

모두를 위한 R 데이터 분석 입문

2판



Chapter 12

군집화와 분류



목차

1. 군집화와 분류의 개요
2. k-평균 군집화
3. k-최근접 이웃 분류
4. k-fold 교차 검증

Section 01

군집화와 분류의 개요

1. 군집화와 분류의 개요

1. 머신러닝의 등장

- 머신러닝은 방대한 데이터를 컴퓨터가 스스로 분석하고 학습하여 유용한 정보를 얻어내거나 미래를 예측하기 위한 예측모델을 만들어내는 기술
- 머신러닝의 대표적 기술 중의 하나는 군집화(clustering)와 분류(classification)

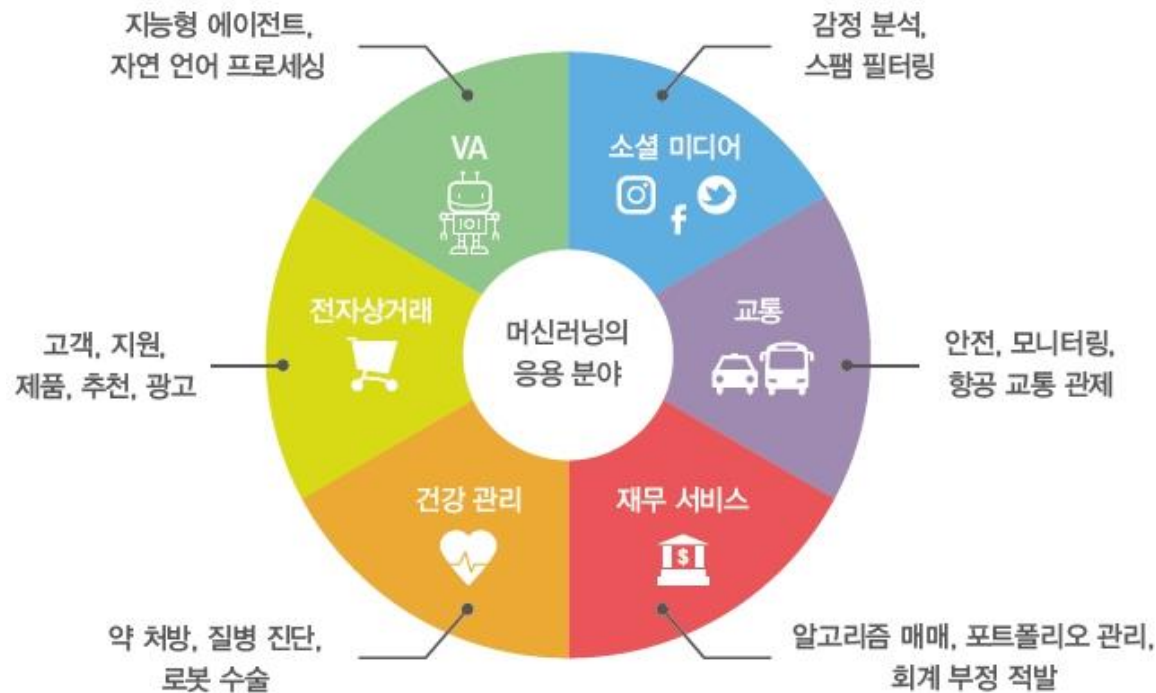


그림 12-1 머신러닝의 응용 분야

1. 군집화와 분류의 개요

2. 군집화와 분류의 개념

- **군집화(clustering):** 주어진 대상 데이터들을 유사성이 높은 것끼리 묶어주는 기술로, 이러한 묶음을 군집, 범주, 그룹 등 다양한 용어로 부름
- **분류(classification):** 그룹의 형태로 알려진 데이터들이 있을 때 그룹을 모르는 어떤 데이터에 대해 어느 그룹에 속하는지를 예측하는 기술

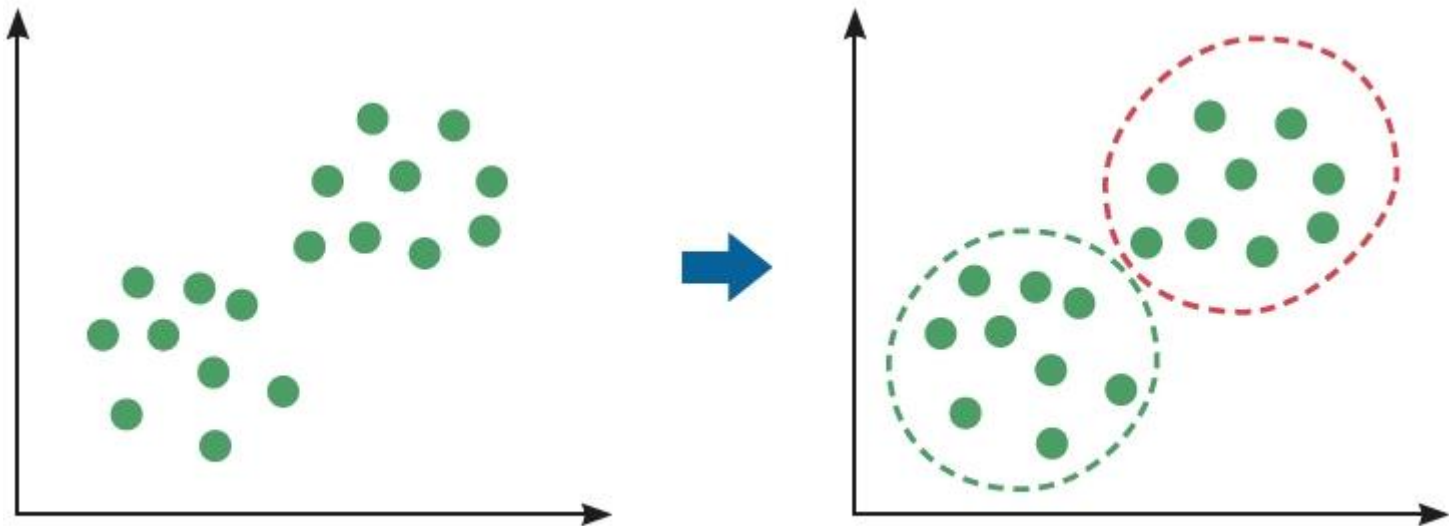


그림 12-2 군집화의 예

1. 군집화와 분류의 개요

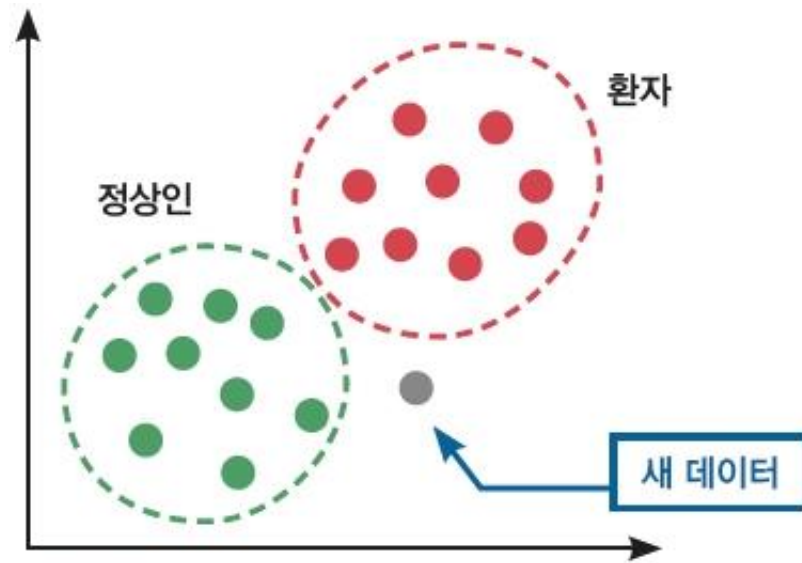


그림 12-3 분류 문제의 예

Section 02

k-평균 군집화

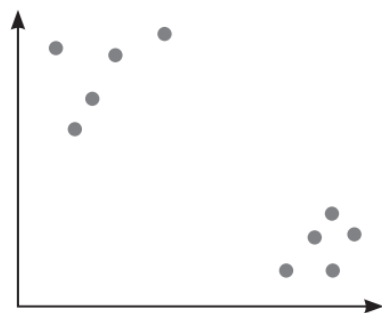
2. k-평균 군집화

1. k-평균 군집화의 과정

- [그림 12-4]는 군집의 개수 k 가 2인 경우를 가정하여 주어진 데이터에서 2개의 군집을 찾는 과정

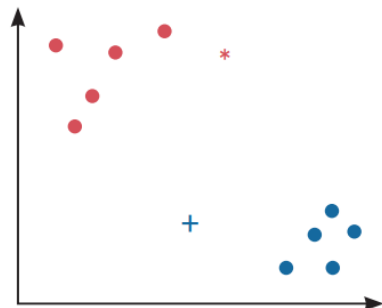
- 1단계: 대상 데이터셋을 준비한다.

이때 산점도 상의
점 하나가
관측값 하나를 의미



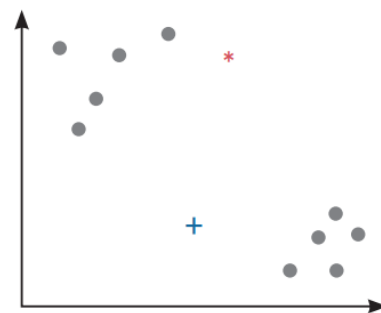
Step. 1

- 3단계: 산점도 상의 점들 하나하나와 임의의 점 2개와의 거리를 계산하여 두 점 중 가까운 쪽으로 군집을 형성. 그 결과 그래프의 왼쪽 위의 점들은 (*) 군집으로, 오른쪽 아래의 점들은 (+) 군집으로 묶임



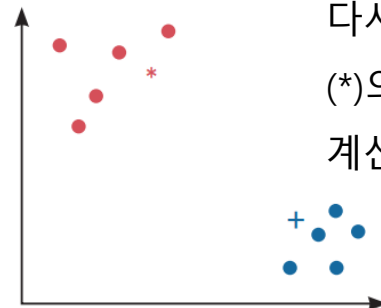
Step. 3

- 2단계: 산점도 상에 임의의 점 2개(*) 와 +)를 만든다. 이 2개의 점은 나중에 군집이 완성되었을 때 각 군집의 중심점이 된다. 따라서 군집의 개수만큼 임의의 점을 생성



Step. 2

- 4단계: 두 개의 군집에서 중심점을 다시 계산(*와 +도 포함하여 계산). (*)의 위치와 (+)의 위치를 새로 계산한 중심점의 위치로 이동



Step. 4

2. k-평균 군집화

- 5단계: 4단계의 과정을 반복
- 6단계: (*)와 (+)의 위치가 더 이상 변동되지 않으면 군집의 중심점에 도달했으므로 반복을 중단

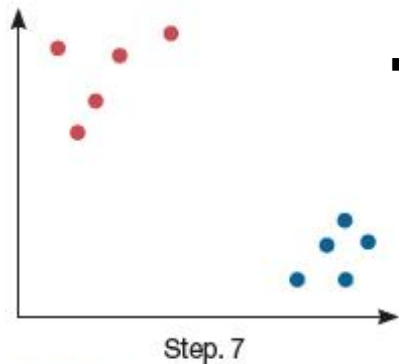
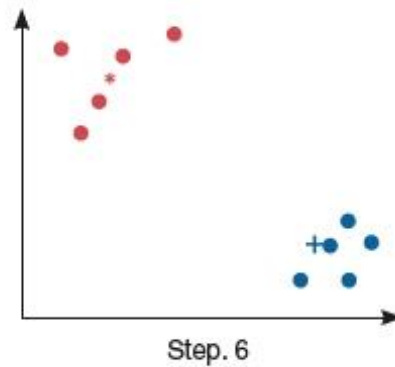
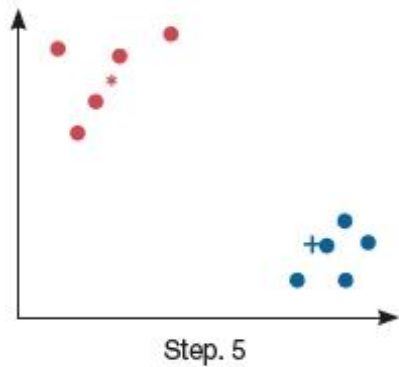


그림 12-4 k-평균 군집화 과정

- 7단계: 마지막으로 (*)와 가까운 점들은 (*) 군집으로, (+)와 가까운 점들은 (+) 군집으로 표시. 군집화를 종료

2. k-평균 군집화

- 1단계: 대상 데이터셋을 준비한다. 이때 산점도 상의 점 하나가 관측값 하나를 의미
- 2단계: 산점도 상에 임의의 점 2개(*) 와 +)를 만든다. 이 2개의 점은 나중에 군집이 완성되었을 때 각 군집의 중심점이 된다. 따라서 군집의 개수만큼 임의의 점을 생성
- 3단계: 산점도 상의 점들 하나하나와 임의의 점 2개와의 거리를 계산하여 두 점 중 가까운 쪽으로 군집을 형성. 그 결과 그래프의 왼쪽 위의 점들은 (*) 군집으로, 오른쪽 아래의 점들은 (+) 군집으로 묶임
- 4단계: 두 개의 군집에서 중심점을 다시 계산(*와 +도 포함하여 계산). (*)의 위치와 (+)의 위치를 새로 계산한 중심점의 위치로 이동
- 5단계: 4단계의 과정을 반복
- 6단계: (*)와 (+)의 위치가 더 이상 변동되지 않으면 군집의 중심점에 도달했으므로 반복을 중단
- 7단계: 마지막으로 (*)와 가까운 점들은 (*) 군집으로, (+)와 가까운 점들은 (+) 군집으로 표시. 군집화를 종료

2. k-평균 군집화

- 즉, k-평균 군집화의 방법을 정리하면 먼저 군집의 중심점을 찾고, 다른 점들은 거리가 가장 가까운 중심점의 군집에 속하는 것으로 결정
- 유클리디안 거리를 이용하면 n차원 상의 점 p, q의 거리는 다음과 같이 계산

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

2. k-평균 군집화

2. R에서의 k-평균 군집화

코드 12-1

```
mydata <- iris[,1:4]                # 데이터 준비

fit <- kmeans(x=mydata, centers=3)
fit
fit$cluster                         # 각 데이터에 대한 군집 번호
fit$centers                        # 각 군집의 중심점 좌표

# 차원 축소 후 군집 시각화
library(cluster)
clusplot(mydata, fit$cluster, color=TRUE, shade=TRUE,
          labels=2, lines=0)

# 데이터에서 두 번째 군집의 데이터만 추출
subset(mydata, fit$cluster==2)
```

2. k-평균 군집화

```
> mydata <- iris[,1:4]           # 데이터 준비  
> fit <- kmeans(x=mydata, centers=3)
```

- **x=mydata**
매개변수 x에는 군집화의 대상이 되는 데이터셋을 지정한다.
- **centers=3**
매개변수 centers에는 군집의 개수를 입력한다.

```
> fit  
K-means clustering with 3 clusters of sizes 62, 38, 50
```

Cluster means:

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width |
|---|--------------|-------------|--------------|-------------|
| 1 | 5.901613 | 2.748387 | 4.393548 | 1.433871 |
| 2 | 6.850000 | 3.073684 | 5.742105 | 2.071053 |
| 3 | 5.006000 | 3.428000 | 1.462000 | 0.246000 |

2. k-평균 군집화

Clustering vector:

```
[1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3  
[34] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1  
[67] 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1  
[100] 1 2 1 2 2 2 2 1 2 2 2 2 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2  
[133] 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2 2 1
```

Within cluster sum of squares by cluster:

```
[1] 39.82097 23.87947 15.15100
(between_SS / total_SS = 88.4 %)
```

Available components:

```
[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

2. k-평균 군집화

[illegible]

그림 12-5 군집화 결과에 대한 주요 정보

[illegible]

K-평균 군집화 결과 해석

```
> mydata <- iris[,1:4]    # 데이터 준비
> fit <- kmeans(x=mydata, centers=3)
> fit
```

K-means clustering with 3 clusters of sizes 38, 62, 50

Cluster means:

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width |
|---|--------------|-------------|--------------|-------------|
| 1 | 6.850000 | 3.073684 | 5.742105 | 2.071053 |
| 2 | 5.901613 | 2.748387 | 4.393548 | 1.433871 |
| 3 | 5.006000 | 3.428000 | 1.462000 | 0.246000 |

Clustering vector:

```
[1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[34] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[67] 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[100] 2 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 1 1 1 1 2 1 2 1 2 1 1 2 2 1 1 1
[133] 1 2 1 1 1 1 2 1 1 1 2 1 1 1 2 1 1 2
```

Within cluster sum of squares by cluster:

```
[1] 23.87947 39.82097 15.15100
(between_SS / total_SS = 88.4 %)
```

클수록 좋음

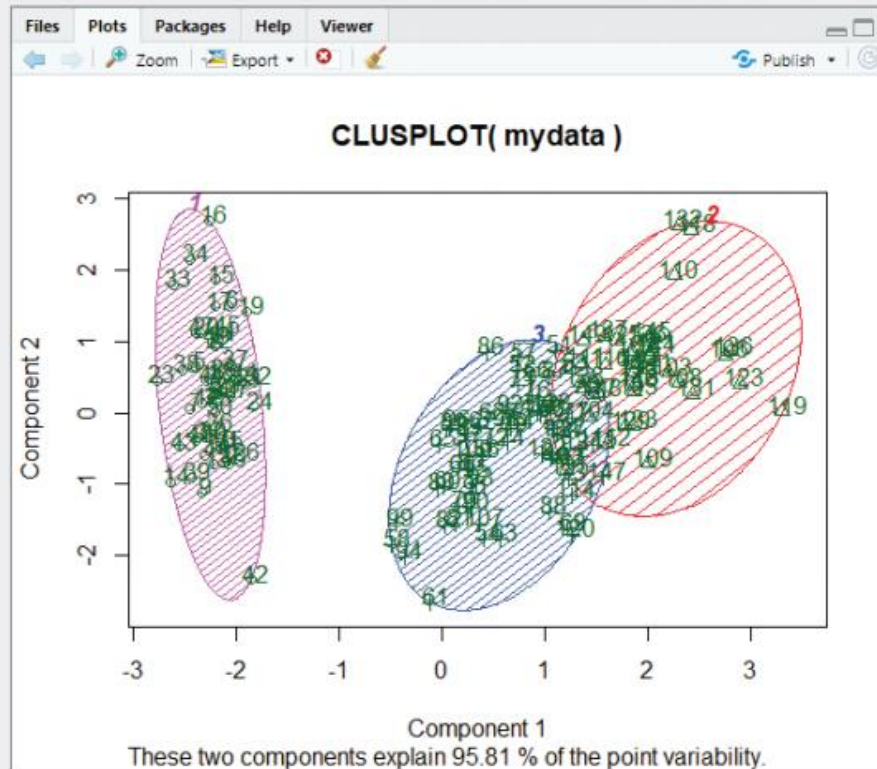
클러스터 내에서 중심과의 차이의
제곱 합: 작을수록 응집도가 높음

Available components:

```
[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

2. k-평균 군집화

```
> # 차원 축소 후 군집 시각화  
> library(cluster)  
> clusplot(mydata, fit$cluster, color=TRUE, shade=TRUE,  
+         labels=2, lines=0)  
>
```



2. k-평균 군집화

- **mydata**

군집화 대상 데이터셋을 지정한다.

- **fit\$cluster**

군집화 결과에서 관측값별 군집 번호이다.

- **color=TRUE**

군집을 표시하는 원의 색깔을 군집별로 다르게 할지 여부를 결정한다.

- **shade=TRUE**

군집을 표시하는 원 안에 빗금을 표시할지 여부를 결정한다.

- **labels=2**

군집화 대상 데이터셋에서 개별 관측값을 그래프상에 어떻게 나타낼지를 지정한다.

- 1: 관측값을 ○, △, +와 같은 기호로 표시
- 2: 관측값을 숫자 번호로 표시

- **lines=0**

군집의 중심점과 중심점을 연결하는 선을 표시할지 여부를 결정한다.

- 0: 표시하지 않음
- 1: 표시함

2. k-평균 군집화

```
> # 데이터에서 두 번째 군집의 데이터만 추출
```

```
> subset(mydata, fit$cluster==2)
```

| | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width |
|----|--------------|-------------|--------------|-------------|
| 51 | 7.0 | 3.2 | 4.7 | 1.4 |
| 52 | 6.4 | 3.2 | 4.5 | 1.5 |
| 54 | 5.5 | 2.3 | 4.0 | 1.3 |
| 55 | 6.5 | 2.8 | 4.6 | 1.5 |
| 56 | 5.7 | 2.8 | 4.5 | 1.3 |
| 57 | 6.3 | 3.3 | 4.7 | 1.6 |

...(이하 생략)

2. k-평균 군집화

3. 대상 데이터 표준화 후 군집화

- 데이터와 데이터의 거리를 계산할 때 발생하는 문제의 예

표 12-1 A, B 학생의 키와 시력

| 학생 | 키 | 시력 |
|----|-----|-----|
| A | 180 | 1.2 |
| B | 170 | 0.9 |

- A와 B 거리의 계산 값

$$\begin{aligned}d(A, B) &= \sqrt{(180 - 170)^2 + (1.2 - 0.9)^2} \\&= \sqrt{(10)^2 + (0.3)^2} \\&= \sqrt{100 + 0.09}\end{aligned}$$

2. k-평균 군집화

- 한계점: 거리 계산에 있어서 키의 값은 많이 반영되는데(100), 시력은 거리 계산에 있어서 거의 영향을 미치지 못함(0.09)
- 즉, 자료의 범위가 큰 변수가 거리 계산에 있어서 더 많은 영향을 미칠 수밖에 없다는 의미
- 분석자들은 모든 변수가 거리 계산에 동등한 영향을 갖도록 하기 위해서 모든 변수의 자료 범위를 0~1 사이로 표준화한 후에 거리 계산
- 변수 A의 값들을 0~1 사이로 표준화하는 공식

$$(x - \min(A)) / (\max(A) - \min(A))$$

- 여기서 x는 변수 A의 임의의 관측값으로, max(A), min(A)는 변수 A의 관측값 중 최댓값과 최솟값

코드 12-2

```
std <- function(X) {                                # 표준화 함수
  return((X-min(X)) /(max(X)-min(X)))
}

mydata <- apply(iris[,1:4], 2, std)                  # 표준화된 데이터 준비

fit <- kmeans(x=mydata, centers=3)
fit
```

Section 03

k-최근접 이웃 분류

3. k-최근접 이웃 분류

1. 분류 문제의 사례

- 분류는 그룹이 있는 데이터에 대해 그룹을 모르는 데이터가 들어왔을 때 어떤 그룹에 속하는지를 예측하는 기술

| No | 운동 시간 | 일하는 시간 | 진단 결과 |
|----|-------|--------|---------|
| 1 | 0.27 | 0.65 | patient |
| 2 | 0.34 | 0.68 | patient |
| 3 | 0.46 | 0.95 | patient |
| 4 | 0.37 | 0.75 | patient |
| 5 | 0.48 | 0.75 | patient |
| 6 | 0.36 | 0.86 | patient |
| 7 | 0.51 | 0.98 | patient |
| 8 | 0.43 | 0.91 | patient |
| 9 | 0.28 | 0.78 | patient |
| 10 | 0.46 | 0.86 | patient |
| 11 | 0.74 | 0.51 | normal |
| 12 | 0.67 | 0.46 | normal |
| 13 | 0.56 | 0.43 | normal |
| 14 | 0.67 | 0.34 | normal |
| 15 | 0.81 | 0.56 | normal |
| 16 | 0.81 | 0.43 | normal |
| 17 | 0.76 | 0.35 | normal |
| 18 | 0.65 | 0.42 | normal |
| 19 | 0.78 | 0.23 | normal |
| 20 | 0.88 | 0.26 | normal |

| 운동 시간 | 일하는 시간 |
|-------|--------|
| 0.45 | 0.61 |

환자 또는 정상?

그림 12-6 분류 문제의 예

3. k-최근접 이웃 분류

2. k-최근접 이웃 분류의 방법

- k-최근접 이웃 분류의 방법
- 1단계: 그룹을 모르는 데이터 P에 대해 이미 그룹이 알려진 데이터 중 P와 가장 가까이 있는 k 개의 데이터를 수집
- 2단계: k개의 데이터가 가장 많이 속해 있는 군집을 P의 군집으로 정함

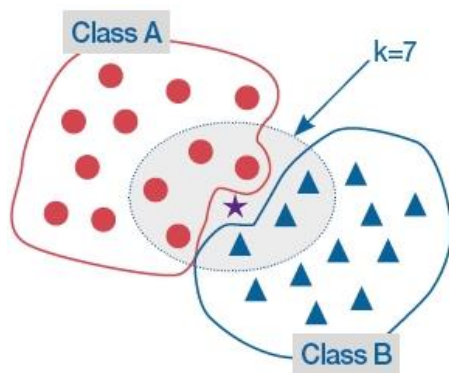


그림 12-7 k-최근접 이웃 알고리즘의 원리(k=7)

- k-최근접 이웃 알고리즘에서 k=7이라고 했을 때, 별점과 가장 가까이 있는 7개의 점들을 찾아보면 회색 타원 안에 있는 점
- 이때 7개의 점 중에서 4개는 Class A에 속하고 3개는 Class B에 속하는데, 다수결에 의해 이웃의 점 4개가 속해 있는 Class A를 별점의 그룹이라고 예측

3. k-최근접 이웃 분류

3. R에서의 k-최근접 이웃 분류

코드 12-3

```
library(class)

# 훈련용 데이터와 테스트용 데이터 준비
tr.idx <- c(1:25, 51:75, 101:125)      # 훈련용 데이터의 인덱스
ds.tr <- iris[tr.idx, 1:4]              # 훈련용 데이터셋
ds.ts <- iris[-tr.idx, 1:4]            # 테스트용 데이터셋
cl.tr <- factor(iris[tr.idx, 5])        # 훈련용 데이터셋의 그룹(품종) 정보
cl.ts <- factor(iris[-tr.idx, 5])      # 테스트용 데이터셋의 그룹(품종) 정보

pred <- knn(ds.tr, ds.ts, cl.tr, k=3, prob=TRUE)
Pred

acc <- mean(pred==cl.ts)                # 예측 정확도
Acc

table(pred, cl.ts)                      # 예측값과 실제값 비교 통계
```

3. k-최근접 이웃 분류

```
> library(class)
> # 훈련용 데이터와 테스트용 데이터 준비
> tr.idx <- c(1:25,51:75, 101:125)      # 훈련용 데이터의 인덱스
> ds.tr <- iris[tr.idx, 1:4]            # 훈련용 데이터셋
> ds.ts <- iris[-tr.idx, 1:4]           # 테스트용 데이터셋
> cl.tr <- factor(iris[tr.idx, 5])       # 훈련용 데이터셋의 그룹(품종) 정보
> cl.ts <- factor(iris[-tr.idx, 5])      # 테스트용 데이터셋의 그룹(품종) 정보
> pred <- knn(ds.tr, ds.ts, cl.tr, k=3, prob=TRUE)
```

3. k-최근접 이웃 분류

- **ds.tr**

훈련용 데이터셋을 지정한다.

- **ds.ts**

테스트용 데이터셋을 지정한다.

- **cl.tr**

훈련용 데이터셋의 그룹 정보를 지정한다.

- **k=3**

최근접 이웃의 개수를 지정한다.

- **prob=TRUE**

예측된 그룹에 대한 지지 확률을 표시할지 여부를 결정한다. 위와 같이 $k=3$ 인 경우 최근접 이웃이 Class A가 2개, Class B가 1개라면 예측된 그룹은 Class A이고 Class A에 대한 지지 확률은 $2/3(=0.6666)$ 이다.

3. k-최근접 이웃 분류

```
> pred
```

```
[1] setosa      setosa      setosa      setosa      setosa      setosa
```

```
[7] setosa      setosa      setosa      setosa      setosa      setosa
```

```
[13] setosa      setosa      setosa      setosa      setosa      setosa
```

```
... (중간 생략)
```

```
[61] virginica   virginica   virginica   versicolor virginica   virginica
```

```
[67] virginica   virginica   virginica   virginica   virginica   virginica
```

```
[73] virginica   virginica   virginica
```

```
attr(,"prob")
```

```
[1] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

```
[7] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

```
[13] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
```

```
... (중간 생략)
```

```
[67] 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.6666667
```

```
[73] 1.0000000 1.0000000 0.6666667
```

```
Levels: setosa versicolor virginica
```

3. k-최근접 이웃 분류

```
> acc <- mean(pred==cl.ts)           # 예측 정확도
> acc
[1] 0.9333333
> table(pred,cl.ts)                  # 예측값과 실제값 비교 통계
```

| | cl.ts | | |
|------------|--------|------------|-----------|
| pred | setosa | versicolor | virginica |
| setosa | 25 | 0 | 0 |
| versicolor | 0 | 23 | 3 |
| virginica | 0 | 2 | 22 |

여기서 잠깐! k-최근접 이웃 분류의 궁금한 점

1. k 값은 어떻게 정하는가?

데이터셋의 관측값의 개수(행의 수)가 100이라면 k는 10보다 작은 것이 좋음
보통은 1~7의 값을 차례로 실험해보고, 예측모델의 정확도 가 가장 높게 나오는 k를 선택

2. 그룹을 예측할 때 다수결에서 동수가 나오면 어떻게 하는가?

knn() 함수에서는 둘 중 하나를 임의로 선택

3. k-최근접 이웃 알고리즘의 단점은 무엇인가?

k개의 최근접 이웃을 알아내기 위해서는 그룹을 모르는 데이터 p와 그룹 정보가 알려진
훈련용 데이터셋의 모든 데이터들과 거리 계산해야 함

Section 04

K-fold 교차 검증

4. k-fold 교차 검증

1. k-fold 교차 검증의 방법

- 예측모델을 개발하려면 그룹 정보가 포함된 데이터셋이 확보되어야 하는데, 데이터셋에 포함된 관측값의 개수는 많으면 많을수록 좋음
- 확보된 데이터셋은 훈련용 데이터와 테스트용 데이터로 나눌 수 있음
- 훈련용 데이터를 이용하여 분류(예측)모델을 개발하고, 테스트용 데이터를 이용하여 예측을 실시
- 머신러닝의 목표 중 하나는 예측 정확도가 높은 모델을 만드는 것

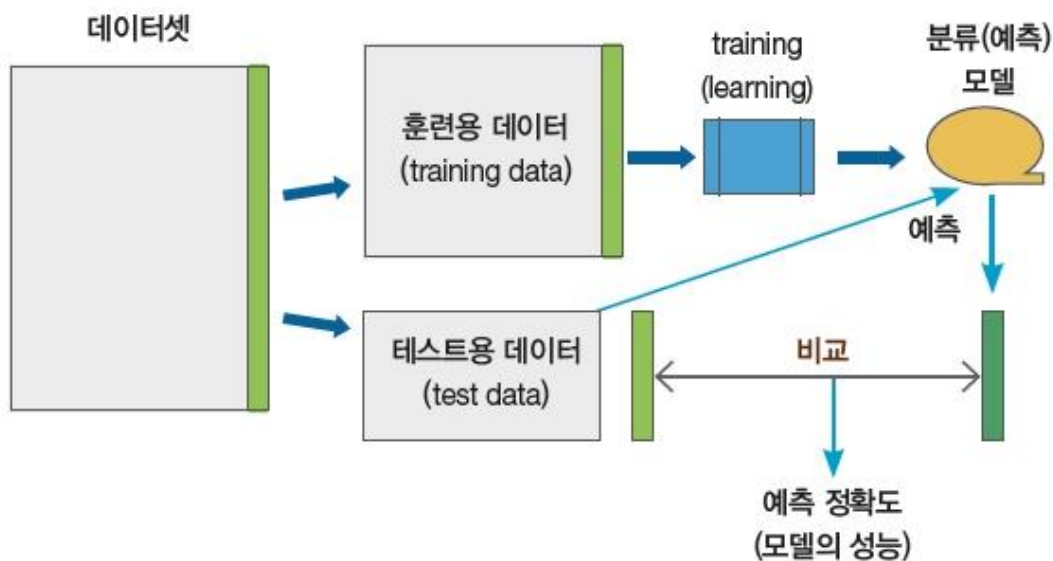


그림 12-8 머신러닝에서 예측모델의 개발 절차

4. k-fold 교차 검증

1. k-fold 교차 검증의 방법

- 단, 훈련용과 테스트용으로 데이터를 나눌 때 데이터가 어떻게 나누어졌는가에 따라 모델의 성능이 달라지는 문제가 있음
- 이 문제를 해결하는 방법은 데이터를 임의로 훈련용&테스트용으로 나누어 모델을 개발하는 과정을 여러 번 반복하여 그곳에서 도출되는 예측 정확도의 평균을 구하는 것
- 이것을 체계화한 방법론이 k-fold 교차 검증(k-fold cross validation)



그림 12-9 k-fold 교차 검증(k=4)

4. k-fold 교차 검증

2. R에서의 k-fold 교차 검증

코드 12-4

```
library(cvTools)                                # cvFolds() 함수 지원

k = 10                                           # 10-fold
folds <- cvFolds(nrow(iris), K=k)              # 폴드 생성

acc <- c()                                       # 폴드별 예측 정확도 저장용 벡터
for (i in 1:k) {
  ts.idx <- folds$which == i                    # 테스트용 데이터의 인덱스
  ds.tr <- iris[-ts.idx, 1:4]                  # 훈련용 데이터셋
  ds.ts <- iris[ts.idx, 1:4]                   # 테스트용 데이터셋
  cl.tr <- factor(iris[-ts.idx, 5])            # 훈련용 데이터셋의 그룹(품종) 정보
  cl.ts <- factor(iris[ts.idx, 5])             # 테스트용 데이터셋의 그룹(품종) 정보

  pred <- knn(ds.tr, ds.ts, cl.tr, k = 5)
  acc[i] <- mean(pred==cl.ts)                  # 예측 정확도
}
acc                                              # 폴드별 예측 정확도
mean(acc)                                       # 폴드평균 예측 정확도
```

4. k-fold 교차 검증

```
> library(cvTools)                # cvFolds() 함수 지원
> k = 10                           # 10-fold
> folds <- cvFolds(nrow(iris), K=k) # 폴드 생성
> acc <- c()                       # 폴드별 예측정확도 저장용 벡터
> for (i in 1:k) {
>   ts.idx <- folds$which == i      # 테스트용 데이터의 인덱스
>   ds.tr <- iris[-ts.idx, 1:4]     # 훈련용 데이터셋
>   ds.ts <- iris[ts.idx, 1:4]      # 테스트용 데이터셋
>   cl.tr <- factor(iris[-ts.idx, 5]) # 훈련용 데이터셋의 그룹(품종) 정보
>   cl.ts <- factor(iris[ts.idx, 5]) # 테스트용 데이터셋의 그룹(품종) 정보
>   pred <- knn(ds.tr, ds.ts, cl.tr, k = 5)
>   acc[i] <- mean(pred==cl.ts)     # 예측 정확도
> }
> acc                              # 폴드별 예측 정확도
[1] 0.9333333 1.0000000 0.9333333 0.9333333 1.0000000 1.0000000
[7] 0.9333333 1.0000000 1.0000000 0.9333333
> mean(acc)                        # 폴드평균 예측 정확도
[1] 0.9666667
```

Thank you!