Chap08.
Classification

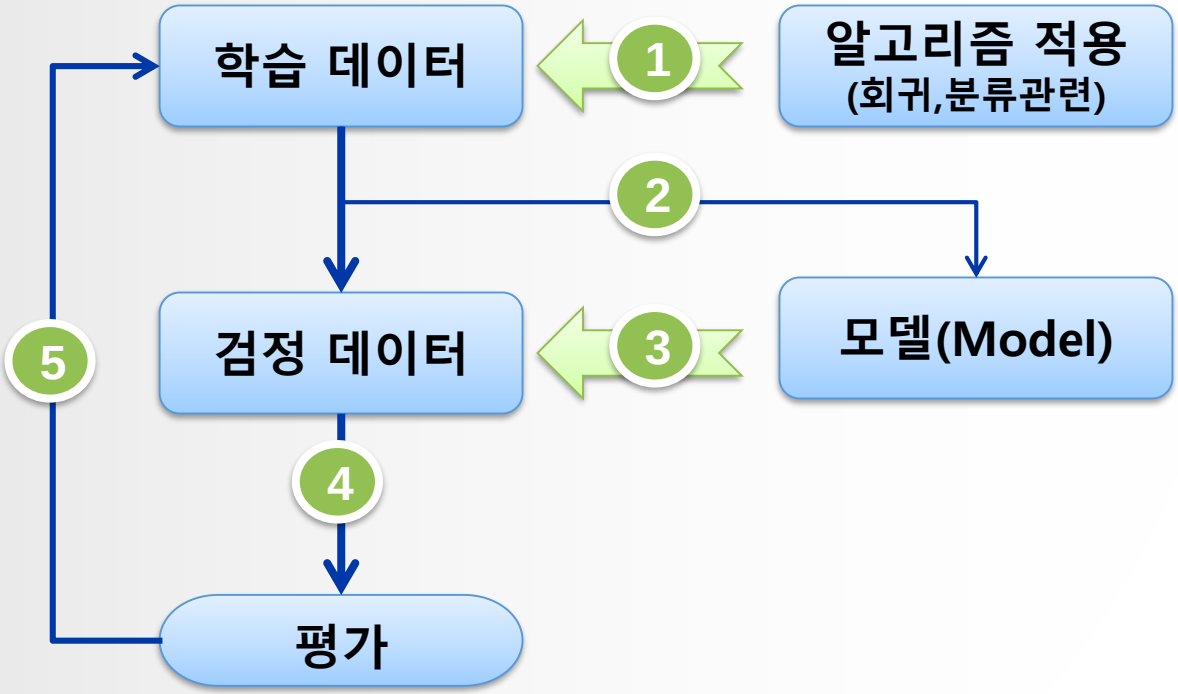
작성자 : 김진성

8. 분류예측 모형

1. k-Nearest Neighbors
2. Naive Bayes
3. SVM(Support Vector Machine)
4. Decision Tree

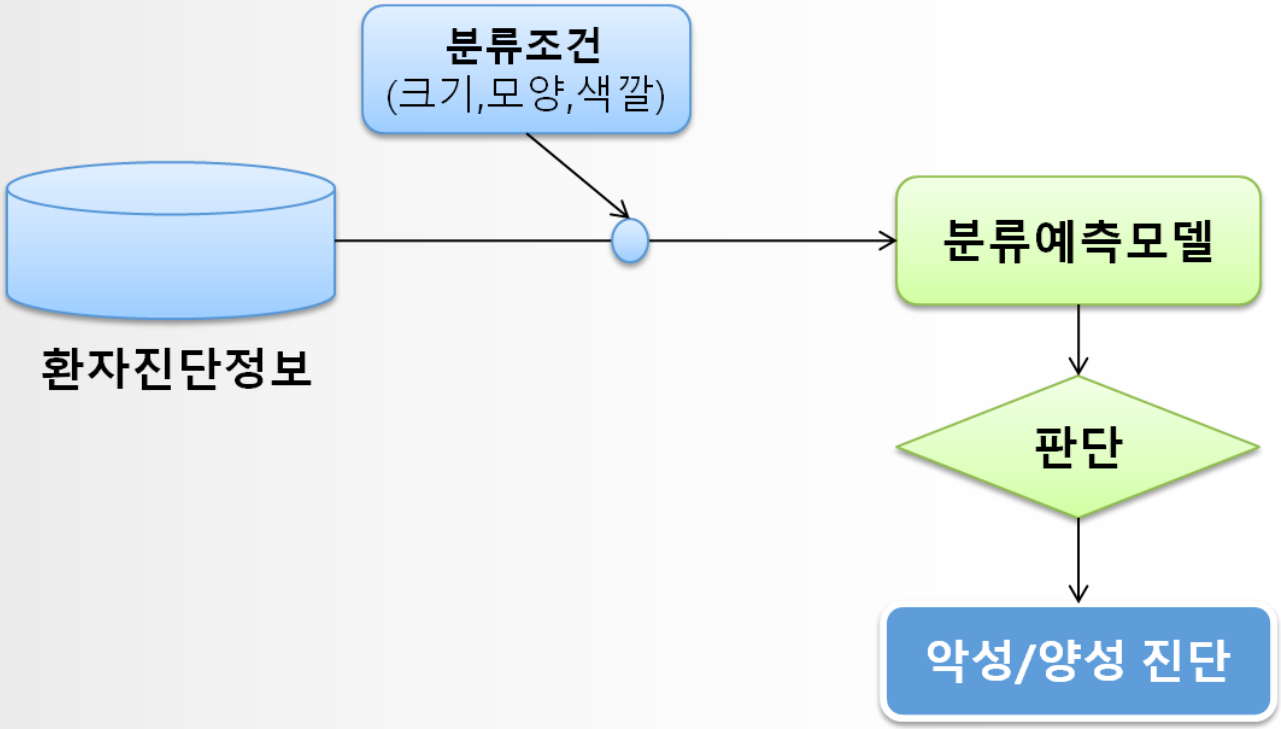
지도학습 모델 생성 과정

- 지도학습(Supervised Learning) 절차



분류예측 모형 적용 예

❖ 의.생명분야에서 분류분석 사례



1. kNN 알고리즘

1. 알려진 범주로 알려지지 않은 범주 분류(해석 용이)
2. 기존에 범주가 존재해야 함 - 식료품(과일, 채소, 단백질 등)
3. 학습하지 않음 : 게으른 학습
4. 결측치(NA)/이상치 전처리 중요
5. 많은 특징을 갖는 데이터 셋은 부적합
6. 유클리드 거리(Euclidean distance) 계산식 이용
   - ✓ 가장 유사한 범주를 가장 가까운 거리로 선택
7. 적용 분야
   - ✓ 개인별 영화 추천
   - ✓ 이미지/비디오에서 얼굴과 글자 인식
   - ✓ **유전자 데이터 패턴 식별(종양 식별)**

$$\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

kNN 알고리즘 예

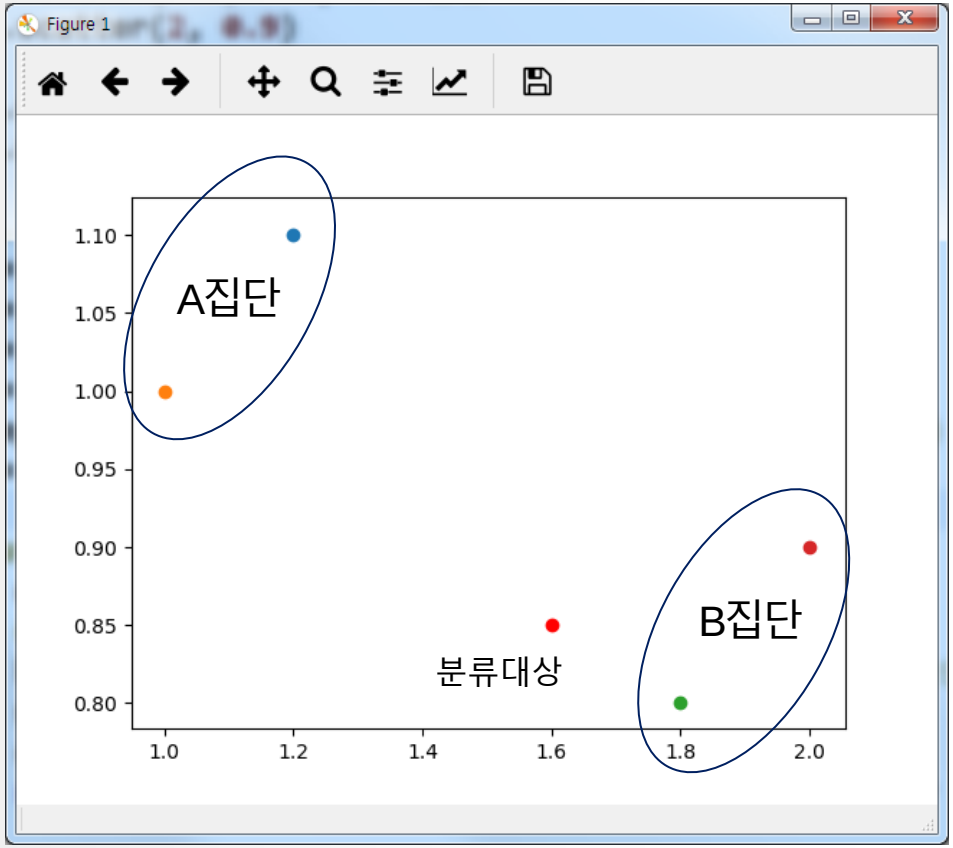


Figure 1
1.10
1.05
1.00
0.95
0.90
0.85
0.80
1.0
1.2
1.4
1.6
1.8
2.0
A집단
B집단
분류대상

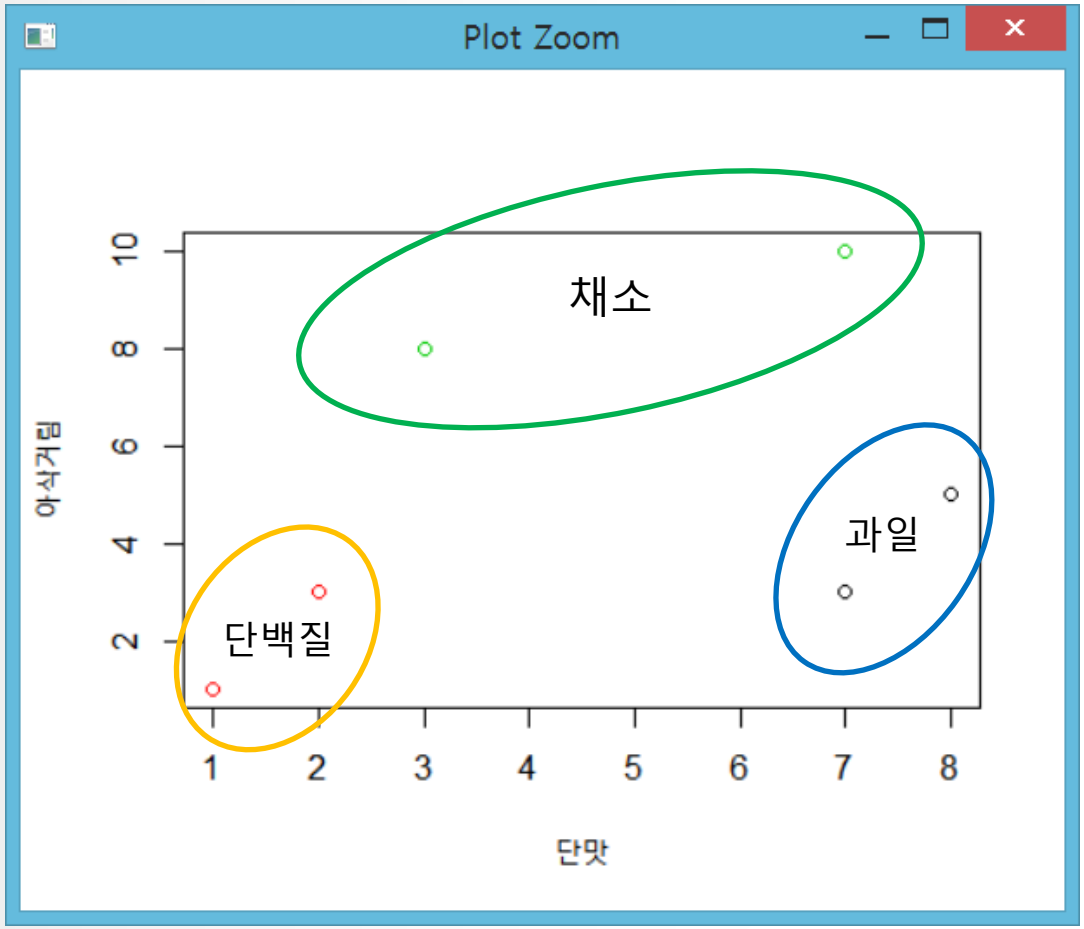
kNN 알고리즘 예

- 식료품 분류 예

식품명	단맛	아삭거림	분류
포도(grape)	8	5	과일
물고기(fish)	2	3	단백질
당근(carrot)	7	10	채소
오랜지(orange)	7	3	과일
셀러리(celery)	3	8	채소
치즈(cheese)	1	1	단백질

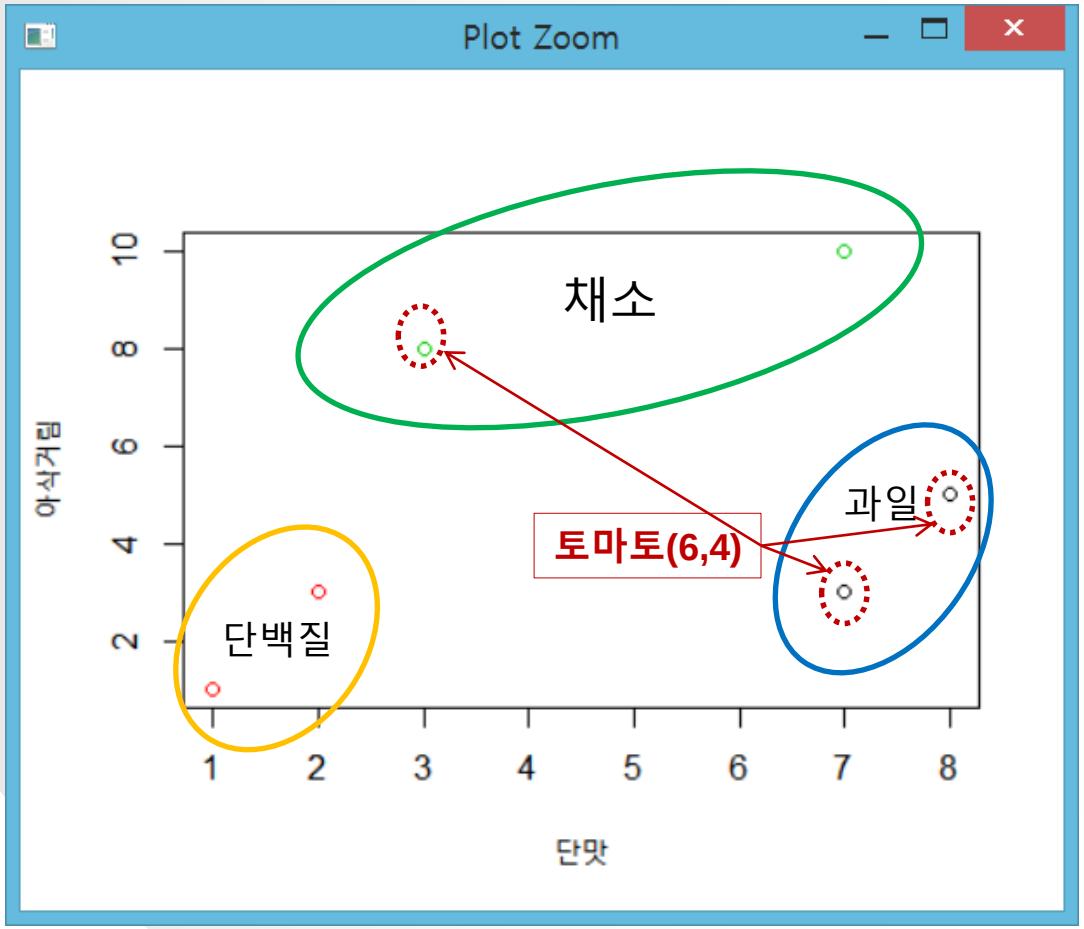
kNN 알고리즘 예

- plot() 함수 적용 결과



kNN 알고리즘 예

- 토마토는 어느 분류에 속하는가?



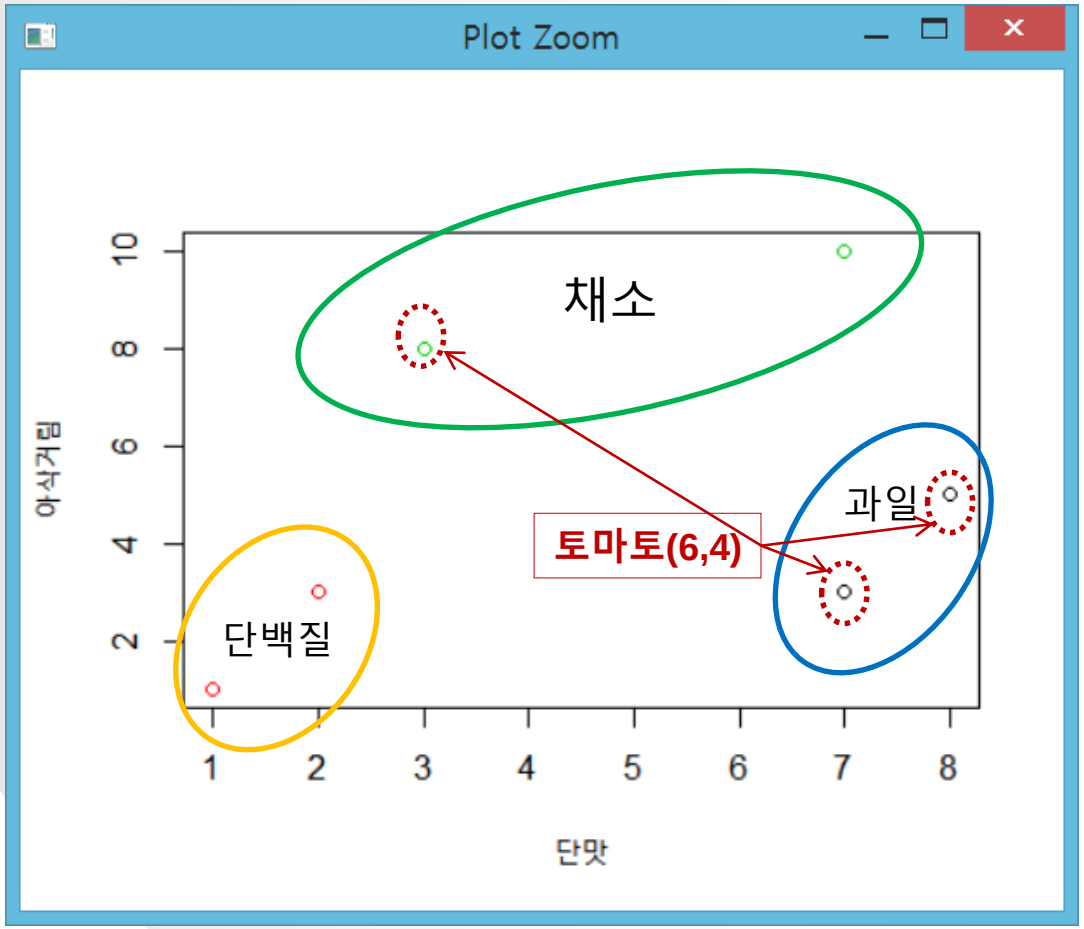
유클리드 거리

1NN : 오랜지

3NN : 오랜지,포도,셀러리

kNN 알고리즘 예

- 토마토는 어느 분류에 속하는가?



유클리드 거리

1NN : 오랜지

3NN : 오랜지,포도,셀러리

2. Naive Bayes 알고리즘

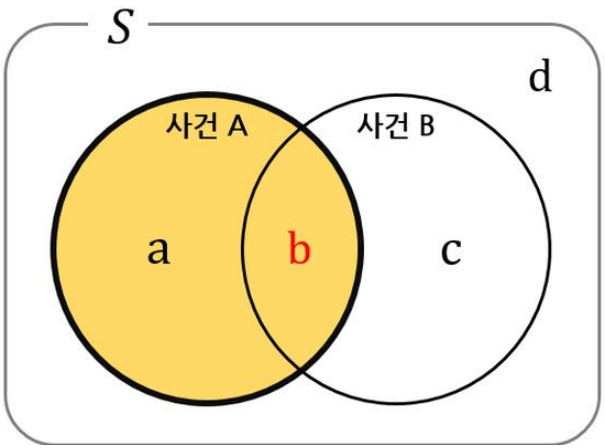
1. 통계적 분류기
   - ✓ 주어진 데이터가 특정 클래스에 속하는지를 확률을 통해서 예측
   - ✓ 조건부 확률 이용 : P(B|A)
2. 베이즈 확률 정리(Bayes' theorem)을 적용한 기계학습 방법
   - ✓ 두 확률 변수(사전 확률과 사후 확률) 사이의 관계를 나타내는 이론
   - ✓ 사전확률 : 사건이 발생하기 전에 알려진 확률
   - ✓ 사후확률 : 베이즈 이론에 근거한 확률
3. 특정 영역에서는 DT나 kNN 분류기 보다 성능이 우수
4. 텍스트 데이터 처럼 희소한 고차원인 경우 높은 정확도와 속도 제공
5. 적용분야
   - ✓ **Spam 메일 분류, 문서(주제) 분류, 비 유무**
   - ✓ 컴퓨터 네트워크에서 침입자 분류(악성코드 유무)

조건부 확률(Conditional Probability)

- 사건 A가 발생했다는 전제 하에서 다른 사건 B가 발생할 확률

$$P(B|A) = \frac{P(A \cap B)}{P(A)} \quad (\text{단 } P(A) > 0)$$

- 벤다이어그램 표현



$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\frac{b}{N}}{\frac{a+b}{N}}$$

N = a+b+c+d 일 때

Bayes' Theorem

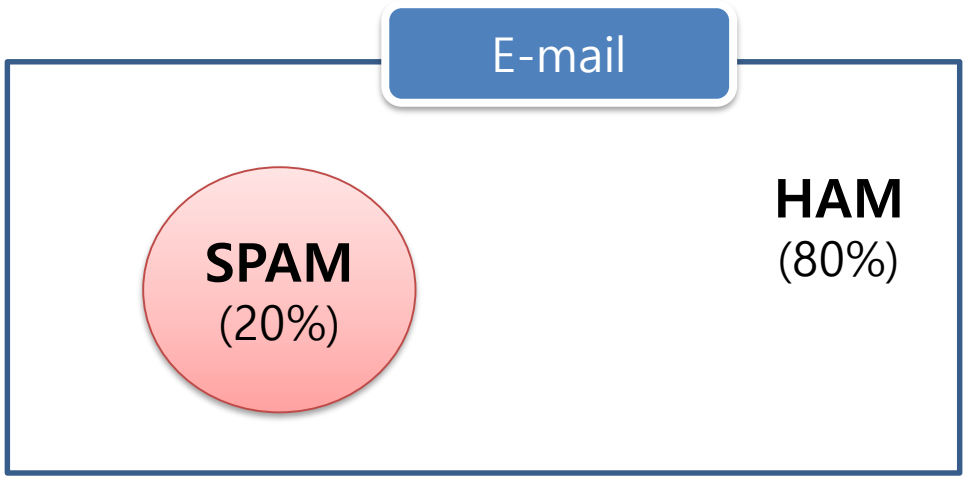
- 베이즈 정리 : 과거의 경험(사건 B)과 현재의 증거(사건 A)를 토대로 어떤 사건의 확률을 예측(추론)하는 이론
- P(A) : 사건 A 사전확률 : 현재의 증거
- P(B) : 사건 B 사전확률 : 과거의 경험
- P(B|A) : 사건 A의 증거에 대한 사후확률 : 사건 A가 일어났다는 것을 알고, 그것이 사건 B로부터 일어난 것이라고 생각되는 조건부 확률

  예) 비아그라 단어(A사건)가 포함될 때 스팸(B사건) 메시지일 확률

  - 사전확률 : 확률 실험 이전에 사건 발생에 대해 이미 알고 있는 사전 지식
  - 사후확률 : 어떤 사건을 인지한 후 이들이 어떤 원인에 의해 출현한 것이라고 생각되는 조건부 확률

SPAM 메시지 분류 예

- 상호배타적, 포괄적 사건
  - ✓ Email 메시지에서 이미 알려진 Spam과 Ham 발생 비율 예



<독립사건인 경우 사전확률>

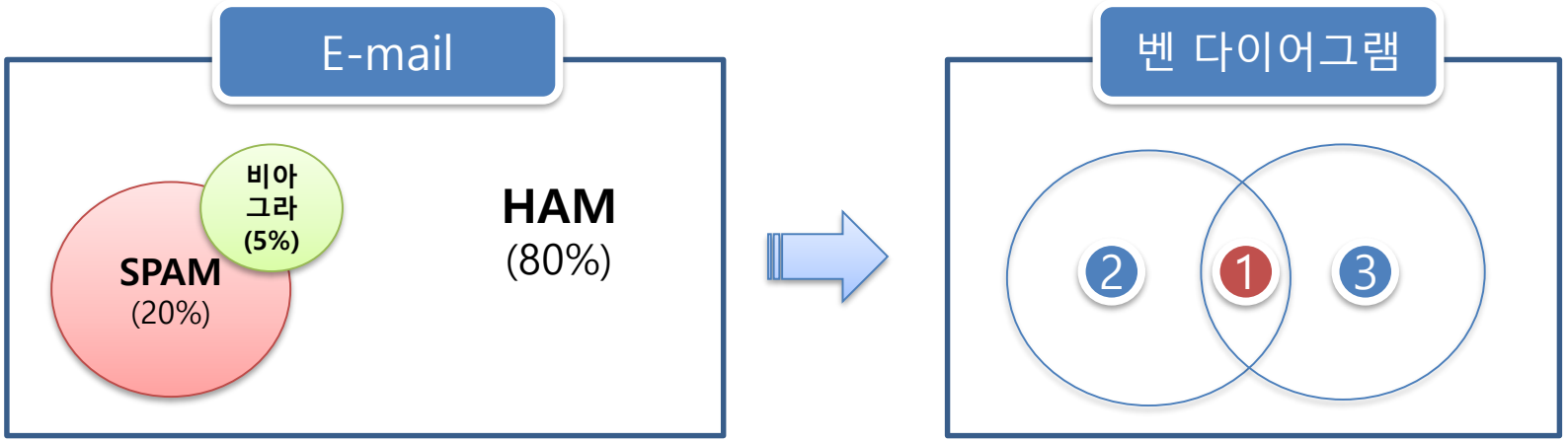
✓ SPAM과 HAM의 확률

- P(spam) = 0.2(20%)
- P(~spam) = 1-0.2=0.8(80%)

1. 상호배타적 : 동시에 두 사건이 일어나지 않음(독립사건)
   - ✓ 예) SPAM이 발생하면 HAM 발생하지 않음
2. 포괄적 : 두 가지 결과만 발생
   - ✓ 예) SPAM 또는 HAM 사건만 발생

● 비 상호배타적 사건

✓ Email 메시지에서 비아그라 속성 추가 예



1. 비상호배타적 : 동시에 두 사건이 일어남
2. 결합확률 : 두 사건이 동시에 일어날 확률

예) SPAM 메일에 비아그라 단어가 포함될 확률

예) HAM 메일에 비아그라 단어가 포함될 확률

● 벤 다이어그램 : 원소 집합의 중첩 표현

① : SPAM/HAM에 비아그라 단어 포함(5%)

② + ① : SPAM 메일(비아그라 단어 출현)

③ + ① : HAM 메일(비아그라 단어 출현)

- **사전확률(주변확률) : 조건 없이 사건A가 발생할 확률**
  - ✓ SPAM 확률 : 20/100 = 0.2
  - ✓ HAM 확률 : 80/100 = 0.8
  - ✓ VIAGRA 확률 : 5/100 = 0.05

- **결합확률 : 두 사건 A와 B가 동시 일어날 확률**
  - ✓ Viagra 단어가 포함된 Spam 메일 확률 : 4/20 = 0.2
  - ✓ Viagra 단어가 포함된 Ham 메일 확률 : 1/80 = 0.0125

구분	VIAGRA		합계
	Yes	No	
SPAM	4	16	20/100
HAM	1	79	80/100
합계	5/100	95/100	100

사전

주변

[사전확률 표]

구분	VIAGRA		합계
	Yes	No	
SPAM	4/20	16/20	20
HAM	1/80	79/80	80
합계	5/100	95/100	100

[결합확률 표]

- 조건부 확률 : 사건 A가 일어나는 조건하에서 사건 B가 일어날 확률
  - ✓ P(B|A) = P(A|B) * P(B) / P(A) = P(A∩B) / P(A)
- **베이즈 이론 적용** : 비아그라(A) 단어가 출현할 때 스팸(B)일 확률 : 80%
  - ✓ P(스팸|비아그라) = P(비아그라|스팸) * P(스팸) / P(비아그라)
  - ✓ 사후확률 = 결합확률 * 사전확률(사건B) / 사전확률(사건A)
  - ✓ P(스팸|비아그라) = (4/20) * (20/100) / (5/100) = 0.8

결합확률: 동시에 일어날 확률

사전확률: 현재의 증거

구분	VIAGRA		합계	주변 확률
	Yes	No		
SPAM	4/20	16/20	20	20/100
HAM	1/80	79/80	80	80/100
합계	5/100	95/100	100	

사전확률 : 과거의 경험

[결합확률 표]

∴ 비아그라 단어가 포함된 Message가 스팸 일 확률은 80%

3. SVM(Support Vector Machine)

- 다양한 데이터 셋에서 잘 동작하는 강력한 모델
  - ✓ 저차원, 고차원 데이터 모두 잘 동작
- 데이터의 특징이 적어도 복잡한 결정 경계 생성
- 모든 특징과 스케일이 비슷한 경우 유리함
  - ✓ 데이터 전처리와 매개변수 설정에 주의
- 샘플(관측치)이 많은 경우 불리함(100,000개 이상)
- 모델 분석이 어려움(블랙 box), 예측과정 이해가 어려움

SVM 알고리즘 특징

- 2000년대 초반에 많이 사용되는 분류 알고리즘
- SVM 알고리즘 - 이진분류 : 두 범주를 직선으로 분류
- 선형분리 - 2개의 집합(초평면:Hyperplane)을 직선으로 분리
  - 초평면 : 2차원 이상의 고차원 공간을 의미
  - 직사각형의 넓이(Margin) : Margin의 최대값을 구하는 것이 관건
  - Support Vectors : Margin과 가장 가까운 점들
- kNN과 선형회귀 모델링 기법이 적용 : 분류와 수치 예측 가능
- 비선형 분류를 위해서 데이터를 고차원 공간 사상(커널 트릭)

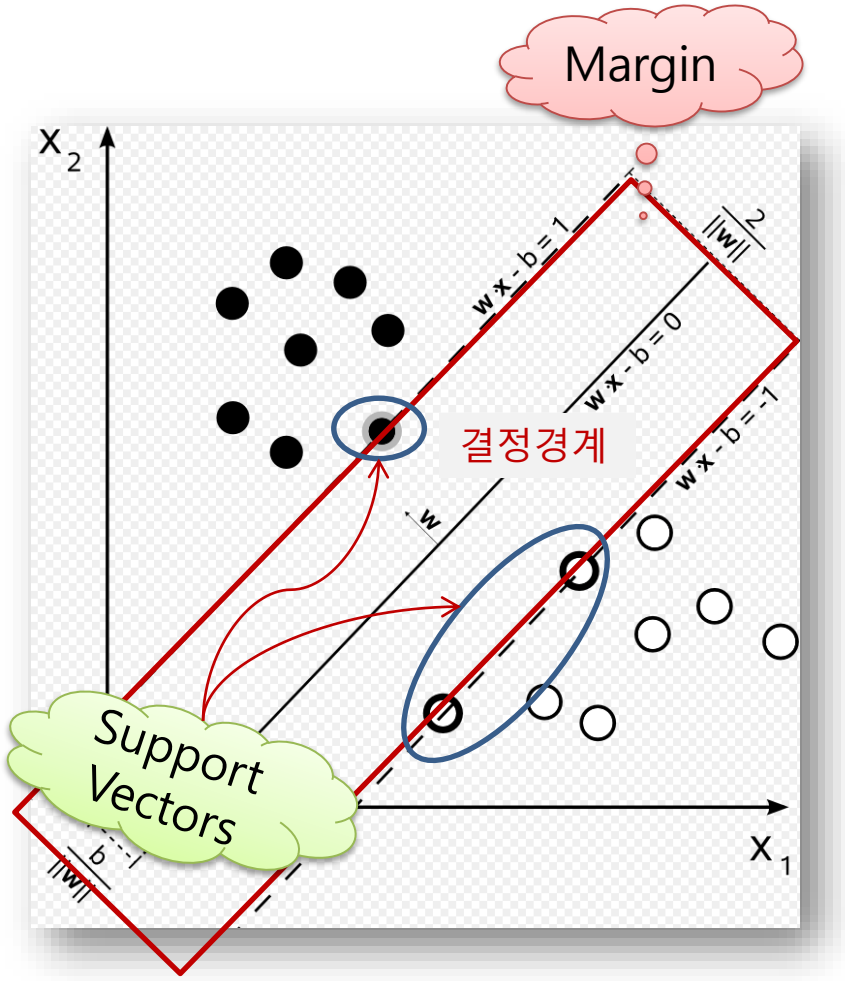
SVM 알고리즘 특징

- 적용 분야
  - ✓ 바이오인포매틱스의 마이크로 유전자 데이터 분류
  - ✓ 인간의 얼굴, 문자, 숫자 인식
    - 이미지 데이터 패턴 인식에 적합

    예) 스캐너로 스캔 된 문서 이미지를 문자로 인식

● 결정경계와 마진(Margin)

SVM 알고리즘 : 가상 결정경계 중심으로 최대한 거리 계산하여 최대의 직사각형 형태(Margin)로 영역 넓힘



Margin: 점에 닿기 전까지의 선형 분류기의 폭

Support Vectors: Margin과 닿는 점(pointer)

W : **초평면**(2차원 이상의 고차원 공간)

w.x – b = 0 : 두 집합을 분류하는 결정경계(가상 직선)

w.x – b = 1 : ● 레이블을 지나는 초평면
● 레이블 집합에서 가장 가까운 포인터

w.x - b = -1 : ○ 레이블을 지나는 초평면
○ 레이블 집합에서 가장 가까운 포인터

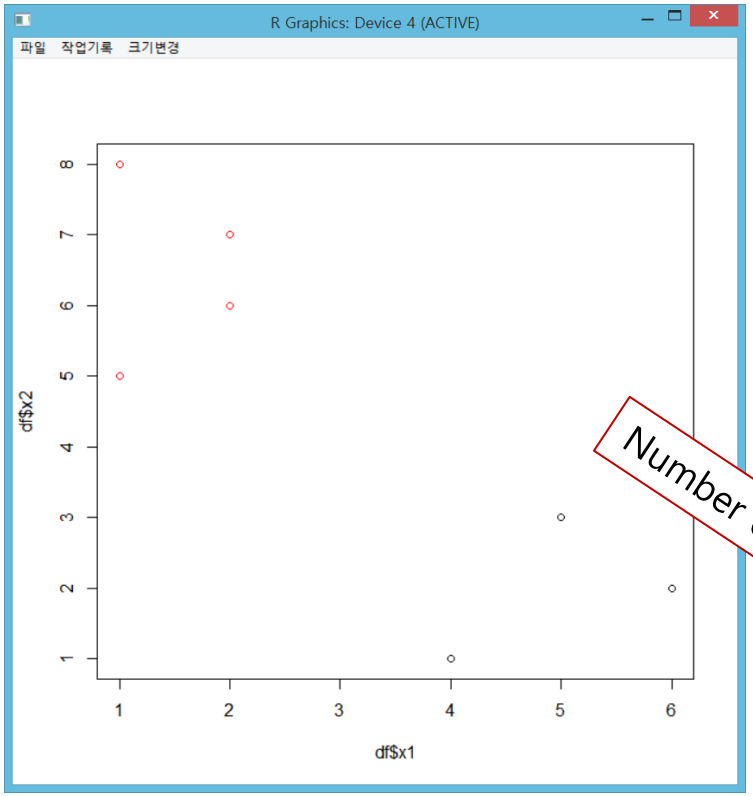
초평면 **마진**($\frac{2}{\|w\|}$) : 두 초평면 사이의 거리
(각 서포트 벡터를 지나는 초평면 사이의 거리)

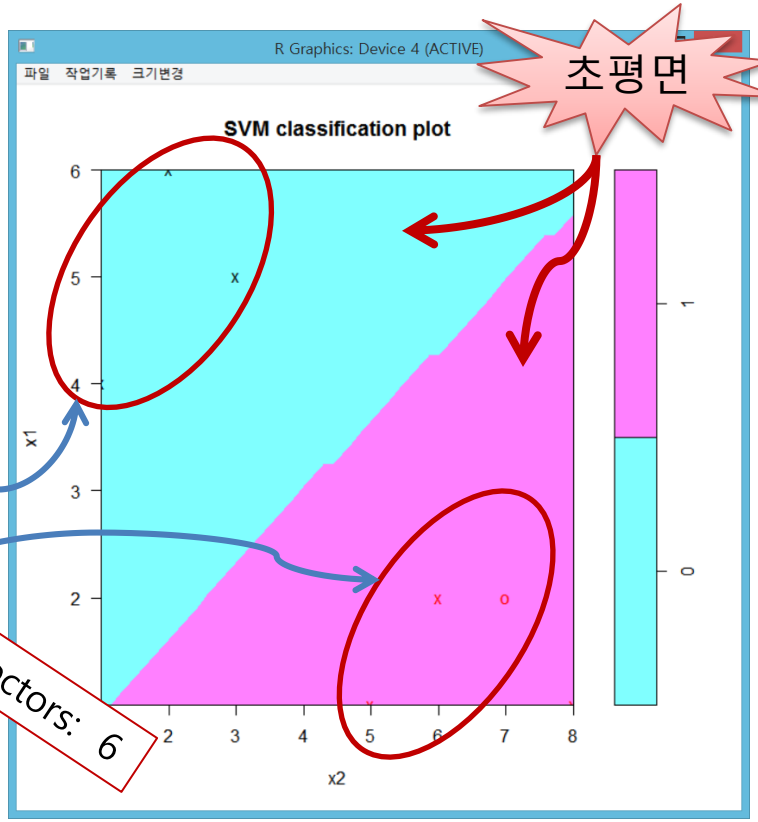
서포트 벡터 : 결정경계를 지지하는 벡터
(마진에 닿는 포인터)

● 초평면(Hyperplane)

서포트 벡터 머신은 분류 또는 회귀 분석에 사용 가능한 초평면(hyperplane) 또는 초평면 들의 집합으로 구성

2개의 집합 직선으로 분리



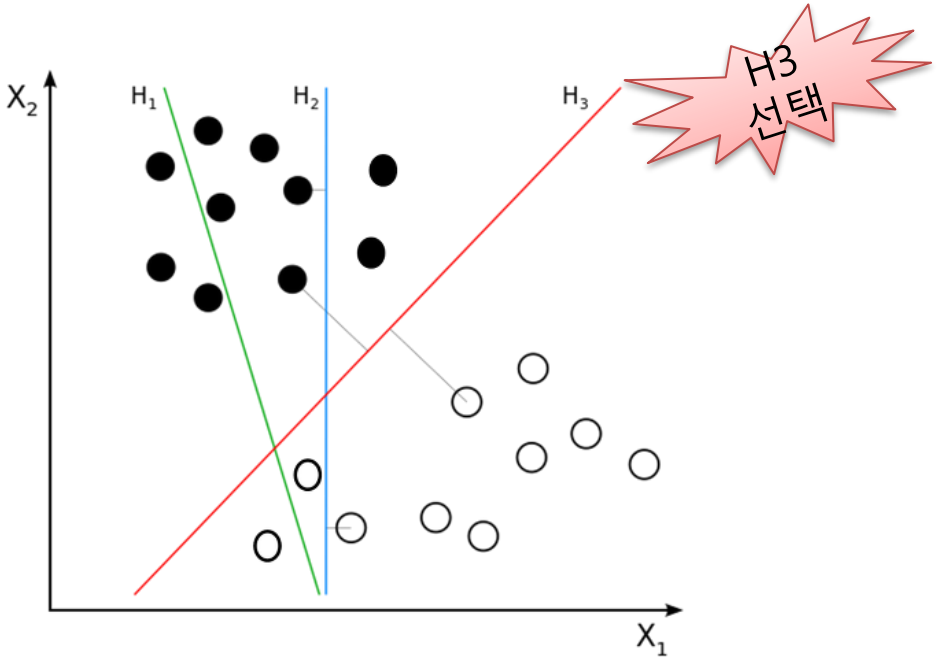


Number of Support Vectors: 6

Margin과 가장 가까운 점 6개

1) 선형 SVM(Linear SVM)

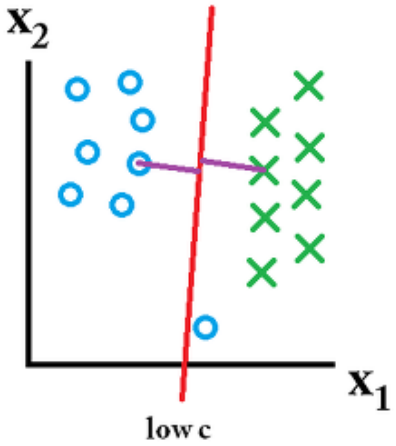
- ✓ 데이터를 선형으로 분리하는 최적의 선형 결정 경계를 찾는 알고리즘
- ✓ 마진이 가장 큰 결정 경계 찾음



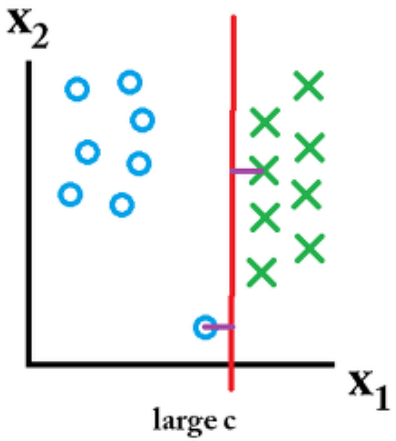
[참고] https://bskyvision.com/163

● C 파라미터

- ✓ 선형 SVM 하이퍼 파라미터 : 기본값 C = 1.0
- ✓ 오분류(Cost)에 벌칙을 적용하는 파라미터
  - 값을 작게 하는 경우 : 모델의 일반화, 과소적합 발생
  - 값을 크게 하는 경우 : 모델의 오분류 최소화, 과적합 발생



C값 작게 한 경우

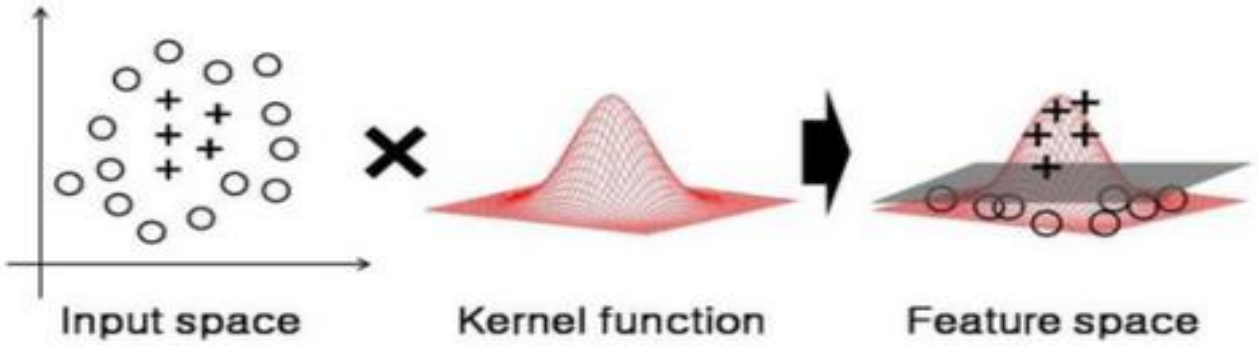


C값 크게 한 경우

2) 비선형 SVM

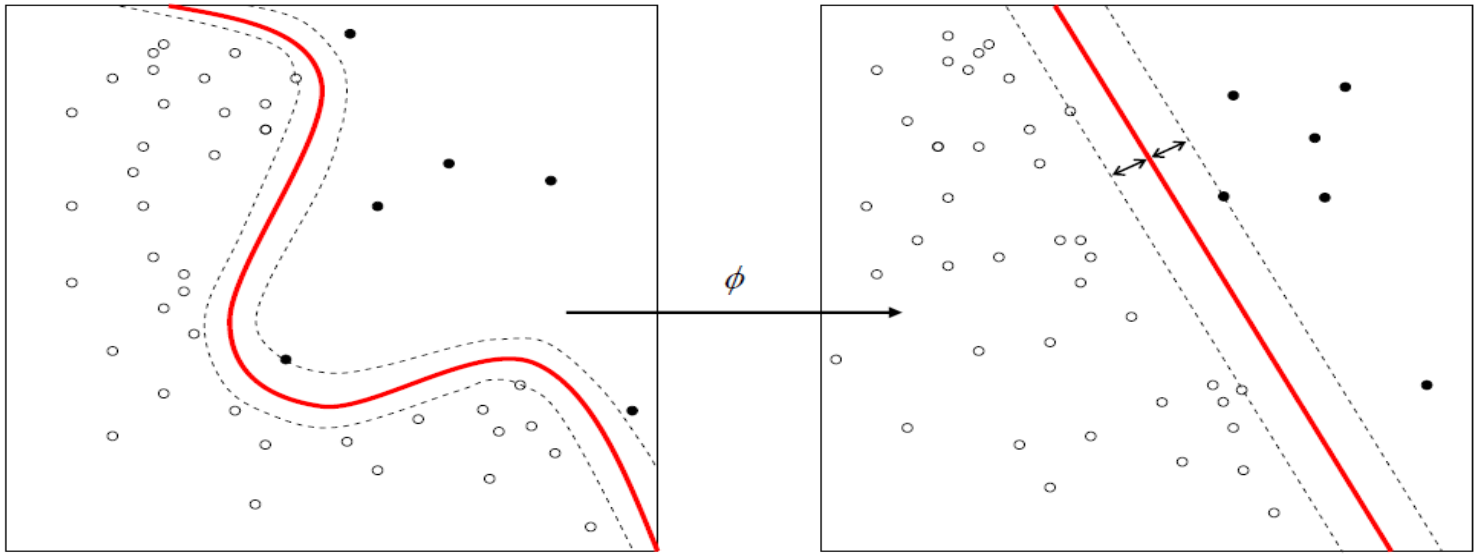
1992년 Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik 제안
선형 분리가 어려운 문제에 커널 트릭(Kernel Trick) 적용 비선형 분류 제안

- 커널 트릭(Kernel Trick) : 비선형(Non Linear) 관계를 선형으로 변환 역할
- 커널 함수(kernel function) : 커널 트릭에 사용되는 함수
- 커널 함수 종류 : Polynomial, Sigmoid, 가우시안 RBF



❖ 주어진 데이터를 고차원 특징 공간으로 사상(mapping)해서 선형 분류하는 기법

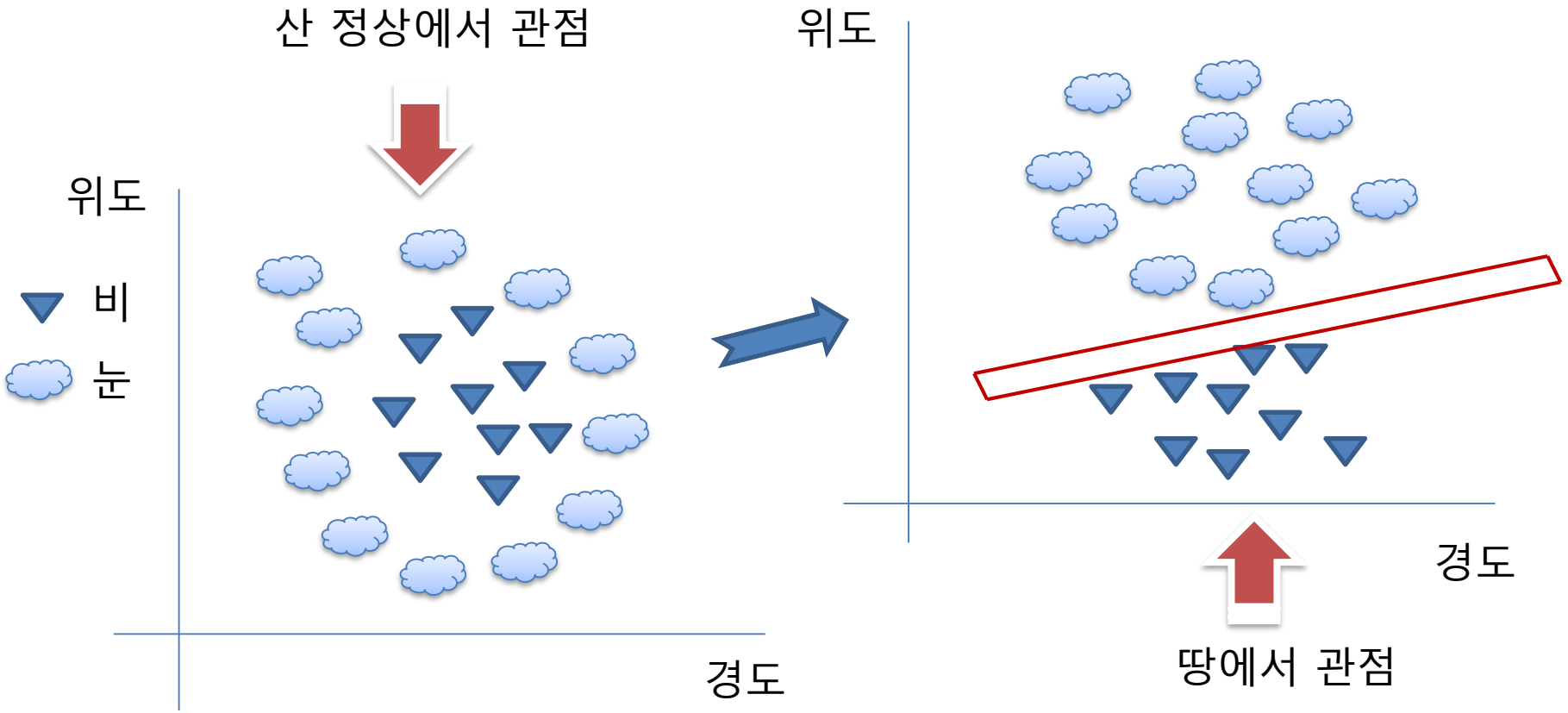
● 비선형 커널 트릭 예



선형 분리가 어려운 문제

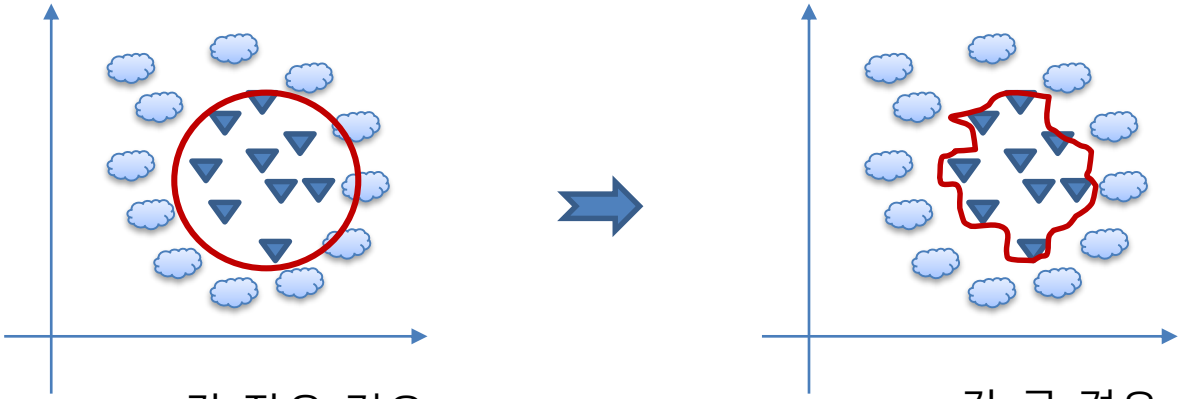
커널 트릭 적용 비선형 분류

● 눈과 비 관계를 나타낸 비선형 커널 트릭 예



● C와 gamma 파라미터

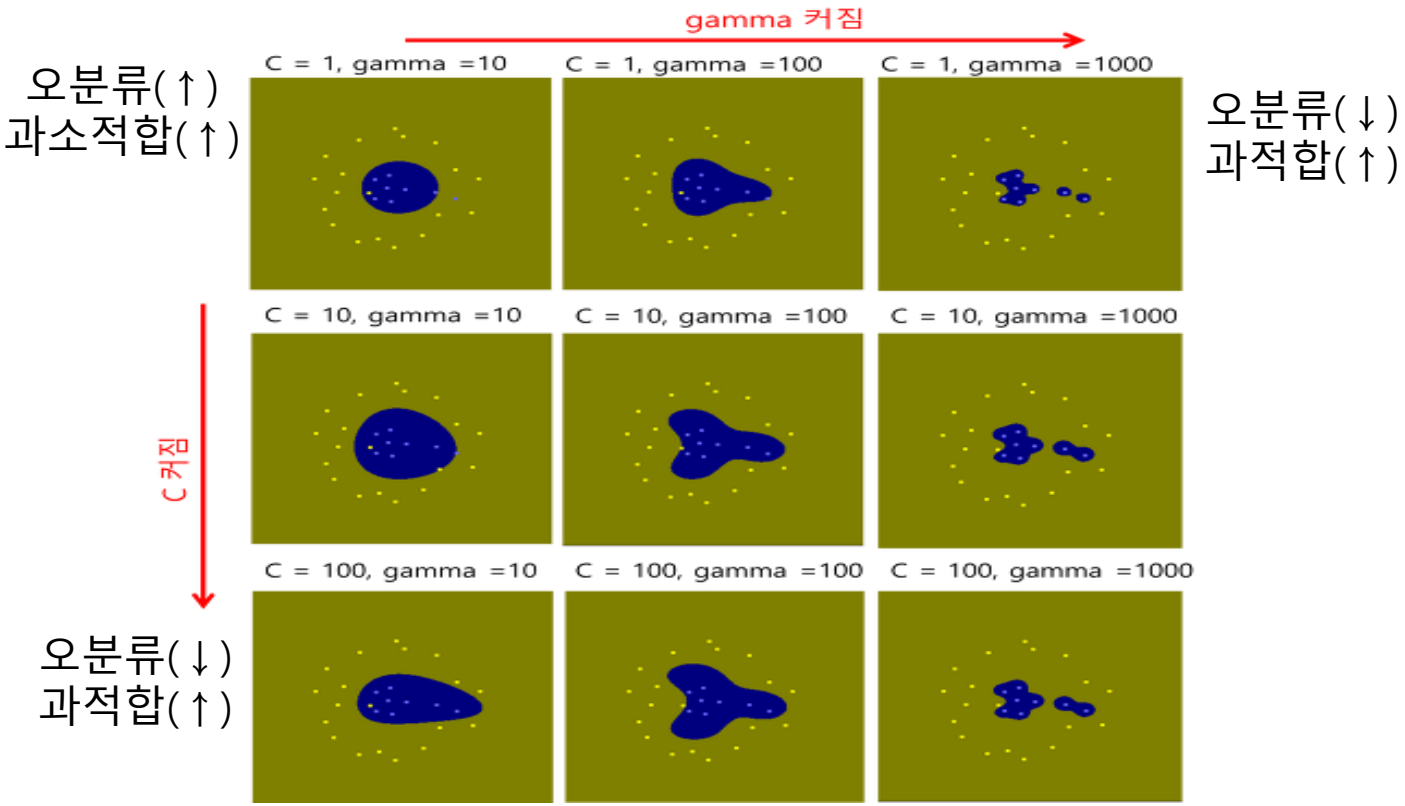
- ✓ 비선형 SVM 하이퍼 파라미터 : 기본값 C = 1.0, gamma = 'auto'
- ✓ C : 오분류(Cost)에 벌칙을 적용하는 파라미터
  - ➢ 값을 작게 하는 경우 : 모델의 일반화, 과소적합 발생
  - ➢ 값을 크게 하는 경우 : 모델의 오분류 최소화, 과적합 발생
- ✓ gamma : 결정경계 모양을 결정하는 파라미터
  - ➢ 값을 작게 하는 경우 : 결정경계, 원의 크기 커짐, 타원 모양
  - ➢ 값을 크게 하는 경우 : 결정경계 원의 크기 작아짐, 찌그러진 모양



gamma값 작은 경우

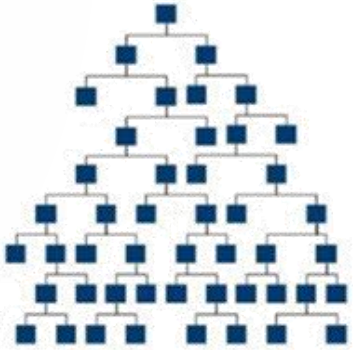
gamma값 큰 경우

- C와 gamma 파라미터
  - ✓ C가 커질수록 오분류의 가능성이 낮다.
  - ✓ gamma가 커질 수록 결정 경계 가까이에 있는 데이터 샘플들에 영향을 크게 받기 때문에 원의 크기는 작아지고, 더 구불구불
  - ✓ 두 파라미터 모두 적정값을 찾아는 것이 과제



4) Decision Tree 알고리즘

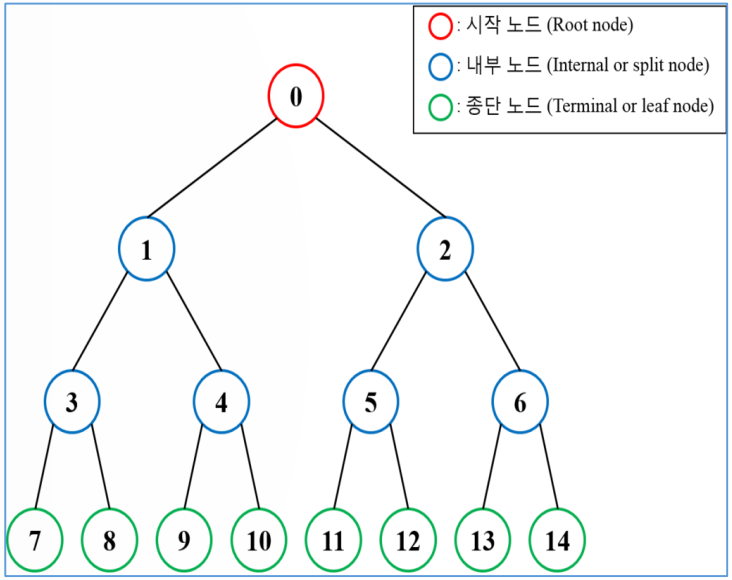
- 모델의 시각화가 쉽고, 가독성 높음
- 특징(변수)의 스케일(정규화나 표준화)조정이 필요 없음
- 이진과 연속 특징이 혼합되어 있어도 잘 동작
- 많은 특징(입력변수)을 갖는 데이터 셋은 부적합
- 단일결정 tree 학습으로 과대적합 발생 우려(일반화 성능 저하)
- 과대적합 해결방안
  - ✓ 각 규칙이 포함한 관측값의 최소 갯수 제한
  - ✓ max_depth=3 : 과적합 조절



깊은 트리 과적합, 오분류 감소

4) Decision Tree 알고리즘

- 트리(Tree) 구조
  - ✓ 계층 구조로 노드와 에지(edge) 집합
  - ✓ 노드는 내부 노드(internal node)와 종단 노드(leaf node) 분류
  - ✓ 모든 노드에서 들어오는 에지는 하나 (그래프와 차이점)
  - ✓ 각 노드에서 나가는 에지는 제한 없음 (주로 두 개의 에지가 있는 것으로 가정)
- 의사결정트리(Decision Tree)
  - ✓ 결정을 내리기 위해 사용하는 트리
  - ✓ 복잡한 문제를 간단한 계층 구조형태로 나누기 위한 기술
  - ✓ 훈련 데이터로부터 트리구조와 매개변수 자동으로 학습
- 종단 노드(Y)에 대한 매개변수(X) 최적화 작업
  - ✓ 예) 노드 분할에 최적인 중요 변수 선정

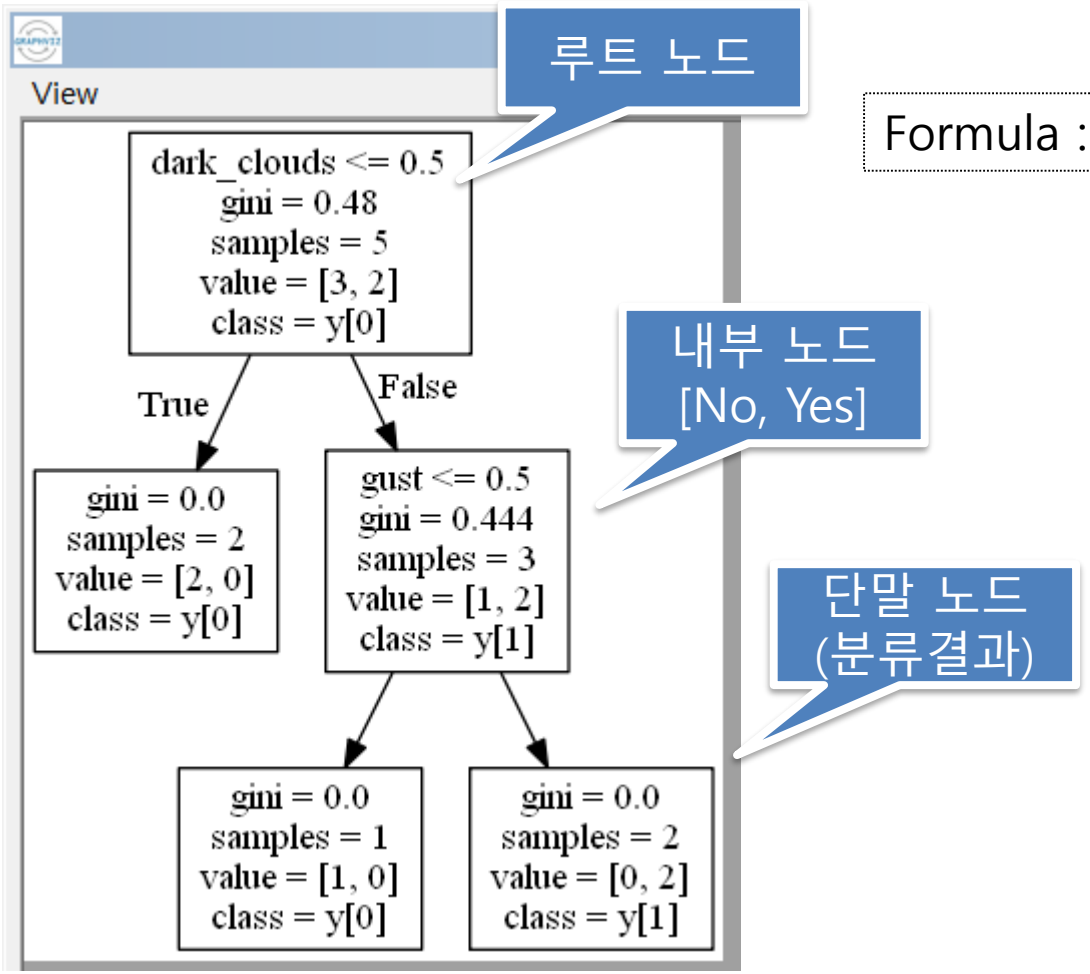


[참고] 위키백과

● Gini Impurity, Entropy

- Tree model에서 중요변수 선정 기준
- 확률 변수 간의 불확실성을 나타내는 수치
- 무질서의 양의 척도, 작을 수록 불확실성이 낮다.
- 입력 변수(x)를 대상으로 중요 변수 선정 시 사용
- 정보이득 = base 지수 - Gini 불순도 or entropy
- 정보이득이 클 수록 중요변수로 본다.
- Gini impurity = sum(p * (1-p))
- Entropy = -sum(p * log(p))

- **Gini 불순도 기준 중요변수 선정** (Graphviz 이용 Tree구조 시각화)



Formula : 비유무 ~ 먹구름, 돌풍

분류조건
Gini 불순도
목표변수 Sample수
목표변수 빈도[No, Yes]
목표변수 분류결과

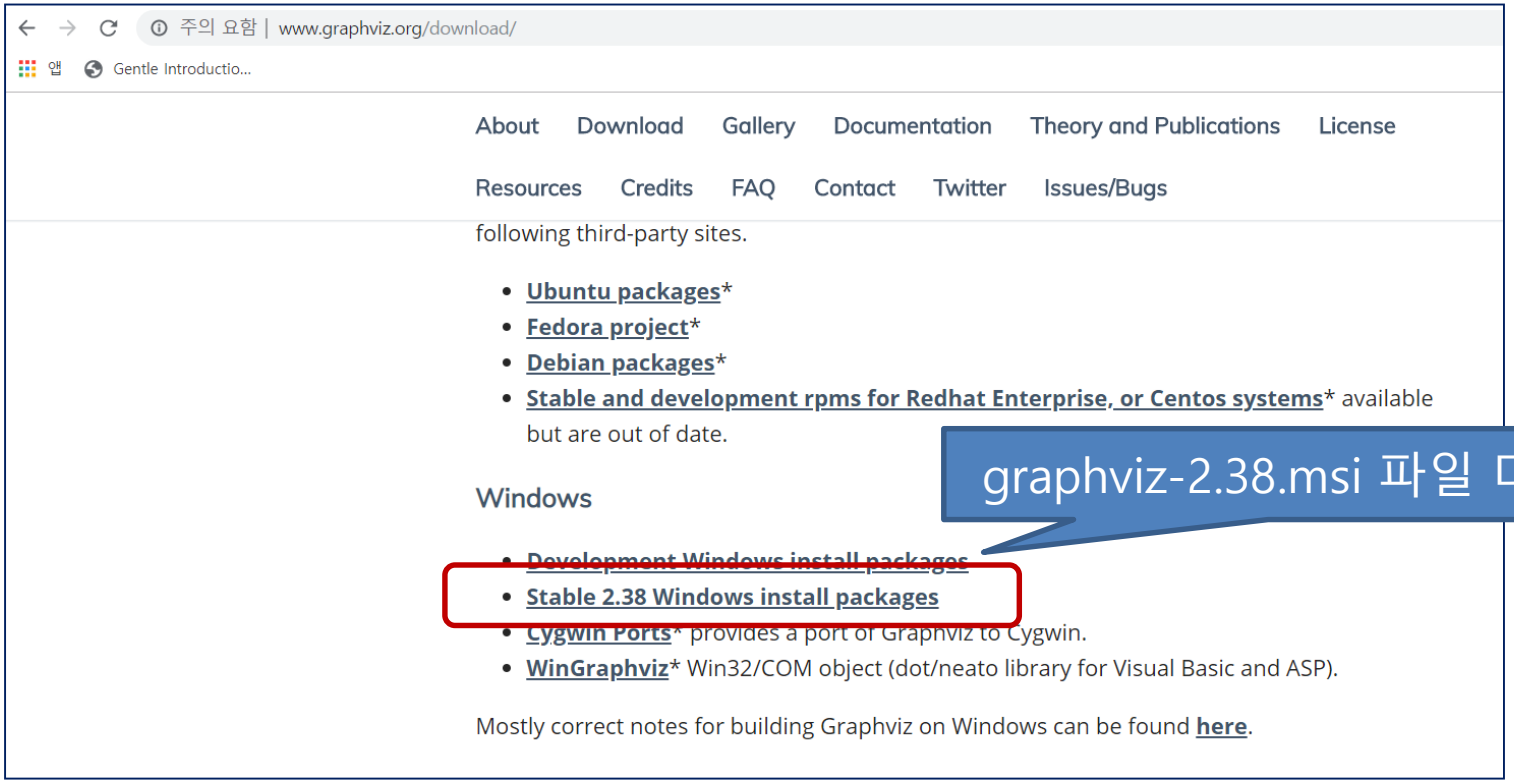
노드(Node)

소프트웨어 다운로드 http://www.graphviz.org/

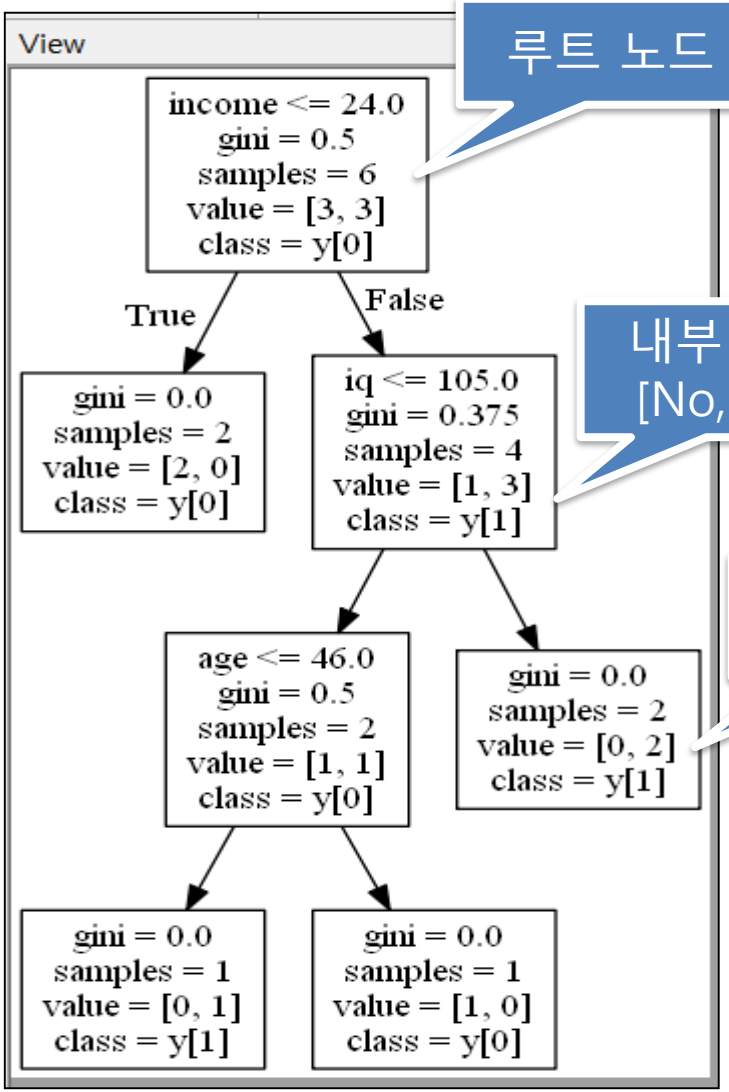
- **Graphviz(Tree model 시각화 소프트웨어) 다운로드**

http://www.graphviz.org/ 다운로드 사이트

Graphviz는 오픈 소스 그래프 시각화 소프트웨어
네트워킹, 생물 정보학, 소프트웨어 엔지니어링, 데이터베이스 및 웹 디자인,
기계 학습 등의 분야에서 시각화를 위한 애플리케이션 제공



- **Gini 불순도 기준 중요변수 선정** (Graphviz 이용 Tree구조 시각화)



루트 노드

내부 노드
[No, Yes]

단말 노드
(분류결과)

Formula :
흡연유무 ~ IQ지수, 수입, 나이, 사업가 유무, 학위소지여부

분류조건
Gini 불순도
목표변수 Sample수
목표변수 빈도[No, Yes]
목표변수 분류결과

노드(Node)

소프트웨어 다운로드 http://www.graphviz.org/

Decision Tree 알고리즘 예

data set

먹구름(x1)	돌풍(x2)	비(y)
1	1	yes
1	1	yes
1	0	no
0	1	no
0	1	no

[해설]

x1=0인 경우 y='no' 2개 분류

x1=1인 경우 x2 변수 이용 y 분류

x2=0 이면 y='no' 분류

x2=1 이면 y='yes' 분류

- 'yes' 확률 2/5=0.4이면 엔트로피 : 0.528771
- 'no' 확률 3/5=0.6 이면 엔트로피 : 0.45

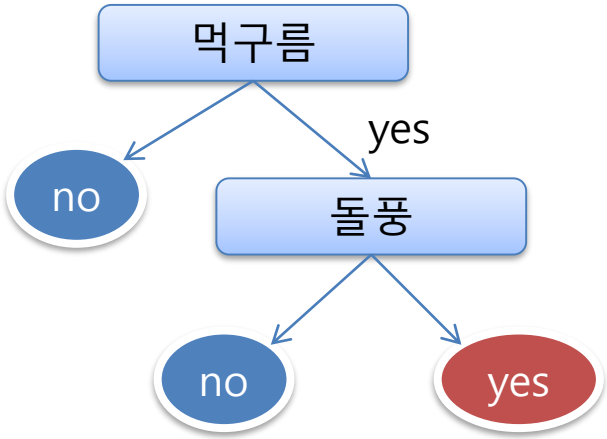
```
totEnt = totEnt -prob*log(prob,2)
```

전체 엔트로피= 0.970950594454686

```
newEnt += prob * calcEnt(dataSet)
infoGain = totEnt – newEnt
```

newEnt : 각 변수 엔트로피

infoGain : 정보이득[가장 큰 변수]



```
from math import log

# 1. data set 생성 함수
def createDataSet():
    dataSet = [[1, 1, 'yes'],
               [1, 1, 'yes'],
               [1, 0, 'no'],
               [0, 1, 'no'],
               [0, 1, 'no']]
    columns = ['dark_clouds','gust'] #
X1,X2,label
    return dataSet, columns

dataSet, columns = createDataSet()
# print(dataSet); print(columns)
```

<< 출력 결과 >>

```
[[1, 1, 'yes'], [1, 1, 'yes'], [1, 0, 'no'], [0, 1, 'no'], [0, 1, 'no']]
['dark_clouds', 'gust']
```

```
# 2. 새넌엔트로피(엔트로피) 계산 함수
def calcShannonEnt(dataSet):
    numEntries = len(dataSet) # 5

    labelCounts = {} # 빈 set
    for featVec in dataSet: # 행 단위 넘김-[1, 1, 'yes']
        currentLabel = featVec[-1] # 3번째 원소(label)
        labelCounts[currentLabel] = labelCounts.get(currentLabel,0)+1
        #print(labelCounts) # {'yes': 2, 'no': 3}

    totEnt = 0.0
    for key in labelCounts.keys(): # 'yes' or 'no'
        # 'yes' : 2/5=0.4(확률), 'no' : 3/5=0.6(확률)
        prob = float(labelCounts[key])/numEntries # 확률
        totEnt = totEnt -  prob * log(prob,2) # 확률 이용 엔트로피계산식
    return totEnt

totEnt = calcShannonEnt(dataSet)
print('전체 엔트로피=', totEnt) # 0.9709505944546686
```

<<출력 결과>>
전체 엔트로피= 0.9709505944546686
[해설] 확률를 이용하여 새넌엔트로피 계산 함수(확률이 클 수록 엔트로피는 작아짐)
전체 label 5개 중에서 'yes' : 2/5=0.4(확률), 'no' : 3/5=0.6(확률)
'yes'의 확률이 0.4이면 shannon엔트로피 0.5287712379549449
'no'의 확률이 0.6 이면 새넌엔트로피는 0.45이다.
따라서 전체 엔트로피는 0.9709505944546686
즉 y변수의 yes/no를 이용하여 전체 엔트로피 계산하고, 전체 엔트로피는
정보이득을 계산하는 기본 엔트로피로 사용된다.

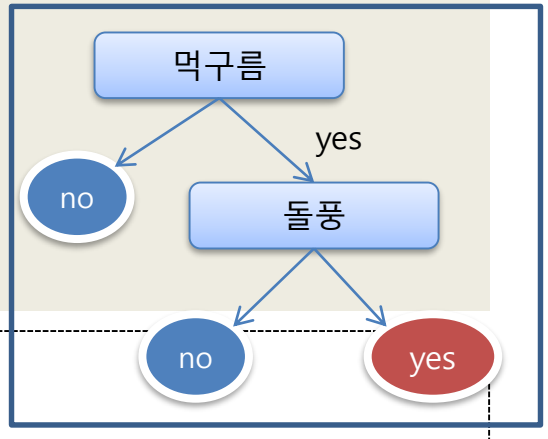
```python
# 3. data set 분할 함수  : 변수와 변수의 값으로 data 분류
def splitDataSet(dataSet, column, value):
    reDataSet = []
    for row in dataSet:
        if row[column] == value : # 칼럼 원소 == value
            blankList = [] # 빈 list
            blankList.extend(row[column+1:]) # [1, 'yes']
            # extend -> append
            reDataSet.append(blankList) # [[1, 'yes']]
    return reDataSet
```

```python
print('data set:', dataSet)
print('(dataSet, 0, 1):',splitDataSet(dataSet, 0, 1))# 전체,x1,x1=1
print('(dataSet, 0, 0):',splitDataSet(dataSet, 0, 0))# 전체,x1,x1=0
print('(dataSet, 1, 1):',splitDataSet(dataSet, 1, 1))# 전체,x2,x2=1
print('(dataSet, 1, 0):',splitDataSet(dataSet, 1, 0))# 전체,x2,x2=0
```

<<출력 결과>>
data set: [[1, 1, 'yes'], [1, 1, 'yes'], [1, 0, 'no'], [0, 1, 'no'], [0, 1, 'no']] -> 전체 data set
(dataSet, 0, 1): [[1, 'yes'], [1, 'yes'], [0, 'no']] -> x1 변수 값이 0인 경우 x2,y값
(dataSet, 0, 0): [[1, 'no'], [1, 'no']] -> x1 변수 값이 1인 경우 x2,y값
(dataSet, 1, 1): [['yes'], ['yes'], ['no'], ['no']] -> x2 변수 값이 1인 경우 y값
(dataSet, 1, 0): [['no']] -> x2 변수 값이 0인 경우 y값

```
# 4. 중요도가 높은 변수 선택 함수 : 정보이득 이용
def chooseBestFeatureToSplit(dataSet):
    columnNum = len(dataSet[0]) - 1  # 2=3-1(x칼럼 수)
    baseEntropy = calcShannonEnt(dataSet) # 전체 엔트로피 = 기준엔트로피
    bestInfoGain = 0.0
    bestFeature = -1
    for column in range(columnNum):  # range(2) : 0~1
        featList = [data[column] for data in dataSet]
        uniqueVals = set(featList)  # 중복 list 제거
        newEntropy = 0.0
        for value in uniqueVals: # {0, 1} -> 각 칼럼/원소 단위 data set 분할
            subDataSet = splitDataSet(dataSet, column, value)
            prob = len(subDataSet)/float(len(dataSet))
            newEntropy += prob * calcShannonEnt(subDataSet) # 변수 엔트로피
        infoGain = baseEntropy - newEntropy # 정보이득 : 0.97-변수엔트로피
        if (infoGain > bestInfoGain): # max 알고리즘
            bestInfoGain = infoGain
            bestFeature = column # x1(0),x2(1)
    return bestFeature

infoGain = chooseBestFeatureToSplit(dataSet) # 0
print('정보이득이 높은 속성 : ', columns[infoGain])
```



<<출력 결과>>

정보이득이 높은 속성 : dark_clouds

[해설] y의 label을 가장 잘 분류하는 변수를 선택하기 위해서 정보이득을 이용한다. 기준 엔트로피에서 각 변수의 엔트로피를 뺀 값에서 가장 큰 값을 정보이득이 가장 높은 변수로 선택한다. 여기서는 x1 변수가 x2 변수 보다 y를 더 잘 분류하고 있다. 즉 중요 변수가 x1(먹구름) 변수가 된다.