의사결정나무

Decision Tree

의사결정나무

- **의결결정나무 란?**
  - 학습 데이터를 분석하여 데이터에 내재되어 있는 패턴을 통해 새롭게 관측된 데이터를 예측 및 분류하는 모델
  - 개념적으로 질문을 던져서 대상을 좁혀 나가는 '스무고개' 놀이와 비슷한 개념
    - 목적(Y)과 자료(X)에 따라 적절한 분리 기준과 정지 규칙을 지정하여 의사결정나무를 생성
  - 의사결정방식 과정의 표현법이 '나무'와 같다고 해서 의사결정나무라고 불림
    - 의사결정 규칙을 나무 모델로 표현

의사결정나무

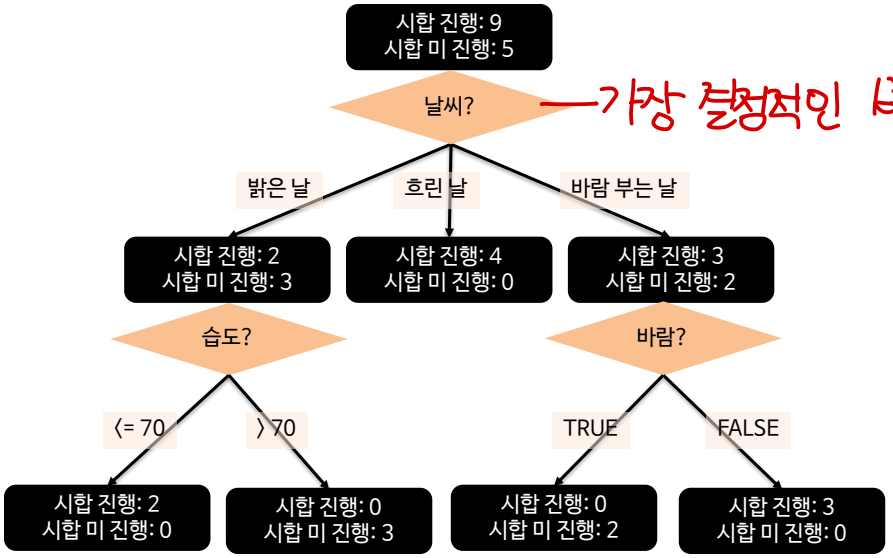
- **스무 고개로 알아보는 의사결정나무**

"호랑이"

의사결정나무

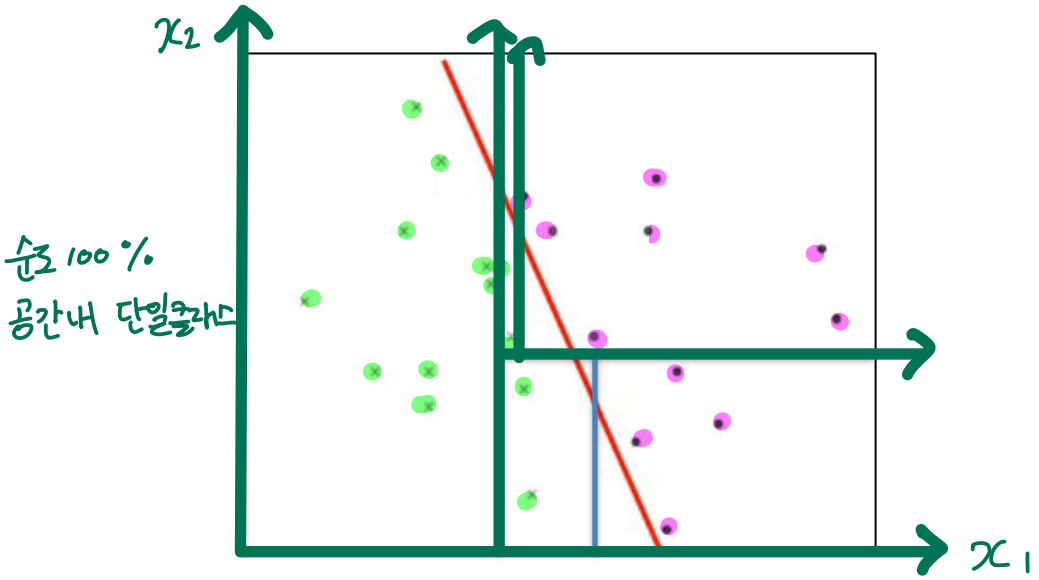
- **의사결정나무의 장점**
  - 이해하기 쉽고 적용하기 쉬움
    - 나무 구조 (If-then 규칙)에 의해 표현 때문에 모델을 쉽게 이해할 수 있음
  - 의사결정과정에 대한 설명(해석) 가능
    - 오늘 야구 경기의 취소 사례의 이유 설명 가능
  - 중요한 변수 선택에 유용 (상단에서 상용된 설명 변수가 중요한 변수)
  - 데이터의 통계적 가정이 필요 없음 (예. LDA 가정: 데이터 정규성)

공분산 구조가 같아야하는 가정 필요 X

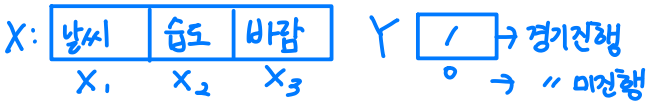


의사결정나무

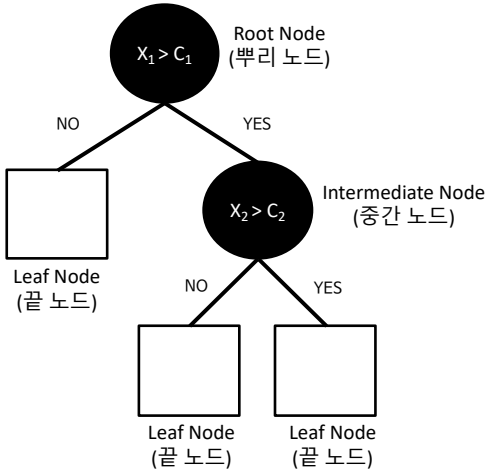
- **의사결정나무의 단점**
  - 좋은 모형을 만들기 위해 많은 데이터가 필요
  - 모형을 만드는데 상대적으로 시간이 많이 소요 (Tree building)
  - 데이터의 변화에 민감(데이터에 따라 모델이 변화함)
    - 학습과 테스트 데이터의 도메인이 유사해야 함 (domain gap 이 작아야 함)
  - 선형 구조 형 데이터 예측 시 더 복잡
    - 예시) 붉은선: 선형 회귀 결정 경계, 푸른선:의사결정나무 결정 경계



의사결정나무



- 의사결정나무를 활용한 데이터 분석
  - 순서: 데이터 ➔ 모델 학습 ➔ 추론



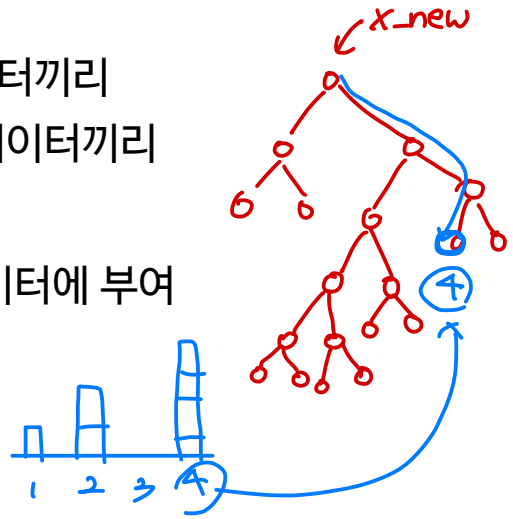
Input				Output
X_1	X_2	...	X_p	Y

  - 데이터: 다변량 변수 사용
  - 모델 학습 (트리 구조 이용)
    - ① 한번에 설명 변수 하나씩 데이터를 선택
    - ② 2개 혹은 그 이상의 부분집합으로 분할하여
    - ③ 데이터 순도가 균일해지도록 재귀적 분할(Recursive Partitioning)

      (재귀적 분할 종료 조건)

      분류 문제: 끝 노드에 비슷한 범주(클래스)를 갖고 있는 관측데이터끼리

      예측 문제: 끝 노드에 비슷한 수치(연속된 값)를 갖고 있는 관측데이터끼리
  - 추론 (판별)
    - 분류: 끝 노드에서 가장 빈도가 높은 종속변수(y)를 새로운 데이터에 부여
    - 회귀: 끝 노드 의 종속변수(y)의 평균을 예측 값으로 반환



의사결정나무

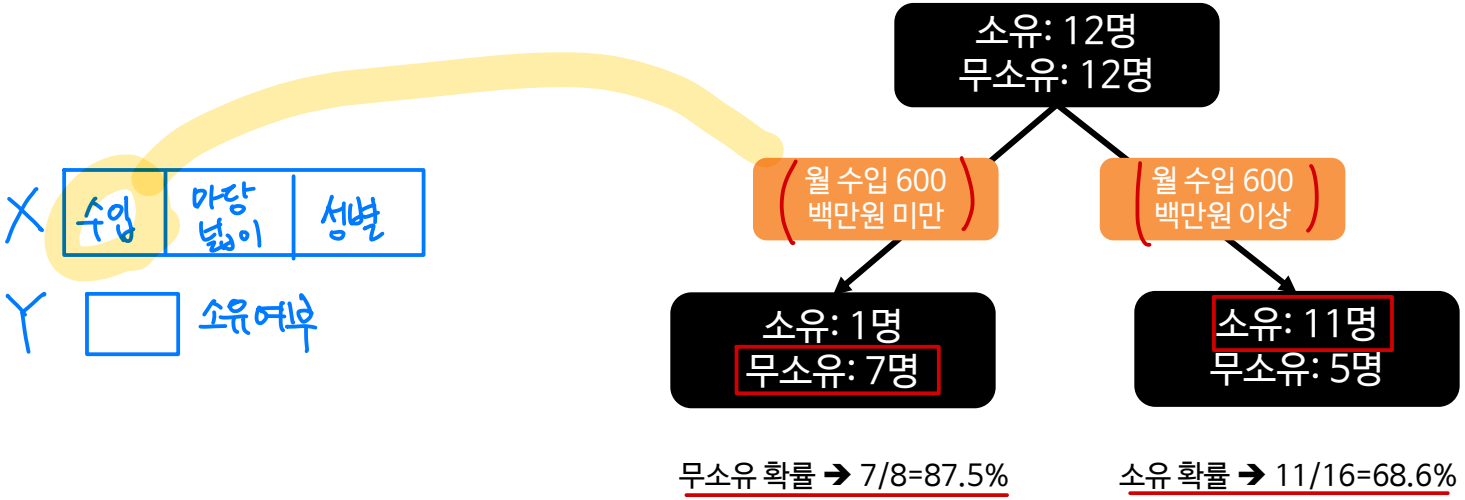
- **구분**
  - 분류 나무 (classification tree): 목표 변수가 범주형 변수 ➔ 분류
  - 회귀 나무 (regression tree): 목표 변수가 수치형 변수 ➔ 예측

- **재귀적 분할 알고리즘**
  - CART (Classification And Regression Tree)
  - C4.5, C5.0
  - CHAID (Chi-square Automatic Interaction Detection)

- **불순도 알고리즘 (분할 기준)**
  - 지니 지수 (Gini index)
  - 엔트로피 지수(Entropy index), 정보 이익(Information Gain)
  - 카이제곱 통계량 (Chi-Square Statistic)

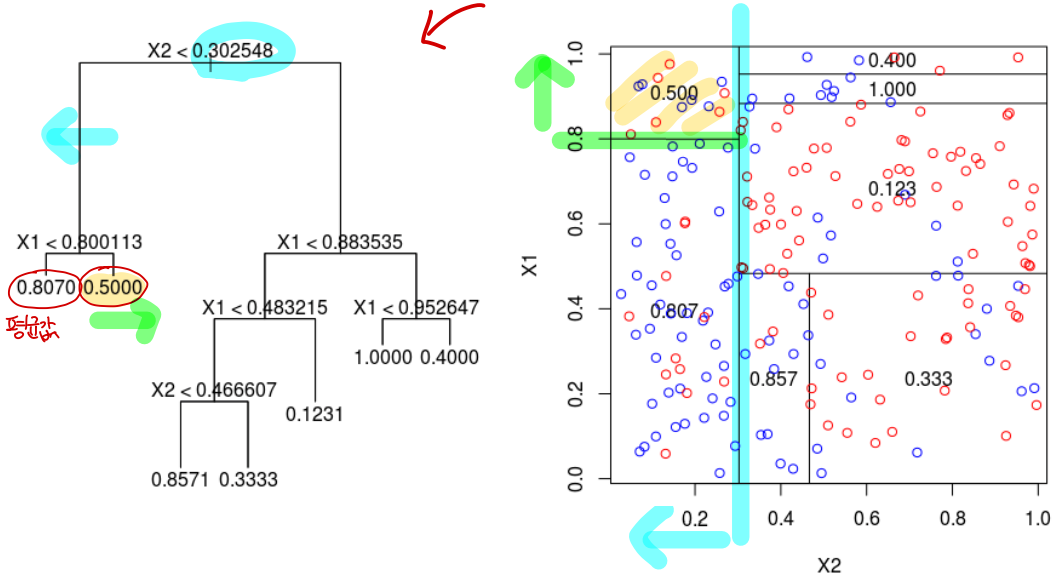
의사결정나무

- **분류 나무 (Classification Tree)**
  - 목표 변수: 범주형 변수 (분리)
  - 분류 알고리즘과 불순도 지표
    - CART: 지니 지수 (Gini Index)
    - C4.5: 엔트로피(Entropy index), 정보 이익(Information gain), 정보이익비율(Information gain ratio)
    - CHAID: 카이 제곱 통계량 (Chi-Square statistic)
  - 분류 결과 (판별, 추론)
    - 소속 집단 판단, 경향성도 확률로 표현 가능



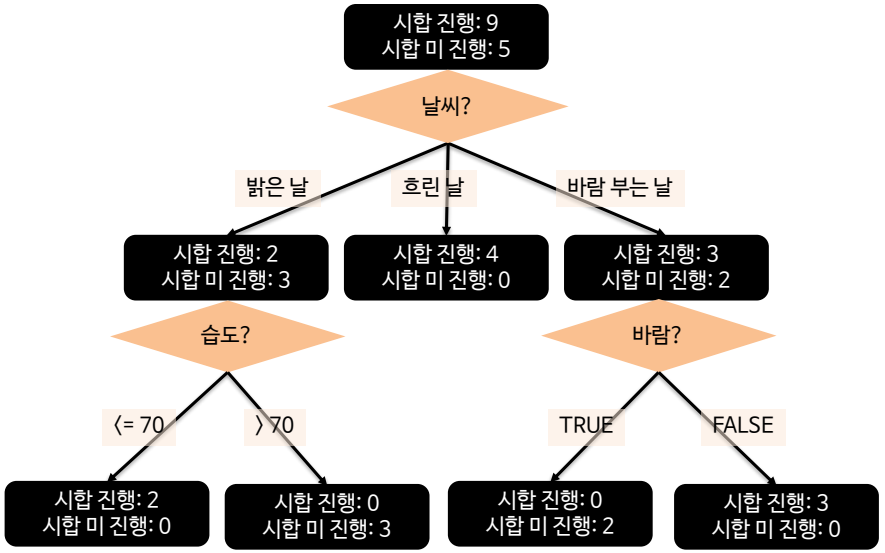
의사결정나무

- **회귀 나무** (Regression Tree)
  - 목표 변수: 수치형 변수 (예측) ex) 키
  - 회귀 알고리즘과 불순도 지표
    - CART: F 통계량과 분산 감소량 (실제 값과 예측 값의 평균 차이가 작도록!)
  - 회귀 결과 (classification and regression tree)
    - 끝 마디 집단의 평균 ($\arg\min|Y-\hat{Y}|_2$ 실제-예측 작아지도록)
    - *예측일 경우 회귀나무보다 신경망 또는 회귀분석이 더 좋음



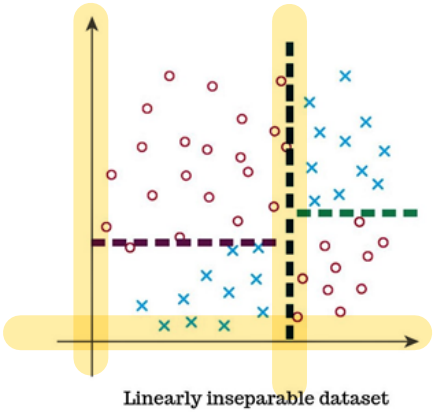
의사결정나무

- 분할
  - 이진 분할(binary split) : CART
  - 다중 분할(multi-way split) : CHAID, C4.5, C5.0 etc

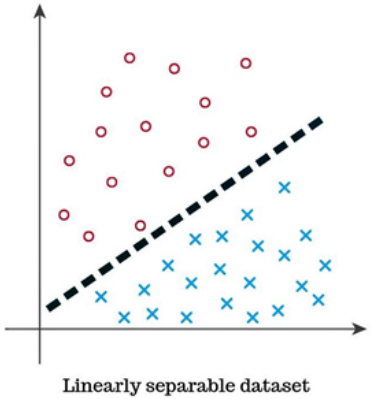


의사결정나무

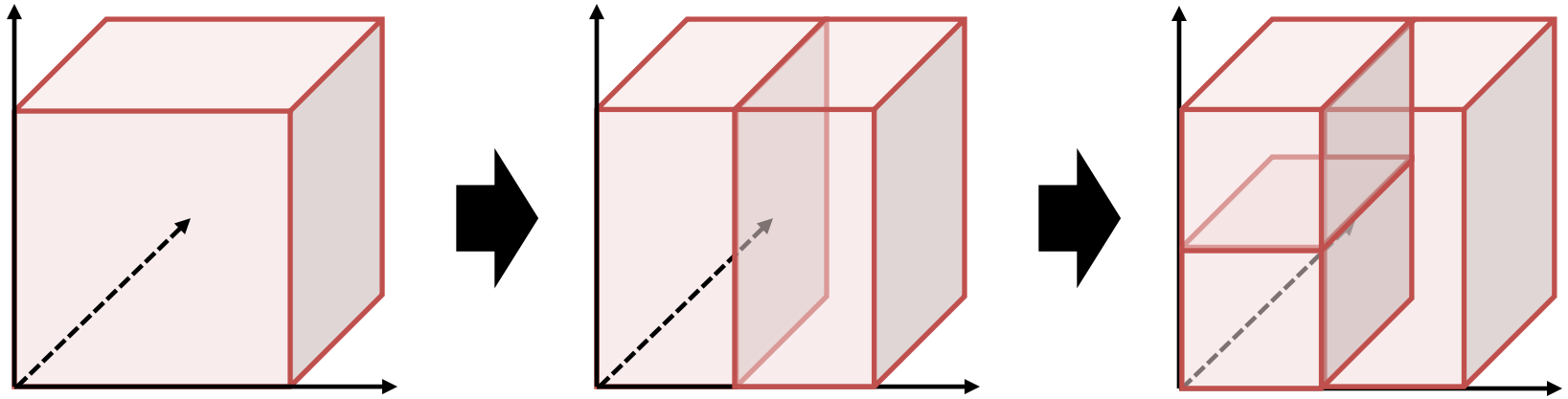
- 이진 분할 (분할이란?)
  - 2차원 데이터 분할



VS

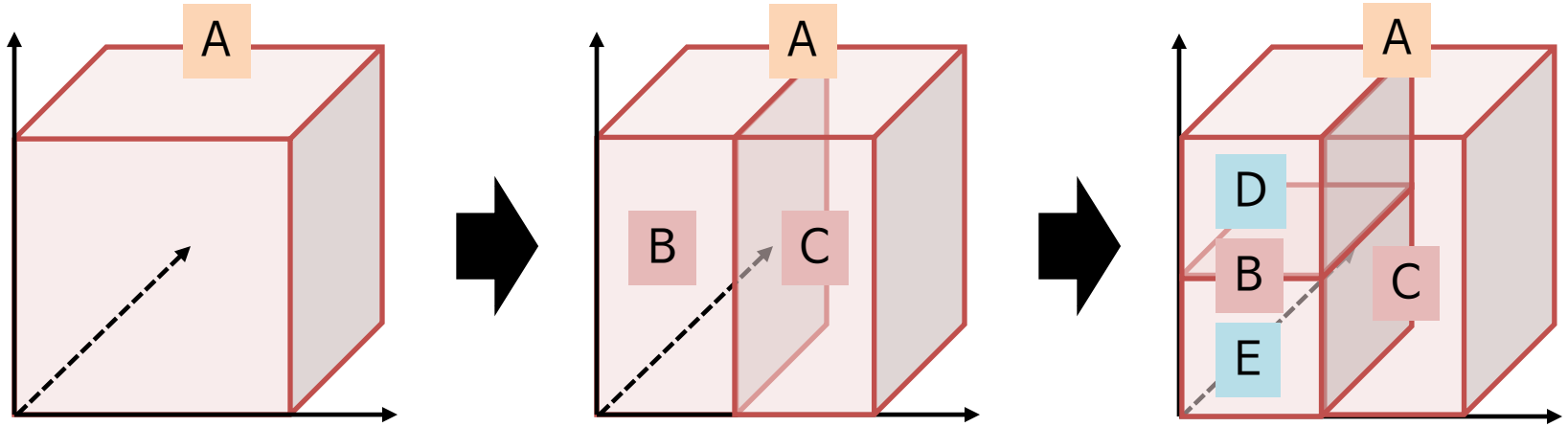


  - 3차원 데이터 분할

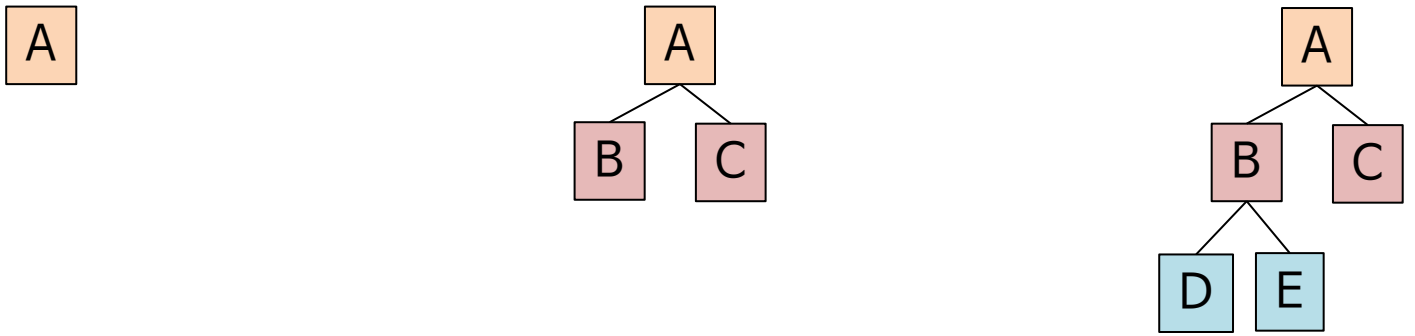


의사결정나무 표현

- 트리 구조를 이용한 데이터 분할 표현법
  - 3차원 데이터 분할 표현법



  - 전 차원 데이터 분할 표현법 (일반화) 트리구조



의사결정나무 II

Decision Tree

의사결정나무

- **재귀적 분할 알고리즘 정리**

	CART	C4.5	CHAID
분류 나무(분류)	O	O	O
회귀 나무(예측)	O	O	X
예측변수	범주, 수치	범주, 수치	범주
불순도 알고리즘	Gini index	Entropy	Chi-square, 통계량
분리	Binary	Multi-way	Multi-way
나무성장	완전 모형 생성(full tree) 후 가지치기		최적 모형 개발 stop (즉, 완전모형생성 없음)
가지치기 (교차검증)	학습시 학습 데이터 검증시 검증 데이터	학습 데이터만 사용	X
개발연도	1984	1993	1980
	2	3	1

분류 나무 : CART

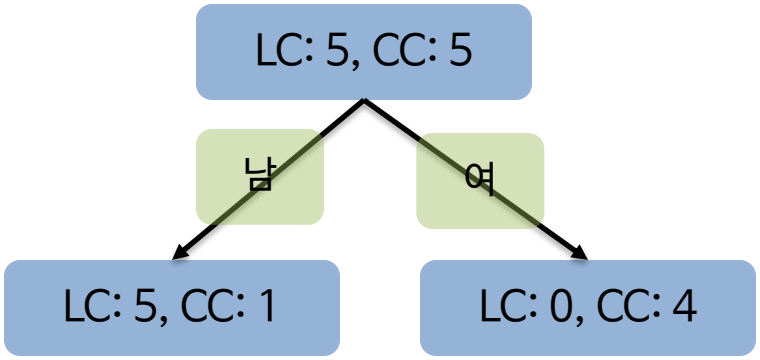
- CART (Classification & Regression Tree)
  - Breiman 등이 개발
  - 종류: 분류 나무, 회귀 나무
  - 분리: 이진 분할
  - 가지치기 (교차 타당도) : 학습 데이터로 나무 생성, 검증용 데이터로 가지치기
  - 불순도 알고리즘: Gini index (불확실성) 는 낮아지는게 좋음

가지치기 개꿀

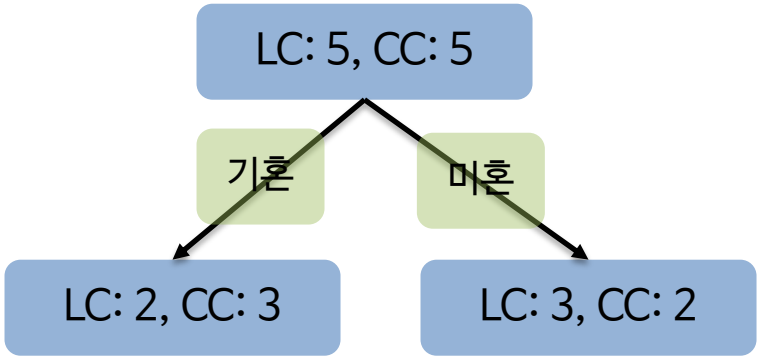
$$G.I(A) = \sum_{i=1}^{d} \left(R_i \left(1 - \sum_{k=1}^{m} p_{ik}^2 \right) \right)$$

분류 나무: CART

- 예제
  - 우리 홈쇼핑에서 충성 고객과 탈퇴 고객을 구분하는 규칙을 생성하고자 함
    - 충성 고객(LC: Loyal Customer)
    - 탈퇴 고객(CC: Churn Customer)
  - 총 10명의 고객을 대상으로 성별, 결혼유무 중 어느 변수가 더 분류를 잘하는 변수인지 찾고, 분류 규칙을 찾고자 함



성별에 의한 분류



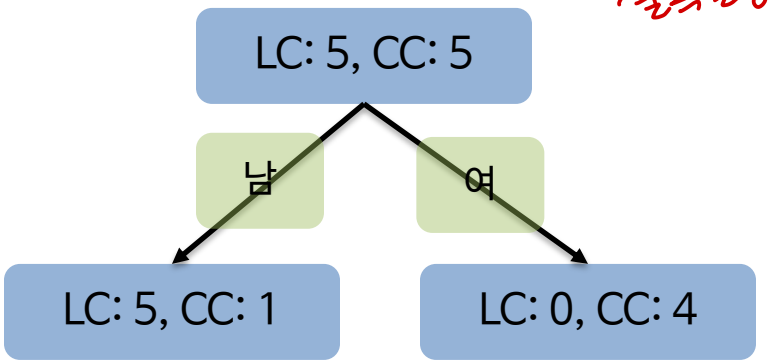
결혼유무에 의한 분류

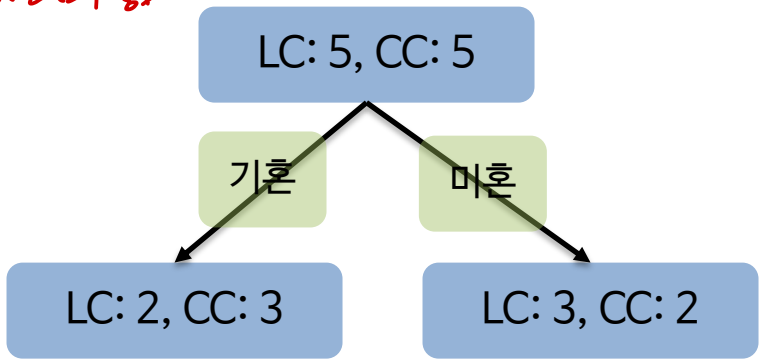
분류 나무: CART

$$G.I(A) = \sum_{i=1}^{d}\left(R_i\left(1-\sum_{k=1}^{m} p_{ik}^2\right)\right)$$

가중치

→ 불확실성 – 낮을수록 좋음





성별에 의한 분류

$1 - \sum_{k=1}^{m} p_{ik}^2$ (i=2, class 4)

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

$G(남) = 1-\left(\frac{5}{6}\right)^2-\left(\frac{1}{6}\right)^2 = 0.278$

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

$G(성별) = 가중치 * G(남) + 가중치 * G(여)$

$= \left(\frac{6}{10}\right)(0.278) + \left(\frac{4}{10}\right)(0) = 0.167$

많으로 나뉘도록 데이터의 전처리

→ 성별우선 선택

결혼유무에 의한 분류

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

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

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

$G(결혼) = 가중치 * G(기혼) + 가중치 * G(미혼)$

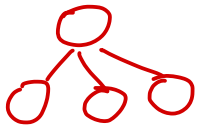
$= \left(\frac{5}{10}\right)(0.48) + \left(\frac{5}{10}\right)(0.48) = 0.48$

낮은수록 좋음

분류 나무: C4.5

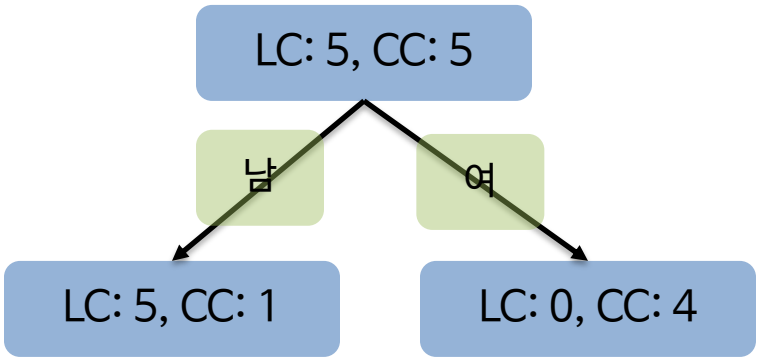
$$Entropy(A) = \sum_{i=1}^{d} R_i \left(- \sum_{k=1}^{m} p_k \log_2 (p_k) \right)$$

- C4.5, C5.0
  - Quinlan 등이 개발
  - 종류: 분류 나무, 회귀 나무
  - 분리: 다중 분할
  - 불순도 알고리즘: 엔트로피(불확실성), 정보이론, 정보 이득률
  - 가지치기 (교차 타당도) : 학습 데이터만 이용하여 나무 성장 및 가지치기 수행
  - 정보 이론➔ 엔트로피
    - Log_2로 계산하는 이유 ➔ bit 수로 정보 계산
      - $Log_2(8)=3bit$
    - -log2로 계산하는 이유: log2(1/2) = -1이기 때문에 + 로 전환 필요
  - 정보 이익 (IG: Information Gain) *정보의 가치 높아야 좋음
    - IG = E(before) - E(After)



분류 나무: C4.5

$$Entropy(A) = \sum_{i=1}^{d} R_i \left(- \sum_{k=1}^{m} p_k \log_2 (p_k) \right)$$



성별에 의한 분류

$E(root) = -\left(\frac{5}{10}\right) log_2 \left(\frac{5}{10}\right) - \left(\frac{5}{10}\right) log_2 \left(\frac{5}{10}\right) = 1$

$E(남) = -\left(\frac{5}{6}\right) log_2 \left(\frac{5}{6}\right) - \left(\frac{1}{6}\right) log_2 \left(\frac{1}{6}\right) = 0.65$

$E(여) = -\left(\frac{0}{4}\right) log_2 \left(\frac{0}{4}\right) - \left(\frac{4}{4}\right) log_2 \left(\frac{4}{4}\right) = 0$

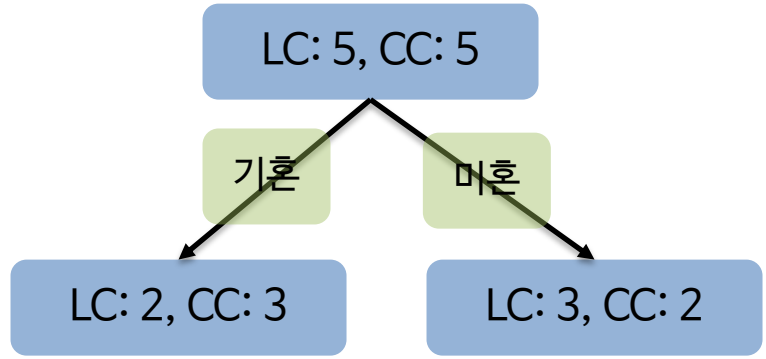
$E(성별) = 가중치 * E(남) + 가중치 * E(여)$

$= \left(\frac{6}{10}\right)(0.65) + \left(\frac{4}{10}\right)(0) = 0.39$

1 - 0.39

IG (성별) = E(Root) - E(성별) = 0.61

← 불 확실성 감소량 (클수록 좋음)



결혼유무에 의한 분류

$E(root) = -\left(\frac{5}{10}\right) log_2 \left(\frac{5}{10}\right) - \left(\frac{5}{10}\right) log_2 \left(\frac{5}{10}\right) = 1$

$E(기혼) = -\left(\frac{2}{5}\right) log_2 \left(\frac{2}{5}\right) - \left(\frac{3}{5}\right) log_2 \left(\frac{3}{5}\right) = 0.971$

$E미혼) = -\left(\frac{3}{5}\right) log_2 \left(\frac{3}{5}\right) - \left(\frac{2}{5}\right) log_2 \left(\frac{2}{5}\right) = 0.971$

$E(결혼) = 가중치 * E(남) + 가중치 * E(여)$

$= \left(\frac{5}{10}\right)(0.971) + \left(\frac{5}{10}\right)(0.971) = 0.971$

IG (결혼) = E(Root) - E(결혼) = 0.029

← 거의 변화가 없음. 작, 결혼 여부는 큰 영향을 주지 못함

지니인덱스랑 다름 !!!

분류 나무: C4.5

- 정보 이득율 (Information gain ratio)
  - C4.5 에서는 information gain ➔ information gain ratio 추가 도입
  - 가지수가 많을 수록 information gain 이 높아지는 경향을 보임
    - 이진분할 vs 다중 분할
  - 단점 보안 위해 IV(Intrinsic Value) 도입하여 정보 이득율을 정규화
    - 가지가 많으면 감점

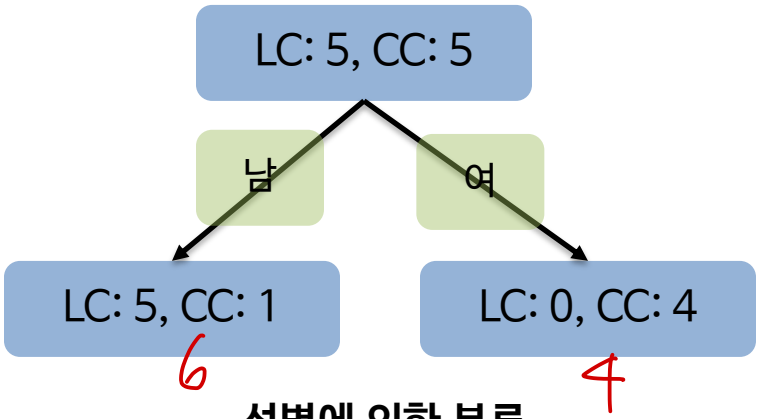
$$IV(A) = -\sum_{k=0}^{n} \frac{1}{n} log_2\left(\frac{1}{n}\right)$$

  - 이득율

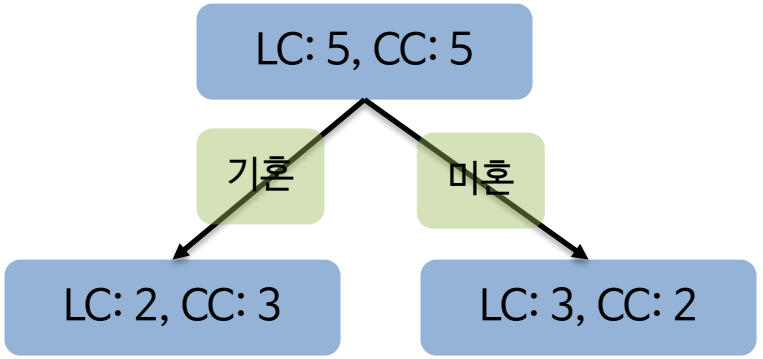
$$IGR(A) = \frac{IG(A)}{IV(A)}$$

분류 나무: C4.5

$$IV(A) = -\sum_{k=0}^{n} \frac{1}{n} log_2(\frac{1}{n})$$



성별에 의한 분류



결혼유무에 의한 분류

IG (성별) = E(Root)-E(성별)=0.61

IV(성별) = $-\left(\frac{6}{10}\right) log_2\left(\frac{6}{10}\right) - \left(\frac{4}{10}\right) log_2\left(\frac{4}{10}\right)$ = 0.97

IGR(성별) = IG(성별)/IV(성별) = 0.61/0.97 = 0.63

IG (결혼) = E(Root)-E(결혼)= 0.029

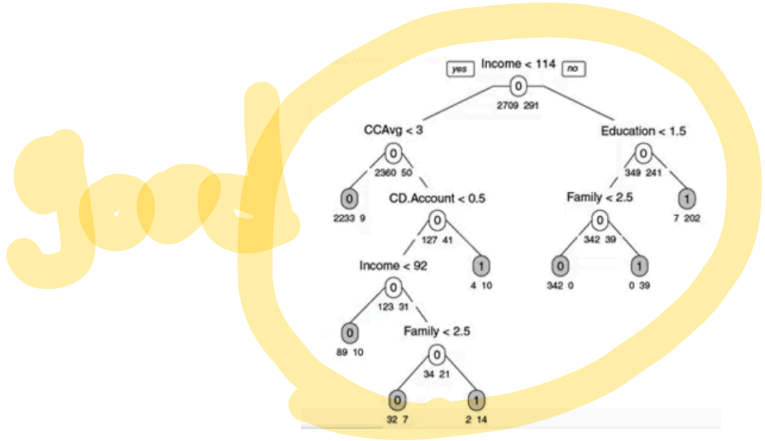
IV(결혼) = $-\left(\frac{5}{10}\right) log_2\left(\frac{5}{10}\right) - \left(\frac{5}{10}\right) log_2\left(\frac{5}{10}\right)$ = 1

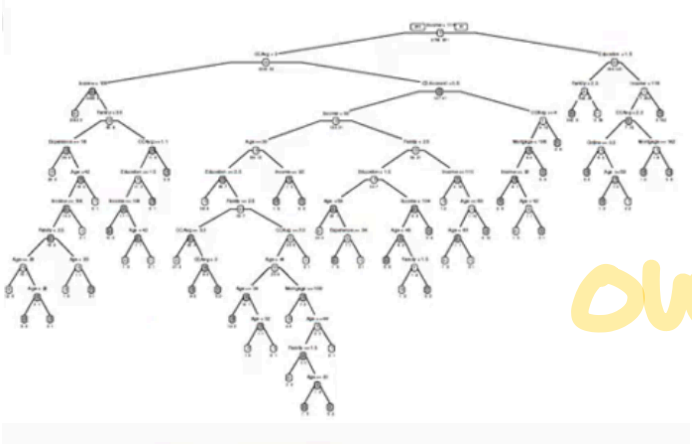
IGR(결혼) = IG(결혼)/IV(결혼) = 0.029/1 =0.029

**다중 분할 예시로 변경하면 - log2()이 계속 붙게 되고 IV가 1을 넘기도 함*

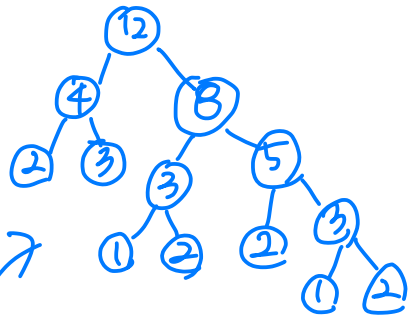
분류 나무

- 끝 없는 분할의 단점
  - 과적합 (Overfitting)
    - 학습용 데이터에 완전히 적합
    - 학습용 데이터에 잡음도 포함되므로 테스트 데이터에서 오차는 일반적으로 증가
  - 과적합 피하는 법
    - 나무 성장 중단
- 모델 학습의 목적
  - (잘못된 학습) 학습용 데이터에서는 높은 성과 ➔ 평가용 데이터에서는 낮은 성과
  - (올바른 학습) 현재 데이터의 설명 ➔ 미래 데이터 예측

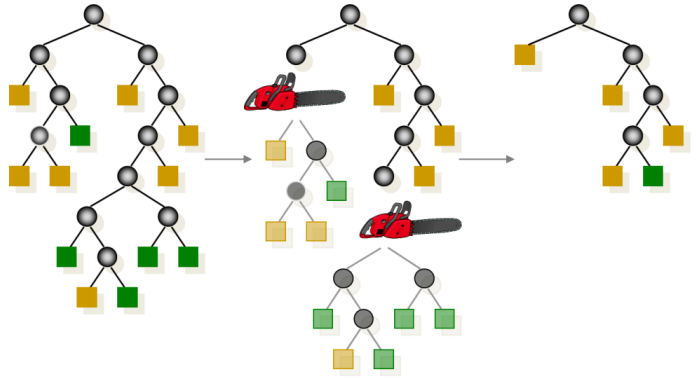




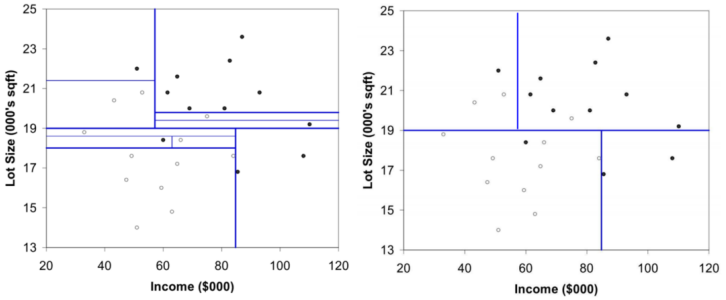
분류 나무: 과적합 방지



- 성장 멈추기 (Stop condition)
  - 나무 모델의 깊이 파라미터로 설정 depth ← 트리깊이
  - 나무 모델을 성장시키면서 특정 조건에서 성장을 중단
  - 노드 내의 최소 관측치의 수 : 2
  - 불순도 최소 감소량 Δt < 2 → 분할안함 → stop
    - CHAID에서 사용
    - 가지치기 사용하지 않고 종료



- 가지치기 (Pruning): 다해보고 결정하자!
  - 완전 모형 생성 후 가지치기
  - 데이터 버리는 개념 아닌 합치(merge)는 개념
  - 나무 모델 생성 후 필요 없는 가지 제거
  - 성정 멈추기 보다 성능 우수
  - 가지치기 비용함수를 최소로 하는 분기를 찾음



과적합 X

의사결정나무 III

Decision Tree

회귀나무

- **입력 데이터(변수 값)의 결과 예측**
  - 데이터가 도달한 끝 노드 데이터들의 평균으로 결정

- **불순도 측정 방법**
  - 제곱 오차 합 (the sum of the squared errors)
  - 오차 = 실제 값-예측 값

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

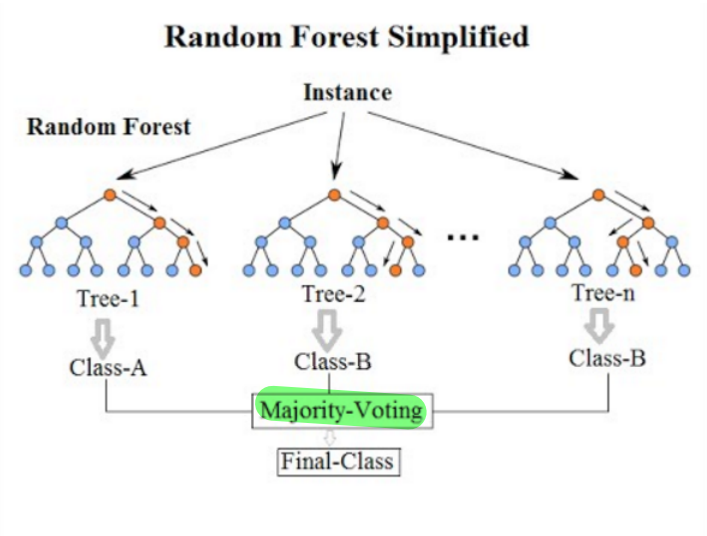
- **성능 평가 방법**
  - 예측모델 평가 방법: RMSE

실제값

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

앙상블(Ensemble)

- 앙상블
  - 여러 모델을 함께 사용하자!!
    - 의사결정나무, KNN, LDA, 로지스틱 등
    - 퀴즈쇼에서 사용되는 힌트: 100인의 정답
  - 설명보다는 예측이 중요할 경우에 사용
  - 예측 알고리즘을 조합하여 예측 성능을 향상
  - 랜덤 숲(Random Forest), Boosted Trees
    - 좋은 의사결정나무를 모아서 숲을 만들자!

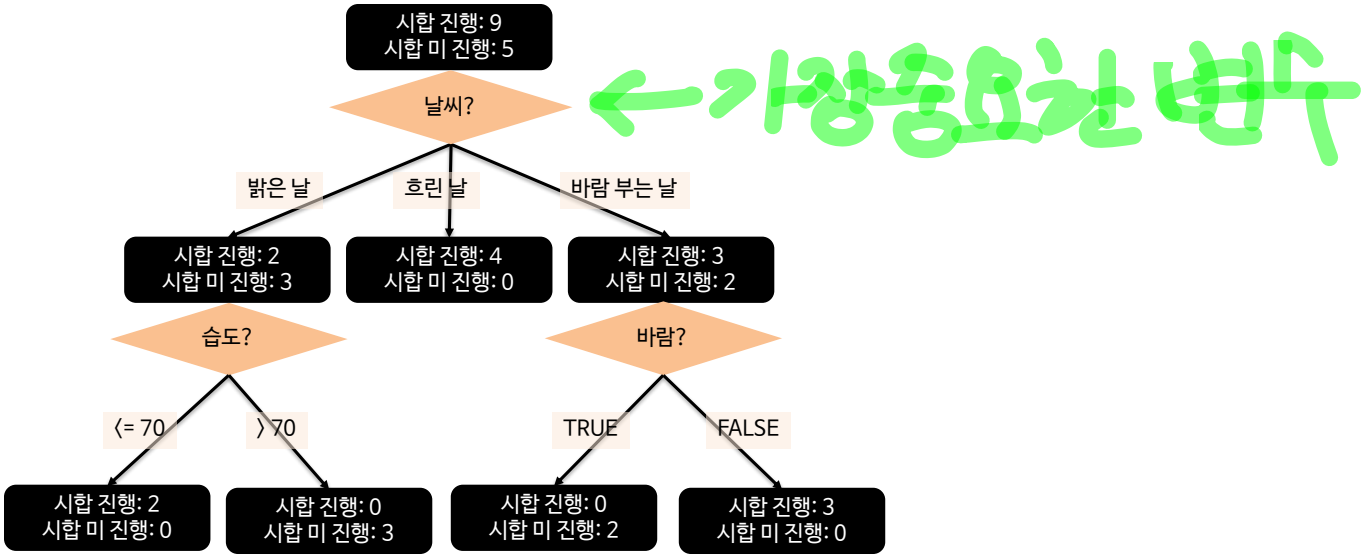


앙상블(Ensemble)

- Random Forest
  - Bootstrap 사용
    - 데이터로부터 복원 추출(뽑은 표본 원복)을 이용하여 여러 샘플을 추출
  - Forest 생성
    - 무작위로 예측 변수를 선택하여 모델 구축
      - 즉, 의사결정나무는 예측 변수 선택 시 기준 지표를 사용하였으나 무작위 숲에서는 무작위로 선택함
  - 앙상블 결과 결합
    - 분류 문제➔투표, 예측 문제➔평균화
  - 비록 나무 구조이지만, 숲이 되면서 해석 가능한 모델의 장점은 사라짐
  - 그러나 결과 분석을 통해 설명 변수 중 중요한 변수 판별 가능

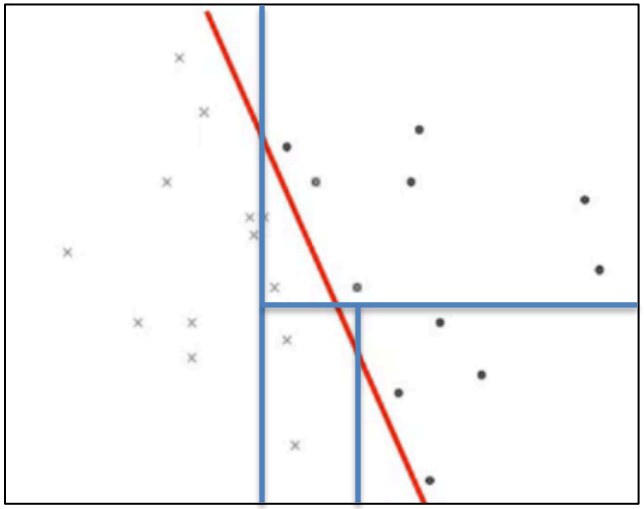
의사결정나무 장점 (복습)

- **의사결정나무의 장점**
  - 이해하기 쉽고 적용하기 쉬움
    - 나무 구조 (If-then 규칙)에 의해 표현 때문에 모델을 쉽게 이해할 수 있음
  - 의사결정과정에 대한 설명(해석) 가능
    - 오늘 야구 경기의 취소 사례의 이유 설명 가능
  - 중요한 변수 선택에 유용 (상단에서 상용된 설명 변수가 중요한 변수)
  - 데이터의 통계적 가정이 필요 없음 (예. LDA 가정: 데이터 정규성)



의사결정나무 단점 (복습)

- **의사결정나무의 단점**
  - 좋은 모형을 만들기 위해 많은 데이터가 필요
  - 모형을 만드는데 상대적으로 시간이 많이 소요 (Tree building)
  - 데이터의 변화에 민감(데이터에 따라 모델이 변화함)
    - 학습과 테스트 데이터의 도메인이 유사해야 함 (domain gap 이 작아야 함)
  - 선형 구조 형 데이터 예측 시 더 복잡
    - 예시) 붉은선: 선형 회귀 결정 경계, 푸른선:의사결정나무 결정 경계



요약

- **주요 방법**
  - Trees and Rules 구조
    - 규칙은 나무 모델로 표현
    - 결과는 규칙으로 표현
  - 재귀적 분할(Recursive Partitioning)
    - 의사결정나무 생성 과정
    - 그룹이 최대한 동질 하도록 반복적으로 하위 그룹으로 분리
  - 가지치기(Pruning the trees)
    - 생성된 나무를 자르는 과정 (정교화)
    - 과적합(overfitting)을 피하기 위해 필요 없는 가지를 간단히 정리하는 과정
  - 구분
    - 분류 나무: 목표 변수가 범주형 변수
    - 회귀 나무: 목표 변수가 수치형 변수

요약

- **재귀적 분할 알고리즘**
  - CART, C4.5, CHAID
- **불순도 알고리즘**
  - 지니 지수, 엔트로피 지수, 카이제곱 통계량
- **의사결정나무 과정**
  - 나무 모델 생성➔과적합문제해결➔검증➔해석 및 예측
    - 생성: CART(Gini), C4.5(Entorpy), CHAID(Chi-Square)
    - 과적합문제: 완전나무모형생성 후 가지치기 (CART, C4.5)
    - 검증: 교차타 당성을 이용하여 의사결정나무 평가
    - 해석 및 예측: 의사결정나무를 해석하고 예측 모형 설정
- **앙상블**
  - Random Forest

의사결정나무 실습

의사결정나무(DT) 실습 #1 (분류)

- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2)

- 데이터 로더, 범주형 데이터 변환, 데이터 분할
  - 3, 4,5장 실습과 동일

```
# Iris data 불러오기
import seaborn as sns # seaborn을 불러옴.
iris=sns.load_dataset('iris') # iris라는 변수명으로 Iris data를 download함.
X=iris.drop('species',axis=1) # 'species'열을 drop하고 특성변수 X를 정의함.
y_=iris['species'] # 'species'열을 label y를 정의함.

from sklearn.preprocessing import LabelEncoder # LabelEncoder() method를 불러옴
classle=LabelEncoder()
y=classle.fit_transform(iris['species'].values) # species 열의 문자형을 범주형 값으로 전환

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y, test_size=0.3, random_state=123, stratify=y)

```

의사결정나무(DT) 실습 #1 (분류)

- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2)

- 의사결정나무 API: Manual
  - DecisionTreeClassifier 클래스 호출
    - criterion : 불순도 알고리즘 옵션, default='gini' 가능 파라미터={entropy,gini}

```
# Classification Tree# 또는 from sklearn import DecisionTreeClassifier

# 과적합 사례 확인 법: max_depth 3=> 5=> 7 높일 수록 학습 데이터 정확도 올라라고 테스트 정확도 고정
#dtc = tree.DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=1)
dtc = tree.DecisionTreeClassifier(criterion='entropy', max_depth=7, random_state=1)
dtc.fit(X_train, y_train)
y_train_pred = dtc.predict(X_train)  # Training accuracy
y_test_pred = dtc.predict(X_test)    # Test accuracy

```

의사결정나무(DT) 실습 #1 (분류)

- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2)

- 의사결정나무 API: Accuracy & Confusion Matrix
  - 3, 4, 5 장 실습과 동일

```
[20]  1 from sklearn.metrics import accuracy_score
      2 print(accuracy_score(y_train, y_train_pred))
      3 print(accuracy_score(y_test, y_test_pred))
```

```
0.9904761904761905
0.9777777777777777
```

```
[21]  1 from sklearn.metrics import confusion_matrix
      2 print(confusion_matrix(y_test,y_test_pred))
```

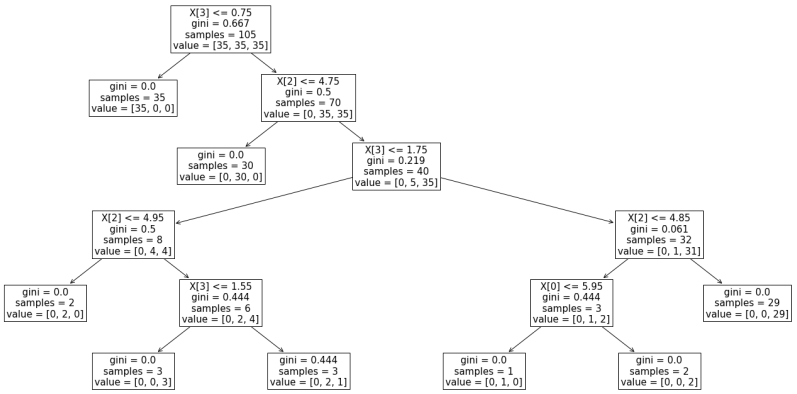
```
[[15  0  0]
 [ 0 15  0]
 [ 0  1 14]]
```

의사결정나무(DT) 실습 #1 (분류)

- 데이터셋: Iris 데이터
- 학습/시험 데이터: x_train/x_test
- 학습/시험 데이터 라벨: y_train/y_test ➔(0,1,2)

- Plot: tree.plot_tree [API]

```
1 import matplotlib.pyplot as plt
2 #tree.plot_tree(dtc.fit(X_train,y_train))
3 fig, ax = plt.subplots(figsize=(25, 12))
4 tree.plot_tree(dtc.fit(X_train, y_train), fontsize=15)
5 plt.savefig('tree_high_dpi', dpi=100)
```

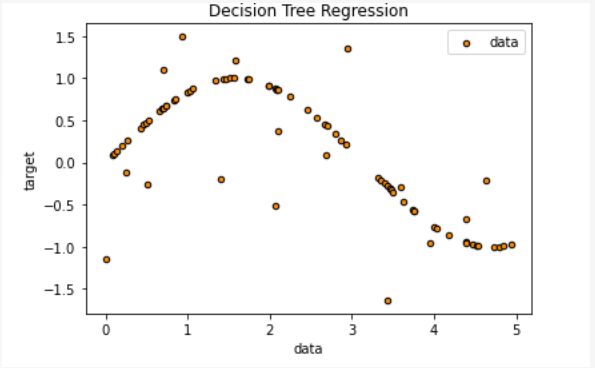


Values =: 각 클래스별 데이터 수

의사결정나무(DT) 실습 #2 (회귀)

- 데이터셋: 합성데이터

```
# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))

```



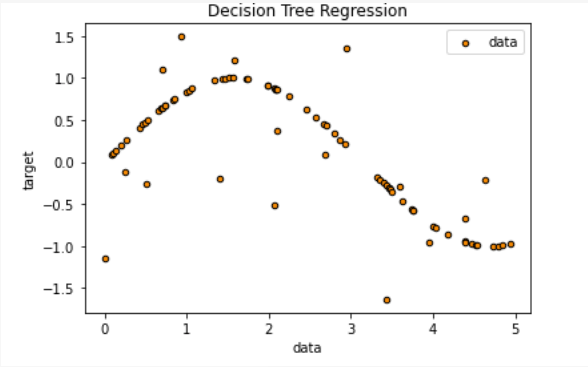
- 의사결정나무 API: [Manual]
  - DecisionTreeRegressor 클래스 호출
    - criterion : 불순도 알고리즘, {“mse”, “friedman_mse”, “mae”}, default=”mse”

```
[22]  # Fit regression model
      regr_1 = DecisionTreeRegressor(max_depth=2)
      regr_2 = DecisionTreeRegressor(max_depth=5)
      regr_1.fit(X, y)
      regr_2.fit(X, y)

      y_pred_1 = regr_1.predict(X)
      y_pred_2 = regr_2.predict(X)

```

의사결정나무(DT) 실습 #2 (회귀)

- 데이터셋: 합성데이터

```
# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))

```



- 의사결정나무 API: MSE/RMSE [API]
  - Mean_squared_error: squared=True ➔ MSE, False➔RMSE

```
from sklearn.metrics import mean_squared_error

print(mean_squared_error(y, y_pred_1))
print(mean_squared_error(y, y_pred_2))
```

```
0.12967126328231798
0.025236948989861896
```