

Naïve Bayes & Decision Tree

21기 공서연

22기 양가현

24.08.03. 정규세션

목차

1. 나이브베이지스

- Statistical Modeling
- What is Naïve Bayes?
- Based on Bayes Rule
- 'naïve' assumption
- Naïve Bayes Classification
- NB 특징

2. 의사결정나무

- 분할과 정복
- 정보이득
- 엔트로피 & 지니계수
- 회귀트리
- 실습
- DT 특징
- Covering Algorithm

Statistical Modeling

- 기본적으로 분류란, 각 데이터에 클래스 라벨을 붙여주는 것
 - 클래스 라벨 뿐만 아니라, 예측에 관한 '신뢰도'를 측정할 수 있을까?
- 속성과 클래스 라벨 간의 관계를 나타낼 수 있는 "확률 이론"을 사용하자
- "나이브베이지스 = 확률적 분류 모델"

What is Naïve Bayes?

- $P(\text{play} = \text{yes} \mid \text{outlook} = \text{rainy}, \text{temp} = \text{cool}, \text{humidity} = \text{high}, \text{windy} = \text{true}) = ?$
- 각각의 속성값을 증거(evidence)로 생각한다
- 사건은 'play = yes'

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Cool	High	True	?

Basic strategy

- 모든 y 에 대해, $P(y|x_1, x_2, \dots, x_d)$ 계산
- $P(y|x_1, x_2, \dots, x_d)$ 을 최대로 하는 y 선택

Based on Bayes Rule

- $P(x, y)$: joint probability
- $P(y|x) = \frac{P(x,y)}{P(x)}$, $P(x|y) = \frac{P(x,y)}{P(y)}$
- $P(y|x) = \frac{P(x|y)P(y)}{P(x)}$

→ 즉, $P(x_1, x_2, \dots, x_d | y)$ 로부터 $P(y | x_1, x_2, \dots, x_d)$ 를 구할 수 있음
→ 그런데, $P(x_1, x_2, \dots, x_d | y)$ 계산은 문제가 없을까?

'naive' assumption

- $P(x_1, x_2, \dots, x_d | y)$ 계산상의 문제
- 속성의 수가 증가할수록, 특정 '증거'에 대한 데이터가 충분하지 않기 때문에 훈련이 충분히 되지 못함
 - 각 특성 x_1, x_2, \dots, x_d 는 각각 독립적일 것이라는 나이브한 가정을 두자
- $P(x_1, x_2, \dots, x_d | y) \approx p(x_1 | y) \cdot p(x_2 | y) \cdot \dots \cdot p(x_d | y)$
- $P(y | x_1, x_2, \dots, x_d) = \frac{p(x_1 | y) \cdot p(x_2 | y) \cdot \dots \cdot p(x_d | y) \cdot p(y)}{P(x_1, x_2, \dots, x_d)}$

Naive Bayes Classification

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

Yes: 9
No: 5

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Cool	High	True	?

$$p(\text{Rainy}|\text{Yes}) = 2/9$$

$$p(\text{Rainy}|\text{No}) = 3/5$$

$$p(\text{Cool}|\text{Yes}) = 3/9$$

$$p(\text{Cool}|\text{No}) = 1/5$$

$$p(\text{High}|\text{Yes}) = 3/9$$

$$p(\text{High}|\text{No}) = 4/5$$

$$p(\text{True}|\text{Yes}) = 3/9$$

$$p(\text{True}|\text{No}) = 2/5$$

$$\begin{aligned}
p(\text{Yes} \mid \text{Rainy}, \text{Cool}, \text{High}, \text{True}) &\approx p(\text{Rainy} \mid \text{Yes}) \cdot p(\text{Cool} \mid \text{Yes}) \cdot p(\text{High} \mid \text{Yes}) \cdot p(\text{True} \mid \text{Yes}) \cdot p(\text{Yes}) \\
&= 2/9 \cdot 3/9 \cdot 3/9 \cdot 3/9 \cdot 9/14 = \mathbf{0.00529}
\end{aligned}$$

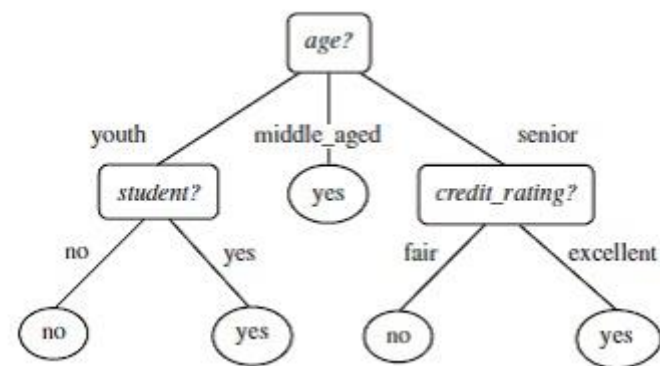
$$\begin{aligned}
p(\text{No} \mid \text{Rainy}, \text{Cool}, \text{High}, \text{True}) &\approx p(\text{Rainy} \mid \text{No}) \cdot p(\text{Cool} \mid \text{No}) \cdot p(\text{High} \mid \text{No}) \cdot p(\text{True} \mid \text{No}) \cdot p(\text{No}) \\
&= 3/5 \cdot 1/5 \cdot 4/5 \cdot 3/5 \cdot 5/14 = \mathbf{0.02057}
\end{aligned}$$

나이브 베이즈 특징

- 나이브베이즈 가정을 사용함으로써, 고차원 환경에서도 빠르게 계산 가능 (+)
- 잡음과 누락 데이터를 잘 처리함 (누락된 변수만 제거해서 계산) (+)
- 모든 클래스에 대해 비슷한 값을 갖는 irrelevant한 속성은 최종 확률 계산에 크게 영향을 주지 않음 (+)
- 모든 속성들이 독립적이라는 무리한 가정이 잘못된 경우가 자주 있음 (-)

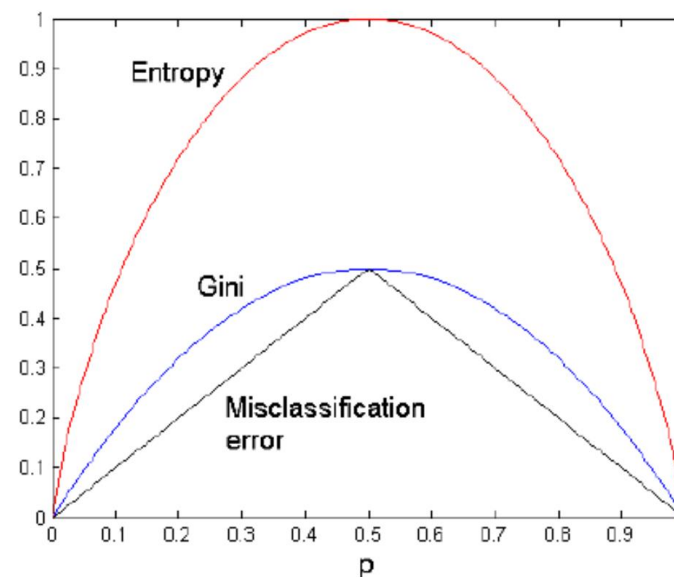
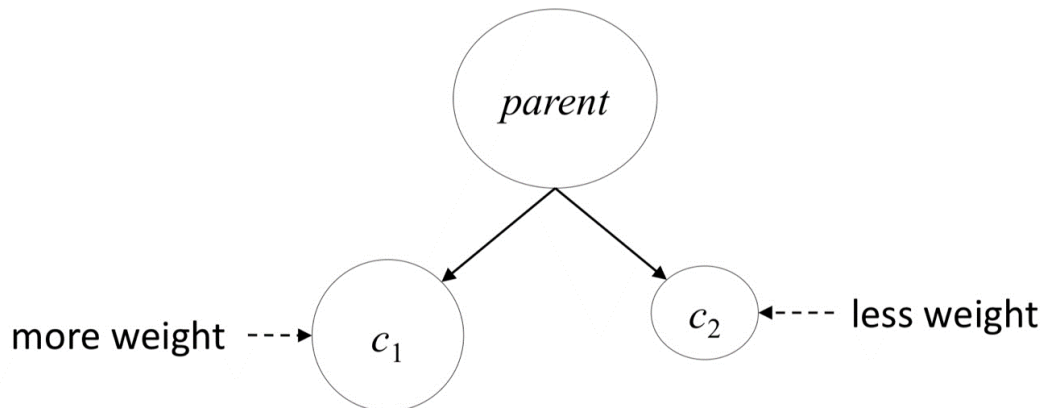
Decision Tree – 분할과 정복

- 클래스 라벨이 결정될 때까지, 인스턴스의 속성에 대해 계속해서 질문함
 - 현재 노드에서 데이터를 가장 잘 분할하는 특성을 선택
 - 선택된 특성에 대해 '**분할 기준**'을 설정하여 데이터를 하위 집합으로 나눔
 - 적절한 수준에서 분할 중단
-
- 각 분할에서 '**정보이득을 최대화(불순도 감소)**'하고자 함



정보이득

- 분할 전후의 불순도를 계산하여, 불순도가 가장 많이 감소하는 분할 선택
- 불순도 계산:
타겟이 이산형 / 카이제곱 통계량(CH2ID), 엔트로피(C4.5), 지니계수(CART)
타겟이 연속형 / ANOVA F-통계량(CH2ID), 분산감소량(CART)





엔트로피


- 노드 내에서 하나의 값을 관측함으로써 얻게 되는 평균 정보량

- $H(S) = p_1 \log_2 \frac{1}{p_1} + p_2 \log_2 \frac{1}{p_2} + p_3 \log_2 \frac{1}{p_3} + \dots$

- $IG = H(\text{parent}) - [p(c_1) \cdot H(c_1) + p(c_2) \cdot H(c_2) + \dots]$


$$-\frac{0}{6} \log_2 \frac{0}{6} - \frac{6}{6} \log_2 \frac{6}{6} = 0$$





$$-\frac{1}{6} \log_2 \frac{1}{6} - \frac{5}{6} \log_2 \frac{5}{6} = 0.65$$


$$-\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1$$

지니계수

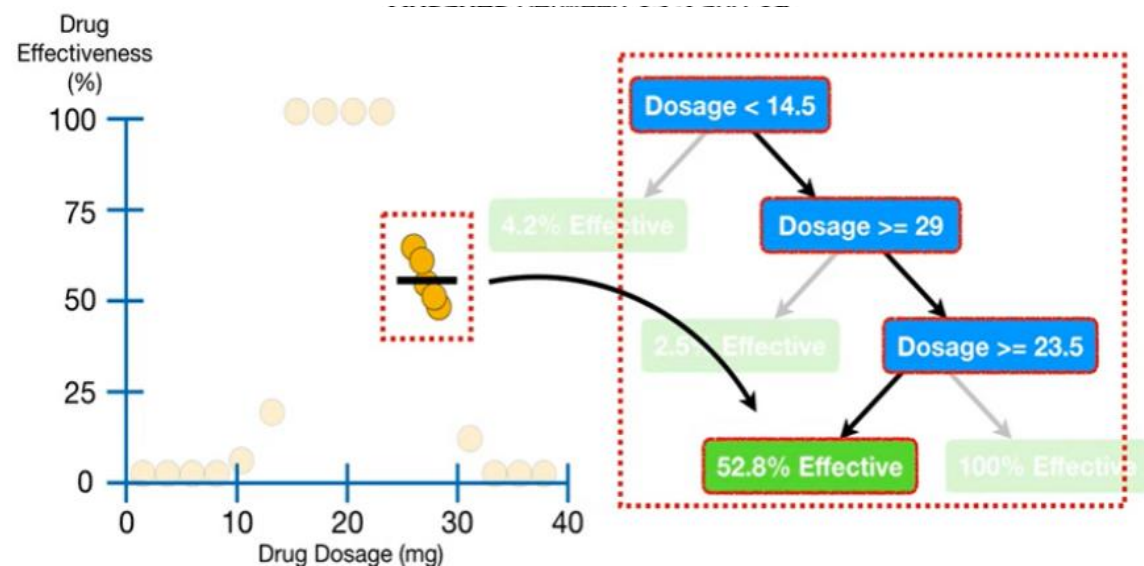
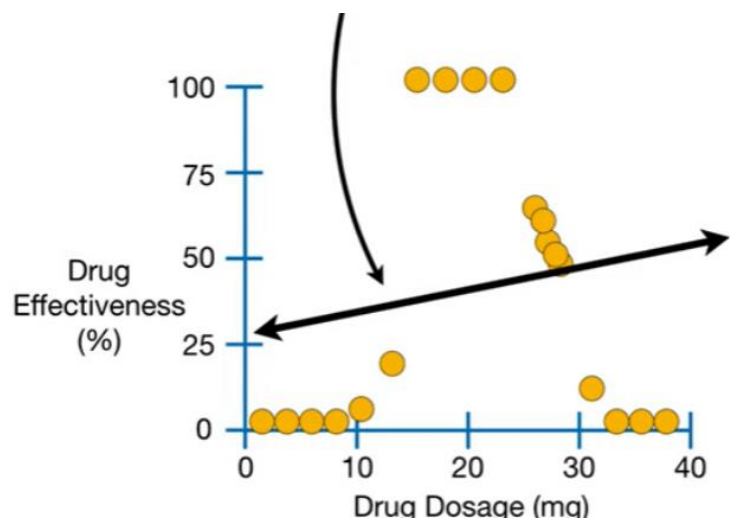
- 값들의 inequality한 정도를 나타냄
- 특정 클래스로 쏠림이 커질수록 제공값이 커짐
- 즉, 분리가 잘 됐다면 지니계수는 0에 가까워짐

• $Gini\ index = 1 - (p(c_1)^2 + p(c_2)^2 + \dots)$

	$1 - (\frac{0^2}{6} + \frac{6^2}{6}) = 0$
	$1 - (\frac{1^2}{6} + \frac{5^2}{6}) = 0.278$
	$1 - (\frac{3^2}{6} + \frac{3^2}{6}) = 0.5$

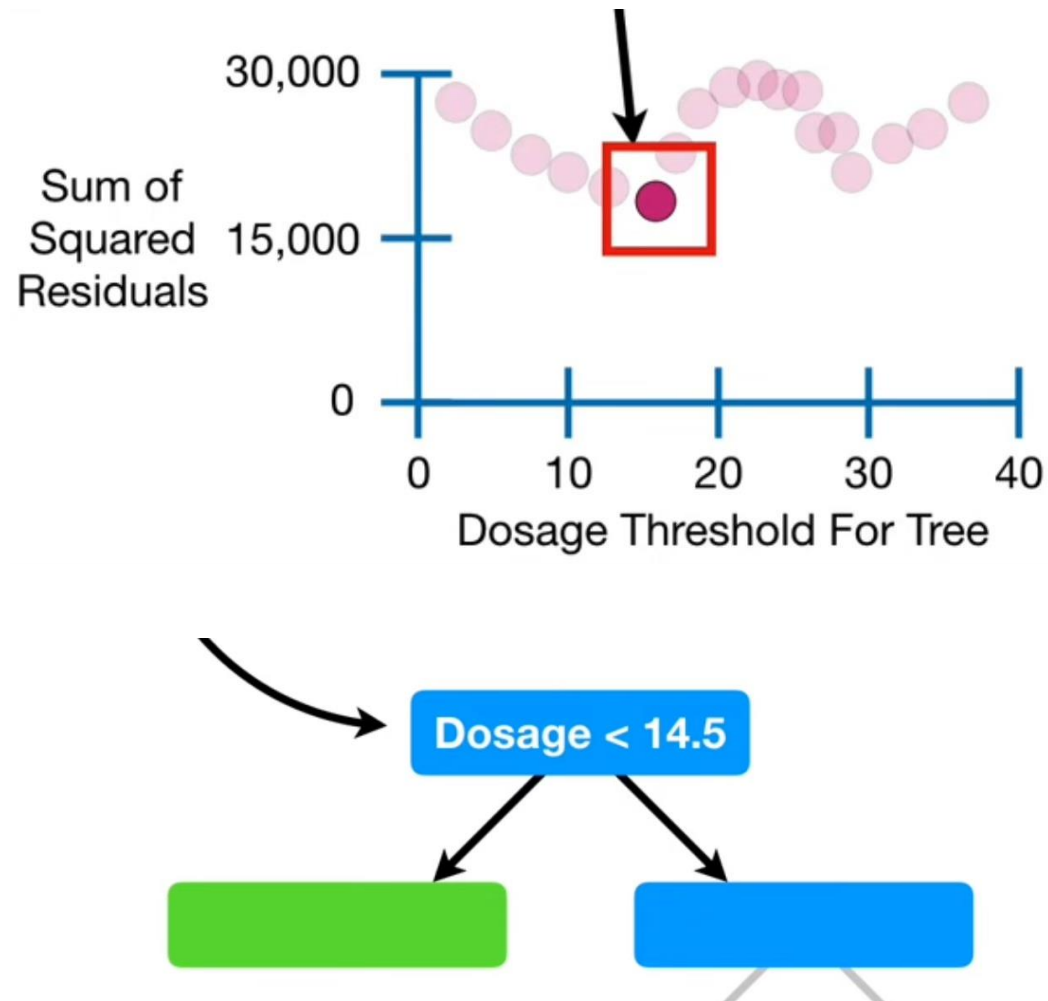
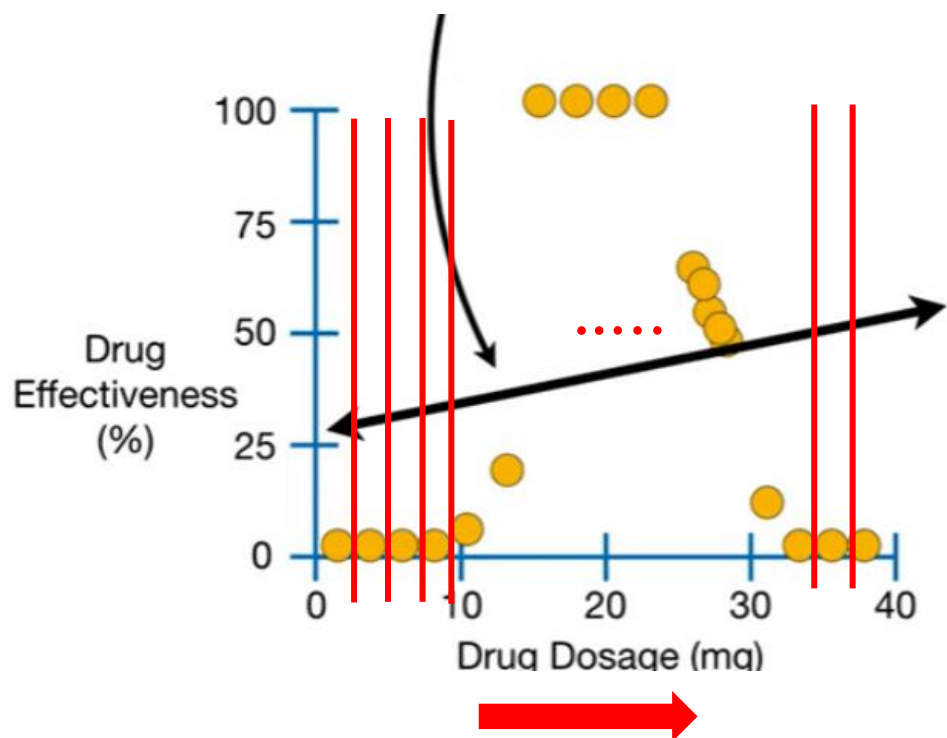
회귀트리

- 선형 예측모델이 적절하지 않을 때 사용
- 출력 데이터가 numeric values (vs. 분류 트리)
- Ex) 투약량(x)과 약효(y)의 관계 관찰



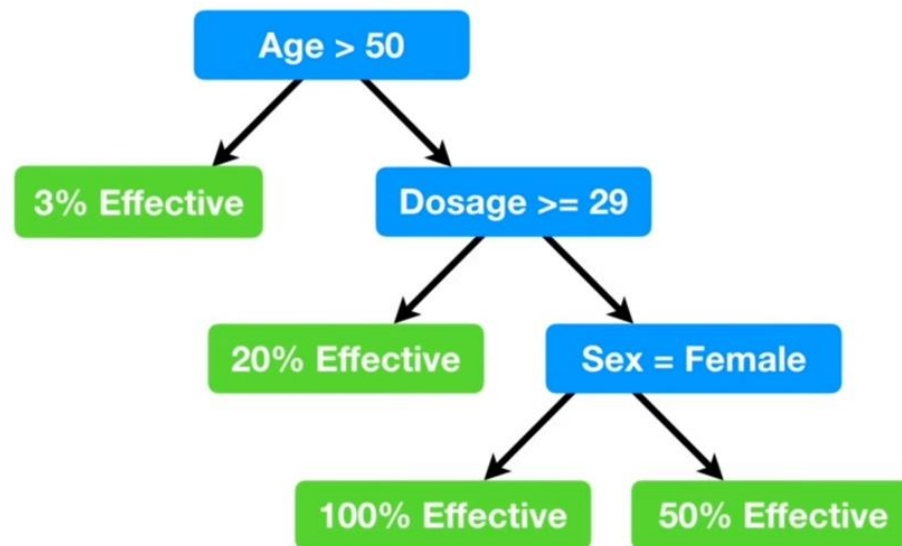
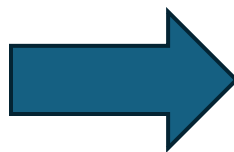
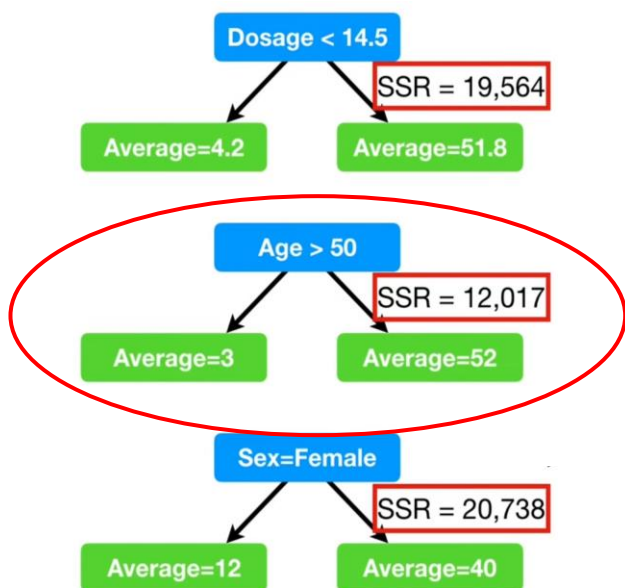
회귀트리

- 분할기준: 잔차제곱합(SSR)



회귀트리

- 독립변수가 여러 개인 경우
- Ex) x_1 (투약량), x_2 (나이), x_3 (성별) ..
- 각 독립변수 별 최적의 threshold를 찾고, 그 중 SSR이 가장 작은 독립변수를 선택



분류트리 실습

- 유방암 데이터 (load_breast_cancer)
- 569개 행과 32개의 열, 누락 데이터X

```
#Decision Tree 학습

from sklearn import tree
from sklearn.model_selection import train_test_split

X = cancer.iloc[:, :-1]
y = cancer.iloc[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)

clf = tree.DecisionTreeClassifier(random_state = 100)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)
```

```
import sklearn.metrics as mt

print('Train_Accuracy: ', clf.score(X_train, y_train))
print('Test_Accuracy: ', clf.score(X_test, y_test))
```

```
Train_Accuracy: 1.0
Test_Accuracy: 0.9298245614035088
```

```
#교차검증
from sklearn.model_selection import cross_val_score
```

```
#각 폴드의 스코어
scores = cross_val_score(clf, X, y, cv = 5)
print(scores)
print('교차검증 평균: ', scores.mean())
```

```
[0.9122807 0.9122807 0.90350877 0.92982456 0.91150442]
교차검증 평균: 0.9138798323241734
```

분류트리 실습

- 최적의 파라미터 서치 & 교차검증

#그리드서치

```
from sklearn.model_selection import GridSearchCV
```

```
parameters = {'max_depth': [3, 5, 7], 'min_samples_split': [3, 5]}  
print(parameters)
```

```
gridcv = GridSearchCV(clf, param_grid = parameters, cv = 5, refit = True)
```

```
gridcv.fit(X_train, y_train)
```

```
scores_df = pd.DataFrame(gridcv.cv_results_['params'])
```

```
scores_df['mean_test_score'] = gridcv.cv_results_['mean_test_score']
```

```
scores_df.sort_values(by = 'mean_test_score', ascending = False)
```

```
{'max_depth': [3, 5, 7], 'min_samples_split': [3, 5]}
```

	max_depth	min_samples_split	mean_test_score
3	5	5	0.924684
0	3	3	0.924652
1	3	5	0.924652
2	5	3	0.922152
4	7	3	0.919589
5	7	5	0.914684

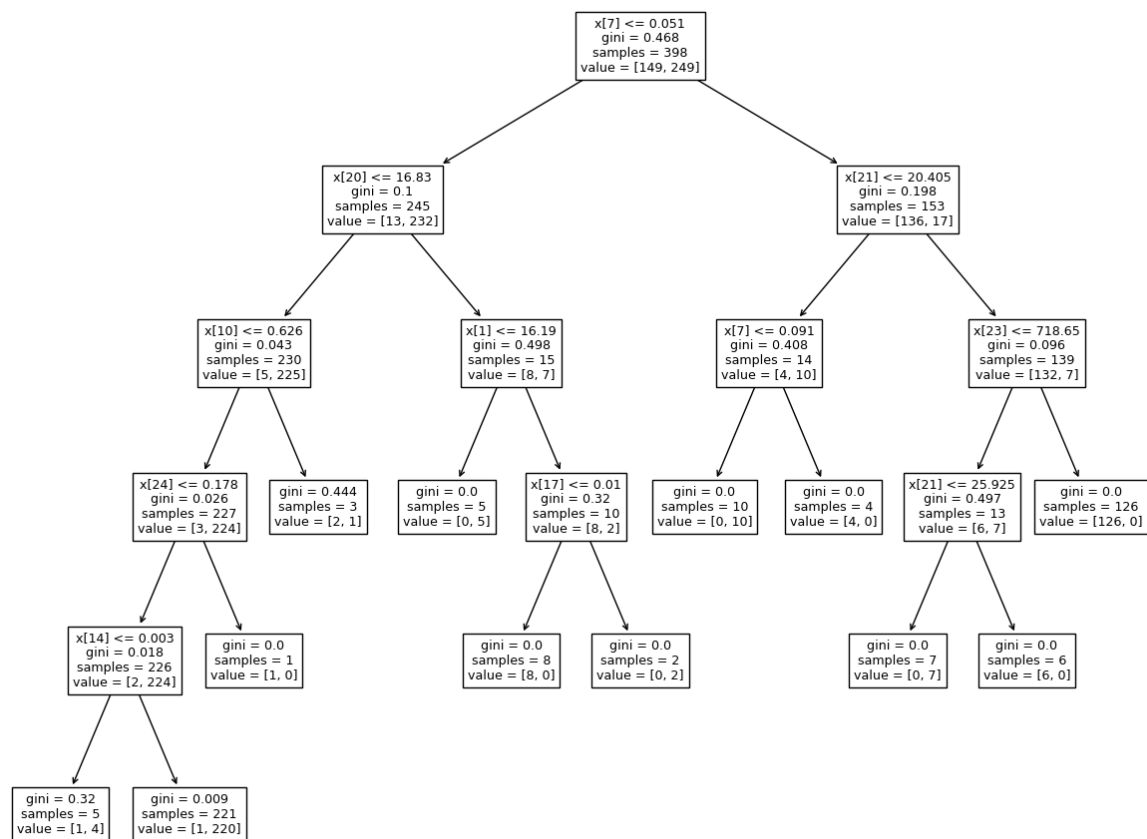
분류트리 실습

```
best_clf = gridcv.best_estimator_  
best_clf.fit(X_train, y_train)  
score = best_clf.score(X_test, y_test)  
print(score)
```

0.9532163742690059

#시각화

```
import matplotlib.pyplot as plt  
  
plt.figure(figsize = (15, 12))  
tree.plot_tree(best_clf)  
plt.show()
```

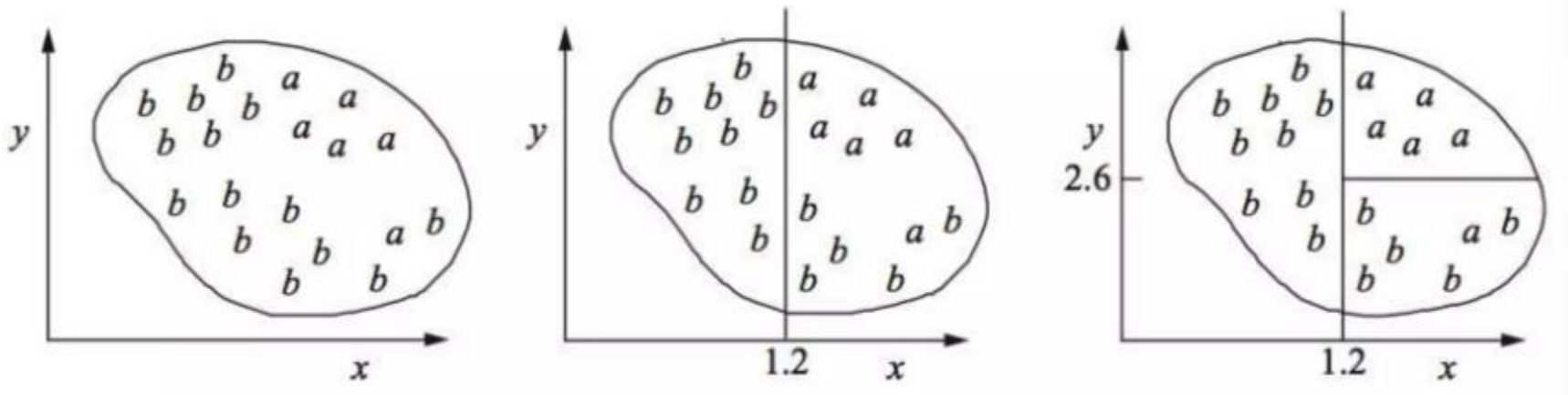


결정트리 특징

- 범주형/숫자형 데이터 모두 적용 가능 (+)
- 휴리스틱 기법 기반으로, 계산 효율성이 좋음 (+)
 - ↔ 항상 최적의 트리를 찾한다고 장담할 수 없음 (-)
- 누락 처리 불가 (데이터 손실 가능성) (-)
- 우연히 irrelevant 속성이 선택될 가능성 O (-)
- 중복된 속성의 경우 비슷한 이득을 주기 때문에 자동적으로 하나만 남음 (+)
Ex) 평수와 제곱미터 중 하나가 선택되면 나머지 하나는 자동으로 후보군에서 제거됨
- 과적합 가능성 O (-)
 - 최대 깊이, 리프노드 개수, 노드 내 데이터 개수 제한 등..

Covering Algorithm – Constructing Rules

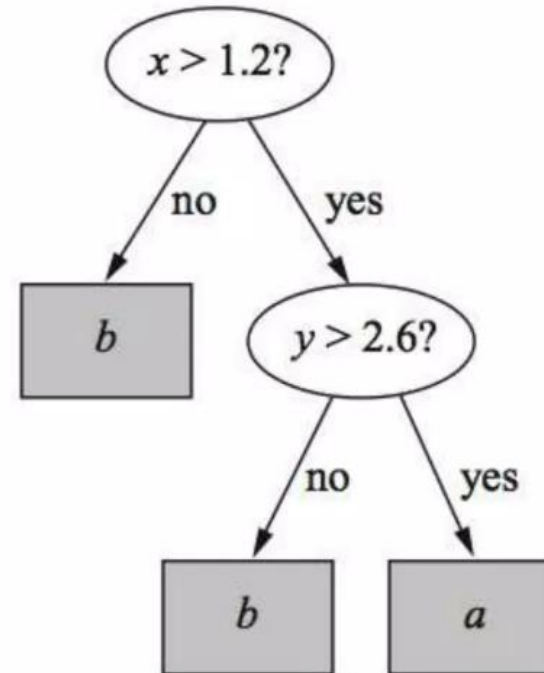
- 분류 능력을 최대화



- Possible rule set for class "a"
 - **If** $x > 1.2$ **then** class = a
 - **If** $x > 1.2$ and $y > 2.6$ **then** class = a

Covering Algorithm – Constructing Rules

- Tree = Instance에 대한 속성, 속성값에 집중
- Rule = 각각의 class에 집중



Covering Algorithm – Constructing Rules

For each class C *#play = Yes 에 대해*
Initialize E to the instance set *#E는 'play = Yes' 인 인스턴스의 모임*
While E contains instances in class C
 Create a rule R with an empty left-hand side that predicts class C *#IF (empty), THEN "play = yes"*
 Until R is perfect (or there are no more attributes to use) do
 For each attribute A not mentioned in R, and each value v, *#outlook = sunny*
 Consider adding the condition A=v to the LHS of R
 Select A and v to maximize the accuracy p/t *#(positive examples) / (instance)*
 (break ties by choosing the condition with the largest p)
 Add A=v to R
 Remove the instances covered by R from E

들어주셔서 감사합니다

