Chap_03
게으른 학습(Lasy learning)

최근접 이웃을 사용한 분류

학습내용

- 최근접 이웃 분류기를 정의하는 주요 개념과 게으른 학습자로 간주되는 이유
- 거리를 이용한 두 예시의 유사도 측정 방법
- k-NN이라 불리는 유명한 최근접 이웃 분류기 적용 방법

최근접 이웃 분류의 이해

- 유유상종(類類相從)
  - 데이터를 유사하거나 가장 가까운(nearest) 이웃과 동일한 범주에 배치한다. (기본개념)
- 근면한 학습자(eager learner)
  - 훈련 데이터 셋이 주어져 있다면 테스트 데이터 셋을 전달받기 전에 기존 데이터의 일반 모델을 작성하는 부류의 학습자
    - 베이지안 분류, 결정트리, 규칙 기반 분류, SVM
- 게으른 학습자(lazy learner)
  - 훈련 데이터 셋을 전달받으면 우선 쌓아두기만 하고 테스트 데이터 셋이 도달하기를 기다린다. 미지의 데이터 셋이 도착하면 이전의 데이터와 유사성을 통해 테스트 데이터 셋을 분류할 수 있도록 저장해 두었던 훈련 데이터 셋의 일반화를 시작한다.
  - 인스턴스 기반 학습자(instance-based learner)
    - k-최근접 이웃 분류자

최근접 이웃 분류의 이해

- 최근접 이웃 분류기
  - 1950년대 초에 등장. 1960년대 이후 패턴 인식 분야에서 널리 활용됨.
  - 레이블이 없는 예시를 레이블된 유사한 예시의 클래스로 할당해 분류하는 방법
    - 클래스 : 레이블의 범주, 분류에서 예측해야 할 목표 특징
  - 컴퓨터가 현재 상황에 대한 결론을 얻고자 과거의 경험을 되살리는 사람과 같은 회상 능력을 적용(유추의 방식으로 학습하는 알고리즘)

- 사용 예
  - 정지 영상(이미지)과 동영상에서 광학 글자 인식과 얼굴 인식을 포함하는 컴퓨터 비전 응용
  - 개인별 추천 영화 예측
  - 특정 단백질과 질병 발견에 사용 가능한 유전자 데이터의 패턴 인식

k-NN(k-근접 이웃) 알고리즘

- k-NN(k-근접 이웃) 알고리즘
  - 데이터 분류를 돕는 지도 학습 알고리즘의 또 다른 타입.
  - 비분류된(혹은 처음 본) 데이터 점들에 그 데이터와 가장 비슷하게 분류된 데이터 점들(트레이닝 샘플)의 클래스를 할당해 분류하는 과정.
  - 훈련 데이터가 n가지 속성으로 정의될 때
    - 인스턴스 하나는 n차원의 패턴 공간상의 한 점으로 투사할 수 있다.
    - 모든 훈련 데이터를 n차원 패턴 공간 안에 저장한다.
    - 미지의 테스트 데이터 셋이 주어졌을 때, k-최근접 이웃 분류자는 패턴 공간상에서 전달받은 데이터와 가장 가까운 k개의 훈련 데이터를 탐색한다.
    - '가장 가까운 이웃' k개를 찾는다.
    - 패턴 공간에서 '가까움'은 기하 거리(euclidean distance)등의 거리 측정법으로 정의.

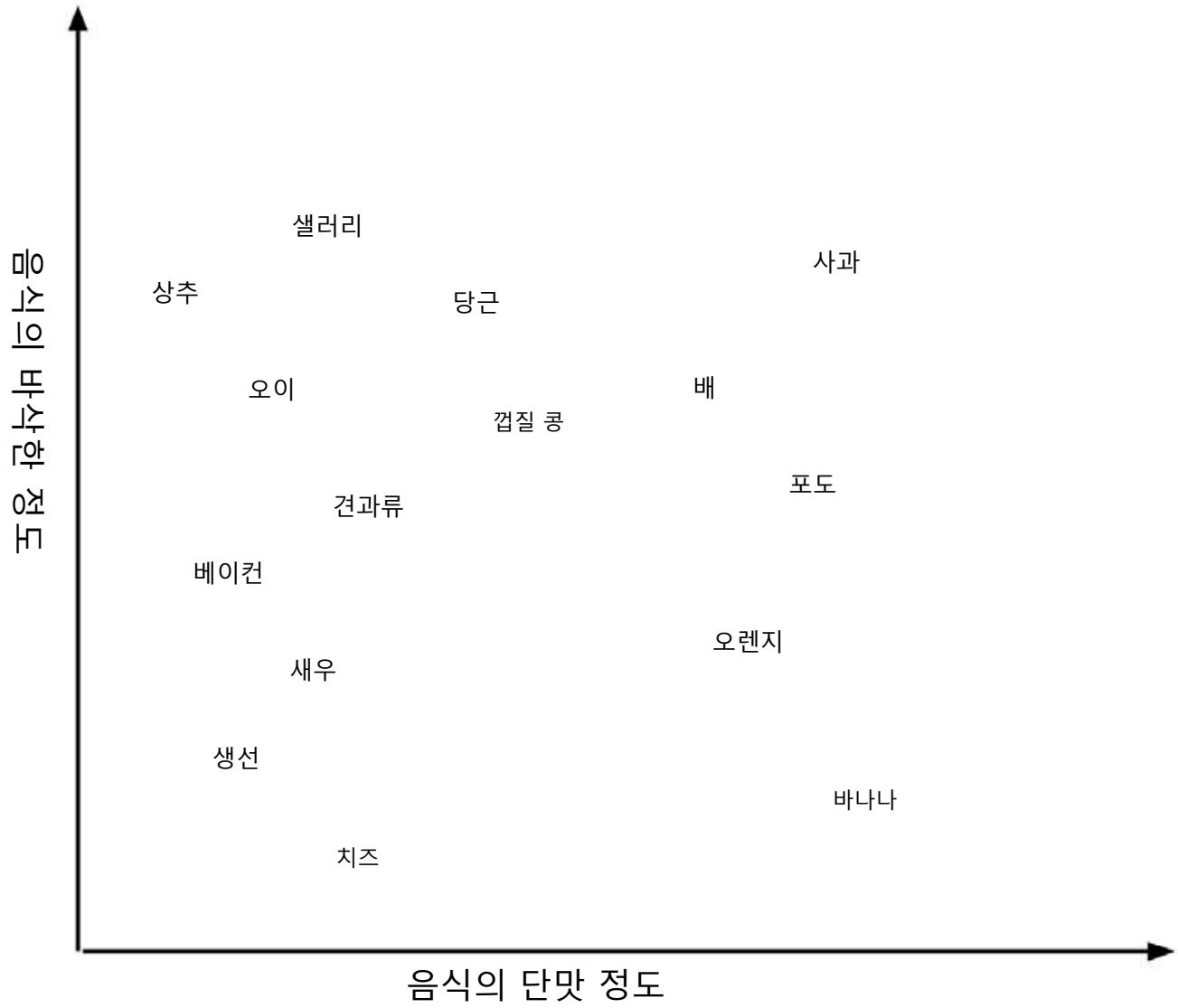
k-NN 알고리즘

- k-최근접 이웃(k-NN, k-nearest neighbors)
  - 장점
    - 단순하고 효율적이다.
    - 기저 데이터 분포에 대한 가정을 하지 않는다.
    - 훈련 단계가 빠르다.
  - 단점
    - 모델을 생성하지 않아 특징과 클래스 간의 관계를 이해하는 능력이 제약을 받는다.
    - 적절한 k의 선택이 필요하다
    - 분류 단계가 느리다.
    - 명목 특징과 누락 데이터용 추가 처리가 필요하다.

k-NN 알고리즘

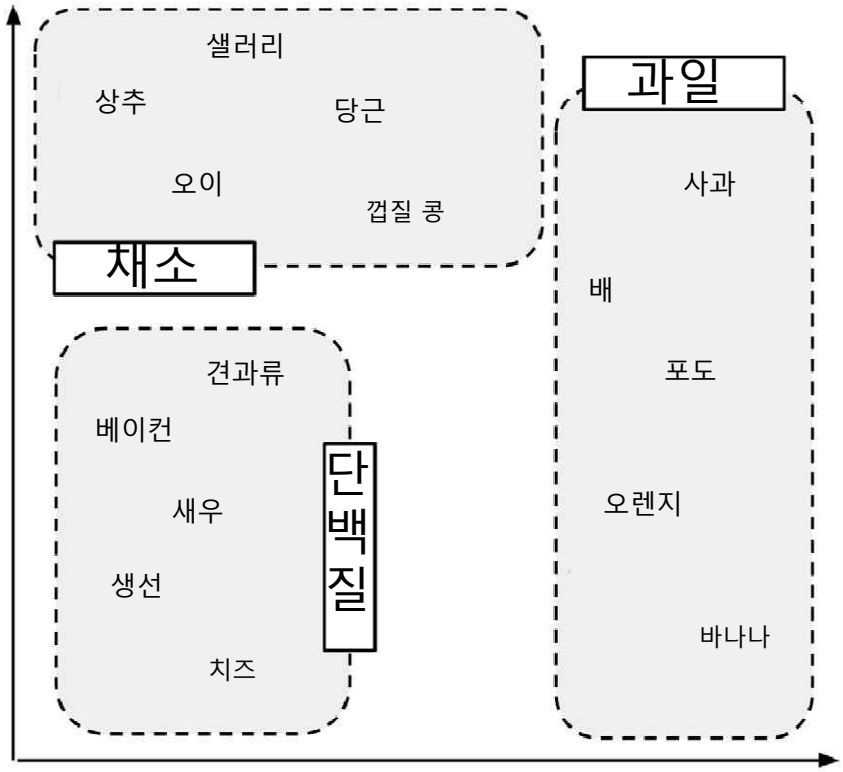
- k : 최근접 이웃의 개수
- 음식 데이터셋 : 2개의 속성(2차원 공간, x축: 단맛, y축: 바삭한 맛)

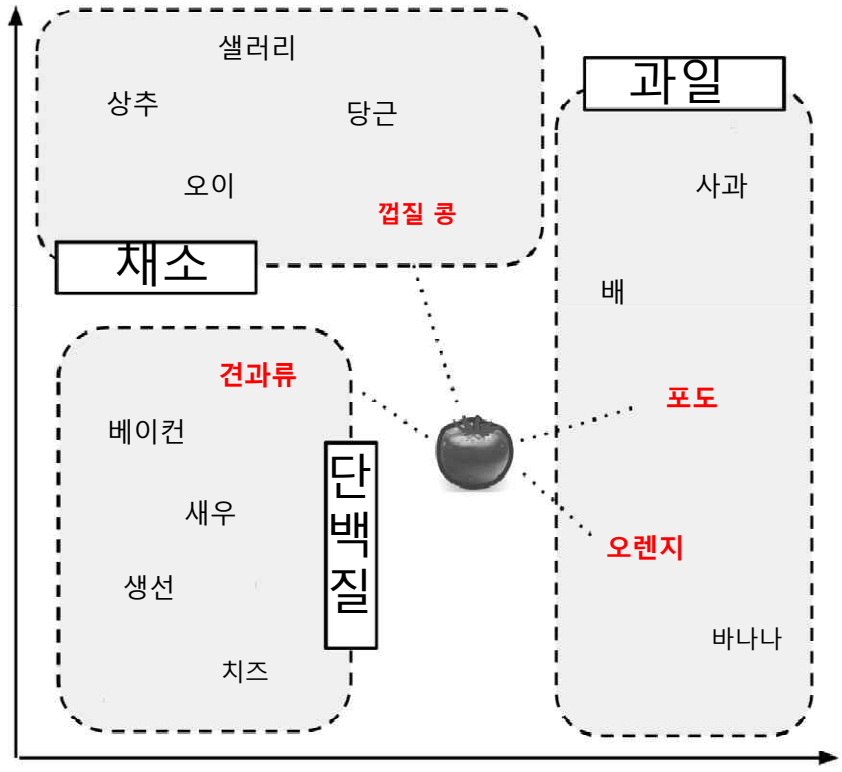
재료	단맛	바삭한 맛	음식 종류
사과	10	9	과일
베이컨	1	4	단백질
바나나	10	1	과일
당근	7	10	채소
샐러리	3	10	채소
치즈	1	1	단백질



선택된 음식의 바삭함과 달콤한 정도를 나타낸 산포도

질문 : 토마토는 과일인가 채소인가?





거리로 유사도 측정

- 토마토의 최근접 이웃을 찾기 위해 거리 함수 또는 두 인스턴스 간의 유사도 측정
- 유클리드 거리
  - 두 점 간의 가장 짧은 직선 거리
  - 토마토와 이웃을 연결하는 점선으로 표시된 것처럼 유클리드 거리는 두 점을 연결하고자 눈금자를 사용할 수 있는 경우에 측정한 거리.

$$\text{dist}(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2}$$

예제) 토마토(단맛=6, 바삭한 맛=4)와 껍질 콩(단맛=3, 바삭한 맛=7)의 거리 계산

$$\text{dist}(\text{tomato}, \text{green bean}) = \sqrt{(6-3)^2 + (4-7)^2} = 4.2$$

거리로 유사도 측정

예제) 토마토(단맛=6, 바삭한 맛=4)와 껍질 콩(단맛=3, 바삭한 맛=7)의 거리 계산

$$\text{dist}\left(\text{tomato}, \text{green bean}\right) = \sqrt{\left(6-3\right)^2 + \left(4-7\right)^2} = 4.2$$

재료	단맛	바삭한 맛	음식 종류	토마토와의 거리
포도	8	5	과일	2.2
껍질 콩	3	7	채소	4.2
견과류	3	6	단백질	3.6
오렌지	7	3	과일	1.4

k=1 (1-NN 알고리즘) ; 오렌지 → 과일로 분류
k=3 (3-NN 알고리즘) ; 오렌지, 포도, 견과류 → 과일이 2개 이므로 과일로 분류

적절한 k 선택

- k-NN에서 k의 개수는 미래 데이터에 대해 일반화되는 능력을 결정

- 편향-분산 tradeoff (또는 딜레마)
  - 지도 학습 알고리즘이 training set의 범위를 넘어 지나치게 일반화 하는 것을 예방하기 위해 두 종류의 오차(편향, 분산)를 최소화할 때 겪는 문제
  - 편향(bias)은 학습 알고리즘에서 잘못된 가정을 했을 때 발생하는 오차. 과소적합(underfitting) 문제 발생.
  - 분산(variance)은 트레이닝 셋에 내재된 작은 변동 때문에 발생하는 오차. 과대적합(overfitting) 문제 발생.
  - k를 큰 값으로 선택하면 노이즈가 많은 데이터로 인한 영향이나 분산은 감소하지만, 작더라도 중요한 패턴을 무시하는 위험을 감수하는 학습자로 편향될 수 있다.

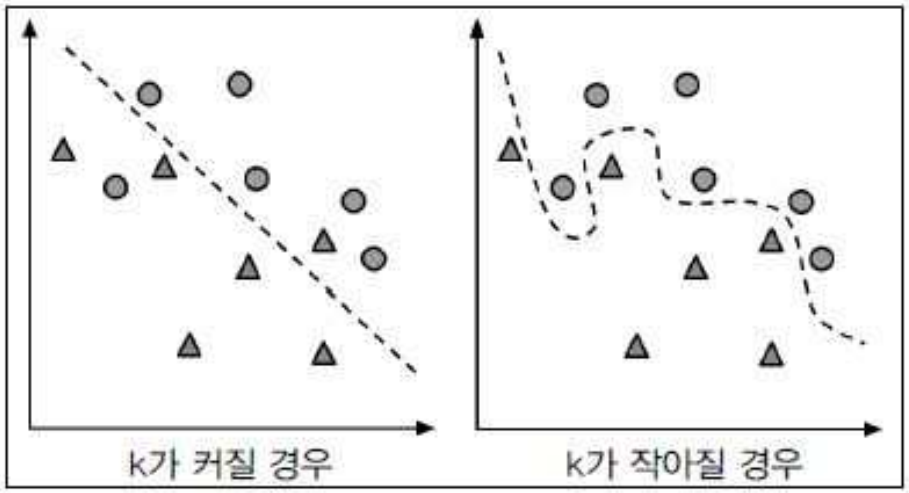


그림 3.4: k가 커지면 편향은 커지고 분산은 줄어든다.

k 값이 작을수록 더 복잡한 결정 경계가 만들어져 훈련 데이터에 세밀하게 맞춰진다.
k는 학습될 개념의 난이도와 훈련 데이터의 레코드 개수에 의존한다.
k = sqrt(훈련 레코드 개수)

데이터에 노이즈가 많지 않고 훈련 데이터셋이 클 경우 k는 크게 중요하지 않다.
k는 실험을 통해 결정한다.
k=1부터 시작해서 테스트 데이터 셋을 통해 분류기의 에러율을 측정.

k-NN 사용을 위한 데이터 준비

속성이 모두 수치형 데이터인 경우,

최소-최대 정규화(min-max normalization)

모든 값이 0에서 1사이 범위에 있도록 특징을 변환.

정규화 공식

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

Z-점수 표준화(z-score standardization)

$$X_{new} = \frac{X - \mu}{\sigma} = \frac{X - \text{Mean}(X)}{\text{StdDev}(X)}$$

k-NN 사용을 위한 데이터 준비

속성이 모두 범주형 데이터인 경우,

더미 코딩(dummy coding)

값 1은 해당 범주를 나타내고, 0은 다른 범주를 나타낸다.

2-범주 이진 지시 변수 :성별 변수의 더미 코딩

$$\text{male} = \begin{cases} 1 & \text{if x} = \text{male} \\ 0 & \text{otherwise} \end{cases}$$

n-범주 명목형 변수: (n-1)레벨의 이진 지시 변수 생성

예 : 온도변수(뜨거움, 보통, 차가움) , 2(=3-1)개의 특징으로 더미 코딩

$$\text{hot} = \begin{cases} 1 & \text{if x} = \text{hot} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{medium} = \begin{cases} 1 & \text{if x} = \text{medium} \\ 0 & \text{otherwise} \end{cases}$$

k-NN 알고리즘으로 유방암 진단

실 습

알고리즘 루틴

1. 데이터 수집
2. 데이터 탐색과 준비
3. 모델 훈련
4. 모델 성능 평가
5. 모델 성능 개선

데이터 수집

실습자료

- k-NN 알고리즘으로 유방암 진단
  - 자료출처 : http://archive.ics.uci.edu/ml에 있는 UCI 머신러닝 리파지토리의 위스콘신 유방암 진단 데이터셋.
  - 목적 : 머신러닝을 통한 암세포 식별
  - 암진단 3단계 [ai.x 2020]
    - 임상진단
    - 영상진단
    - 조직진단(병리학)
      - 현미경을 통한 조직세포 확인
      - 인공지능 딥러닝 기술로 확장

실습자료

- Data 설명
  - Instances : 569개의 암 조직 검사 데이터
  - Variables : 32개의 특징
  - 수치 측정치(세포핵 특성치)
    - 반지름, 질감, 둘레, 넓이, 매끄러움, 조밀성, 오목함, 오목점, 대칭성, 프랙탈 차원 들의 평균, 표준오차, 최대값으로 구성.
  - Target variable : diagnosis(진단)
    - M(Malignant, 악성)
    - B(Benign, 양성)

데이터 탐색 및 준비

```
wbcd <- read.csv("wisc_bc_data.csv", stringsAsFactors = FALSE)
str(wbcd)
wbcd <- wbcd[-1] #id 제외
table(wbcd$diagnosis) # B 양성, M 음성
wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B","M"),
                                          labels = c("Benign", "Malignant"))
round(prop.table(table(wbcd$diagnosis))*100, digits = 1)
summary(wbcd[c("radius_mean","area_mean","smoothness_mean")])

#수치 데이터 정규화
normalize <- function(x){
  return((x-min(x))/(max(x)-min(x)))
}
wbcd_n <- as.data.frame(lapply(wbcd[2:31], normalize))
summary(wbcd_n$area_mean)
```

```
> summary(wbcd[c("radius_mean","area_mean","smoothness_mean")])
  radius_mean       area_mean      smoothness_mean
 Min.   : 6.981   Min.   : 143.5   Min.   :0.05263
 1st Qu.:11.700   1st Qu.: 420.3   1st Qu.:0.08637
 Median :13.370   Median : 551.1   Median :0.09587
 Mean   :14.127   Mean   : 654.9   Mean   :0.09636
 3rd Qu.:15.780   3rd Qu.: 782.7   3rd Qu.:0.10530
 Max.   :28.110   Max.   :2501.0   Max.   :0.16340
> summary(wbcd_n[c("radius_mean","area_mean","smoothness_mean")])
  radius_mean       area_mean      smoothness_mean
 Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
 1st Qu.:0.2233   1st Qu.:0.1174   1st Qu.:0.3046
 Median :0.3024   Median :0.1729   Median :0.3904
 Mean   :0.3382   Mean   :0.2169   Mean   :0.3948
 3rd Qu.:0.4164   3rd Qu.:0.2711   3rd Qu.:0.4755
 Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
>
```

훈련 및 테스트 데이터셋 생성

- Dataset partition
  - Train dataset
    - k-NN모델을 구축하고자 사용되는 훈련 데이터셋
    - 데이터의 70~90%사용,
    - 469개의 인스턴스 사용.
  - Test
    - 모델의 예측 정확도를 평가하고자 사용되는 테스트 데이터셋
    - 데이터의 10~30%사용.
    - 100개의 인스턴스 사용.

```
#훈련 데이터와 테스트 데이터 생성

wbcd_train <- wbcd_n[1:469,]

wbcd_test <- wbcd_n[470:569,]


#knn모델훈련 -> class label벡터생성

wbcd_train_labels <- wbcd[1:469,1]

wbcd_test_labels <- wbcd[470:569,1]
```

모델 훈련

kNN classification syntax

using the `knn()` function in the `class` package

Building the classifier and making predictions:

```
p <- knn(train, test, class, k)
```

- `train` is a data frame containing numeric training data
- `test` is a data frame containing numeric test data
- `class` is a factor vector with the class for each row in the training data
- `k` is an integer indicating the number of nearest neighbors

The function returns a factor vector of predicted classes for each row in the test data frame.

class 패키지의 knn()함수 이용 :

1. 테스트 데이터의 각 인스턴스별로 유클리드 거리를 이용해 k-최근접 이웃을 식별.
2. k는 사용자 지정 숫자.
3. 테스트 인스턴스는 이웃의 다수 클래스로 배정.
4. 동점 표의 경우, 무작위로 선택.

```
#data로 모델훈련

install.packages("class")

library(class)

wbcd_test_pred <- knn(train = wbcd_train,
                      test=wbcd_test,
                      cl=wbcd_train_labels,
                      k=21)
```

knn()함수는 wbcd_test 데이터셋의 각 instance에 대해 예측된 레이블의 팩터 벡터를 출력하며, wbcd_test_pred 에 대입한다.

모델 성능 평가

wbcd_test_pred 벡터에 있는 예측된 클래스가 wbcd_test_labels 벡터에 있는 실제 값과 얼마나 잘 일치하는가를 평가

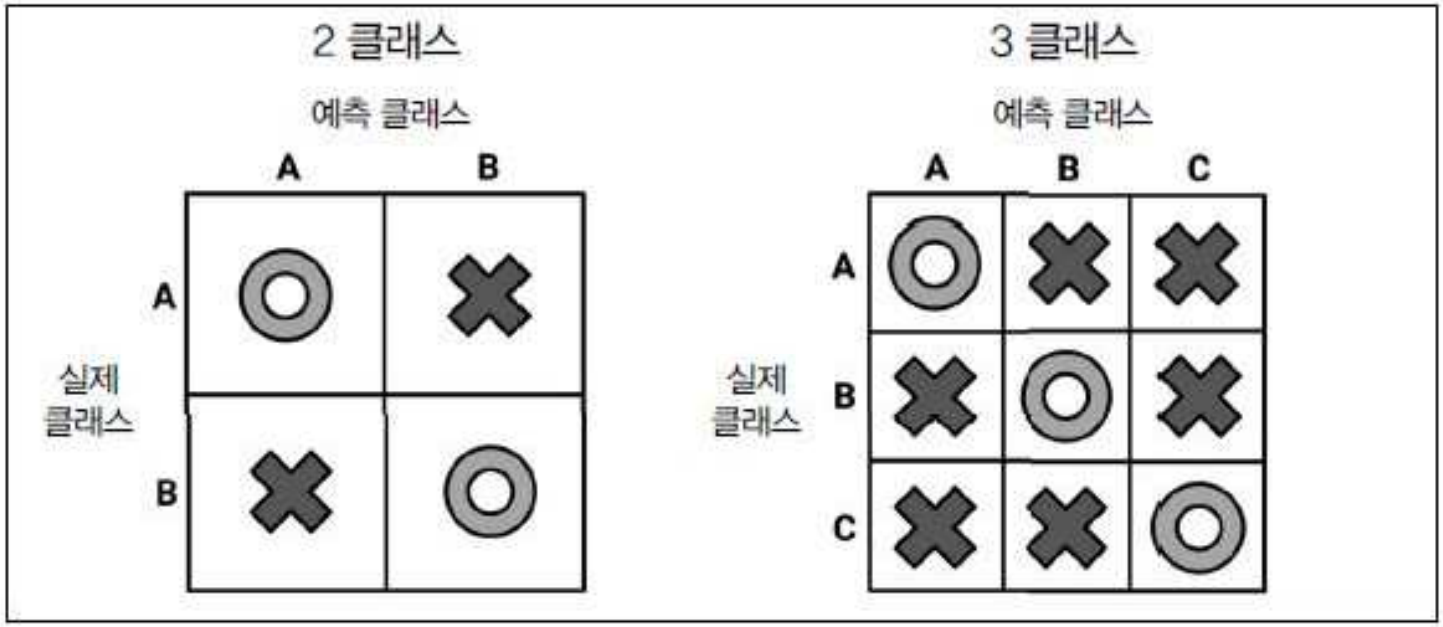
```
#모델 성능 평가

#install.packages("gmodels")

library(gmodels)

CrossTable(x=wbcd_test_labels, y=wbcd_test_pred, chisq = TRUE)
```

wbcd_test_labels	wbcd_test_pred Benign	Malignant	Row Total
Benign	61	0	61
	1.000	0.000	0.610
	0.968	0.000	
	0.610	0.000	
Malignant	2	37	39
	0.051	0.949	0.390
	0.032	1.000	
	0.020	0.370	
Column Total	63	37	100
	0.630	0.370	

오류율 = 2 / 100 = 0.02

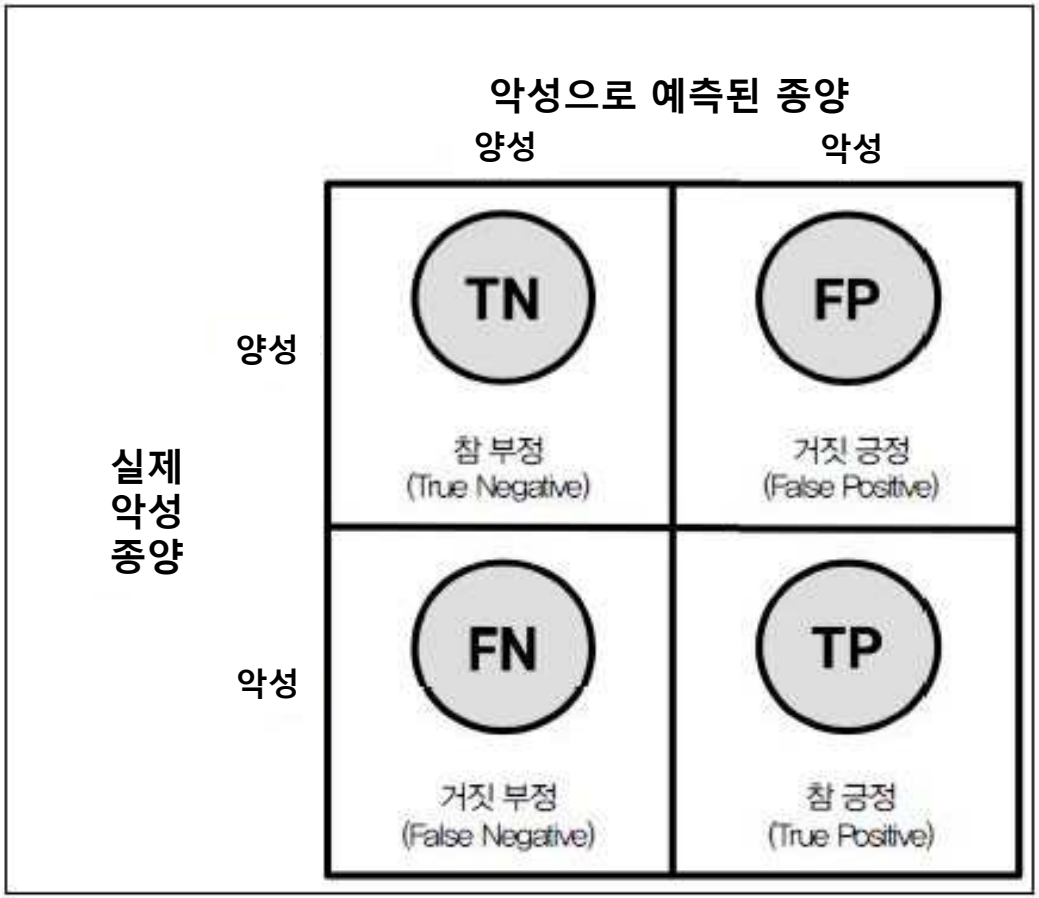
정확도 = 98 / 100 = 0.98

Confusion matrix(혼동행렬)



혼동 행렬은 예측 클래스가 실제 값과 일치하거나 불일치하는 경우의 수를 센다.

Confusion matrix(혼동행렬)



긍정과 부정 클래스 사이를 구분하면 혼동 행렬에 상세 사항이 추가된다.

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{r rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = 1 - \text{accur}$$

모델 성능 개선

모델 성능 개선

- 모델 성능 개선 방향
  - 예측 정확성을 향상.
  - 오류율을 줄임.

- 모델 성능 개선 방법
  - 수치 특성을 재변환
    - Z-점수 표준화
  - K에 몇 가지 다른 값을 시도

```
#모델 성능 개선  # 1. 데이터 표준화 z-score

wbcd_z <- as.data.frame(scale(wbcd[-1]))

summary(wbcd_z$area_mean)

wbcd_z_train <- wbcd_z[1:469,]

wbcd_z_test <- wbcd_z[470:569,]

wbcd_z_train_labels <- wbcd[1:469,1]

wbcd_z_test_labels <- wbcd[470:569,1]

wbcd_z_test_pred <- knn(train = wbcd_z_train, test = wbcd_z_test,

                                cl = wbcd_z_train_labels, k=21)

CrossTable(x=wbcd_z_test_labels, y= wbcd_z_test_pred, prop.chisq = FALSE)
```

	wbcd_test_pred		
wbcd_test_labels	Benign	Malignant	Row Total
Benign	61	0	61
	1.000	0.000	0.610
	0.924	0.000	
	0.610	0.000	
Malignant	5	34	39
	0.128	0.872	0.390
	0.076	1.000	
	0.050	0.340	
Column Total	66	34	100
	0.660	0.340	

```
# 2. k개수 조정
#k=1
wbcd_test_pred3 <- knn(train = wbcd_train, test=wbcd_test,
cl=wbcd_train_labels, k=1)
CrossTable(x=wbcd_test_labels, y=wbcd_test_pred3, chisq = TRUE)
#k=5
wbcd_test_pred3 <- knn(train = wbcd_train, test=wbcd_test,
cl=wbcd_train_labels, k=5)
CrossTable(x=wbcd_test_labels, y=wbcd_test_pred3, chisq = TRUE)
#k=11
wbcd_test_pred3 <- knn(train = wbcd_train, test=wbcd_test,
cl=wbcd_train_labels, k=11)
CrossTable(x=wbcd_test_labels, y=wbcd_test_pred3, chisq = TRUE)
#k=15
wbcd_test_pred3 <- knn(train = wbcd_train, test=wbcd_test,
cl=wbcd_train_labels, k=15)
CrossTable(x=wbcd_test_labels, y=wbcd_test_pred3, chisq = TRUE)
#k=21
wbcd_test_pred3 <- knn(train = wbcd_train, test=wbcd_test,
cl=wbcd_train_labels, k=21)
CrossTable(x=wbcd_test_labels, y=wbcd_test_pred3, chisq = TRUE)
#k=27
wbcd_test_pred3 <- knn(train = wbcd_train, test=wbcd_test,
cl=wbcd_train_labels, k=27)
CrossTable(x=wbcd_test_labels, y=wbcd_test_pred3, chisq = TRUE)
```

k	FN	FP	오류율
1	1	3	4%
5	2	0	2%
11	3	0	3%
15	3	0	3%
21	**2**	**0**	**2%**
27	4	0	4%