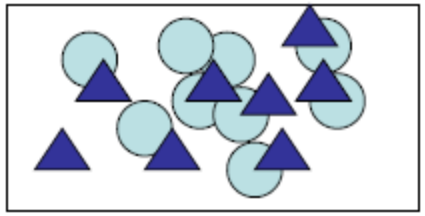
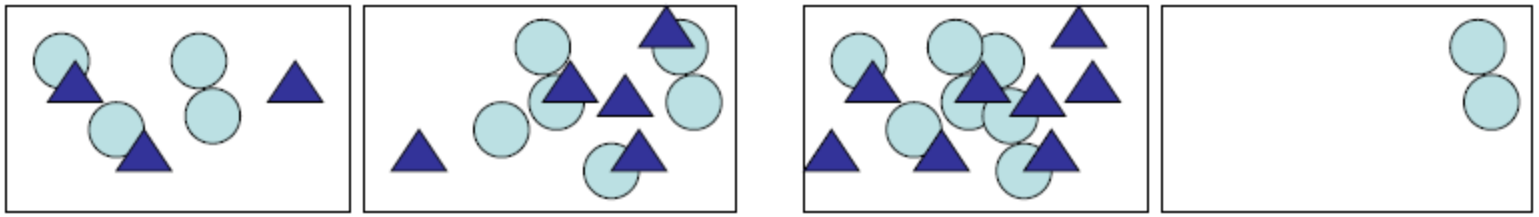
Introduction to Tree Methods

- 스무고개(Twenty Questions)
  - 일련의 질문
  - 앞선 질문의 답이 그 다음 질문을 결정
- 의사결정 나무
  - 연속된 질문들
  - 뿌리 노드
  - 자식 노드
  - 말단 노드
    - 특정 분류값을 가짐

- 모든 의사결정나무 알고리즘들은 기본 절차에 있어서는 공통점을 가지고 있음
  - 목표변수 측면에서 부모노드보다 더 순수도(purity)가 높은 자식노드들이 되도록, 데이터를 반복적으로 더 작은 집단으로 나눈다(repeatedly split)

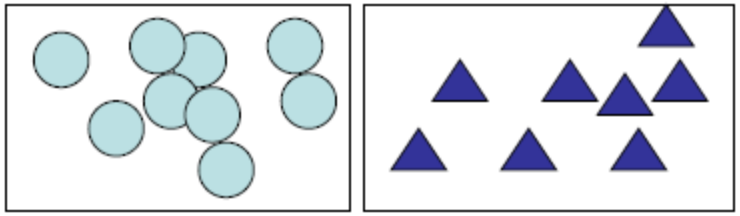


Original Data



Poor Split

Poor Split



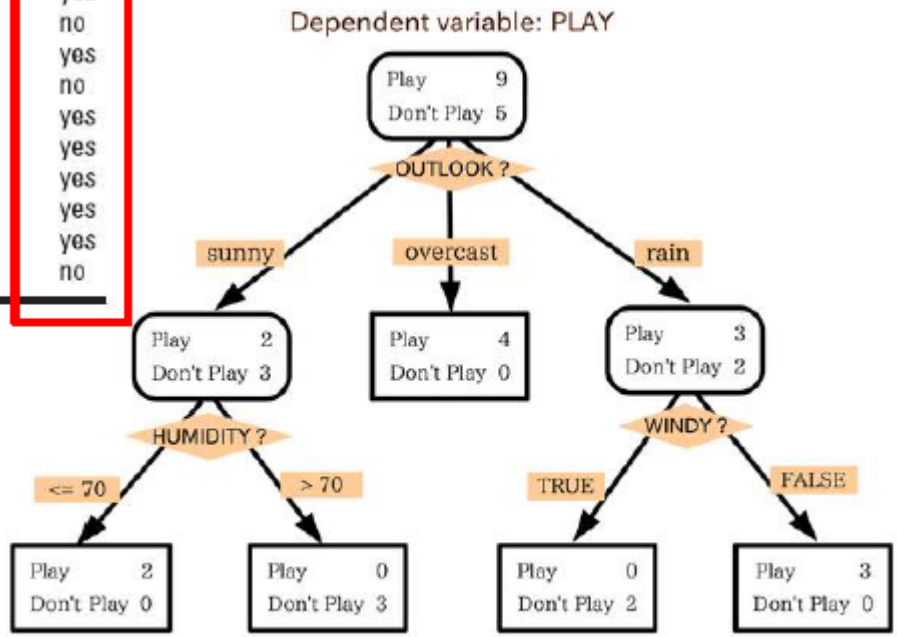
Good Split

- 분할 시도의 중단
  - 주어진 노드의 순수도를 유의한 수준으로 증가시키는 분할이 없을 때
  - 해당 노드에 속한 레코드의 수가 미리 설정한 하한선에 도달했을 때
  - 나무의 깊이가 미리 정해 놓은 한계에 도달했을 때
- 최대나무(Full tree)는 일반적으로 새로운 레코드들의 집합을 분류하는 데 최상의 성과를 제공하는 나무는 아님

- **의사결정나무의 예**

타겟변수

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

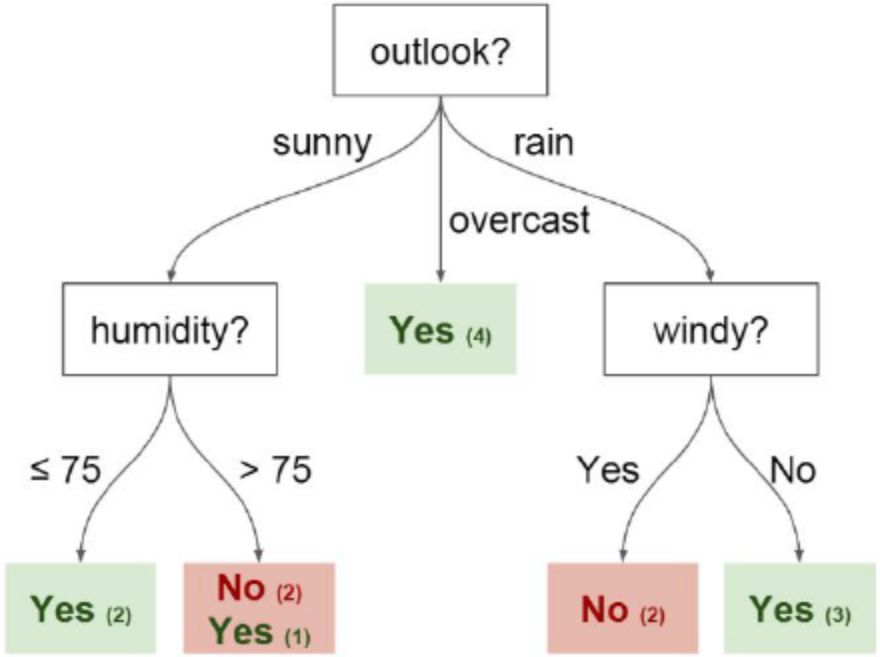


Leaf node

Temperature	Outlook	Humidity	Windy	Played?
Mild	Sunny	80	No	Yes
Hot	Sunny	75	Yes	No
Hot	Overcast	77	No	Yes
Cool	Rain	70	No	Yes
Cool	Overcast	72	Yes	Yes
Mild	Sunny	77	No	No
Cool	Sunny	70	No	Yes
Mild	Rain	69	No	Yes
Mild	Sunny	85	Yes	Yes
Mild	Overcast	77	Yes	Yes
Hot	Overcast	74	No	Yes
Mild	Rain	77	Yes	No
Cool	Rain	73	Yes	No

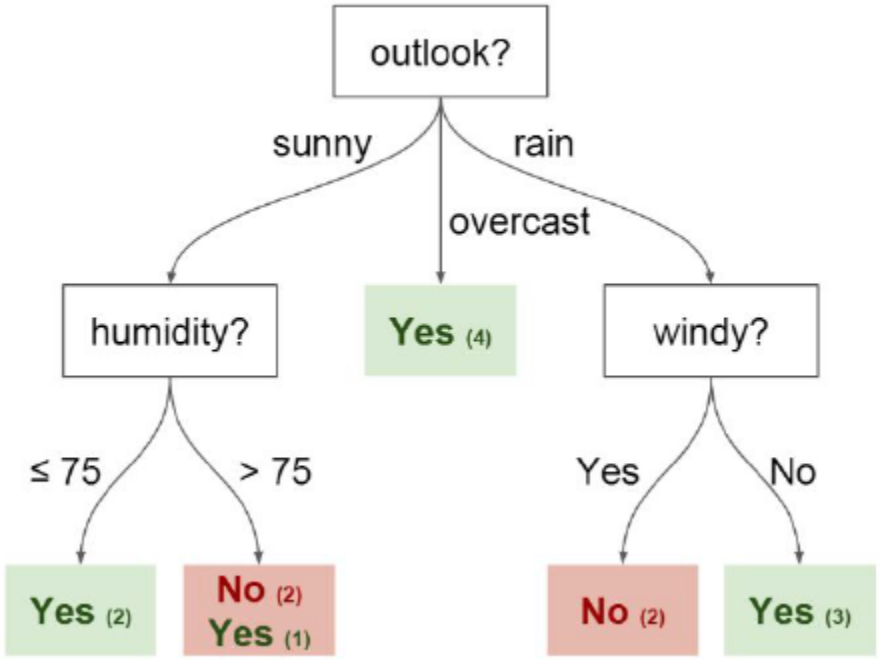
I want to use this data to predict whether or not he will show up to play.

An intuitive way to do this is through a Decision Tree



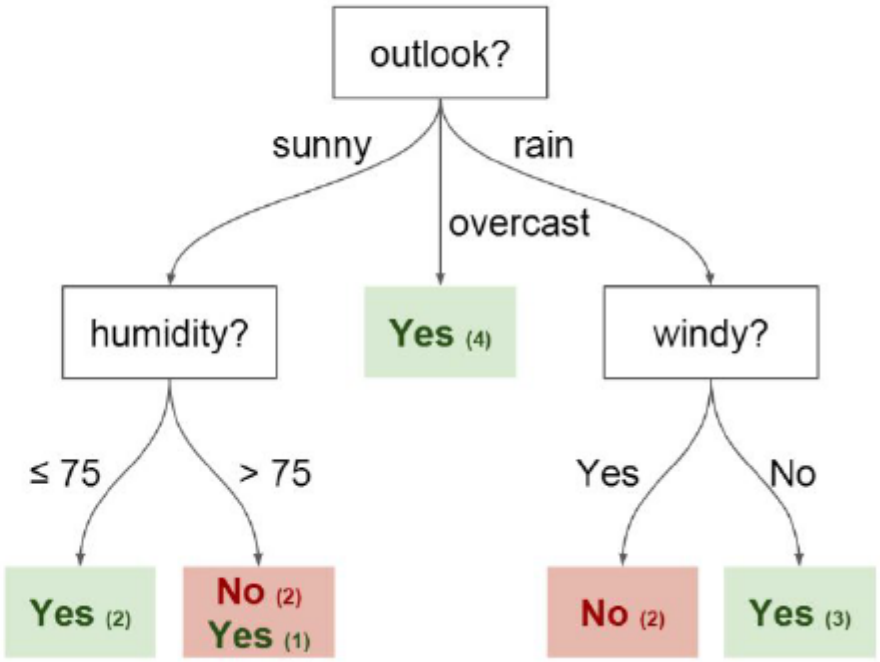
In this tree we have:

- Nodes
  - Split for the value of a certain attribute
- Edges
  - Outcome of a split to next node



In this tree we have:

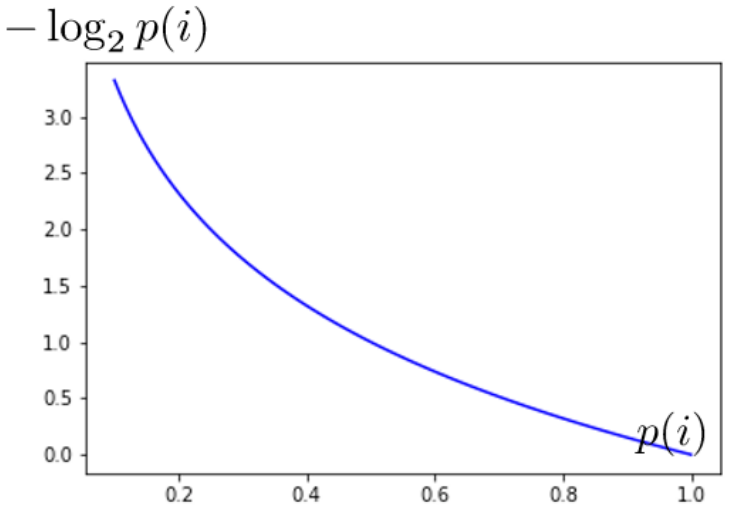
- Root
  - The node that performs the first split
- Leaves
  - Terminal nodes that predict the outcome



Entropy

$$h(D) = -\sum_{i=1}^{m} p_i log_2(p_i) \qquad \text{where} \begin{cases} D & \text{Data set} \\ p_i & \text{Probability of label } i \end{cases}$$

- 확률이 1이면 Entropy 0
- 확률이 작을수록 커짐



Entropy

- Entropy는 목적 달성을 위한 경우의 수를 정량적으로 표현하는 수치 ➔ 작을 수록 경우의 수가 적음
- Higher Entropy → Higher uncertainty
- Lower Entropy → Lower uncertainty

Entropy가 작으면 얻을 수 있는 정보가 많다.

	age	income	student	credit_rating	class_buys_computer
0	youth	high	no	fair	no
1	youth	high	no	excellent	no
2	middle_aged	high	no	fair	yes
3	senior	medium	no	fair	yes
4	senior	low	yes	fair	yes
5	senior	low	yes	excellent	no
6	middle_aged	low	yes	excellent	yes
7	youth	medium	no	fair	no
8	youth	low	yes	fair	yes
9	senior	medium	yes	fair	yes
10	youth	medium	yes	excellent	yes
11	middle_aged	medium	no	excellent	yes
12	middle_aged	high	yes	fair	yes
13	senior	medium	no	excellent	no

$$h(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$h(D) = -\frac{9}{14}\log_2\frac{9}{14} + -\frac{5}{14}\log_2\frac{5}{14}$$
$$= 0.940\text{bits}$$

```
x = np.array([9/14,5/14])
y = np.log2(x)

- sum(x * y)
```

```
0.94028595867063114
```

Growing a Decision Tree

- Decision Tree를 성장(만드는) 시키는 알고리즘이 필요
- 어떻게 하면 가장 잘 분기(branch)를 만들수 있는가?
- Data의 attribute를 기준으로 분기를 생성
- 어떤 attribute를 기준이면 가장 entropy가 작은가?
- 하나를 자른 후에 그 다음은 어떻게 할 것인가?

Algorithms of Decision Tree

- 크게 두 가지 형태의 decision tree 알고리즘 존재
- 알고리즘 별 attribute branch 방법이 다름
- ID3 → C4.5(Ross Quinlan), CART
- 연속형 변수를 위한 regression tree도 존재

Information Gain

- Entropy 함수를 도입하여 branch splitting

- Information Gain: Entropy를 사용하여 속성별 분류시 Impurity를 측정하는 지표

- (전체 Entropy – 속성별 Entropy)로 속성별 Information Gain을 계산함

Information Gain

$$Info(D) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

전체 데이터 D의 정보량

$$Info_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{D} * Info(D_j)$$

속성 A로 분류시 정보량

$$Gain(A) = Info(D) - Info_A(D)$$

A 속성의 정보 소득

$$\text{분기 전 엔트로피} = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = 1$$

$$\text{분기 후 엔트로피} = \frac{1}{24}(-\log_2 1) + \frac{23}{24}\left(-\frac{12}{23}\log_2\left(\frac{12}{23}\right) - \frac{11}{23}\log_2\left(\frac{11}{23}\right)\right) \approx 0.96$$

$$\text{정보획득} = 1 - 0.96 = 0.04$$

Growing a Decision Tree

Age $Gain(age) = Info(D) - Info_age(D)$

Credit $Gain(credit) = Info(D) - Info_{credit}(D)$

Income $Gain(Income) = Info(D) - Info_{Income}(D)$

Student $Gain(Studnet) = Info(D) - Info_{Student}(D)$

Growing a Decision Tree

Age

$$Gain(age) = Info(D) - Info_age(D)$$

$$Info_{age}(D) = -\sum_{j=1}^{v} \frac{|D_j|}{D} * Info(D$$

$$Info_{age}(D) = \frac{5}{14} \times \left(-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5}\right) + \frac{4}{14} \times \left(-\frac{4}{4}\log_2\frac{4}{4}\right) + \frac{5}{14} \times \left(-\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5}\right)$$

	age	income	student	credit_rating	class_buys_computer
0	youth	high	no	fair	no
1	youth	high	no	excellent	no
2	middle_aged	high	no	fair	yes
3	senior	medium	no	fair	yes
4	senior	low	yes	fair	yes
5	senior	low	yes	excellent	no
6	middle_aged	low	yes	excellent	yes
7	youth	medium	no	fair	no
8	youth	low	yes	fair	yes
9	senior	medium	yes	fair	yes
10	youth	medium	yes	excellent	yes
11	middle_aged	medium	no	excellent	yes
12	middle_aged	high	yes	fair	yes
13	senior	medium	no	excellent	no

Growing a Decision Tree

Age

```
get_info(pd_data) - get_attribute_info(pd_data, "age")
```

```
0.24674981977443933
```

$$Gain(age) = Info(D) - Info_age(D)$$

Credit

```
get_info(pd_data) - get_attribute_info(pd_data, "income")
```

```
0.029222565658954869
```

$$Gain(credit) = Info(D) - Info_{credit}(D)$$

Income

```
get_info(pd_data) - get_attribute_info(pd_data, "student")
```

```
0.15183550136234159
```

$$Gain(Income) = Info(D) - Info_{Income}(D)$$

Student

```
get_info(pd_data) - get_attribute_info(pd_data, "credit_rating")
```

```
0.048127030408269489
```

$$Gain(Studnet) = Info(D) - Info_{Student}(D)$$

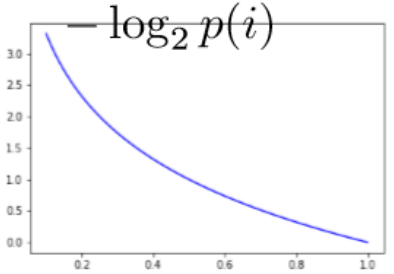
Gain Ratio

- ID3의 발전형인 C4.5 알고리즘에서 사용되는 measure

http://dl.acm.org/citation.cfm?id=152181

- Info(D)의 값을 평준화 시켜 분할 정보 값을 대신 사용

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} * \left(\log_2 \frac{|D_j|}{|D|} \right)$$



$$GainRatio(A) = \frac{Gain(A)}{SplitInfo_A(D)} = \frac{Info(D) - Info_a(D)}{SplitInfo_A(D)}$$

Gini Index

- CART 알고리즘의 split measure

https://www.amazon.com/dp/0412048418?tag=inspiredalgor-20

- 훈련 튜플 세트를 파티션으로 나누었 때 불순한 정도 측정

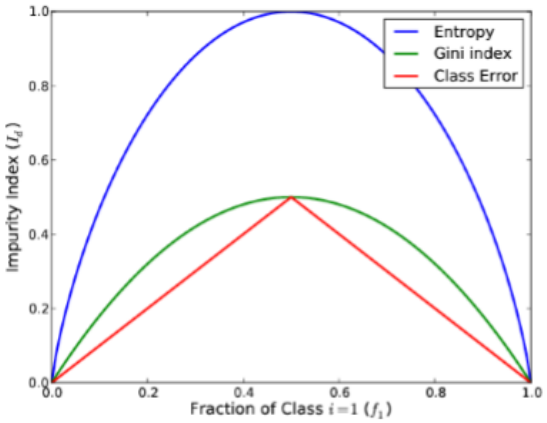
$$\text{Gini}(D) = 1 - \sum_{i=1}^{m} p_i^2 = 1 - \sum_{i=1}^{m} \frac{|C_{i,D}|}{|D|} \qquad \text{where } C_i \text{ is a class}$$

- 데이터의 대상 속성을 얼마나 잘못 분류할지를 계산

Gini Index

- 실제 Gini Index는 Entropy와 비슷한 그래프가 그려짐
- 0.5일 때 Impurity의 최대화, 약 극점에서 0

$$\mathrm{Gini}(D) = \sum_{i=1}^{m} p_i(1-p_i) = 1 - \sum_{i=1}^{m} p_i^2$$



지니 계수(Gini Index): CART (1/2)

- 데이터 셋 T가 n개의 클래스로 이루어진 데이터들을 포함할 때, 지니 계수 $gini(T)$는 다음과 같이 정의

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2$$

  여기서 p_j 는 데이터 셋 T에서 클래스 j의 상대적 빈도수

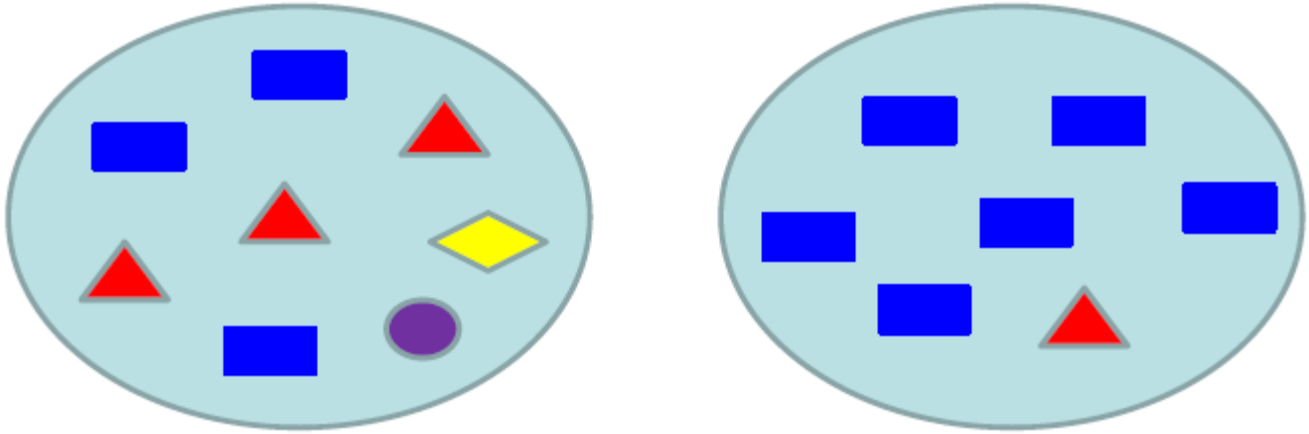
- 데이터 셋 T 가 각각 크기 N_1과 N_2인 두개의 데이서 셋 T_1 와 T_2 로 분할 되었을 경우, 지니 계수 $gini(T)$ 는 다음과 같이 정의

$$gini_{split}(T) = \frac{N_1}{N} gini(T_1) + \frac{N_2}{N} gini(T_2)$$

- 가장 작은 $gini_{split}(T)$ 를 갖는 변수가 분할기준 변수로 선정

지니 계수(Gini Index): CART (2/2)

- 두 개의 nodes가 있다고 가정



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

- 첫 번째 node에 대한 지니 계수 값
  = 1 - $(3/8)^2$ - $(1/8)^2$ - $(3/8)^2$ - $(1/8)^2$ = 0.69
- 두 번째 node에 대한 지니 계수 값 = 1 - $(6/7)^2$ - $(1/7)^2$ = 0.25
- $gini_{split}$= (8/15)*0.69+(7/15)*0.25 = 0.4847
- 동질적인(Purer) node 일수록 더 작은 지니 계수 값을 지님

Information Gain(IG): ID3

- 분할기준 변수로서의 Outlook:
- Root node(Parent node)에서의 Information = I(9,5) =

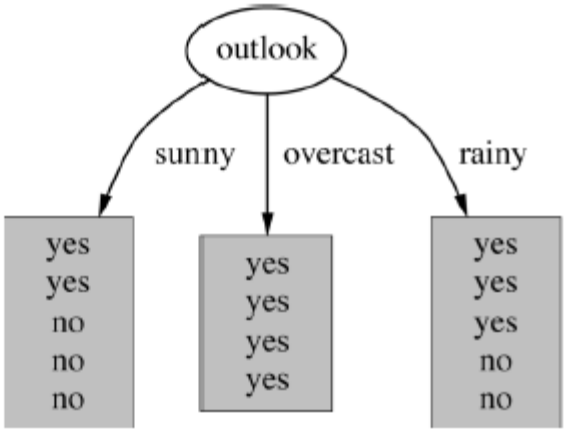
  yes no

  = E(9/14, 5/14) = - (9/14)log(9/14) - (5/14)log(5/14) = 0.940
- Child nodes에서의 평균 information

  =5/14*I(2,3) + 4/14*I(4,0) + 5/14*I(3,2)

  =5/14*E(2/5,3/5) + 4/14*E(4/4,0/4) + 5/14*E(3/5,2/5)

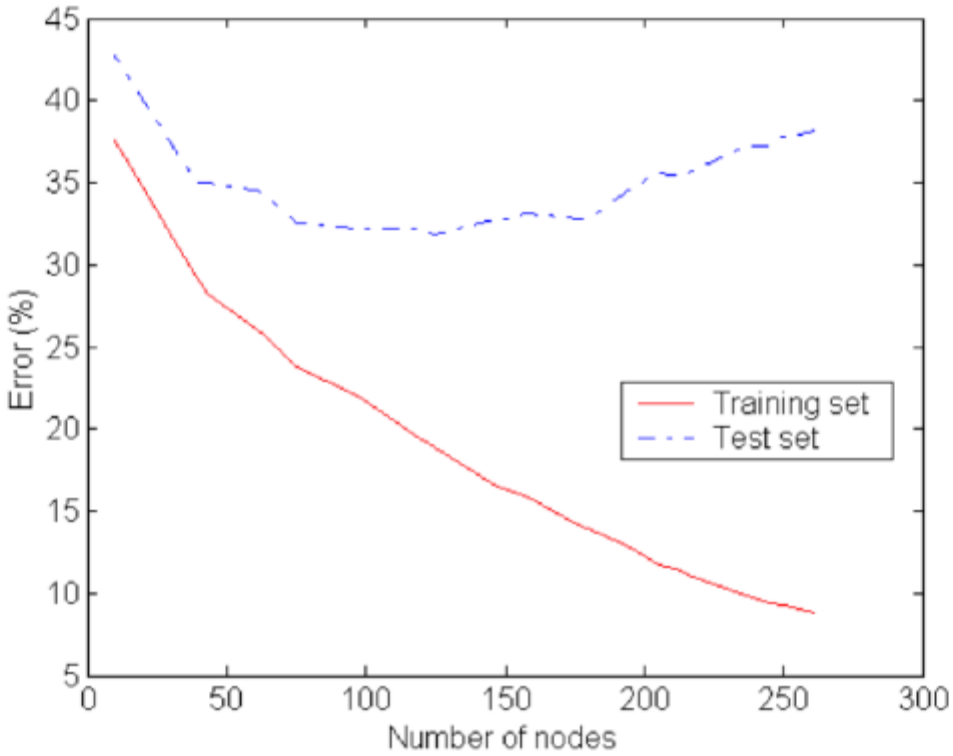
  = 0.693
- IG(**outlook**)= 0.940 - 0.693 = 0.247
- 유사한 방법으로 계산하면:
  - IG(temperature) = 0.029
  - IG(humidity) = 0.152
  - IG(windy) = 0.048



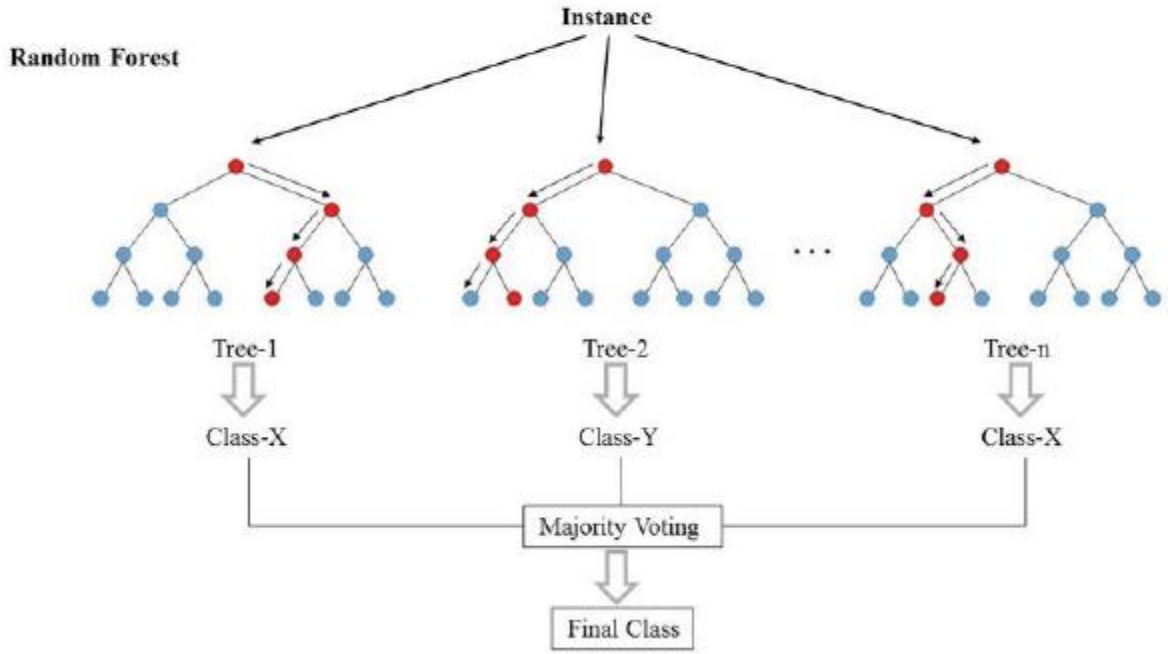
Decision Tree를 만들다 생기는 문제점

- class의 leaf node의 결정
- 너무 많을 경우 → Overfitting
- impurity 또는 variance는 낮은데, 노드에 데이터가 1개
- 어떤 시점에서 트리 가지치기 해야할 지 결정이 중요

오분류율 최소화


45
40
35
30
25
20
15
10
5
Error (%)
0
50
100
150
200
250
300
Number of nodes
Training set
Test set

Random Forest



Random Forest

- correlation 낮은 m개의 subset data로 학습
- Tree의 구성은 binary로 구성
- Split시 검토대상 feature를 random하게 n개 선정
- 전체 feature를 p라 할 때, n = p이면 bagging tree
- Feature의 재사용이 가능, n은 $\sqrt{p}$ 또는 $^{p}/_{3}$
- Variance가 높은 트리 → last node 1 ~ 5

Random Forest

- Bagging + Randomized decision tree
- Variance가 높은 decision tree들의 ensemble
- 여러개의 나무 → Forest
- 가장 간단하면서 높은 성능을 자랑하는 대표적인 모델
- Regressor와 Classifier 모두 지원