# 의사 결정 트리
# &
# 규칙 기반의 분류

# 학 습 목 표

- 트리와 규칙이 데이터를 관심 있는 세그먼트로 분할하는 방법
- C5.0, 1R, RIPPER 알고리즘을 포함하는 가장 보편적인 의사 결정 트리와 분류 규칙 학습자
- 위험한 은행 대출과 독버섯 식별과 같은 실제 분류 작업을 수행하기 위한 알고리즘 사용 방법

분할 정복

# 의사 결정 트리

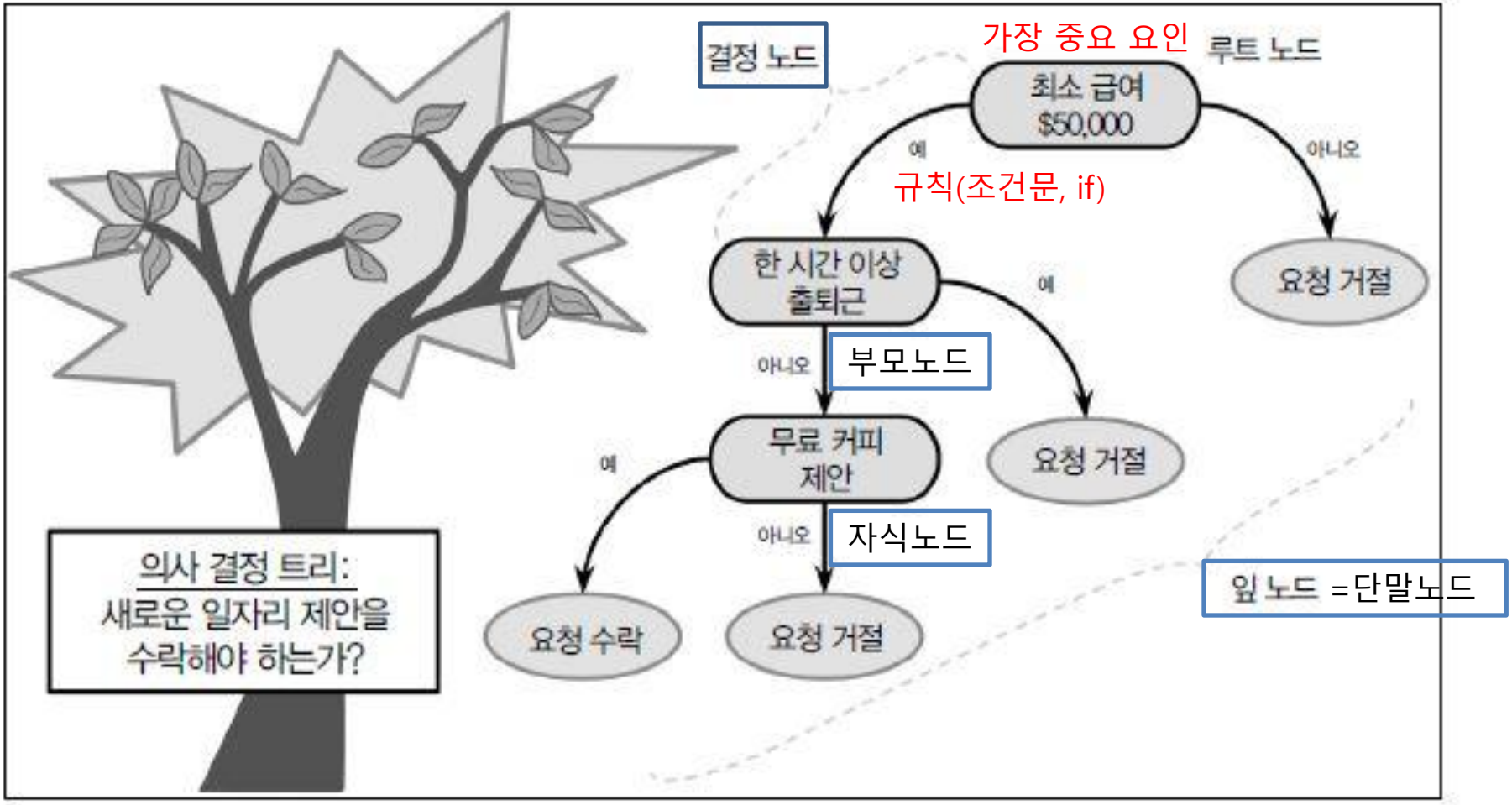



**그림 5.1:** 새로운 일자리 제안을 수락할 것인지 결정하는 절차를 나타낸 결정 트리

# 의사결정나무

- Decision tree
  - 스무고개방식의 기계학습 모형
    - yes/no로 대답할 수 있는 질문을 최대 20개까지 하며 생각하고 답하는 놀이
    - 예측 : 평균값(회귀나무), 분류 : 가장 큰 확률의 집단(분류나무)
  - 특징
    - 규칙(rules)은 나무모델(tree diagrams)로 표현, 결과는 규칙으로 표현(해석용이)
    - 나무를 만들고 키우는 과정( 재귀적 분할)
      - 그룹이 최대한 동질적으로 분리
      - 나무를 키우면서 불필요한 가지 성장을 멈춤(stop rules)
    - 가지치기
      - 만들어진 나무를 정교하게 다듬기 위해 가지를 자르는 과정
      - 과적합 방지
  - 용도
    - 신용 평가 모델 : 은행 대출, 카드발급대상
    - 고객 만족이나 고객 이탈과 같은 고객 행동 마케팅 연구 : 통신회사 이탈고객
    - 질병 진단 : 설명이 중요

# 의사결정나무

- 의사결정나무 구분
  - 분류나무(Classification Tree)
    - 종속변수가 범주형 변수이면 분류, 터미널 노드가 소속집단 표시
  - 회귀나무(Regression Tree)
    - 종속변수가 연속형 변수이면 예측, 터미널 노드가 집단의 평균 표시

- 알고리즘
  - CART(Classification And Regression Tree)
  - C5.0
  - CHAID(Chi-square Automatic Interaction Detection)

- 불순도(impurity)에 따른 분할 측도(명확한 질문인가 평가지표)
  - 지니 지수(Gini Index)
  - 엔트로피 지수(Entropy Index), 정보 이익(Information Gain)
  - 카이제곱 통계량(Chi-Square Statistic)

# 의사결정나무

- 알고리즘
  - C5.0
    - 다지분리(multiple split)이 가능하고 범주형 입력 변수의 범주 수만큼 분리 가능
    - 엔트로피 지수 사용
  - CHAID
    - 가지치기를 하지 않고 적당한 크기에서 나무모형의 성장을 중지
    - 입력변수가 반드시 범주형 변수.
    - 카이제곱 통계량 사용
  - CART(Classification and Regression Tree)
    - 목적변수가 범주형인 경우 지니 지수, 연속형인 경우 분산을 이용해 이진분리(binary split)를 사용
    - 개별 입력 변수 또는 입력변수들의 선형결합들 중 최적의 분리를 찾을 수 있음

분할 정복

# 규칙 기반의 분류

# 분할 정복

- Divide and conquer(분할 정복)
  - 재귀적 분할(recursive partitioning)
  - 나무를 키우는 과정
  - 데이터를 부분집합으로 나누고 그 다음 더 작은 부분집합으로 반복해서 나누며, 알고리즘이 부분집합의 데이터가 충분히 동질적이라고 판단하거나 다른 종료 조건을 만나 과정이 종료될 때까지 계속된다.

– 예: 시나리오 제작 추진 여부 결정
  - 호평작, 주류 히트작, 흥행 실패작 중 어디에 속할지 예측하는 알고리즘
  - 30개 영화의 성공과 실패 요인을 분석
  - 예상 촬영 예산, 출연진 수(A급), 성공 수준 사이의 관계

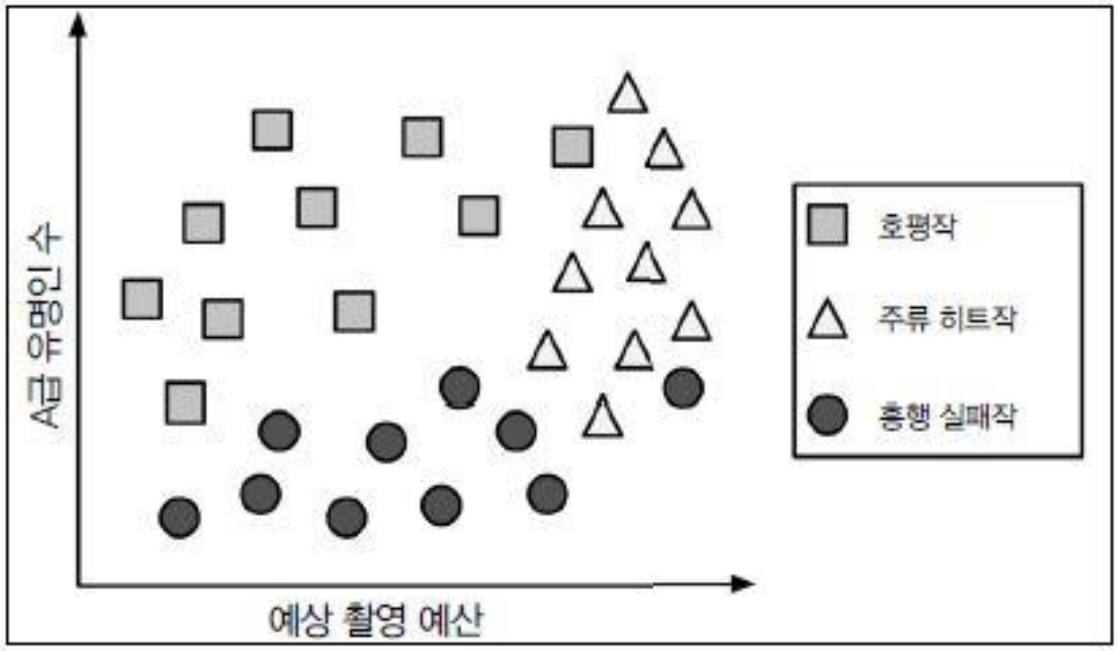



**그림 5.2:** 영화 예산과 연예인 수의 관계를 나타낸 산포도

루트 노드 생성 : A급 스타 수가 많은 그룹과 그렇지 않은 그룹으로 분할

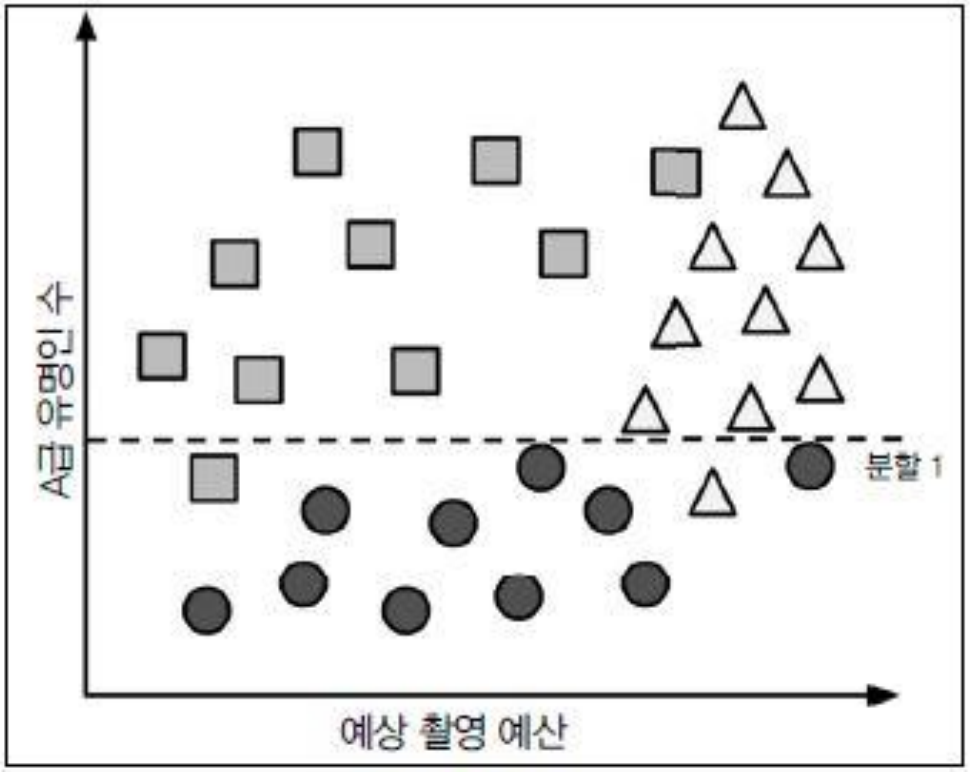



**그림 5.3:** 결정 트리의 첫 번째 분할은 데이터를 연예인 수가 많고 적음에 따라 나눴다.

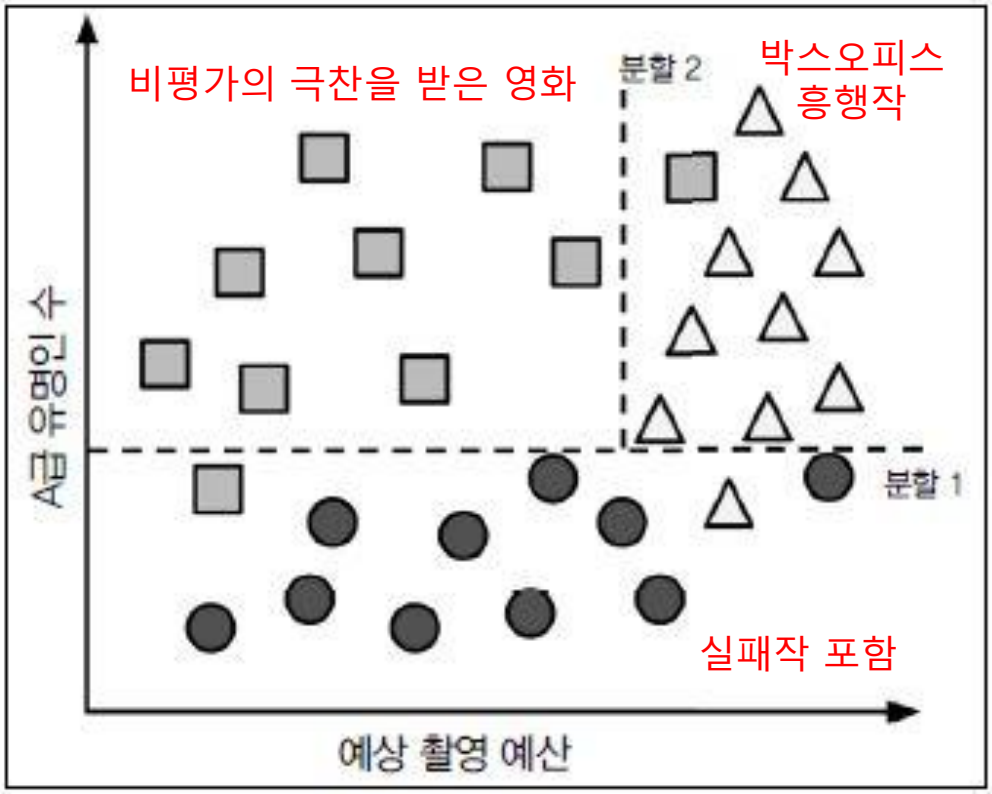



**그림 5.4:** 결정 트리의 두 번째 분할은 많은 연예인이 등장한 영화를 대상으로 추가적으로 낮은 예산과 높은 예산으로 나눴다.

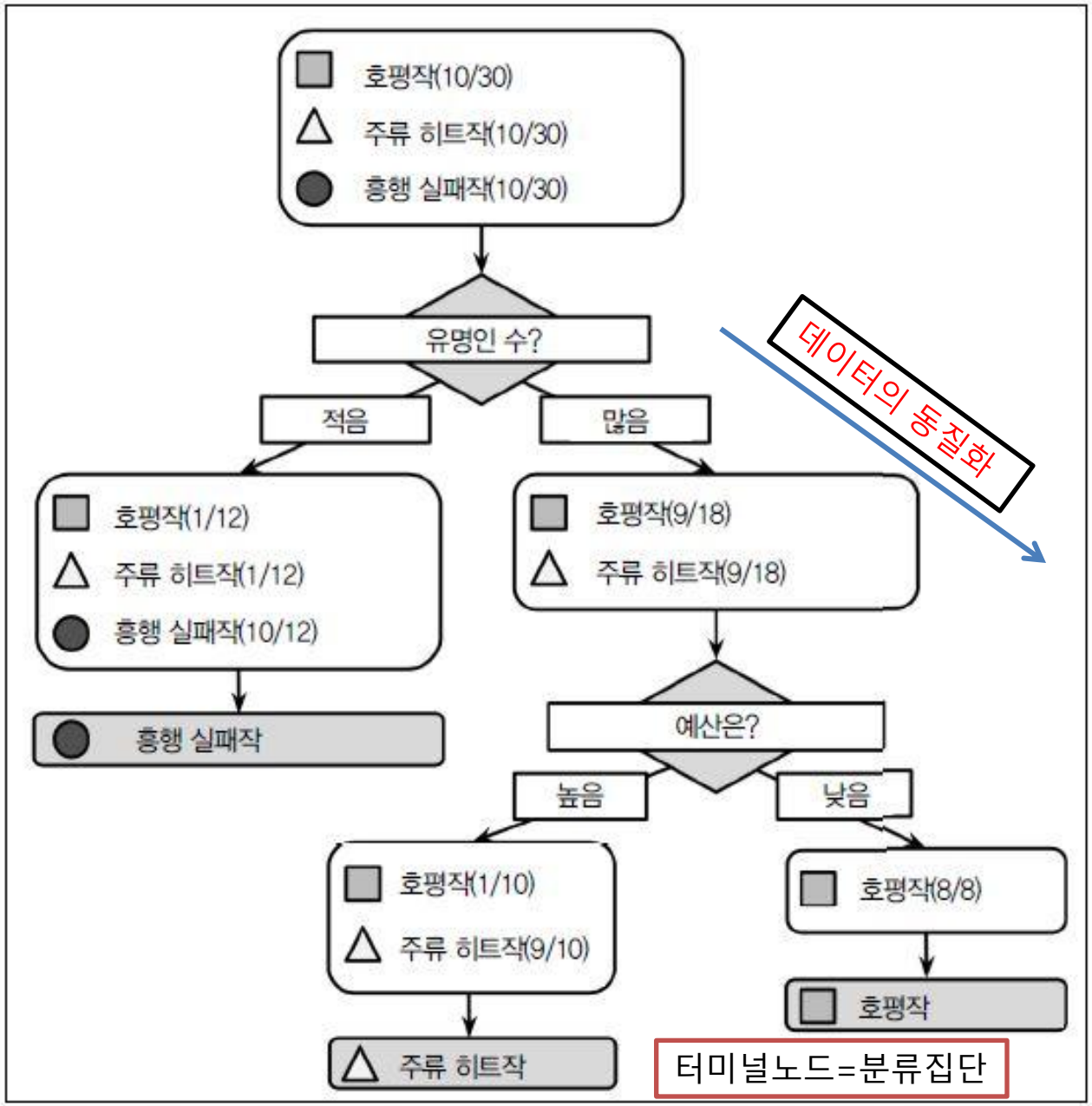



**그림 5.5:** 과거 영화데이터로 구축된 결정 트리는 미래 영화의 흥행을 예측할 수 있다.

# C5.0 알고리즘

- 의사 결정 트리를 생성하기 위한 산업 표준

| 장점 | 단점 |
| --- | --- |
| 많은 유형의 문제에 잘 실행되는 범용 분류기 | 의사 결정 모델이 레벨 수가 많은 특징의 분할로 편향될 가능성 |
| 수치, 명목, 누락 데이터 | 과적합 또는 과소적합 가능성 |
| 중요하지 않은 특징은 제외 | 축 평행 분할에 의존하기 때문에 어떤 관계는 모델링의 어려움 |
| 데이터 크기에 무관 | 훈련 데이터의 작은 변화가 큰 영향을 초래 |
| 수학적 배경 없이 해석 가능한 모델 | 큰 트리는 해석이 어렵고 트리가 만든 결정은 직관적이지 않아 보일 가능성 |
| 다른 복잡한 모델보다 더 효율적 | |

# 엔트로피 지수

- 최고의 분할 선택
  - 분할 조건이 되는 특징 식별
  - 순도(purity)
    - 인스턴스의 부분집합이 단일 클래스를 포함하는 정도
    - 순수(pure) : 단일 클래스로 이뤄진 부분집합
  - 엔트로피(entropy)
    - 클래스 값의 집합 내에서 무작위성이나 무질서를 정량화
    - 컴퓨터 : 시스템 내 정보의 불확실성 정도를 나타냄
    - 비트(bit)단위로 측정(bit : 컴퓨터의 정보처리 단위, 2진수(0,1))

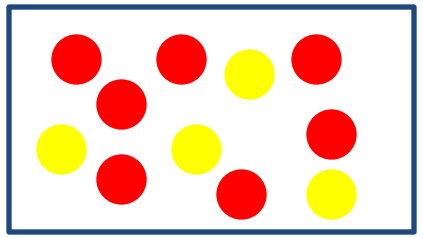
# 엔트로피 지수

- 엔트로피를 줄이고 궁극적으로 그룹 내의 동질성을 증가시키는 분할을 선택
  - 클래스가 2개인 경우 : 0 ~ 1
  - 클래스가 n개인 경우 : 0 ~ $\log_2(n)$
  - 엔트로피 지수가 낮을 수록 정렬되고 동질적인 집단이며 엔트로피 지수가 높을수록 혼잡하고 이질적인 집단이다.

- 엔트로피(S)

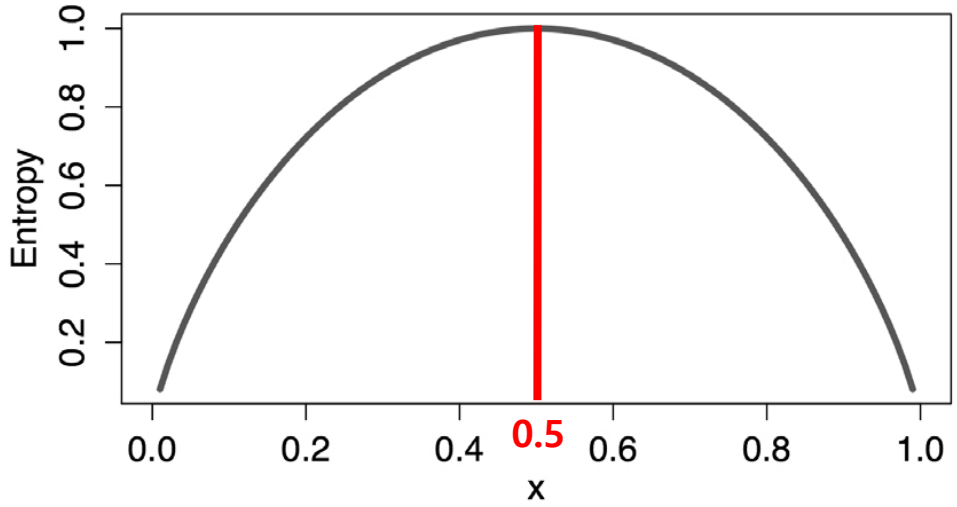
$$S = \sum_{i=1}^{c} - p_i \log_2(p_i)$$

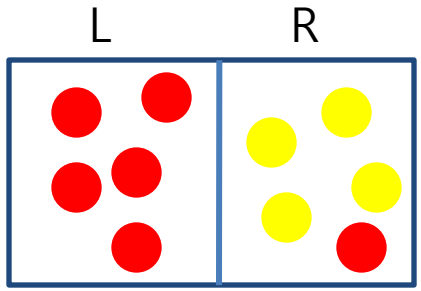
- 예제
  - 빨간색(60%)과 흰색(40%) 두 클래스를 갖는 데이터 파티션이 있다고 가정.
  - 엔트로피 = -0.6 * $\log_2(0.6)$ - 0.4 * $\log_2(0.4)$
    = 0.97



# 엔트로피 지수

- 가능한 모든 2 클래스 배열의 시각화
  - 한 클래스의 인스턴스 비율 : x
  - 나머지 클래스의 인스턴스 비율 : 1-x
  - x=0.5일 때 엔트로피가 최대
  - 하나의 클래스로 100% 분류될 때 엔트로피는 0



# 엔트로피 지수

- 정보 획득량(information gain)
  - 속성 선택 방법, 동질성의 변화량 측정
  - 정보 획득량이 가장 높은 속성을 선택
    - 정보 획득량이 높은 속성이란, 인스턴스를 결과 파티션으로 분할하는데 필요한 정보가 가장 적은 속성으로, 임의성이 가장 적고 파티션의 불순도가 가장 낮은 속성
  - 정보 획득량(F) = 분할 전 엔트로피($S_1$) – 분할 후 엔트로피($S_2$)
  - 정보 획득량이 높을수록 엔트로피는 낮다.
  - 정보 획득량이 낮을수록 엔트로피는 높다.

$$\text{엔트로피}(S) = \sum_{i=1}^{n} w_i \text{엔트로피}(P_i)$$

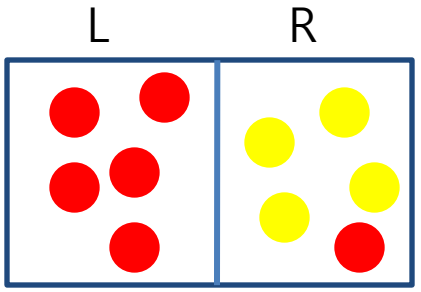
# 정보획득량

- 정보획득량 계산
  - E = -0.6 * $\log_2(0.6)$ - 0.4 * $\log_2(0.4)$ = 0.97
  - E(L) = $-\frac{0}{5}\log_2\frac{0}{5}-\frac{5}{5}\log_2\frac{5}{5}=0$
  - E(R) = $-\frac{4}{5}\log_2\frac{4}{5}-\frac{1}{5}\log_2\frac{1}{5}=0.7219$
  - E(LR)= $(\frac{5}{10})E(L)+(\frac{5}{10})E(R)=0.36095$
  - 정보획득량 = E – E(LR)

    = 1- 0.36095

    = 0.63905

  불확실성이 64% 감소. 값이 클수록 좋다.



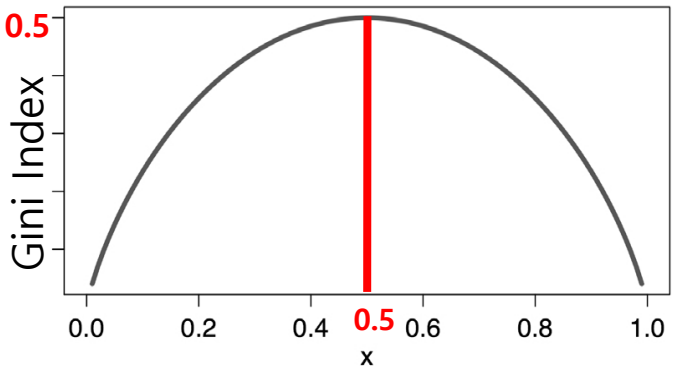
# 가지치기

- 과적합(overfitting)문제 해결
  - 가지치기(pruning)
    - 처음 보는 데이터에 일반화가 잘 되도록 트리의 크기를 줄이는 방법
    - 사전가지치기
      - 결정이 특정 개수에 도달하거나 결정 노드가 포함하는 예시 개수가 아주 작은 숫자가 되면 트리 성장 멈춤.
    - 사후 가지치기(C5.0)
      - 가장 큰 트리를 만든 후에 트리 크기를 적절한 수준으로 줄이고자 잎 노드를 바른다.
      - 방법
        - » 훈련 데이터에 과적합되는 큰 트리로 자라게 한다.
        - » 나중에 분류 오류에 영향이 거의 없는 노드와 분기 제거

# CART

- CART(Classification And Regression Tree)
  - 분류와 예측 모두 가능
  - Gini 지수를 이용
    - 불확실성 측도 (낮을수록 좋다)
  - 이진분리
  - 가지치기
    - 나무를 완전히 성장시킨 후 가지치기(사후 가지치기)
    - 학습 데이터를 이용하여 나무를 성장시키고, 테스트 데이터를 이용하여 가지치기
  - 2번 복원추출 할 경우 나올 수 있는 확률
    - 최대 G(D)=0.5(두 집단이 동일할 때)
    - 그룹이 동질적이면 G(D)가 낮아지고
    - 그룹이 이질적이면 G(D)가 높아진다

$$G(D) = 1 - \sum_{i=1}^{n} p_i^{\,2}$$



# 의사결정나무 모델 과정

- 나무모델 생성
  - 재귀적 분할 방법을 이용하여 나무모형 생성
  - CART(Gini), C5.0(Entropy), CHAID(Chi-square)이용
- 과적합문제 해결
  - 나무모형을 생성하면서 가지치기(CHAID)
  - 나무모형을 완성 후 가지치기(CART, C5.0)
    - Training dataset : C5.0
    - Training dataset으로 나무생성 후 test dataset으로 가지치기 : CART
- 검증
  - 교차타당성으로 의사결정나무 모형 평가
- 해석및예측
  - 의사결정나무 모형을 해석하고 예측모형으로 설정

# 실습예제

- C5.0 의사 결정 트리를 이용한 위험 은행 대출 식별
  - 1단계 : 데이터 수집
    - credit <- read.csv("credit.csv")

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | checking_balance | months_lo | credit_hist | purpose | amount | savings_ba | employme | percent_of | years_at_re | age | other_crec | housing | existing_lo | job | dependen | phone | default |
| 2 | < 0 DM | 6 | critical | furniture/a | 1169 | unknown | > 7 years | 4 | 4 | 67 | none | own | 2 | skilled | 1 | yes | no |
| 3 | 1 - 200 DM | 48 | good | furniture/a | 5951 | < 100 DM | 1 - 4 years | 2 | 2 | 22 | none | own | 1 | skilled | 1 | no | yes |
| 4 | unknown | 12 | critical | education | 2096 | < 100 DM | 4 - 7 years | 2 | 3 | 49 | none | own | 1 | unskilled | 2 | no | no |
| 5 | < 0 DM | 42 | good | furniture/a | 7882 | < 100 DM | 4 - 7 years | 2 | 4 | 45 | none | other | 1 | skilled | 2 | no | no |
| 6 | < 0 DM | 24 | poor | car | 4870 | < 100 DM | 1 - 4 years | 3 | 4 | 53 | none | other | 2 | skilled | 2 | no | yes |
| 7 | unknown | 36 | good | education | 9055 | unknown | 1 - 4 years | 2 | 4 | 35 | none | other | 1 | unskilled | 2 | yes | no |
| 8 | unknown | 24 | good | furniture/a | 2835 | 500 - 100( | > 7 years | 3 | 4 | 53 | none | own | 1 | skilled | 1 | no | no |
| 9 | 1 - 200 DM | 36 | good | car | 6948 | < 100 DM | 1 - 4 years | 2 | 2 | 35 | none | rent | 1 | managem | 1 | yes | no |
| 10 | unknown | 12 | good | furniture/a | 3059 | > 1000 DM | 4 - 7 years | 2 | 4 | 61 | none | own | 1 | unskilled | 1 | no | no |
| 11 | 1 - 200 DM | 30 | critical | car | 5234 | < 100 DM | unemploy | 4 | 2 | 28 | none | own | 2 | managem | 1 | no | yes |
| 12 | 1 - 200 DM | 12 | good | car | 1295 | < 100 DM | < 1 year | 3 | 1 | 25 | none | rent | 1 | skilled | 1 | no | yes |
| 13 | < 0 DM | 48 | good | business | 4308 | < 100 DM | < 1 year | 3 | 4 | 24 | none | rent | 1 | skilled | 1 | no | yes |
| 14 | 1 - 200 DM | 12 | good | furniture/a | 1567 | < 100 DM | 1 - 4 years | 1 | 1 | 22 | none | own | 1 | skilled | 1 | yes | no |
| 15 | < 0 DM | 24 | critical | car | 1199 | < 100 DM | > 7 years | 4 | 4 | 60 | none | own | 2 | unskilled | 1 | no | yes |
| 16 | < 0 DM | 15 | good | car | 1403 | < 100 DM | 1 - 4 years | 2 | 4 | 28 | none | rent | 1 | skilled | 1 | no | no |
| 17 | < 0 DM | 24 | good | furniture/a | 1282 | 100 - 500 | 1 - 4 years | 4 | 2 | 32 | none | own | 1 | unskilled | 1 | no | yes |
| 18 | unknown | 24 | critical | furniture/a | 2424 | unknown | > 7 years | 4 | 4 | 53 | none | own | 2 | skilled | 1 | no | no |
| 19 | < 0 DM | 30 | perfect | business | 8072 | unknown | < 1 year | 2 | 3 | 25 | bank | own | 3 | skilled | 1 | no | no |
| 20 | 1 - 200 DM | 24 | good | car | 12579 | < 100 DM | > 7 years | 4 | 2 | 44 | none | other | 1 | managem | 1 | yes | yes |
| 21 | unknown | 24 | good | furniture/a | 3430 | 500 - 100( | > 7 years | 3 | 2 | 31 | none | own | 1 | skilled | 2 | yes | no |
| 22 | unknown | 9 | critical | car | 2134 | < 100 DM | 1 - 4 years | 4 | 4 | 48 | none | own | 3 | skilled | 1 | yes | no |

# 실습과정

- C5.0 의사 결정 트리를 이용한 위험 은행 대출 식별
  - 1단계 : 데이터 수집
  - 2단계_1 : 데이터 탐색
  - 2단계_2 : 데이터 준비
  - 3단계 : 모델 훈련
  - 4단계 : 모델 성능 평가
  - 5단계 : 모델 성능 개선

# Sample_data

- Variables
  - Checking balance : 수표 계좌 잔고
  - Savings balance : 저축 계좌 잔고
  - Month loan duration : 대출 기간
  - Amount : 대출 금액
  - Default : 상환 또는 채무 불이행

- Random sample
  - Set.seed() : 동일한 결과 생성
  - Sample() : 랜덤 샘플링

- Confusion matrix(혼동행렬)
  - CrossTable() : 교차분석

• 패키지 설치

Install.packages("C50")

Library(C50)

**C5.0 의사 결정 트리 구문**

C50 패키지의 C5.0() 함수 사용

**분류기 구축:**

```
m <- C5.0(train, class, trials = 1, costs = NULL)
```

- train: 훈련 데이터를 포함하는 데이터 프레임 또는 행렬
- class: 훈련 데이터의 각 행에 대한 클래스를 갖는 팩터 벡터
- trials: 부스팅 반복 횟수를 제어하는 숫자 옵션(디폴트 값은 1)
- costs: 다양한 종류의 오류 관련된 비용을 지정하는 행렬 옵션

이 함수는 예측에 사용될 수 있는 C5.0 객체를 반환한다.

**예측:**

```
p <- predict(m, test, type = "class")
```

- m: C5.0() 함수에 의해 훈련된 모델
- test: 분류기를 구축하는 데 사용된 훈련 데이터와 같은 특징을 갖는 테스트 데이터를 포함하는 데이터 프레임 또는 행렬
- type: "class"나 "prob", 예측 결과가 가장 확률이 높은 클래스인지 원시 예측 확률인지를 지정

이 함수는 type 파라미터의 값에 따라 예측 클래스 값이나 원시 예측 확률의 벡터를 반환한다.

**예제:**

```
credit_model <- C5.0(credit_train, loan_default)
credit_prediction <- predict(credit_model,
  credit_test)
```

# 데이터 탐색

```
credit <- read.csv("credit.csv")
str(credit)

table(credit$checking_balance)
table(credit$savings_balance)

summary(credit$months_loan_duration)
summary(credit$amount)

table(credit$default)
```

# 데이터 준비

```
set.seed(1234)
train_sample <- sample(1000, 900)
str(train_sample)
head(train_sample)

credit_train <- credit[train_sample, ]
credit_test  <- credit[-train_sample, ]
head(credit_train)

prop.table(table(credit_train$default))
prop.table(table(credit_test$default))
```

# 모델 훈련 및 구축

```
str(credit_train$default)

credit_train$default <- as.factor(credit_train$default)

str(credit_train$default)

credit_model <- C5.0(credit_train[-17], credit_train$default)

credit_model

summary(credit_model)
```

```
C5.0 [Release 2.07 GPL Edition]
------------------------------

Class specified by attribute `outcome'

Read 900 cases (17 attributes) from undefined.data

Decision tree:

checking_balance in {> 200 DM,unknown}: no (412/50)
checking_balance in {< 0 DM,1 - 200 DM}:
:...credit_history in {perfect,very good}: yes (59/18)
    credit_history in {critical,good,poor}:
    :...months_loan_duration <= 22:
        :...credit_history = critical: no (72/14)
        :   credit_history = poor:
        :   :...dependents > 1: no (5)
        :   :   dependents <= 1:
        :   :   :...years_at_residence <= 3: yes (4/1)
        :   :       years_at_residence > 3: no (5/1)
```

```
Evaluation on training data (900 cases):

      Decision Tree
    ----------------
    Size      Errors

      69   99(11.0%)   <<


     (a)   (b)    <-classified as
    ----  ----
     625    10    (a): class no
      89   176    (b): class yes
```

# 모델 성능 평가

```
credit_pred <- predict(credit_model, credit_test)
credit_pred
library(gmodels)
CrossTable(credit_test$default, credit_pred,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn=c('actual default', 'preidected default'))
```

| | preidected default | | |
|---|---|---|---|
| actual default | no | yes | Row Total |
| no | 64 | 13 | 77 |
| | 0.640 | 0.130 | |
| yes | 15 | 8 | 23 |
| | 0.150 | 0.080 | |
| Column Total | 79 | 21 | 100 |

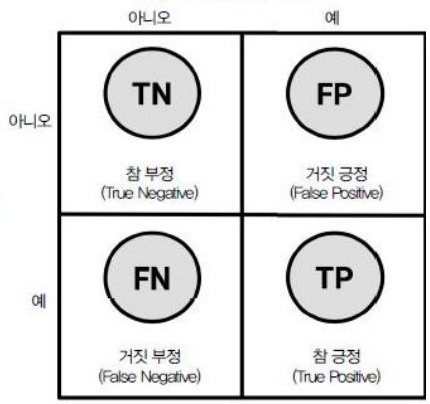
# 모델 성능 개선

```
credit_boost10 <- C5.0(credit_train[-17], credit_train$default, trials = 10)
credit_boost10
summary(credit_boost10)
credit_boost_pred10 <- predict(credit_boost10, credit_test)
CrossTable(credit_test$default, credit_boost_pred10,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn=c('actual default', 'preidected default'))
```

| actual default | preidected default: no | preidected default: yes | Row Total |
|---|---|---|---|
| no | 67<br>0.670 | 10<br>0.100 | 77 |
| yes | 12<br>0.120 | 11<br>0.110 | 23 |
| Column Total | 79 | 21 | 100 |

# 비용행렬

- Cost matrix
  - C5.0 알고리즘의 여러 오류 유형에 패널티를 적용해서 매트릭스 구조로 저장
  - 2 × 2 matrix (yes, no)
    1. predicted no, actual no
    2. Predicted yes, actual no
    3. Predicted no, actual yes
    4. Predicted yes, actual yes




| | | actual | |
|---|---|---|---|
| | | no | yes |
| predicted | no | 0 | 4 |
| | yes | 1 | 0 |

알고리즘 분류기가 no 또는 yes를 정확히 분류할 때는 비용이 0, FN은 FP의 비용 1에 대해 비용 4를 갖는다.

```
matrix_dimensions <- list(c("no","yes"),c("no","yes"))
names(matrix_dimensions) <- c("predicted","actual")
matrix_dimensions
error_cost <- matrix(c(0,1,4,0), nrow=2, dimnames=matrix_dimensions)
error_cost
credit_cost <- C5.0(credit_train[-17], credit_train$default, costs = error_cost)
credit_cost
credit_cost_pred <- predict(credit_cost, credit_test)
CrossTable(credit_test$default, credit_cost_pred,
           prop.chisq = FALSE, prop.c = FALSE, prop.r = FALSE,
           dnn=c('actual default', 'preidected default'))
```

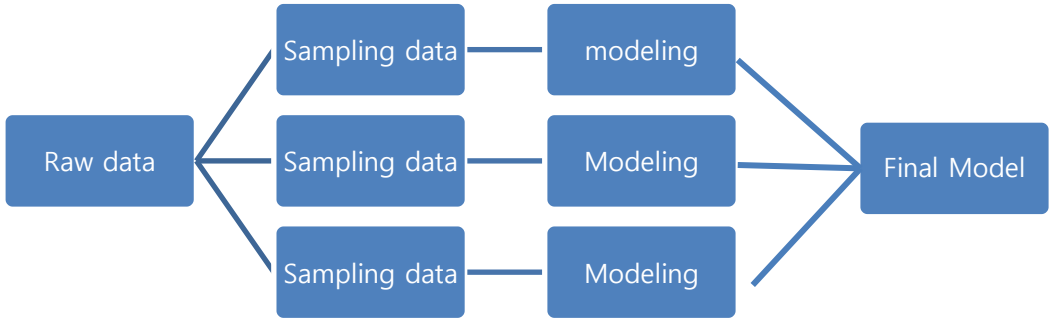
| | preidected default | | |
|---|---|---|---|
| actual default | no | yes | Row Total |
| no | 43 | 34 | 77 |
| | 0.430 | 0.340 | |
| yes | 5 | 18 | 23 |
| | 0.050 | 0.180 | |
| Column Total | 48 | 52 | 100 |

# 메타 알고리즘

- 앙상블
  - 주어진 자료로부터 여러 개의 예측모형들을 만든 후 예측모형들을 조합하여 하나의 최종 예측모형을 만드는 방법
    - 배깅
    - 부스팅
    - 랜덤포레스트
    - 스태킹
  - 학습방법이 불안정성
    - 학습자료의 작은 변화에 의해 예측모형이 크게 변하는 경우, 그 학습방법은 불안정하다.
    - 가장 안정적인 방법
      - 1-nearest neighbor(가장 가까운 자료만 변하지 않으면 예측 모형이 변하지 않음)
      - 선형회귀모형 (최소제곱추정법으로 모형 결정)
    - 가장 불안정한 방법
      - 의사결정나무

# 분류분석

- 앙상블
  - 배깅(bagging : bootstrap aggregating)
    - 주어진 자료에서 여러 개의 붓스트랩(bootstrap)자료를 생성하고 각 붓스트랩 자료에 예측모형을 만든 후 결합하여 최종 예측모형을 만드는 방법
      - 붓스트랩(bootstrap) : 주어진 자료에서 동일한 크기의 표본을 랜덤 복원추출로 뽑은 자료
    - 최적의 의사결정나무를 구축할 때 가장 어려운 부분이 가지치기(pruning)이지만 배깅에서는 가지치기를 하지 않고 최대로 성장한 의사결정나무들을 활용
    - 배깅은 훈련자료를 모집단으로 생각하고 평균예측모형을 구한 것과 같아 분산을 줄이고 예측력을 향상시킬 수 있다.



- 부스팅(boosting)
  - 예측력이 약한 모형들을 결합하여 강한 예측모형을 만드는 방법
  - Adaboost(Adaptive boosting, 에이다부스트)
    - 이진 분류 문제에서 랜덤 분류기 보다 조금 더 좋은 분류기 n개에 각각 가중치를 설정하고 n개의 분류기를 결합하여 최종 분류기를 만드는 방법(가중치의 합은 1)
  - 훈련 오차를 빠르고 쉽게 줄일 수 있다.
  - 배깅에 비해 많은 경우 예측오차가 향상되어 Adaboost의 성능이 배깅 보다 뛰어난 경우가 많다.

- 랜덤포레스트(random forest)
  - 배깅과 부스팅보다 더 많은 무작위성을 주어 약한 학습기들을 생성한 후 이를 선형 결합하여 최종 학습기를 만드는 방법
  - randomForest 패키지는 random input에 따른 forest of tree를 이용한 분류방법이다.
  - 랜덤한 forest에는 많은 트리들이 생성된다.
  - 수천 개의 변수를 통해 변수 제거없이 실행되므로 정확도 측면에서 좋은 성과를 보인다.
  - Category variable의 value 종류가 32개를 넘을 수 없다.
  - 이론적 설명이나 최종 결과에 대한 해석이 어렵다는 단점이 있지만 예측력이 매우 높은 장점이 있다.
  - 입력변수가 많은 경우, 배깅과 부스팅과 비슷하거나 좋은 예측력을 보인다.