
START
머신러닝과 딥러닝
Machine Learning deep Learning

Chapter 6. 비선형분류모형 I

Machine Learning & Deep Learning

손영두

e-mail: youngdoo@dongguk.edu

의사결정나무 (Decision tree) 분류

의사결정나무 분류

- ✓ 널리 사용되는 분류기 중 하나: 복잡한 데이터에 대하여 좋은 성능을 보임
- ✓ 학습 데이터로부터 “설명 가능한” 분류 기준을 결정
- ✓ 데이터를 “뿌리”로부터 말단의 “잎“까지 순차적으로 분류

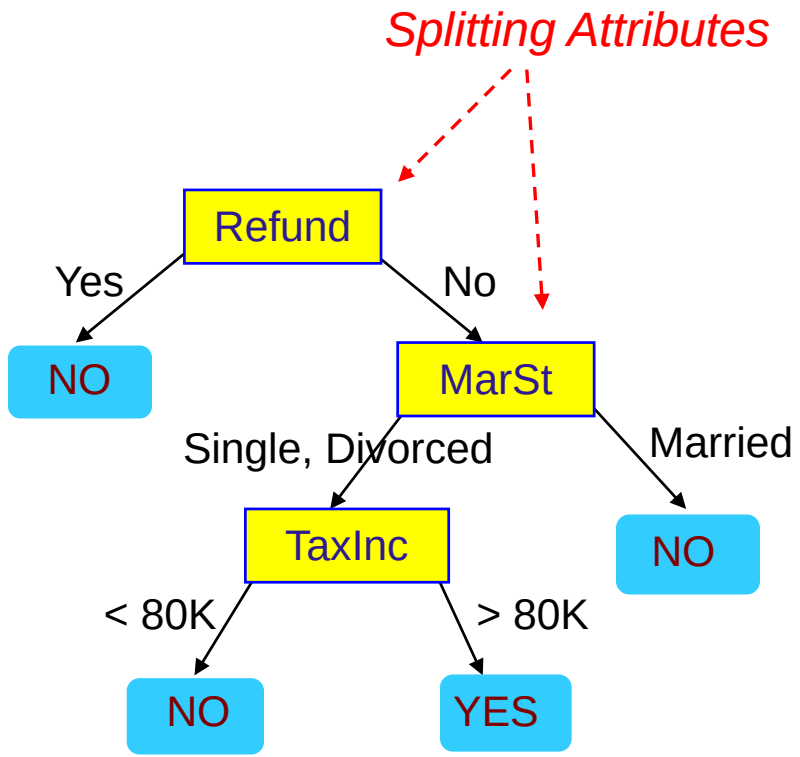
의사결정나무 알고리즘

- ✓ **Hunt’s alogirhm**
- ✓ **CART**
- ✓ **ID3, C4.5**
- ✓ **SLIQ, SPRINT**

의사결정나무 (Decision tree) 분류

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Defaulted (class)
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

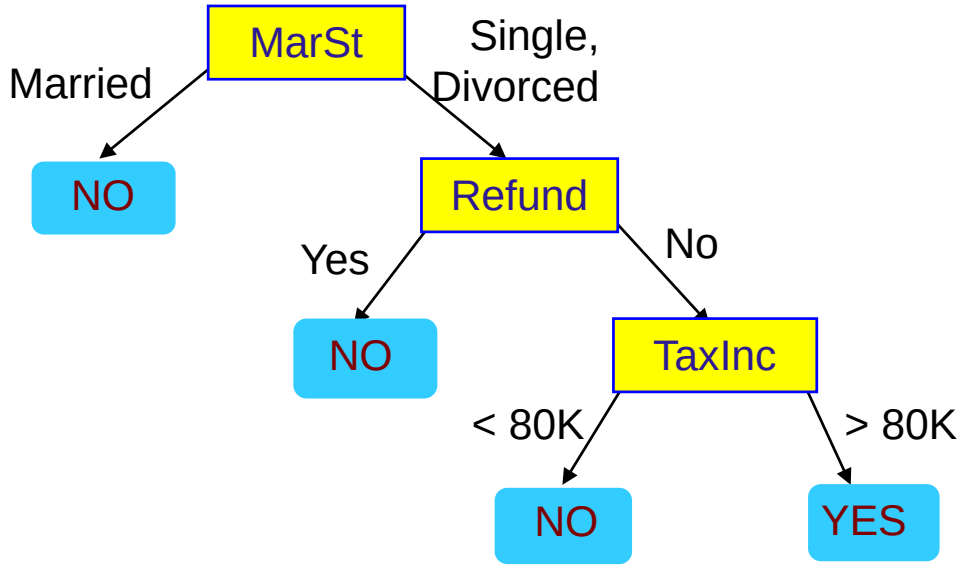
Training Data



Model: Decision Tree

의사결정나무 (Decision tree) 분류

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Defaulted**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



같은 데이터의 분류에 대하여 하나 이상의 서로 다른 트리 학습이 가능 !

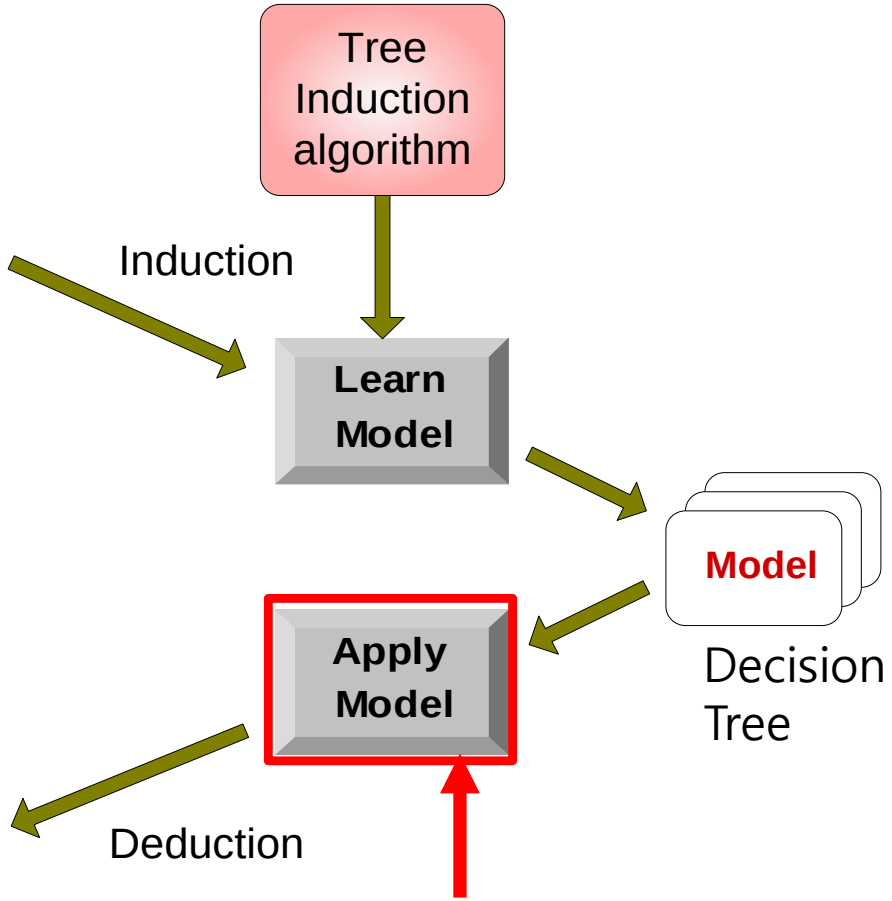
의사결정나무 (Decision tree) 분류

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

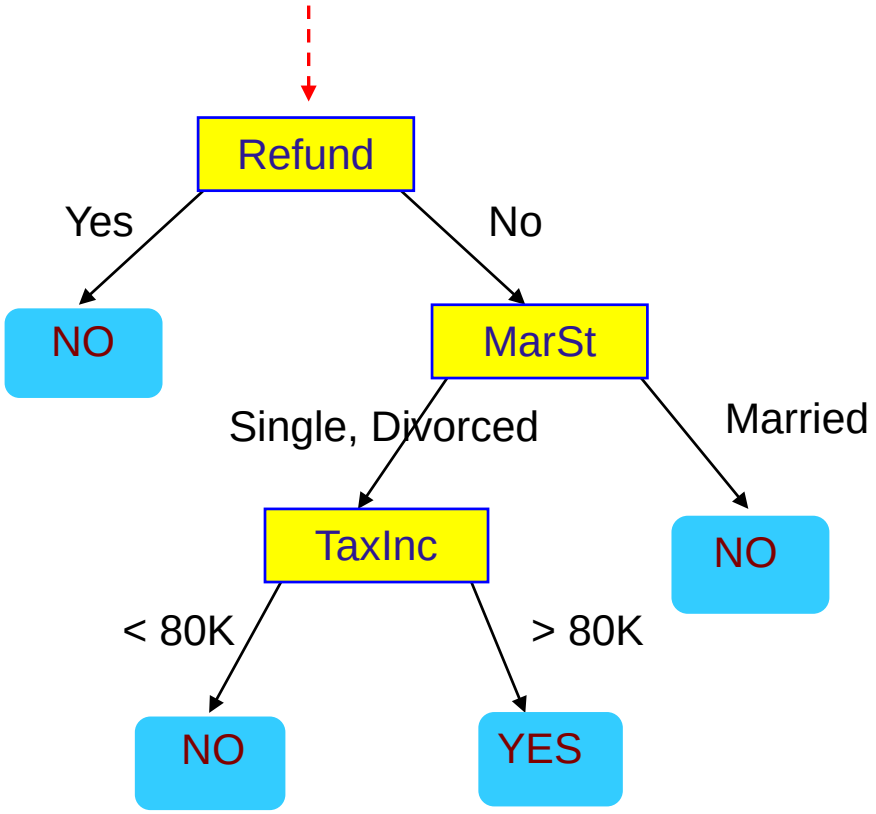
Test Set



의사결정나무 (Decision tree) 분류

테스트 데이터에의 적용

Start from the root of tree.



Test Data

Refund	Marital Status	Taxable Income	Defaulted
No	Married	80K	?

Hunt's Algorithm

✓ D_t : node t 에 있는 학습 데이터의 수

✓ General Procedure:

- 만일 D_t 에 속한 데이터가 모두 하나의 클래스 y_t에 속해있다면,

  t 는 y_t 클래스의 leaf node

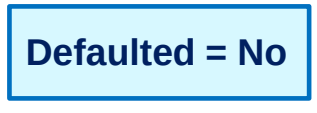
- 만일 D_t 에 속한 데이터가 존재하지 않으면, t 는 기본 클래스 y_d 의 leaf node

- 만일 D_t 에 속한 데이터가 둘 이상의 클래스에 속해 있다면, 특정 기준을 이용하여 데이터들을 더 작은 부분집합으로 분리

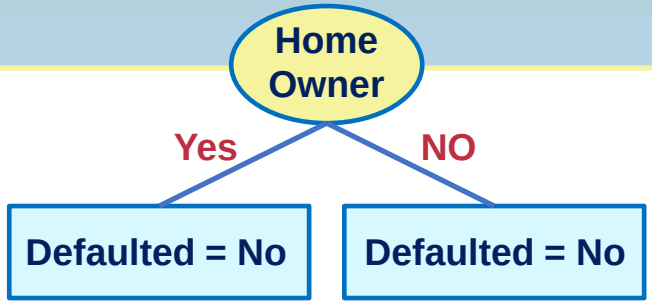
Hunt's Algorithm

	▼ birnary	▼ categorical	▼ continuous	▼ class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

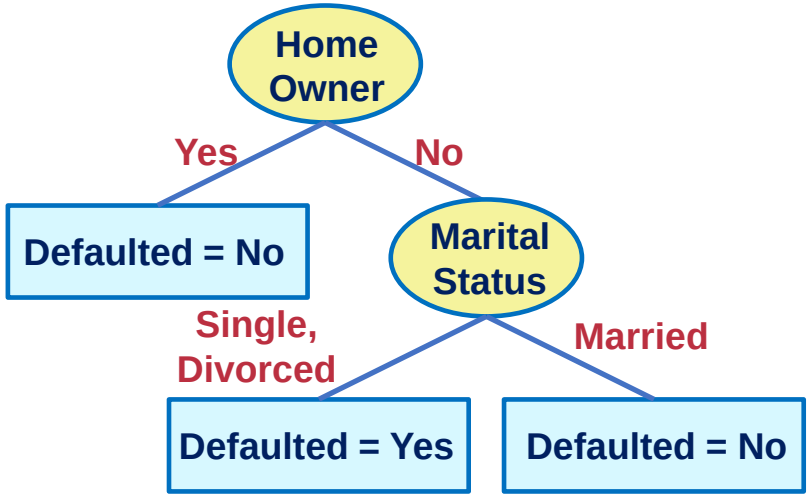
Hunt's Algorithm



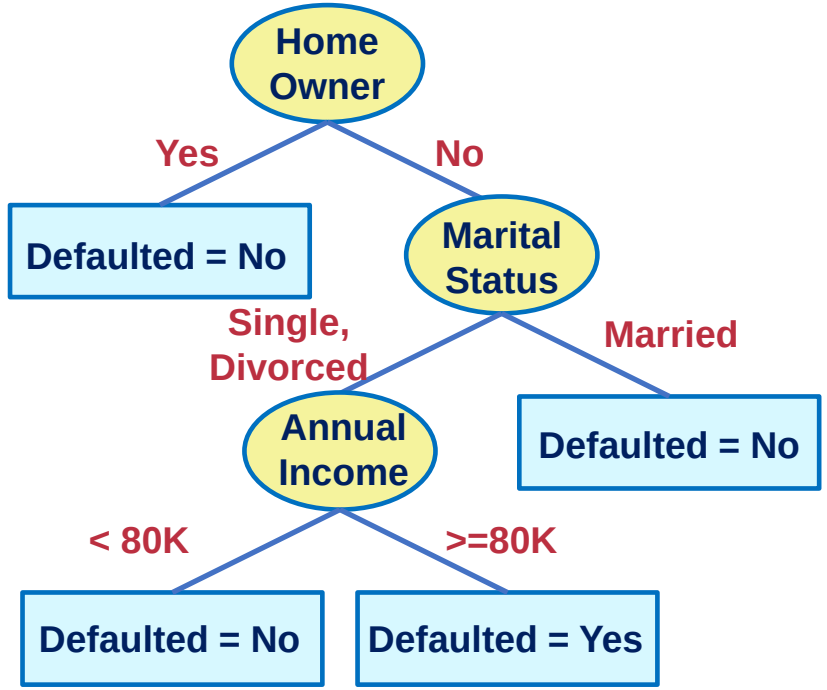
(A)



(B)



(C)



(D)

Tree Induction

Greedy strategy (탐욕적 전략)

- 하나의 입력 변수를 이용하여 특정 최적 기준을 만족하는 분리 조건을 탐색

Issues

- 어떻게 분리할 것인가?
  - ✓ 어떻게 분리 기준 변수를 선택할 것인가?
  - ✓ 최적의 분리는 무엇인가?
- 언제까지 분리할 것인가?

어떻게 분리할 것인가?

✓ Attribute type에 따라

Nominal: 명목형

Ordinal: 순서형

Continuous: 연속형

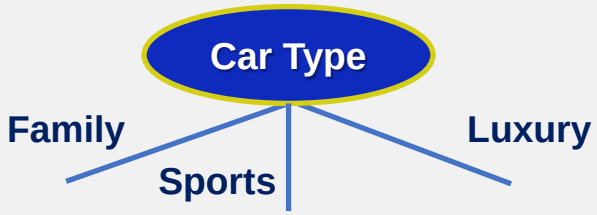
✓ 분리의 수에 따라

2-way split

Multi-way split

Splitting Based on Nominal Attributes

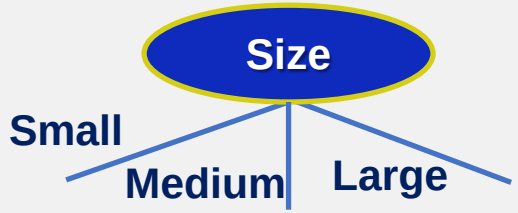
Multi-way split: 서로 다른 값을 모두 사용하여 분리



Binary split: 값 들을 두 개의 부분집합으로 분리. 최적의 분리 기준 탐색 필요

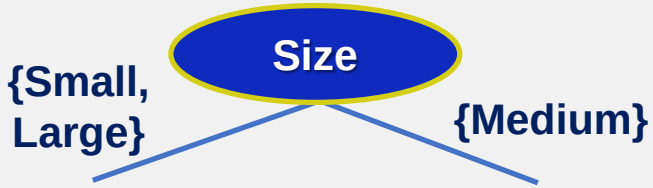
Splitting Based on Ordinal Attributes

Multi-way split: 서로 다른 값을 모두 사용하여 분리



Binary split: 값 들을 두 개의 부분집합으로 분리. 최적의 분리 기준 탐색 필요

Wrong split

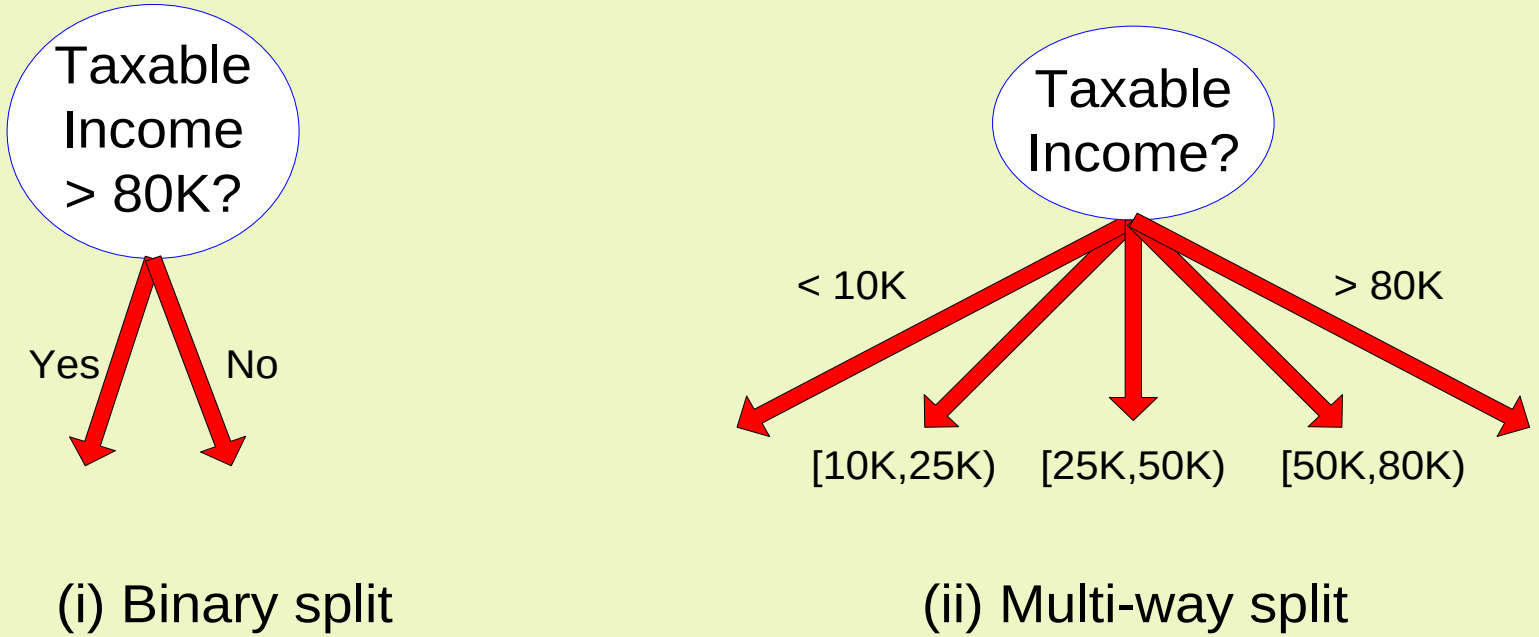


Splitting Based on Nominal Attributes

이산화를 통하여 순서형 변수로 재구성

Binary Split: (A < v) or (A ≥ v)

✓ 모든 기준 값에 대하여 탐색

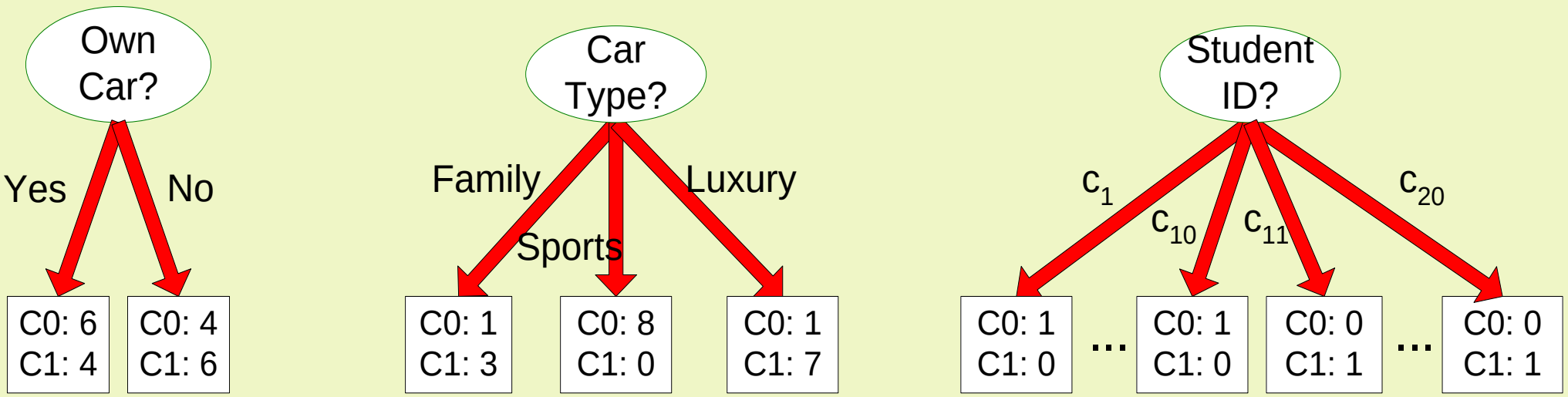


(i) Binary split

(ii) Multi-way split

How to Determine the Best Split

**Before Splitting: 10 records of class 0,
10 records of class 1**



Which test condition is the best?

How to Determine the Best Split

Greedy approach:

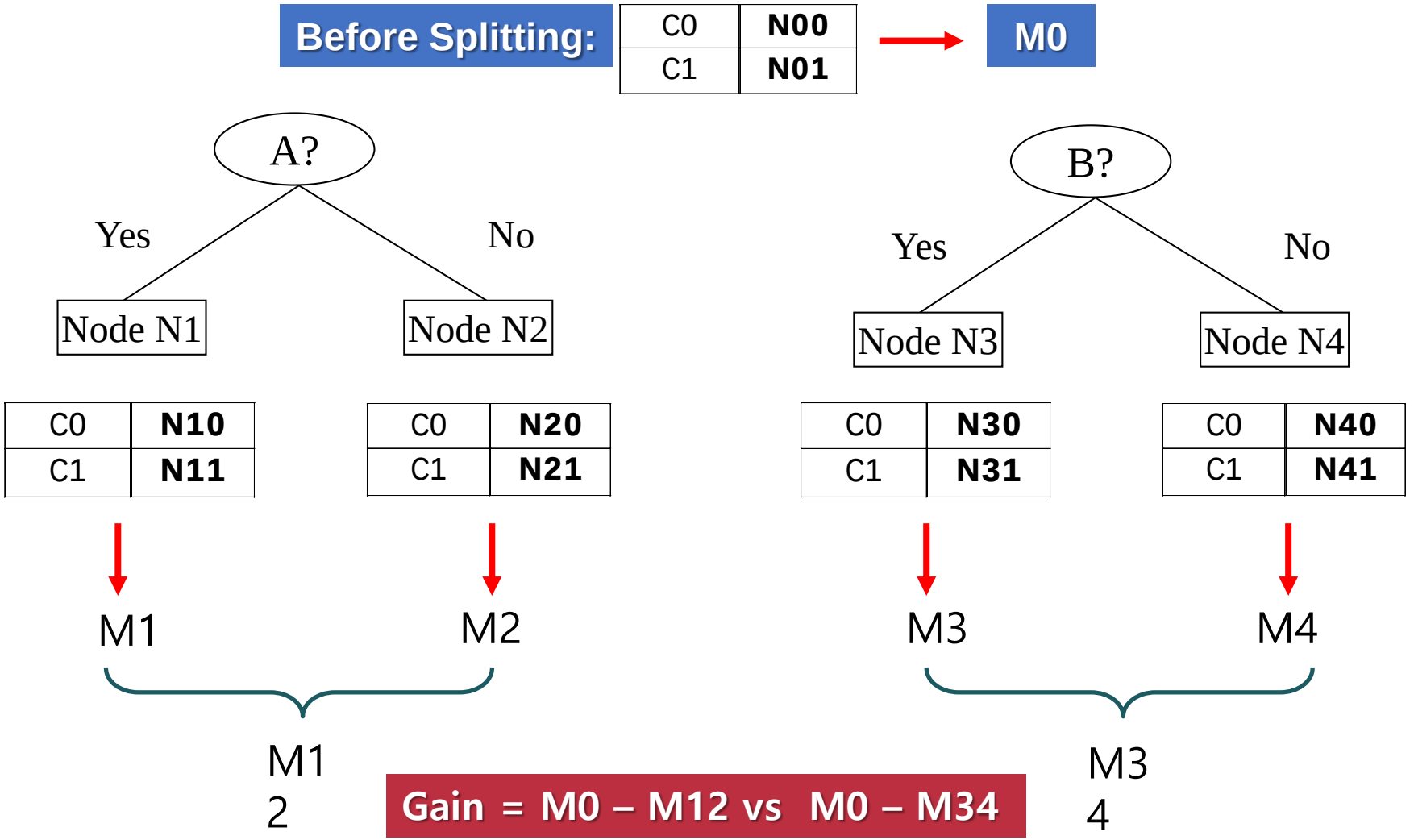
- ✓ 균질(homogeneous)한 class 분포가 선호됨

불순도(impurity)에 대한 측정이 필요

- ✓ Gini Index
- ✓ Entropy
- ✓ Misclassification error

C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, High degree of impurity	**Homogeneous, Low degree of impurity**

How to Find the Best Split



Measure of Impurity: GINI

Node t에서의 Gini Index:

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

- ✓ $p(j \mid t)$: node t에서의 클래스 j의 상대빈도
- ✓ 최대값: $(1 - 1/n_c)$, 데이터들이 모든 클래스에 균일하게 분포할 때
- ✓ 최소값: 0, 모든 데이터가 하나의 클래스에 속해있을 때
- ✓ Used in CART, SLIQ, SPRINT
- ✓ Node p 가 k 개로 분리될 때의 분리 이후의 gini index:

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)² – (5/6)² = 0.278

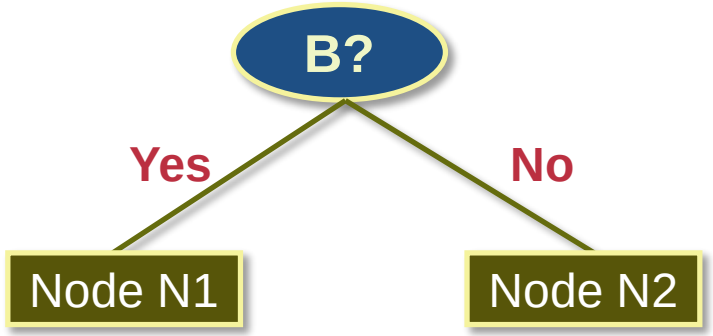
C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)² – (4/6)² = 0.444

Binary Attributes: Computing GINI Index

- 두 개로 분리하는 경우,
- 가중치로 인하여 더 크고 순도가 높은 분리를 탐색



	Parent
C1	6
C2	6
Gini = 0.500	

Gini(N1)
= 1 – $(5/6)^2$ – $(2/6)^2$
= 0.194

Gini(N2)
= 1 – $(1/6)^2$ – $(4/6)^2$
= 0.528

	N1	N2
C1	5	1
C2	2	4
Gini=0.333		

Gini(Children)
= 7/12 * 0.194 +
5/12 * 0.528
= 0.333

Binary Attributes: Computing GINI Index

- 서로 다른 값마다, 각 클래스에 속하는 수를 탐색하여 Count matrix를 구성
- 이를 이용하여 분리 기준을 탐색

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split (find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

- 일반적으로 하나의 값을 이용하여 2-way split을 수행
- 다양한 분리 기준 값이 존재

  분리 기준 값의 수 = 서로 다른 값의 수

- 각 분리 기준 값은 해당하는 count matrix가 존재
  - ✓ 각 분리마다 클래스에 해당하는 수의 데이터를 확인, $A < v$ and $A \geq v$
- 최적의 분리 기준값 v를 선택하는 단순한 방법
  - ✓ 각 v마다 count matri를 계산하고 Gini index를 계산
  - ✓ 많은 계산 시간을 소요

Binary Attributes: Computing GINI Index

현실적인 계산: 각 입력변수 마다,

- ✓ 값을 크기대로 정렬
- ✓ 순차적으로 scan하여 count matri와 gini index를 계산
- ✓ 가장 작은 gini index가 구해진 값으로 분리

	No	No	No	Yes	Yes	Yes	No	No	No	No
	Taxable Income									
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230
	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >	<= >
Yes	0 3	0 3	0 3	0 3	1 2	2 1	3 0	3 0	3 0	3 0	3 0
No	0 7	1 6	2 5	3 4	3 4	3 4	3 4	4 3	5 2	6 1	7 0
Gini	0.420	0.400	0.375	0.343	0.417	0.400	***0.300***	0.343	0.375	0.400	0.420

Measure based on Information: Entropy

✓ Node t에서의 entropy:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

- $p(j \mid t)$: node t에서의 클래스 j의 상대빈도
- 최대값: ($\log n_c$), 데이터들이 모든 클래스에 균일하게 분포할 때
- 최소값: 0, 모든 데이터가 하나의 클래스에 속해있을 때
- Entropy 기반의 계산은 GINI index 기반의 계산과 유사

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on Information

✓ Information Gain:

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

- 분리를 통한 엔트로피의 감소량을 추정. GAIN이 가장 높아지도록 (엔트로피의 감소량이 가장 많아지도록) 분리
- Used in ID3 and C4.5
- Disadvantage: 상대적으로 많은 분리를 하려는 경향이 있음. 작고 순도가 높은 부분집합을 추구

Splitting Based on Information

✓ Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

- 상대적 GAIN을 이용하여 작고 많은 분리를 제어 가능
- Used in C4.5

Measure based on Classification Error

✓Node t 에서의 classification error (분류 오류):

$$Error(t) = 1 - \max_i P(i \mid t)$$

✓ Node에서의 오분류율을 탐지

- 최대값: $(1 - 1/n_c)$, 데이터들이 모든 클래스에 균일하게 분포할 때
- 최소값: 0, 모든 데이터가 하나의 클래스에 속해있을 때

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

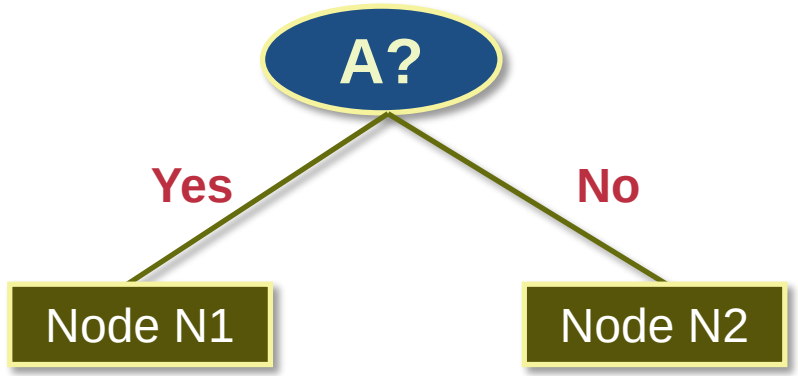
Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

C1	**1**
C2	**5**

P(C1) = 2/6 P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Misclassification Error vs Gini



	Parent
C1	7
C2	3
Gini = 0.42	

Gini(N1)
= 1 – $(3/3)^2$ – $(0/3)^2$
= 0

Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves !!

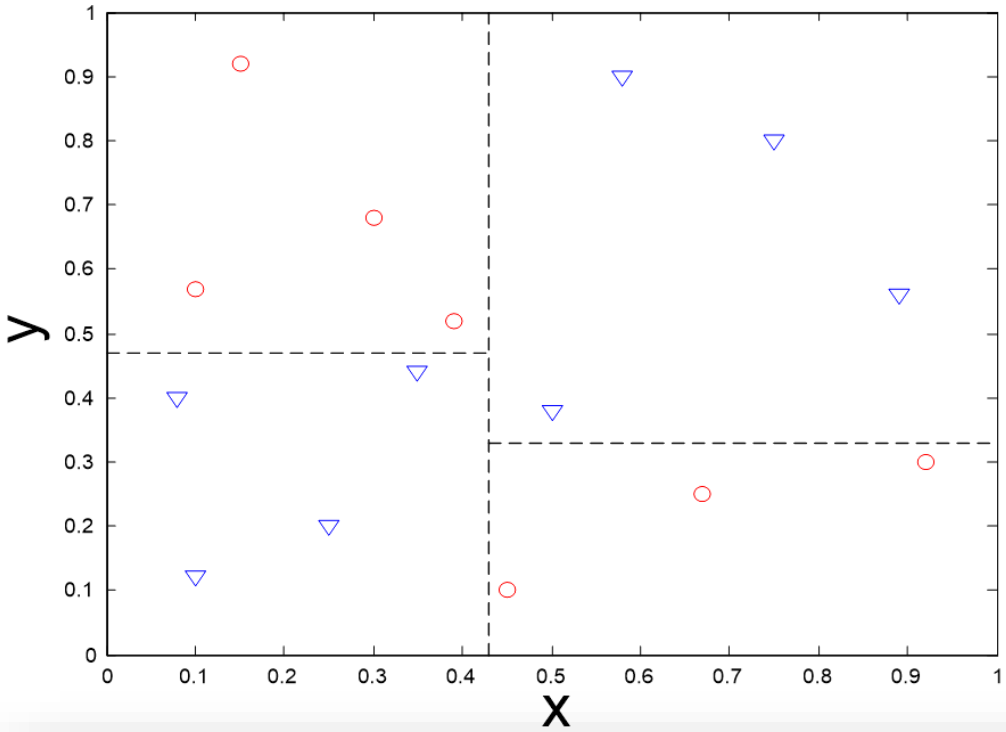
Tree Induction의 중단

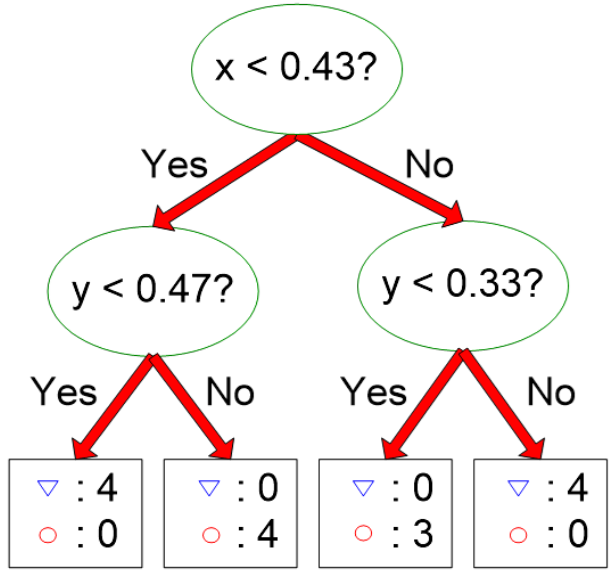
- ✓ 한 노드 안의 데이터가 모두 같은 클래스에 속할 경우 분리를 중단
- ✓ 한 노드 안의 데이터의 입력 값이 모두 비슷한 경우 분리를 중단
- ✓ 조기 종료: overfitting을 방지

Decision Tree 요약

- ✓ 학습이 쉽고 빠름
- ✓ 새로운 데이터에 대한 분류가 빠름
- ✓ 작은 크기의 의사결정나무에 대하여 설명 및 이해가 쉬움
- ✓ 많은 데이터셋에서 우수한 성능을 보임

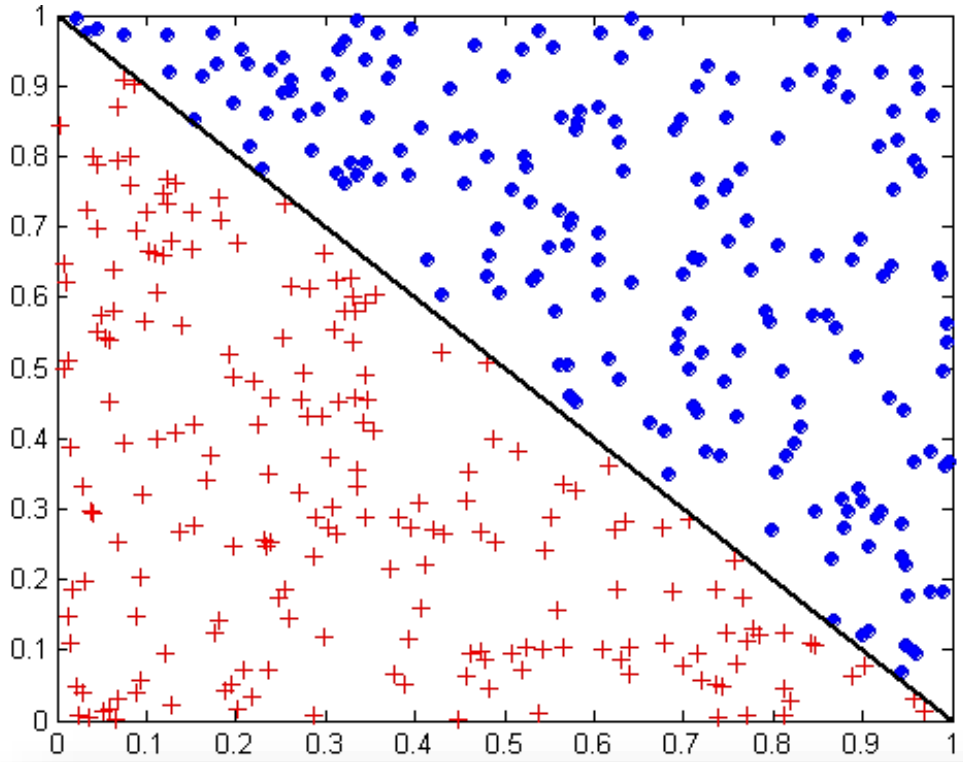
Decision Boundary in Decision Trees

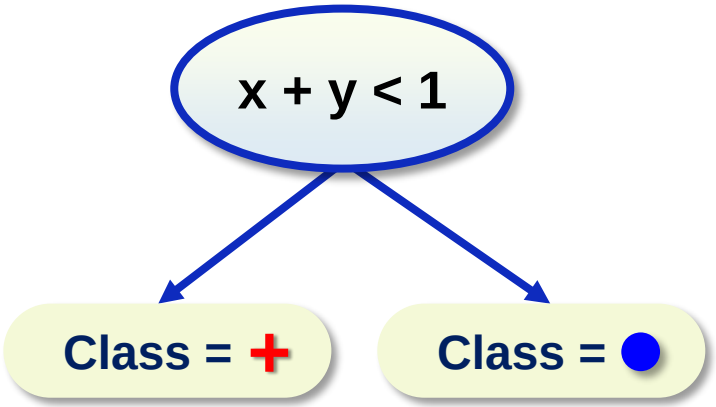




- ✓ Decision boundary: 서로 다른 클래스를 나누는 분류 기준
- ✓ 의사결정 나무에서는 한 번에 하나의 조건으로 데이터를 분리하기 때문에 각 축에 평행한 decision boundary를 생성

Oblique Decision Trees





- ✓ 여러 변수를 동시에 고려하여 분리 조건을 생성
- ✓ 더욱 다양한 표현 가능
- ✓ 최적의 조건을 찾는 것은 매우 높은 계산량을 요구

Decision Tree의 Overfitting

- ✓ Decision tree가 크고 복잡해질수록 overfitting의 발생 경향이 증가: decision tree에서의 가장 고질적인 문제
- ✓ Training error를 통해서는 구성된 tree가 새로운 데이터에 대하여 잘 작동할지 확신할 수 없음
- ✓ Error를 측정하는 새로운 방법이 필요

Estimating Generalization Errors

- ✓ Re-substitution errors: error on training (Σ e(t))
- ✓ Generalization errors: error on testing (Σ e'(t))
- ✓ Generalization error 추정 방법들:
  - Optimistic approach: e'(t) = e(t)
  - Pessimistic approach:
    - ✓ 각 leaf node: e'(t) = (e(t)+0.5)
    - ✓ 총 errors: e'(T) = e(T) + N * 0.5 (N: number of leaf nodes)
    - ✓ 30개의 leaf nodes and 10 개의 training error가 발생하는 tree의 경우:
      Training error = 10/1000 = 1%
      Generalization error = (10 + 30*0.5)/1000 = 2.5%
  - Reduced error pruning (REP):
    - ✓ Uses validation data set to estimate generalization error

Occam's Razor

- ✓ 두 모델이 유사한 generalization error를 가질 경우, 단순한 모델이 복잡한 모델보다 더 선호된다
- ✓ 복잡한 모델의 경우 우연히 현재 샘플된 데이터의 에러에 의해 적합 되었을 확률이 높음
- ✓ 따라서 모델을 평가할 때에는 반드시 모델의 복잡도 또한 함께 고려해야 한다

How to Address Overfitting

✓ Pre-Pruning (Early Stopping Rule)

- 트리가 다 구성되기 전에 알고리즘을 중단
- 일반적인 중단 조건:
  - ✓ 한 노드에 속한 데이터가 모두 같은 클래스일 때
  - ✓ 한 노드에 속한 데이터의 입력 값이 모두 같을 때
- 추가적인 제한 조건:
  - ✓ 한 노드에 속한 데이터가 (사전에 정의된) 일정 수 이하일 때
  - ✓ 한 노드에 속한 데이터의 클래스 분포가 모든 입력 변수에 독립적일 때 (e.g., using χ^2 2 test)
  - ✓ 어떤 분리도 불순도 수치를 개선하지 못할 때 (e.g., Gini or information gain).

How to Address Overfitting

✓ Post-Pruning

- ✓ 의사결정나무를 최대 수치까지 온전히 학습
- ✓ 완성된 의사결정나무의 노드를 제거 (가지치기)
- ✓ 만일 가지치기에 의하여 generalization error가 개선된다면, 가지치기된 상태를 유지
  - Sub-tree가 leaf node로 대체됨
- ✓ 새로운 leaf node의 class label은 포함한 데이터 중 가장 많은 수의 클래스로 결정

Example of Post-Pruning

Class = Yes	20
Class = No	10
Error = 10/30	

Training Error (Before splitting) = 10/30

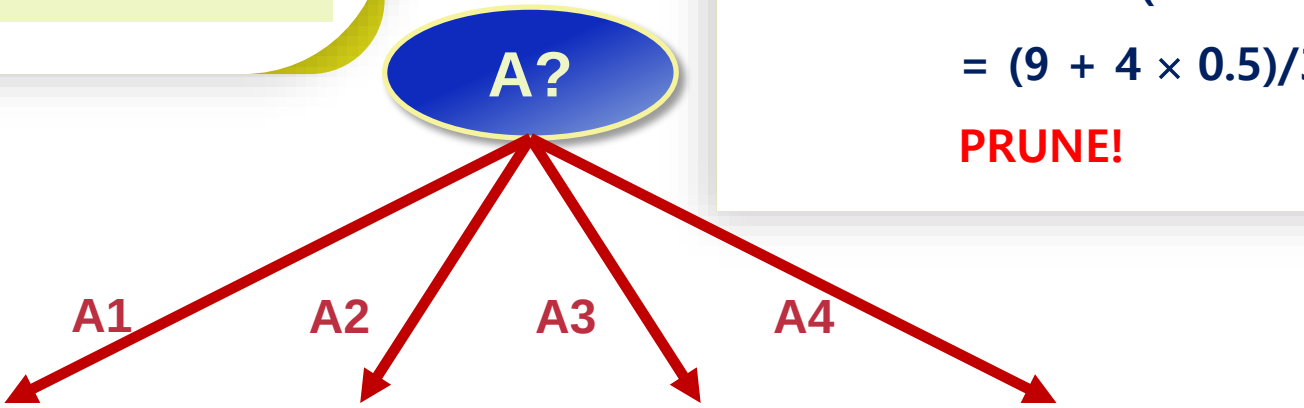
Pessimistic error = (10 + 0.5)/30 = 10.5/30

Training Error (After splitting) = 9/30

Pessimistic error (After splitting)

= (9 + 4 × 0.5)/30 = 11/30

PRUNE!

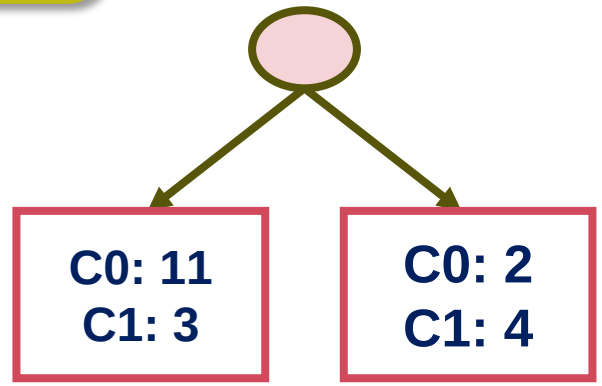


Class = Yes	8	Class = Yes	3	Class = Yes	4	Class = Yes	5
Class = No	4	Class = No	4	Class = No	1	Class = No	1

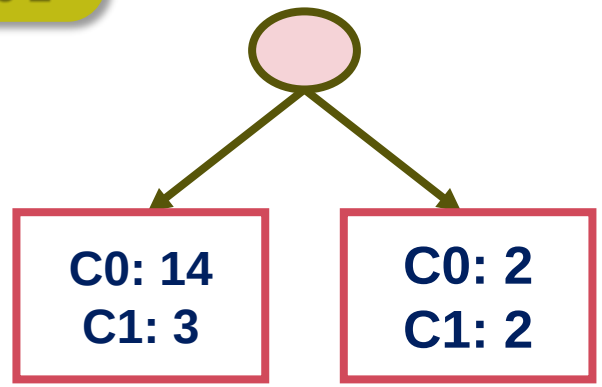
Example of Post-Pruning

- ✓ **Optimistic error?**
  - Don’t prune for both cases
- ✓ **Pessimistic error?**
  - Don’t prune case 1, prune case 2
- ✓ **Reduced error pruning?**
  - Depends on validation set

Case 1



Case 2



실습 – Iris Data load

```
iris = load_iris()
iris_frame=pd.DataFrame(data=np.c_[iris['data'],iris['target']],columns=iris['feature_names'] + ['target'])
iris_frame['target'] = iris_frame['target'].map({0:"setosa",1:"versicolor",2:"virginica"})
X=iris_frame.iloc[:,:-1]
Y=iris_frame.iloc[:,[-1]]
iris_frame
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

실습 – Decision Tree 학습

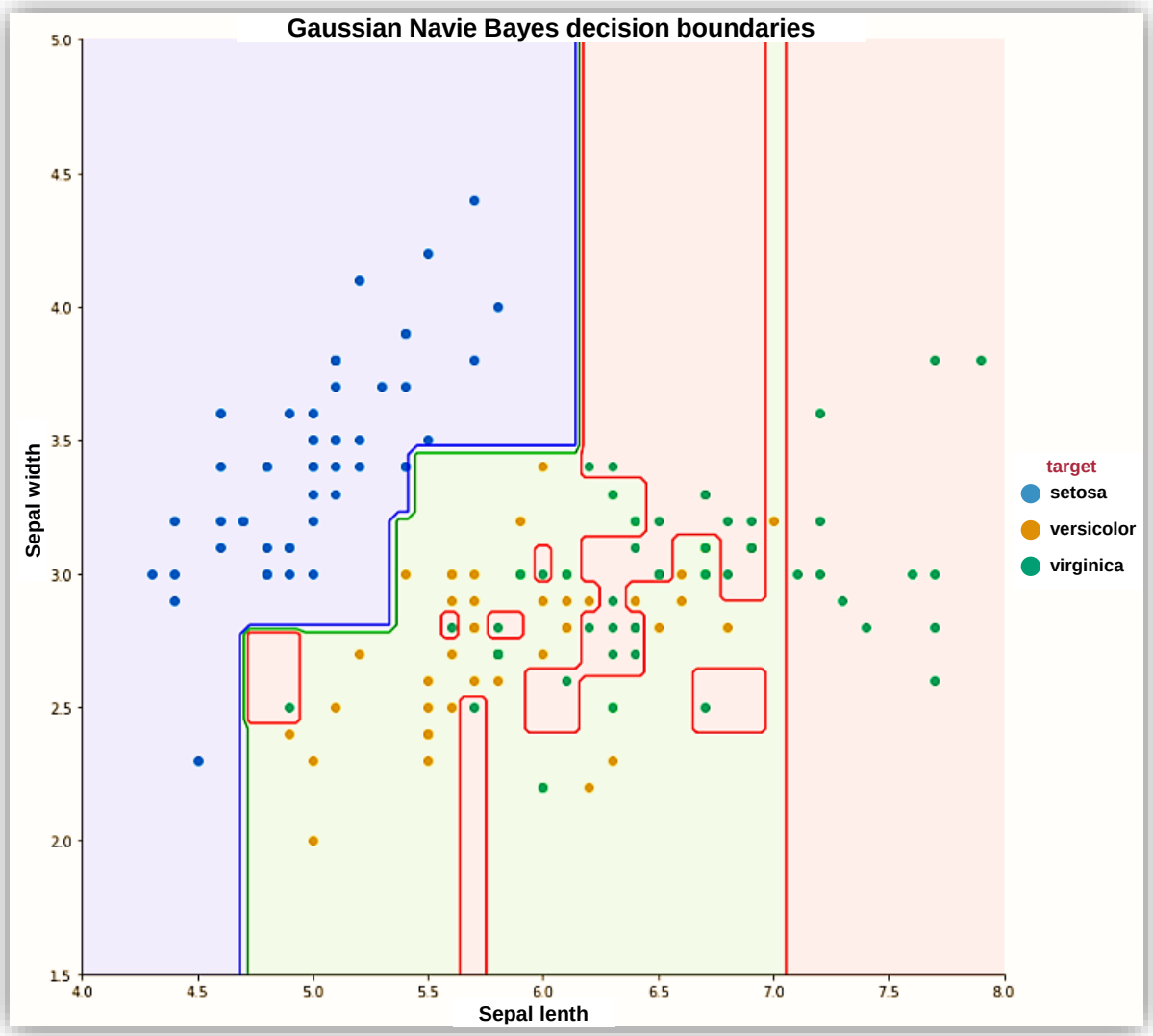
✓ Feature 중 sepal에 관련된 두 개의 feature만 이용해서 학습 (시각화가 비교적 편리함)

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0,criterion='gini',max_depth=5)
import matplotlib.colors as colors
df1 = iris_frame[["sepal length (cm)", "sepal width (cm)",'target']]
X = df1.iloc[:,0:2]
Y = df1.iloc[:,2].replace({'setosa':0,'versicolor':1,'virginica':2}).copy()
clf.fit(X,Y)
N = 100
```

실습 – Decision Tree 학습 시각화

```
X_ = np.linspace(4, 8, N)
Y_ = np.linspace(1.5, 5, N)
X_, Y_ = np.meshgrid(X_, Y_)

color_list = ['Blues','Greens','Reds']
my_norm = colors.Normalize(vmin=-1.,vmax=1.)
g = sn.FacetGrid(iris_frame, hue="target", size=10, palette = 'colorblind') .map(plt.scatter, "sepal length (cm)",
                                                                                  "sepal width (cm)",).add_legend()
my_ax = g.ax
zz = np.array( [clf.predict( [[xx,yy]])[0] for xx, yy in zip(np.ravel(X_), np.ravel(Y_)) ] )
Z = zz.reshape(X_.shape)



#Plot the filled and boundary contours
my_ax.contourf( X_, Y_, Z, 2, alpha = .1, colors = ('blue','green','red'))
my_ax.contour( X_, Y_, Z, 2, alpha = 1, colors = ('blue','green','red'))

# Add axis and title
my_ax.set_xlabel('Sepal length')
my_ax.set_ylabel('Sepal width')
my_ax.set_title('DecisionTree decision boundaries')

plt.show()
```

실습 – Decision Tree 학습 시각화



실습 – Decision Tree 학습 매개변수 조절

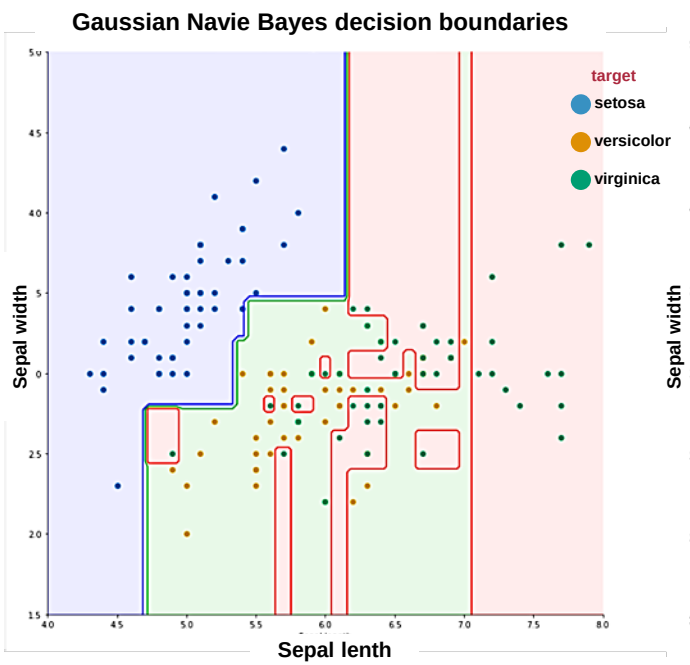
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0,criterion='gini',max_depth=10)
```

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0,criterion='gini',max_depth=5)
```

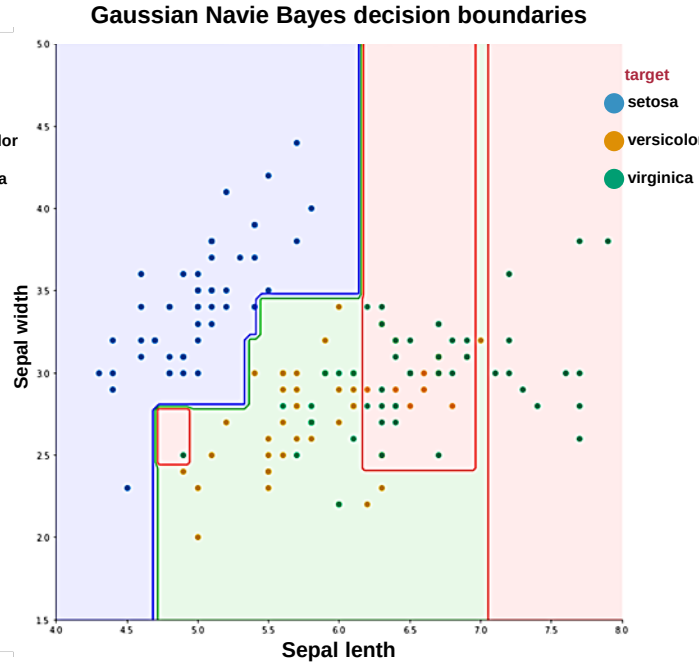
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0,criterion='gini',max_depth=2)
```

최대 깊이 조절

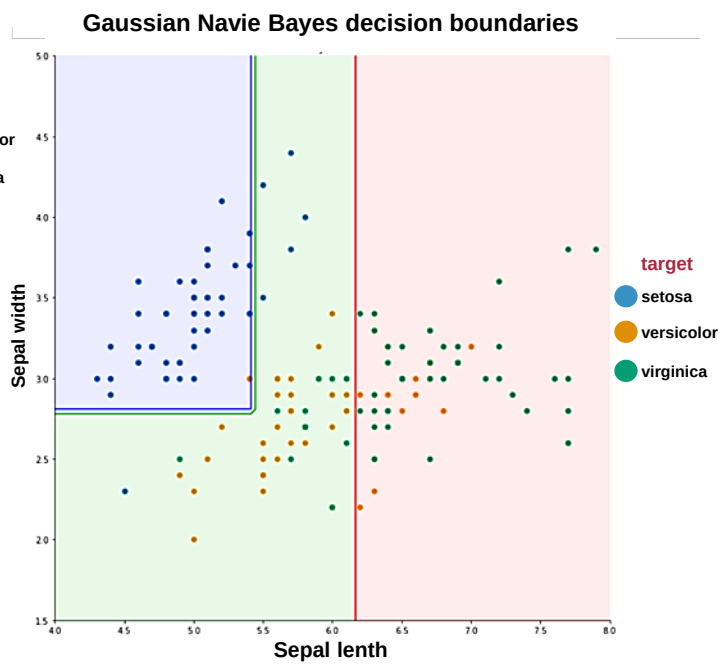
실습 – Decision Tree 학습 매개변수 조절 결과 확인



최대 깊이(max_depth)= 10



최대 깊이(max_depth)= 5



최대 깊이(max_depth)= 2

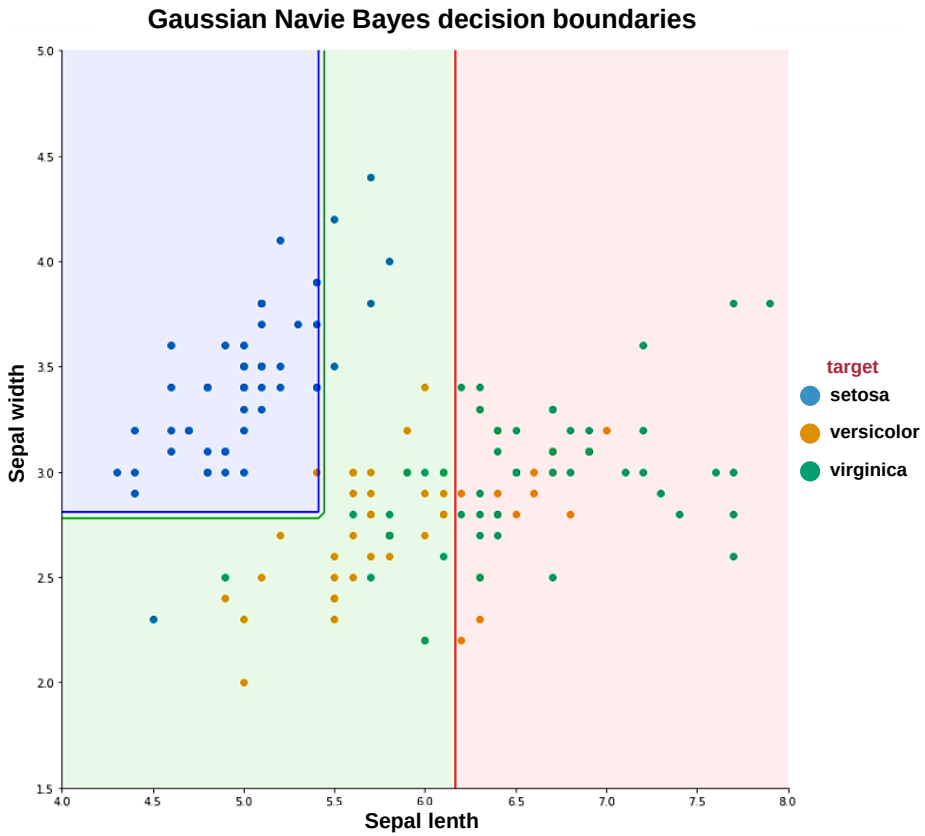
실습 – Decision Tree 학습 매개변수 조절

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0,criterion='gini' max_depth=2)
```

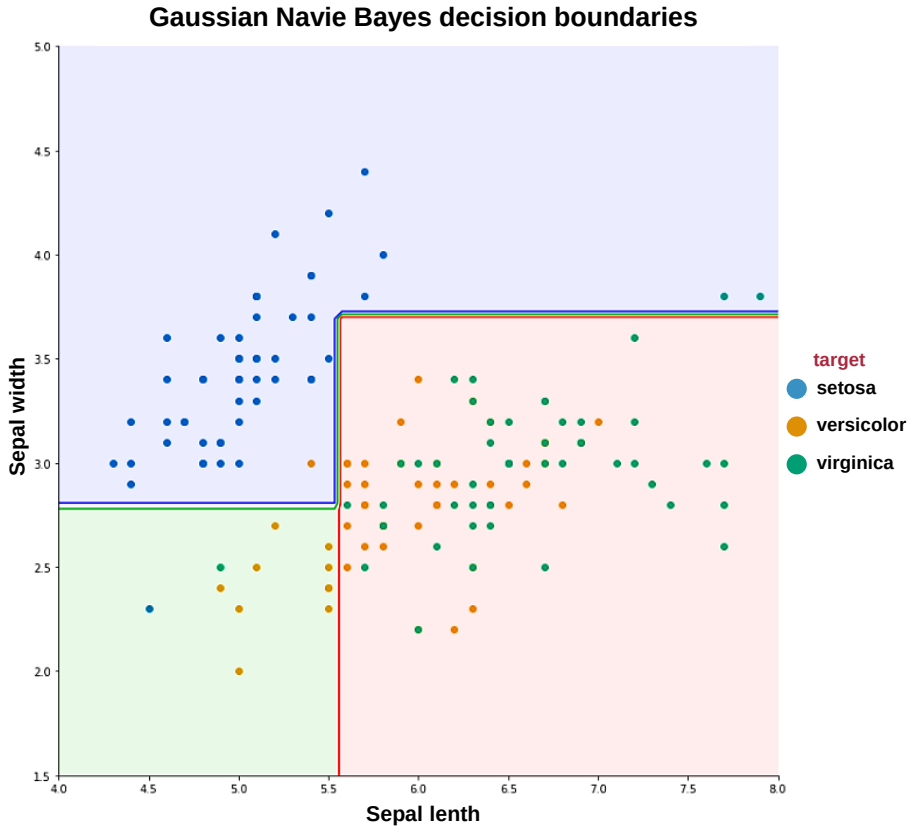
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0,criterion='entropy' max_depth=2)
```

Gini index -> entropy

실습 – Decision Tree 학습 매개변수 조절



Gini index



Entropy

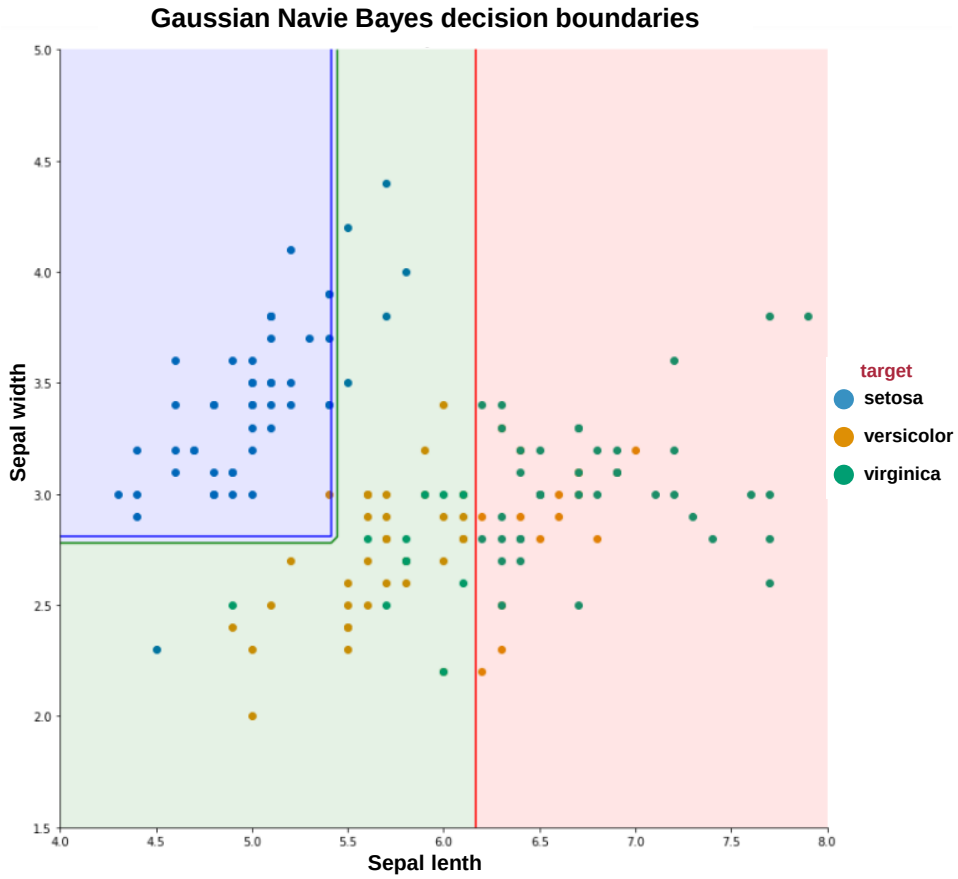
실습 – Decision Tree 학습 매개변수 조절

```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0,criterion='gini' max_depth=2)
```

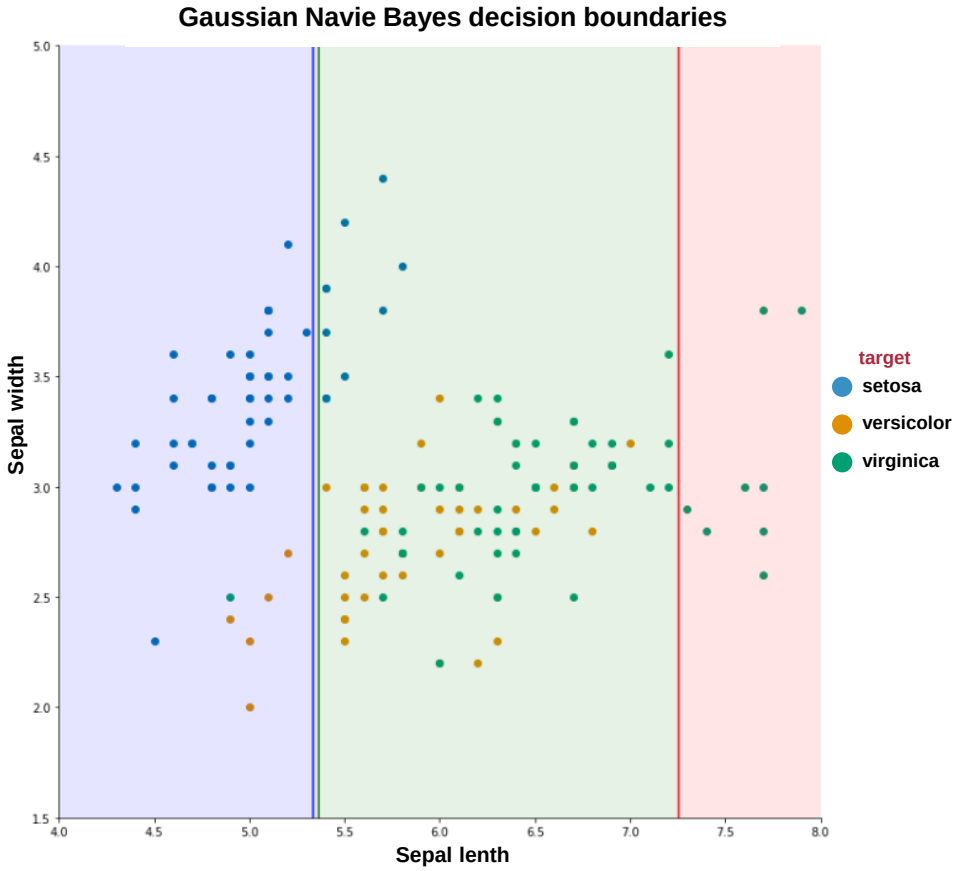
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0,criterion='gini',max_depth=2 splitter='random')
```

기본적으로 splitter= ‘best’가 defaul이고 splitter를 random splitter로 변경 가능

실습 – Decision Tree 학습 매개변수 조절



Best



Random