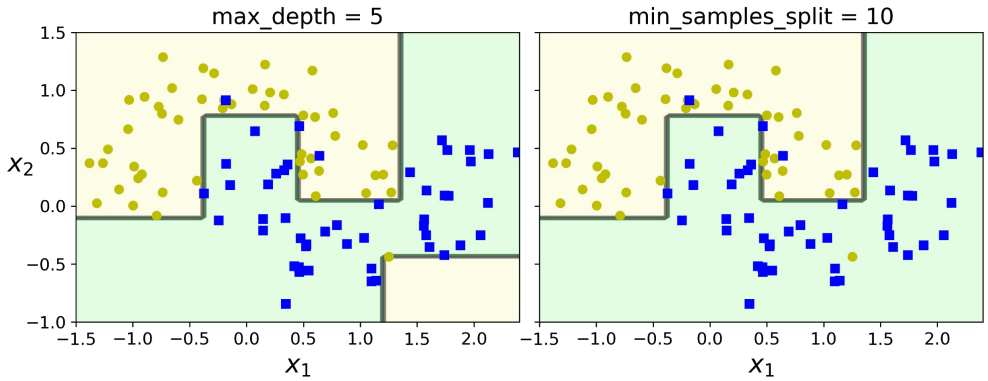
Dicision Tree에서 Precision과 Recall을 조절하는 방법이 있을까요?

PR곡선을 사용해서 그림을 그리면 임계값을 조절해서 모델의 Precision과 Recall을 조절할 수 있습니다.

왼쪽 그림에서 오른쪽 그림으로 변하는 모델 같은 경우 정밀도가 낮아졌다고 볼 수도 있습니다.

max_ 변수를 낮추거나 min_ 변수를 높이면 모델의 규제가 더 심해진다고 볼 수 있는데

위와 같은 경우를 Precision과 Recall을 조절하는 방법이라고 볼 수 있을까요?



1) https://arxiv.org/pdf/1011.0608.pdf

질문

구건모

Dicision Tree에서 Precision과 Recall을 조절하는 방법이 있을까요?

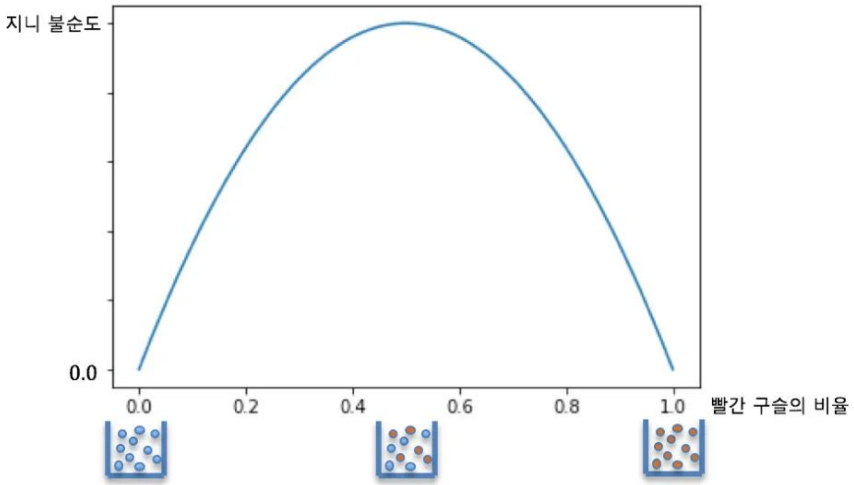
min_ 변수나 max_ 변수를 이용해 규제를 주면 overfitting이나 underfitting 문제를 해결해 모델의 성능을 높게(?) 향상시킬 수 있고, 이 결과로 precision과 recall이 바뀔 것 같다.

하지만 이게 PR 곡선의 임계값 바꾸는 것처럼 직접적인 연관인지 모르겠다. 그리고 규제를 조절한다고 해서 명확하게 직접적으로 precision과 recall이 우리가 원하는 방향으로 바뀌지 않을 것 같다.

1) https://arxiv.org/pdf/1011.0608.pdf

데이터(m < 2000)를 정렬하면 훈련 속도가 빨라지는 이유

presort = True를 설정해서 정렬을 하게 된 후 지니 불순도를 구하게 되면 단조증가/단조감소 함수가 되어서 모든 데이터를 보지 않고 최적의 지니 불순도를 찾을 수 있지 않을까..?



식 6-1: 지니 불순도

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

1) https://tyami.github.io/machine%20learning/decision-tree-4-CART/

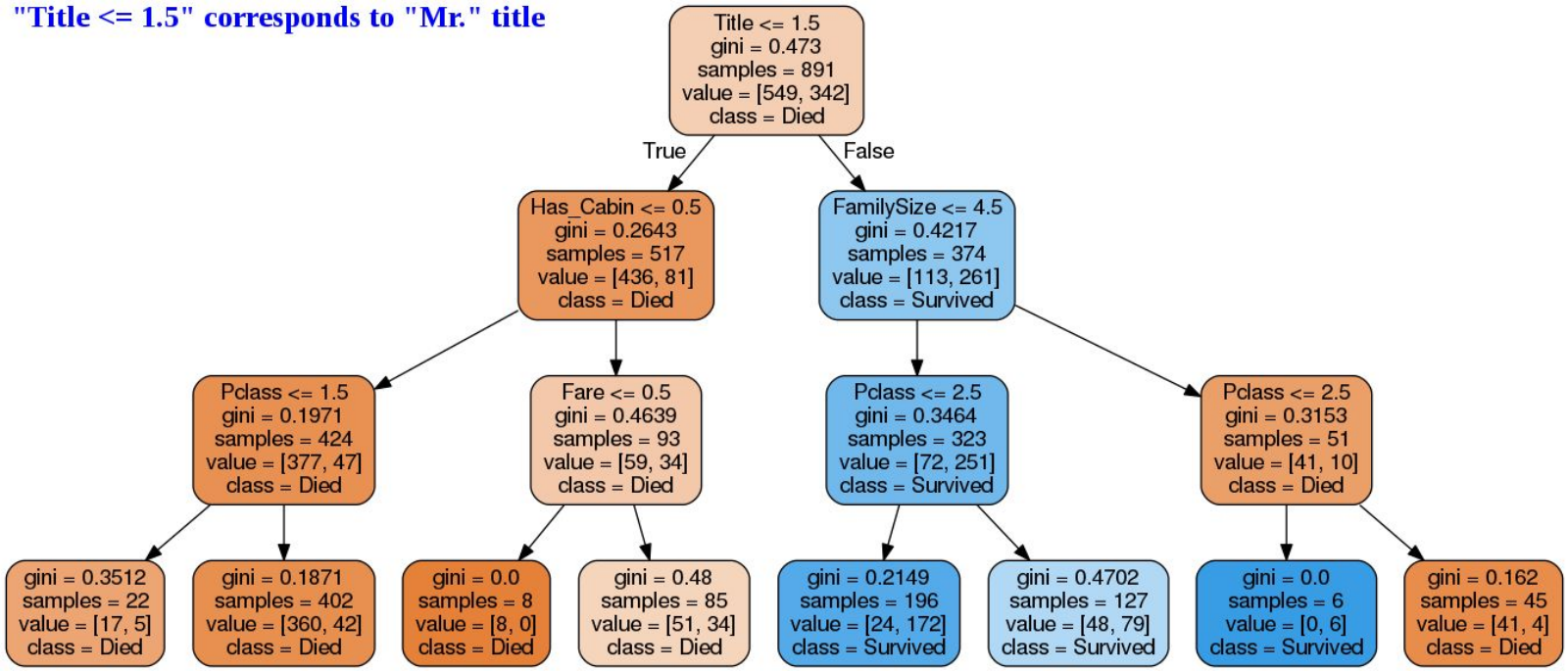
토의

구건모

데이터(m < 2000)를 정렬하면 훈련 속도가 빨라지는 이유

presort=True는 없어진 매개변수
히스토그램 기반 그레디언트 부스팅으로 대체된 듯 하다

Decision Tree로 Titanic 데이터 셋 분석해보기



1) https://www.kaggle.com/dmilla/introduction-to-decision-trees-titanic-dataset

토의

구건모

Decision Tree로 Titanic 데이터 셋 분석해보기

남자인데 돈 없고 짐 없으면 다 죽었다... 씁쓸...

데이터를 Decision tree로 보니까 재미있는 부분이 있었다.
Decision tree는 white box.

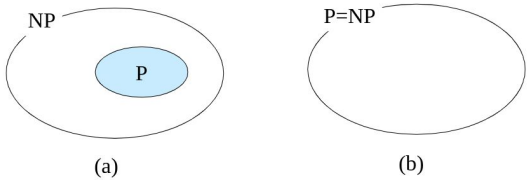
조성진

NP-완전 문제? NP-Complete

다항시간? $O(n^k)$

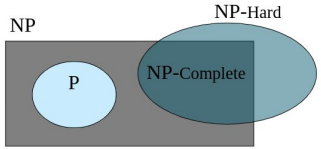
지니 불순도, 엔트로피 불순도 : 낮을수록 불확실성이 줄어든다?
크로스엔트로피
쿨백-라이블러 발산

P와 NP의 포함 관계



✓ 위 (a)인지 (b)인지는 아직 밝혀지지 않음.
✓ (a)일 것이라 강력히 추정됨.
✓ 백만불의 상금이 걸려 있다.

NP와 NP-Complete, NP-Hard의 관계



✓ P 부분은 추정

https://hyunw.kim/blog/2017/10/27/KL_divergence.html

조성진

P, NP, NP-Complete, NP-Hard

P : Polynomial time에 풀 수 있다 - O(n^k)
NP: Non-deterministic polynomial, polynomial에 풀릴지 안풀릴지 알 수 없다
non-polynomial이 아니다

Tractable: Polynomial 안에 풀 수 있다 - P
Intractable: Polynomial 안에 풀 수 없다 - NP, NP-Complete, NP-Hard
Unsolvable: 얼마나 걸릴지 모른다

P - tractable
not in P - intractable or unsolvable

조성진

NP algorithms = two stage procedure

1. Nondeterministic (“guessing”) stage:

generated randomly an arbitrary string that can be thought of a candidate solution (“certificate”) - 임의의 solution 하나 제시

2. Deterministic(“verification”) stage:

take the certificate and the instance to the problem and returns YES if the certificate represents a solution - 우리가 원하는 optimal solution인지 판단하는 과정

NP algorithms - verification stage is polynomial : 2번 과정만 polynomial 시간 걸린다
Solution 찾는 1번 과정에 대해서는 걸리는 시간을 모른다

토의

조성진

P = problems that can be solved in polynomial time
NP = problems for which a solution can be verified in polynomial time

Is sorting in NP? -> Not a decision problem -> Sorting 알고리즘 자체가 얼마나 걸리는지
Is sortedness in NP -> Yes, Easy to verify -> Sort 되어있는가 확인

Is P = NP? -> P $\subseteq$ NP
NP $\subseteq$ P or P = NP인가가 Open question이다

보통의 컴퓨터 과학자들은 false라고 믿는중...

조성진

NP-Completeness (formally)

NP-complete는 hardest problem in NP
Most practical problem은 P나 NP-complete이다

A problem B is NP-complete if

(1) $B \in \mathbf{NP}$

(2) $A \leq_p B$ for all $A \in \mathbf{NP}$

만약 B가 (2)만 만족하면 NP-hard이다
A <= B 이부분은 Polynomial reducible - mapping이 polynomial 안에 가능

조성진

Polynomial Reductions

Given two problems A, B we say that A is polynomial reducible to B if: $A \leq_p B$

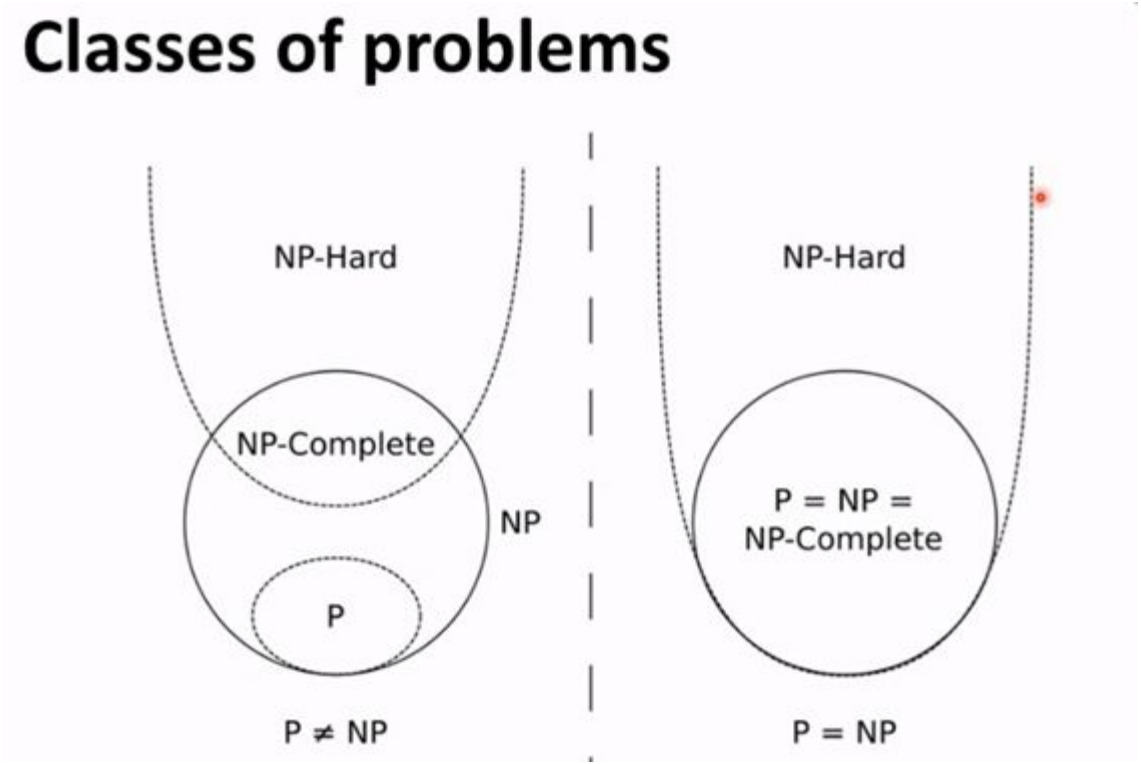
1. There exists a function f that converts the input of A to inputs of B in polynomial time

2. A(i) = YES <-> B(f(i)) = Yes - mapping

function f를 polynomial time 안에 A->B input으로 바꿀 수 있으면

토의

조성진


Classes of problems
NP-Hard
NP-Complete
NP
P
P ≠ NP
NP-Hard
P = NP =
NP-Complete
P = NP