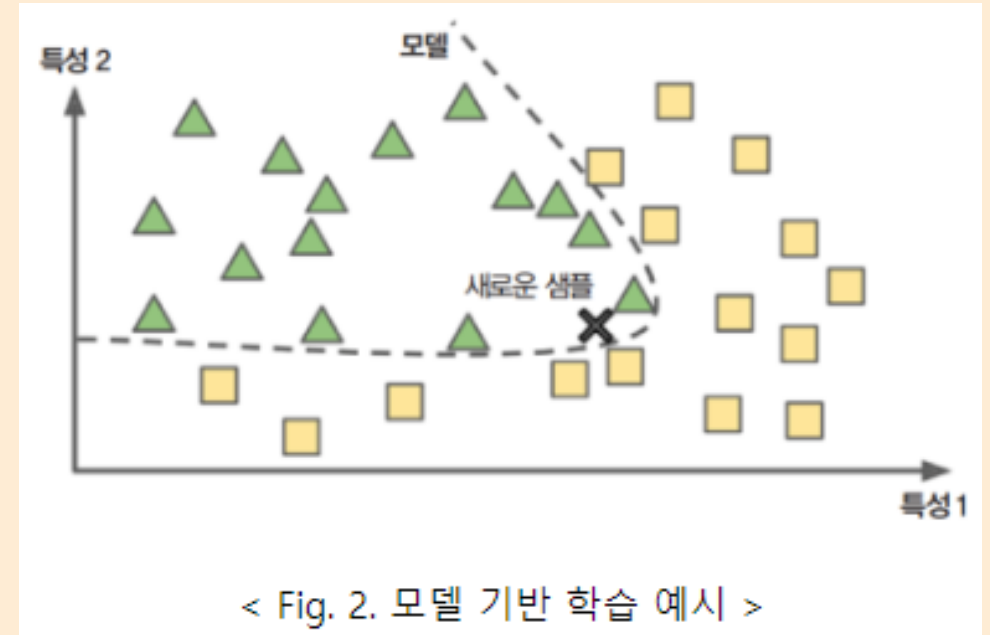




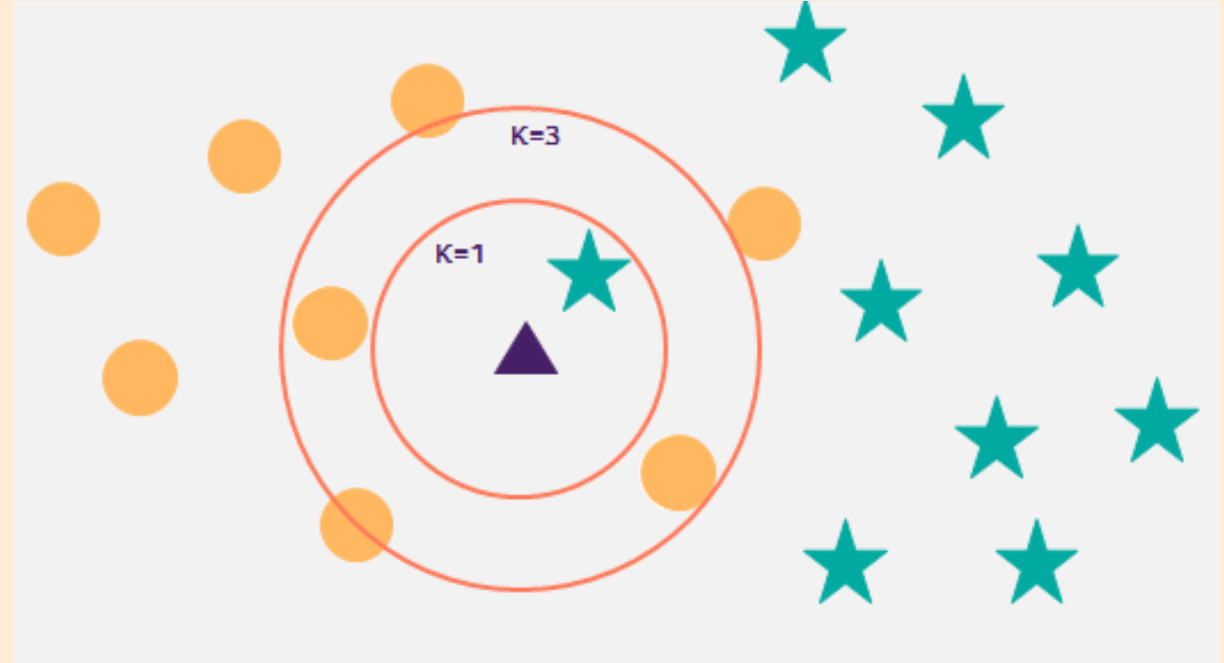
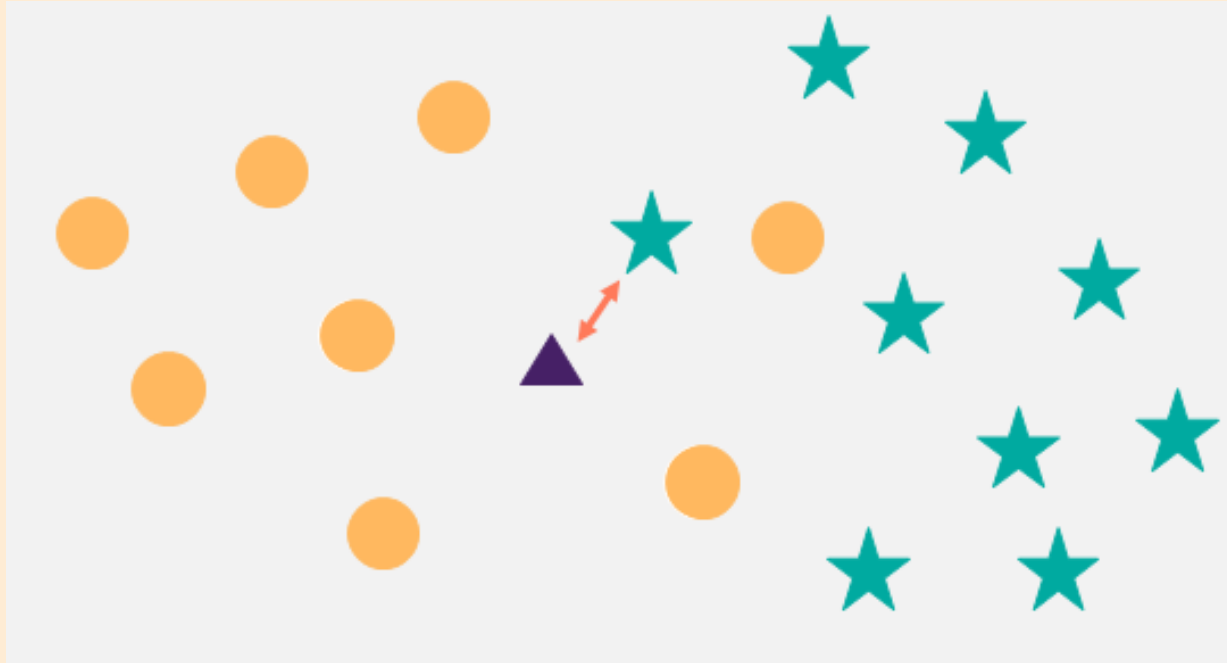
# K-NN & K-means

문혜현, 오민석, 이지선

# 사례 기반 학습? 모델 기반 학습?



# KNN이란?



1. New data가 기존 데이터들과의 거리 측정
2. K 기준 인접한 데이터 개수 확인
3. K개수 기준 다수결의 원칙에 따른 데이터 분류



# 알고리즘 순서

1. 정규화
2. K기반 데이터간의 거리 계산
3. 정답 레이블 선정



# 1. 정규화

도시	인구(명)	미세먼지농도( $\mu\text{g}/\text{m}^3$ )
서울	1000만	200
시애틀	67만	40

도시별 정보를 모아서 유사한 환경을 지닌 도시를 뽑는 문제. 위 표 기준으로는 거리/유사성 측정시 미세먼지농도 정보는 전혀 반영이 되지 않을 것이다. 인구 변수에 해당하는 숫자가 훨씬 크기 때문이다. 보통 min-max scale(최대값 1, 최소값 0)을 사용한다.



## Data 가져오기

```
wbcd <- read.csv('C:/Users/palt1/Desktop/비타민 발표/wbcd.csv')
str(wbcd)
library(ggplot2)
library(caret)
library(dplyr)
wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"),
                        labels = c("Benign", "Malignant"))
```

```
> str(wbcd)
'data.frame': 569 obs. of 31 variables:
 $ diagnosis      : Factor w/ 2 levels "B","M": 2 2 2 1 1 2 2 1 1
 ..
 $ radius_mean    : num  19.53 14.6 20.59 12.81 8.57 ...
 $ texture_mean   : num  18.9 23.3 21.2 13.1 13.1 ...
 $ perimeter_mean : num  129.5 94 137.8 81.3 54.5 ...
 $ area_mean      : num  1217 665 1320 509 221 ...
 $ smoothness_mean : num  0.115 0.0868 0.1085 0.0874 0.1036 ...
 $ compactness_mean : num  0.1642 0.0664 0.1644 0.0377 0.0763 ...
 $ concavity_mean  : num  0.2197 0.0839 0.2188 0.00919 0.02565 ...
 $ concave_points_mean : num  0.1062 0.0527 0.1121 0.0133 0.0151 ...
 $ symmetry_mean   : num  0.179 0.163 0.185 0.147 0.168 ...
 $ fractal_dimension_mean : num  0.0655 0.0542 0.0622 0.0613 0.0713 ...
 $ radius_se      : num  1.111 0.416 0.59 0.289 0.127 ...
```

데이터 구조 확인

## 어떠한 Data?

- 위스콘신 대학교에서 제공한 유방암 진단결과 데이터
- 주로 머신러닝 knn 등 분류 모델의 예제로 자주 사용
- 간단한 칼럼에 대한 설명

diagnosis	양성 여부 (M = 악성, B = 양성)
각 세포에 대한 정보들	
radius	반경 (중심에서 외벽까지 거리들의 평균값)
texture	질감 (Gray-Scale 값들의 표준편차) #gray-scale 값은 광도의 정보를 전달할 수
perimeter	둘레
area	면적
smoothness	매끄러움(반경길이의 국소적 변화)
compactness	조그만 정도(둘레 <sup>2</sup> /면적 - 1)
concavity	오목함(윤곽의 오목한 부분의 정도)
points	오목한 점의 수
symmetry	대칭
dimension	프랙탈 차원(해안선근사 -1)
_mean	3 ~ 12 번까지는 평균값을 의미합니다.
_se	13 ~ 22 번까지는 표준오차(Standard Error) 를 의미합니다.
_worst	23 ~ 32 번까지는 각 세포별 구분들에서 제일 큰 3개의 값을 평균낸 값입니다.

# 정규화

```
#정규화|
normalize <- function(x){
  return ((x-min(x)) / (max(x)-min(x)))
}
normalize(c(1,2,3,4,5))

wbcd_n <- as.data.frame(lapply(select(wbcd,-diagnosis), normalize))
```

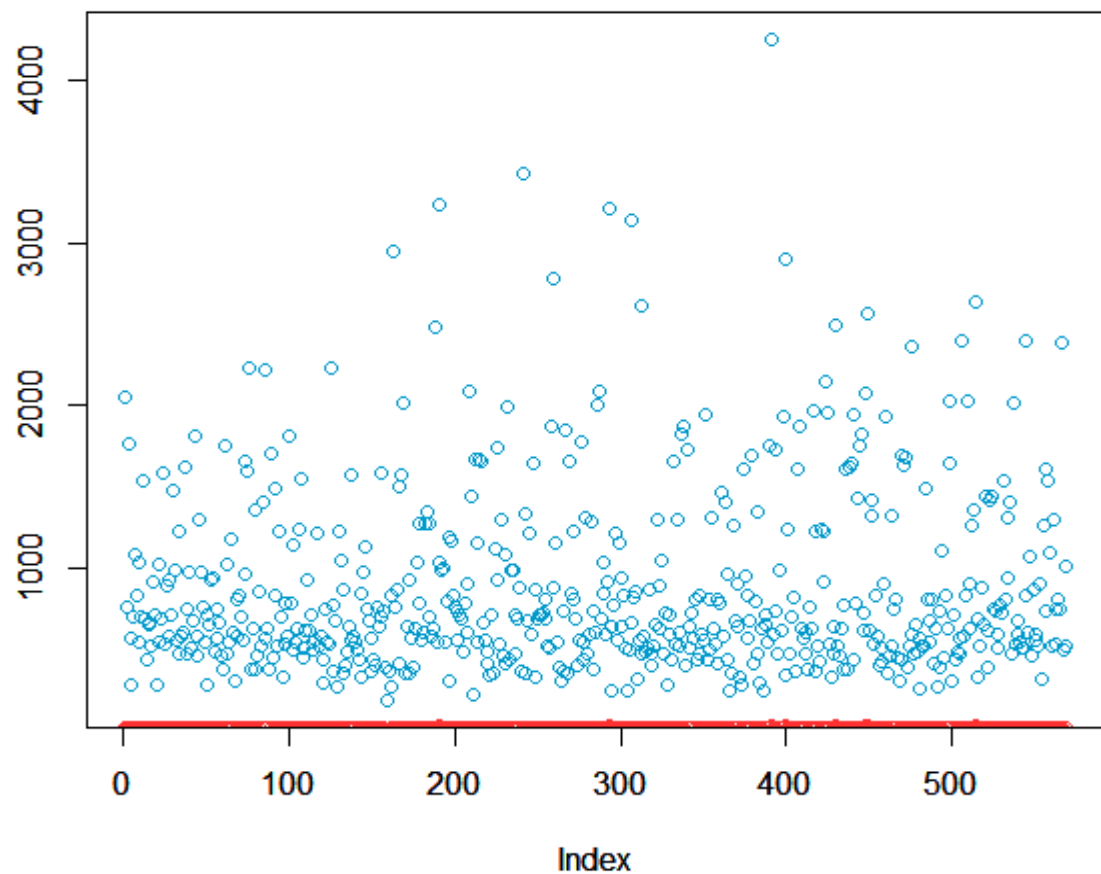




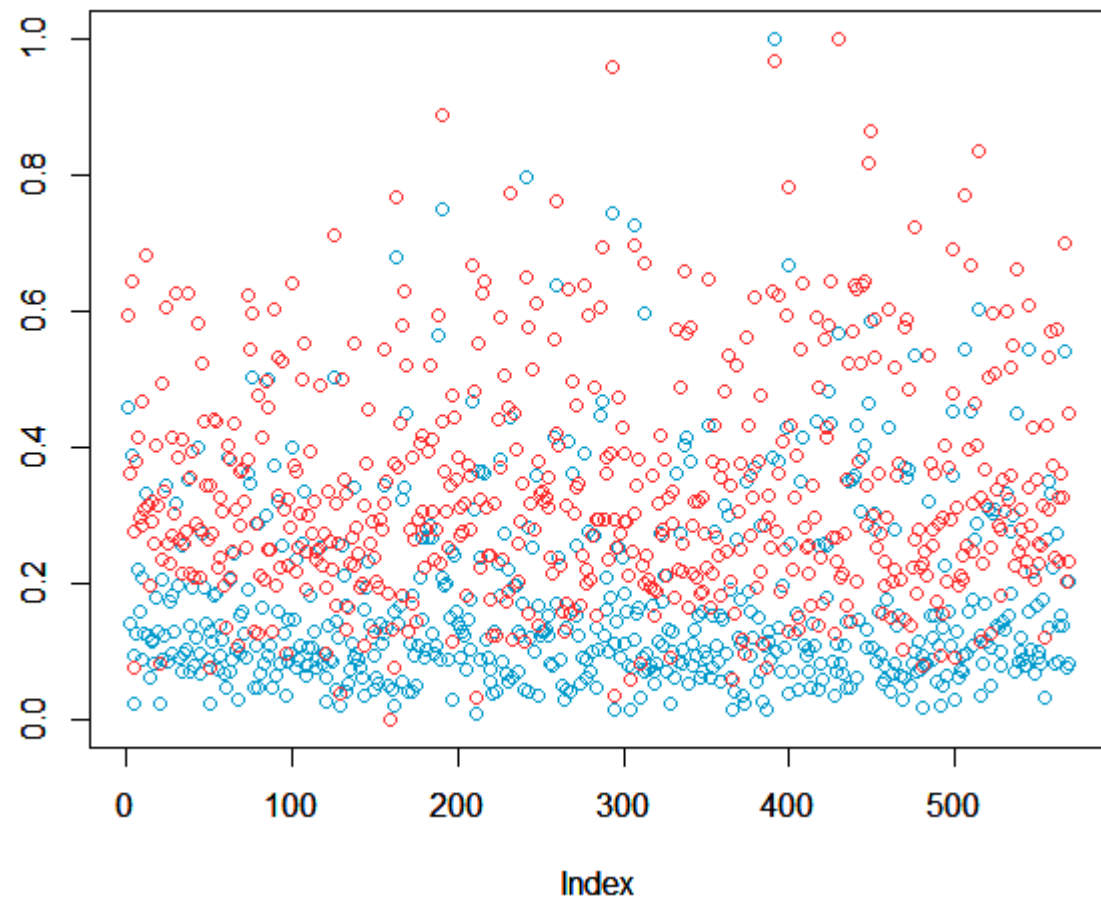
# 정규화

산점도를 통해 칼럼 별 단위가 다르다는 것을 확인  
후 정규화 통해 단위 통일화

```
plot(wbcd$area_worst,col="#009ACD",ylim=range(wbcd$area_worst))  
par(new=T)  
plot(wbcd$radius_mean,col="#FF3030", ylim=range(wbcd$area_worst))
```



```
plot(wbcd_n$area_worst,col="#009ACD",ylim=range(wbcd_n$area_worst))  
par(new=T)  
plot(wbcd_n$radius_mean,col="#FF3030", ylim=range(wbcd_n$area_worst))
```



# 정규화

정규화 전 / 후 비교하기

perimeter_worst	area_worst	smoothness_worst	compactness_worst
171.10	2053.0	0.14950	0.41160
102.20	758.2	0.13120	0.15810
163.20	1760.0	0.14640	0.35970
86.70	570.7	0.11620	0.05445
63.30	275.6	0.16410	0.22350
95.54	698.8	0.09387	0.05131

perimeter_worst	area_worst	smoothness_worst	compactness_worst
0.1888540	0.09199273	0.4756653	0.1455405
0.5617312	0.38704286	0.4967972	0.3225058
0.2688879	0.15859713	0.2326487	0.1108071
0.3734748	0.15913783	0.4670805	0.6613985
0.2733702	0.12799843	0.7034934	0.4919036
0.3196872	0.18184723	0.6183055	0.3049451



센터링 / 표준화 / 정규화

데이터 집합의  
평균을 0으로

센터링

$$x_{new} = \frac{x - \mu}{\sigma}$$

표준화

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

정규화



# Train / Test 나누기

Train, test set 나누기

검증데이터의 target 종류의  
비율이 원본 데이터와 같게 유지

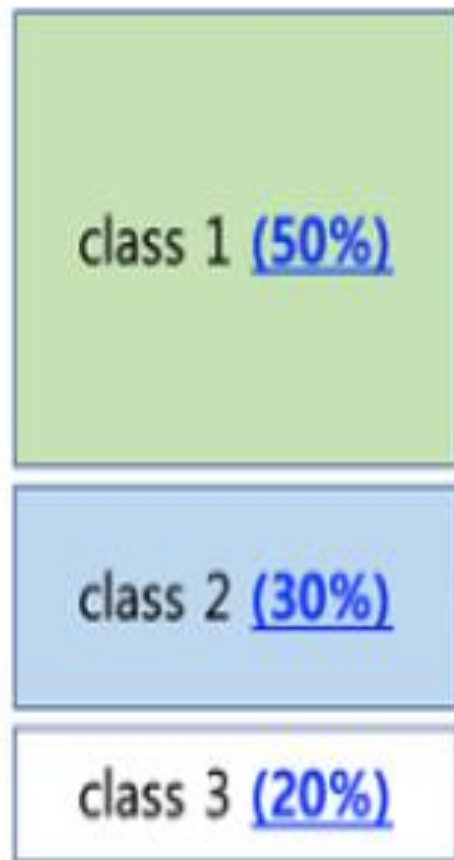
```
train_index <- createDataPartition(wbcd$diagnosis, p=0.7, list=FALSE)
wbcd_train <- wbcd[train_index,]
wbcd_test <- wbcd[-train_index,]
```

```
caret::createDataPartition(
  y,          # 분류(또는 레이블)
  times=1,    # 생성할 분할의 수
  p=0.5,      # 훈련 데이터에서 사용할 데이터의 비율
  list=TRUE,  # 결과를 리스트로 반환할지 여부. FALSE면 행렬을 반환한다.
)
```

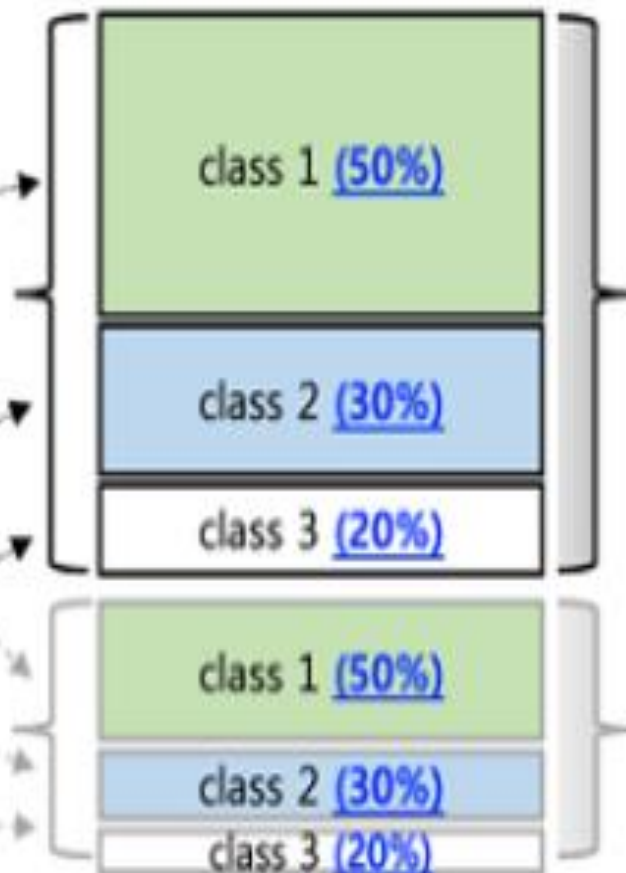


## 각 층의 비율을 고려한 set분할

Raw data(100%)



Train(70%), Test(30%) set



Training set (70%)  
→ *model training*

Test set (30%)  
→ *model evaluation*

## 종속label 비율 확인

```
# 비율 확인
frqtab <- function(x, caption) {
  round(100*prop.table(table(x)), 1)
}

ft_orig <- frqtab(wbcd$diagnosis)
ft_train <- frqtab(wbcd_train$diagnosis)
ft_test <- frqtab(wbcd_test$diagnosis)
ftcmp_df <- as.data.frame(cbind(ft_orig, ft_train, ft_test))
```

```
> ftcmp_df
```

	ft_orig	ft_train	ft_test
Benign	62.7	62.7	62.9
Malignant	37.3	37.3	37.1



## 2. K기반 거리계산 - 거리

### 2) 할당

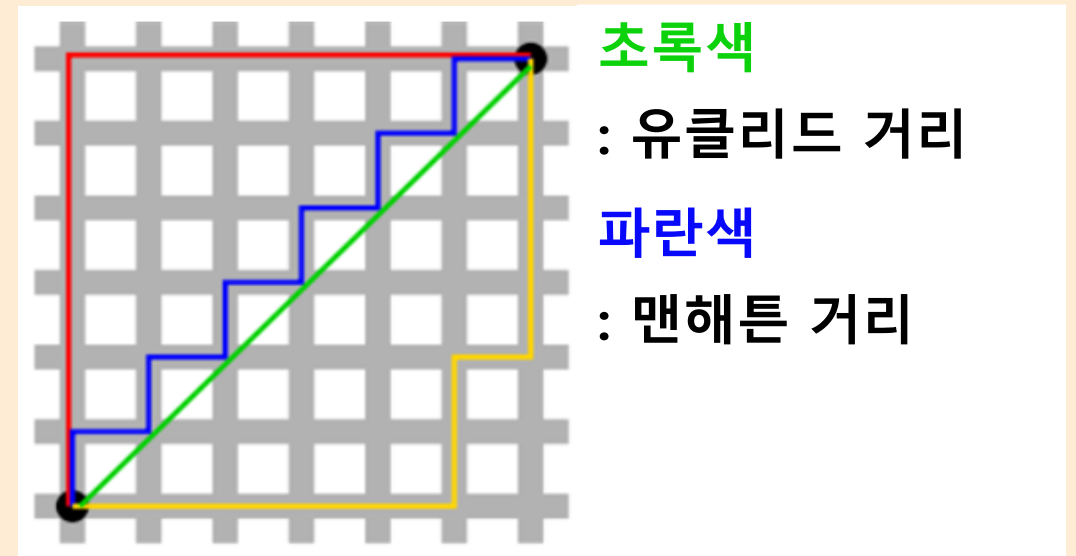
: 각 데이터는 **거리 함수**에 따라 가장 근접한 클러스터 중심에 할당됨

① **유클리드 거리** (전통 방식) → 각 속성들 간의 차이를 모두 고려함

$$\text{dist}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (n = \text{데이터 개수})$$

② **맨해튼 거리**

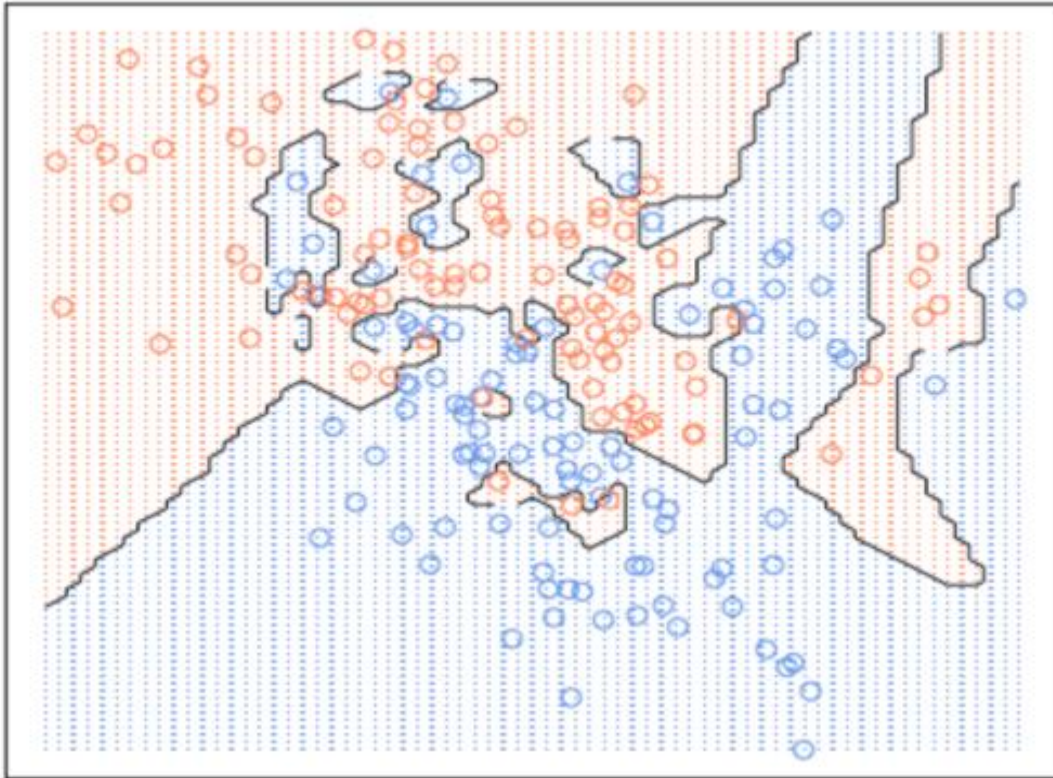
$$L_1 = |x_1 - x_2| + |y_1 - y_2| = \sum_{i=1}^n |a_i - b_i|$$





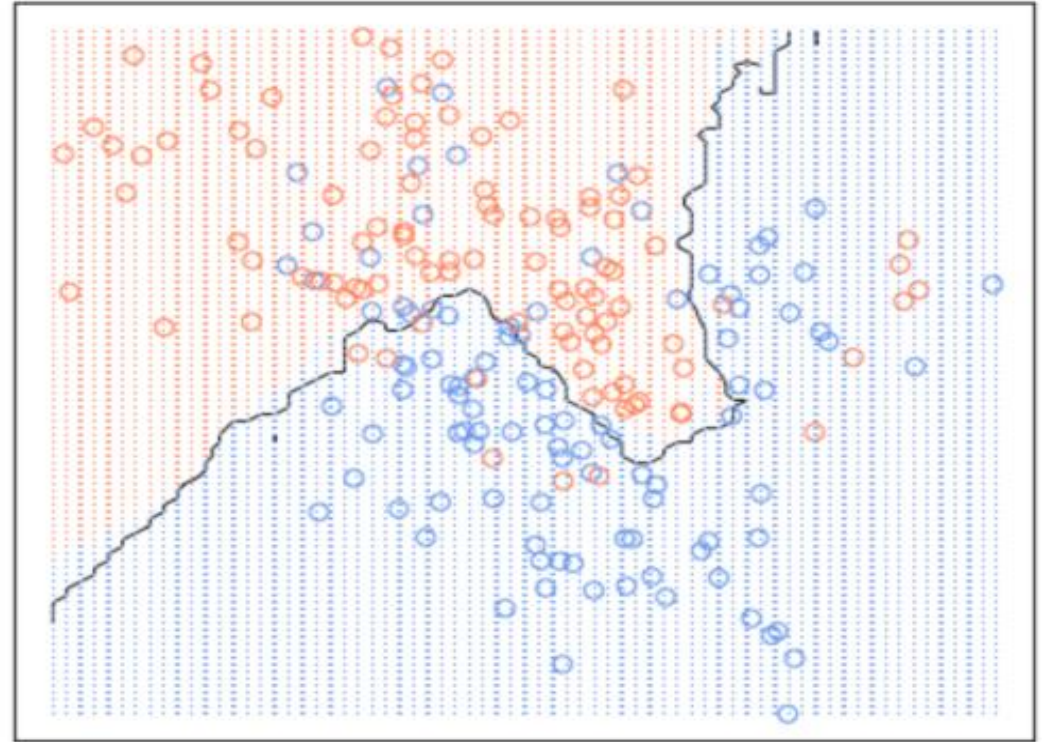
## 2. K기반 거리계산 - K

nearest neighbour ( $k = 1$ )



- k가 작다면?  
데이터 하나하나 고려하기 때문에  
Overfitting

20-nearest neighbour



- k가 크다면?  
세세한 부분에 신경을 안써서  
Underfitting





# Caret 패키지

```
train(  
  X,  
  y,  
  method = "rf",  
  preProcess = NULL,  
  ...,  
  weights = NULL,  
  metric = ifelse(is.factor(y), "Accuracy", "RMSE"),  
  maximize = ifelse(metric %in% c("RMSE", "logLoss", "MAE"), FALSE, TRUE),  
  trControl = trainControl(),  
  tuneGrid = NULL,  
  tuneLength = ifelse(trControl$method == "none", 1, 3)  
)
```

# method : 모델명

# preprocess : 전처리, center, scale, pca 등

# weights : 데이터 가중치

# metric : 평가 메트릭, 회귀  
시 RMSE 자동지정

# trControl : 훈련 파라미터

# tuneLength : 평가된 모델들  
확인 개수 지정



# Caret 패키지

```
trainControl(  
  method = "boot",  
  number = ifelse(grepl("cv", method), 10, 25),  
  repeats = ifelse(grepl("[d_]cv$", method), 1, NA),  
  p = 0.75,
```

# method : 데이터 샘플링 기법으로 boot,  
cv(교차검증),  
repeatedcv(교차검증 반복)

# number : 교차검증 몇 개로 나눌것인지,  
부트스트래핑 몇 회할 것인지

# repeat : 데이터 샘플링 반복 횟수

# p : training 비율



# Caret 패키지

```
confusionMatrix(data, ...)

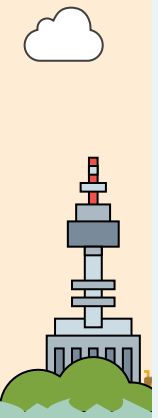
# S3 method for default
confusionMatrix(
  data,
  reference,
  positive = NULL,
  dnn = c("Prediction", "Reference"),
  prevalence = NULL,
  mode = "sens_spec",
  ...
)
```

# Data : predict한 모델

# Reference : 실제 값

# Positive를 어떤 label로  
할 것인지

# Mode : 어떠한 평가 척  
도를 쓸 것인지  
default : sens\_spec



### 3. 정답 레이블 선정

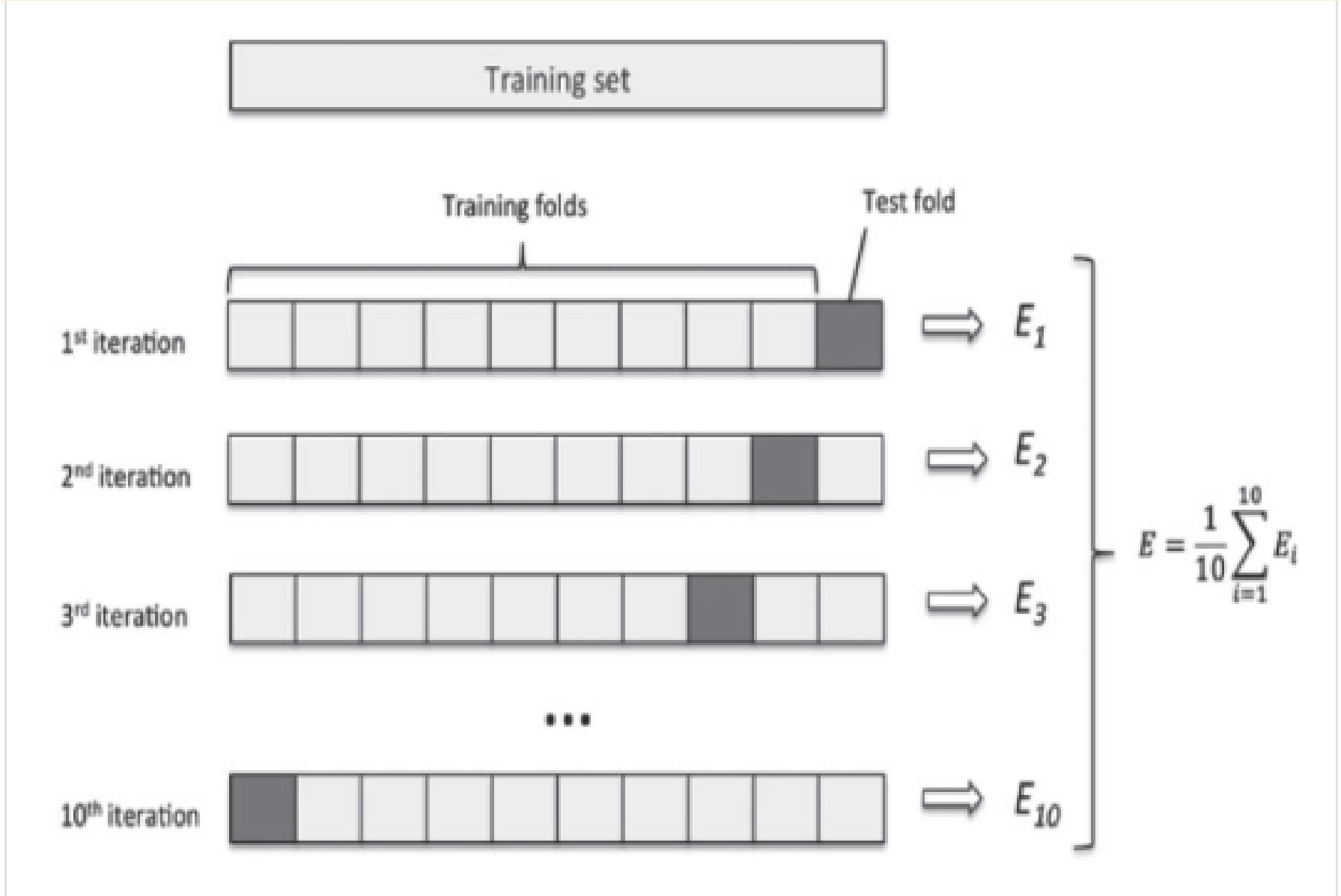
- Caret 패키지의 traincontrol, train이용한 모델 학습

```
# k-fold교차검증
# number : number만큼 쪼개겠다.
# repeat : repeat 수만큼 반복
ctrl <- trainControl(method="repeatedcv", number=10, repeats=2)

#data shuffle
set.seed(12345)

#caret에서 제공하는 preProcess : center(평균을 0), scale(표준화), range(정규화)
# 회기일 경우 metric : RMSE
knnFit1 <- train(diagnosis ~ ., data=wbcd_train, method="knn",
                 trControl=ctrl, metric="Accuracy", tuneLength=10,
                 #preProc=c('range'))
)
knnFit1
```

# 교차검증이란?



## 결과값

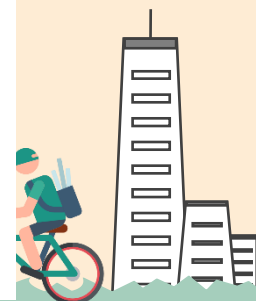
```
> knnFit1
k-Nearest Neighbors

399 samples
 30 predictor
 2 classes: 'Benign', 'Malignant'

Pre-processing: re-scaling to [0, 1] (30)
Resampling: Cross-Validated (10 fold, repeated 2 times)
Summary of sample sizes: 359, 359, 359, 359, 359, 359, ...
Resampling results across tuning parameters:
```

k	Accuracy	Kappa
5	0.9662179	0.9260709
7	0.9636859	0.9203128
9	0.9611859	0.9149104
11	0.9624359	0.9176872
13	0.9574359	0.9064967
15	0.9561538	0.9036794
17	0.9574038	0.9062013
19	0.9561538	0.9033474
21	0.9523397	0.8950564
23	0.9535897	0.8976870

Accuracy was used to select the optimal model using the largest value.  
The final value used for the model was k = 5.



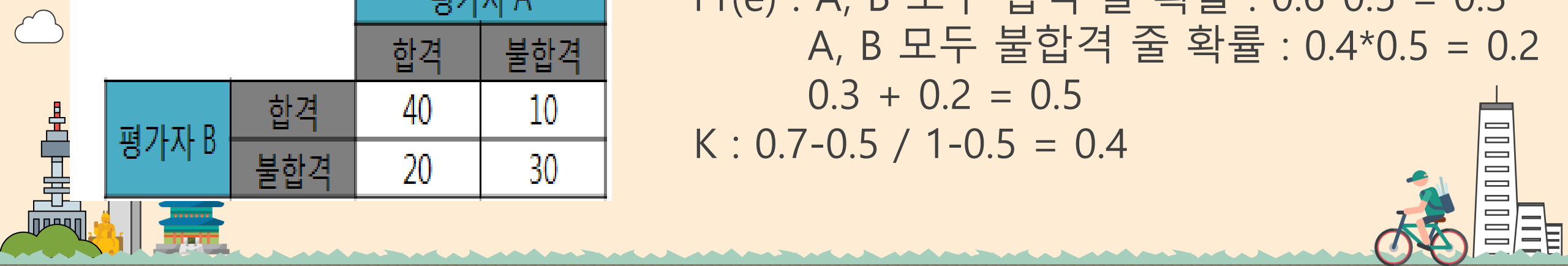
## Kappa 값이란?

두 평가자의 평가가 얼마나 일치하는지 0~1사이의 값을 갖는다.

$$\kappa = \frac{Pr(a) - Pr(e)}{1 - Pr(e)}$$

Pr(a) : 두 평가자의 일치도

Pr(e) : 평가자들이 확률적(우연히)으로  
일치할 확률



		평가자 A	
		합격	불합격
평가자 B	합격	40	10
	불합격	20	30

$$Pr(a) : 40 + 30 / 100 = 0.7$$

$$Pr(e) : A, B \text{ 모두 합격 줄 확률} : 0.6 * 0.5 = 0.3$$

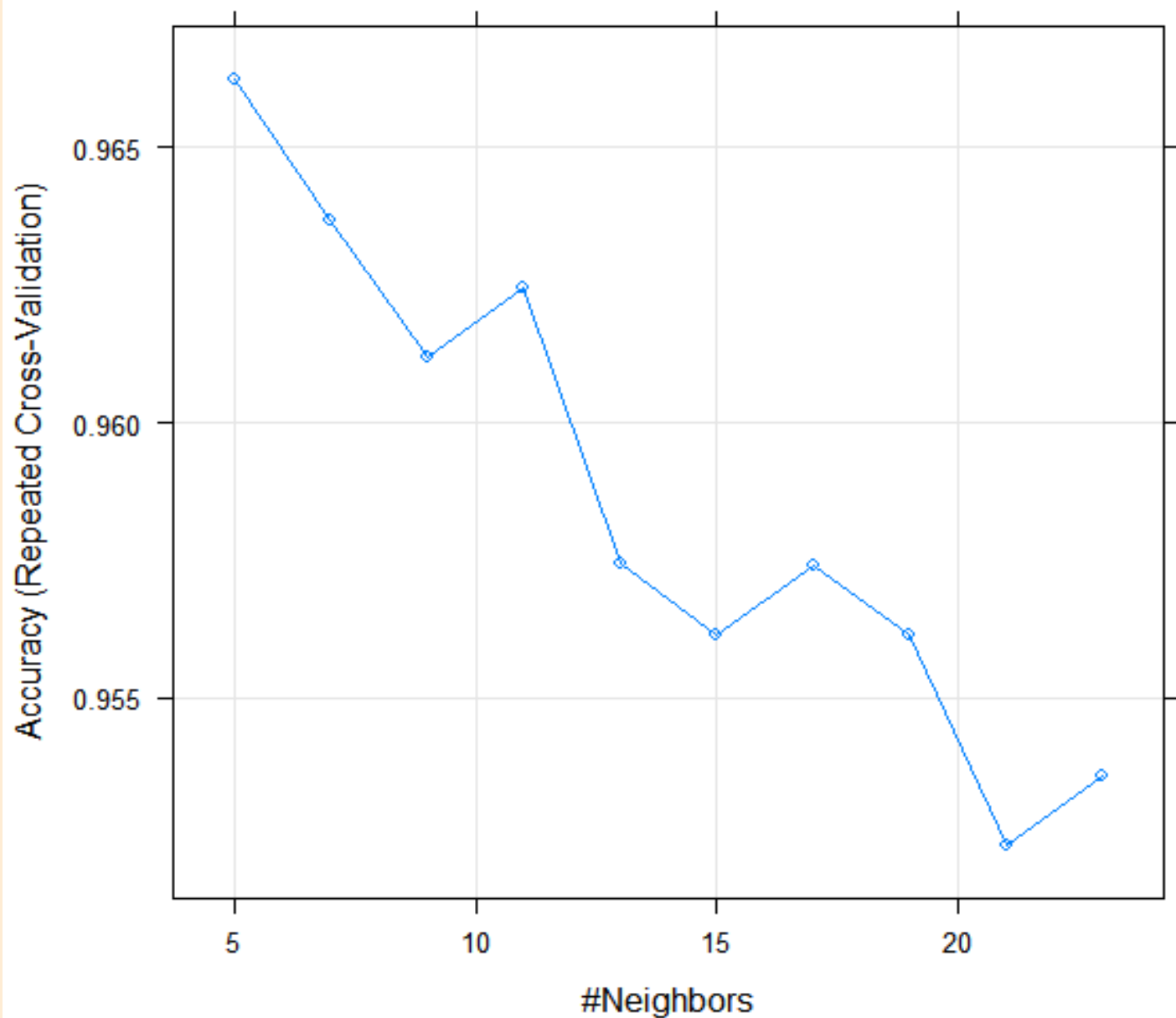
$$A, B \text{ 모두 불합격 줄 확률} : 0.4 * 0.5 = 0.2$$

$$0.3 + 0.2 = 0.5$$

$$K : 0.7 - 0.5 / 1 - 0.5 = 0.4$$

## 최적의 k 구하기

```
# k=5일 때 최적  
plot(knnFit1)  
max(knnFit1$bestTune)
```






## 최적의 k로 예측

```
#예측  
knnPredict1 <- predict(knnFit1, newdata=wbcd_test)  
cmat1 <- confusionMatrix(knnPredict1, wbcd_test$diagnosis,  
                           positive="Malignant")
```

cmat1



No information rate - 가장 많은 값이  
발견된 비율  
 $107 / (107 + 63)$   
단순 분류 알고리즘보다 성능 좋아야  
함.

```
> cmat1
```

Confusion Matrix and Statistics

	Reference	
Prediction	Benign	Malignant
Benign	104	4
Malignant	3	59

Accuracy : 0.9588  
95% CI : (0.917, 0.9833)  
No Information Rate : 0.6294  
P-Value [Acc > NIR] : <2e-16  
  
Kappa : 0.9114

Mcnemar's Test P-Value : 1

Sensitivity : 0.9365  
Specificity : 0.9720  
Pos Pred Value : 0.9516  
Neg Pred Value : 0.9630  
Prevalence : 0.3706  
Detection Rate : 0.3471  
Detection Prevalence : 0.3647  
Balanced Accuracy : 0.9542

'Positive' class : Malignant

# Confusion Matrix 통한 3가지 척도

		True condition	
		Condition positive	Condition negative
Predicted Condition	Predicted condition positive	True positive	False positive
	Predicted condition negative	False negative	True negative

**Sensitivity**

$$\frac{TP}{(TP + FN)}$$

(Recall, 재현도) 실제 TP인 것 중 예측이 TP로 된 확률

**Precision**

$$\frac{TP}{(TP + FP)}$$

정밀도, TP로 예측된것중 실제로도 TP인 확률

**Accuracy**

$$\frac{TP + TN}{(TP + TN + FP + FN)}$$

전체 경우의 수에서 정답 확률



## 전체 코드

```
wbcd <- read.csv('C:/Users/palt1/Desktop/비타민 발표/wbcd.csv')
str(wbcd)
library(ggplot2)
library(caret)
library(dplyr)
wbcd$diagnosis <- factor(wbcd$diagnosis, levels = c("B", "M"),
                        labels = c("Benign", "Malignant"))

qplot(wbcd$radius_mean, wbcd$radius_mean, data=wbcd, geom='boxplot')
qplot(wbcd$perimeter_mean, wbcd$perimeter_mean, data=wbcd, geom='boxplot')

train_index <- createDataPartition(wbcd$diagnosis, p=0.7, list=FALSE)
wbcd_train <- wbcd[train_index,]
wbcd_test <- wbcd[-train_index,]

#정규화
normalize <- function(x){
  return ((x-min(x)) / (max(x)-min(x)))
}
normalize(c(1,2,3,4,5))

wbcd_n <- as.data.frame(lapply(select(wbcd,-diagnosis), normalize))

# 비율 확인
frqtab <- function(x, caption) {
  round(100*prop.table(table(x)), 1)
}

ft_orig <- frqtab(wbcd$diagnosis)
ft_train <- frqtab(wbcd_train$diagnosis)
ft_test <- frqtab(wbcd_test$diagnosis)
ftcmp_df <- as.data.frame(cbind(ft_orig, ft_train, ft_test))
```



## 전체 코드

```
# 교차검증, 몇 겹
ctrl <- trainControl(method="repeatedcv", number=10)
set.seed(12345)
#caret에서 제공하는 preProcess : center(평균을 0), scale(전체 표준 편차0), range
# 회기일 경우 metric : RMSE
knnFit1 <- train(diagnosis ~ ., data=wbcd_train, method="knn",
                 trControl=ctrl, metric="Accuracy", tuneLength=10,
                 preProc=c('range'))
knnFit1

# k=9일 때 최적
plot(knnFit1)
max(knnFit1$bestTune)

#예측
knnPredict1 <- predict(knnFit1, newdata=wbcd_test)
cmat1 <- confusionMatrix(knnPredict1, wbcd_test$diagnosis,
                         positive="Malignant")

cmat1
```



## Knn 다른 함수

```
#정규화
normalize <- function(x){
  return ((x-min(x)) / (max(x)-min(x)))
}

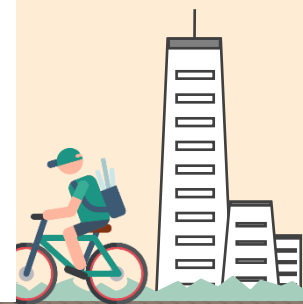
wbcd_n <- as.data.frame(lapply(select(wbcd,-diagnosis), normalize))

#train/test
wbcd_train <- wbcd_n[1:469,]
wbcd_test <- wbcd_n[470:569,]

wbcd_train_labels <- wbcd[1:469,1]
wbcd_test_labels <- wbcd[470:569,1]

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

#crosstable
CrossTable(x=wbcd_test_labels, y=wbcd_test_pred, prop.chisq=TRUE)
|
# 일반 횡수
# 카이 제곱 ( 기대치 비율 )
# 행을 기준으로 비율 값 ( 가로로 읽는다. )
# 컬럼을 기준으로 비율 값 ( 세로로 읽는다. )
# 전체를 기준으로 비율 값
```



## 추가1 - 유의점

설명변수에 민감한 반응을 보인다. 중요도 기반의 tree알고리즘과 달리,  
거리 기반이기 때문에 중요하지 않은 변수가 모델에 추가될수록 정확도가 떨어진다.  
따라서 중요도가 높은 의미있는 설명변수만 추출하여, 정확성을 높여야한다.



## 추가2 - 유의점

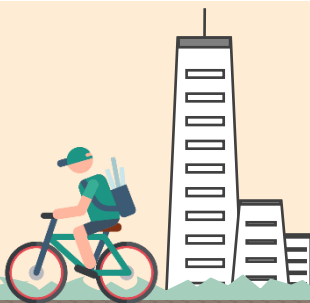
변수가 범주형일 경우 dummy화 시켜줘야 한다.

그 후 그대로 train, test 셋 나누고 모델 학습시켜주면 된다.

Dummies(패키지)는 one-hot encoding 방식

```
Season <- c("s1", "s2", "s3", "s4", "s1", "s2", "s3", "s4")
store<-c('a', 'b', 'c', 'a', 'a', 'b', 'c', 'c')
SalesAmt <- c(300, 800, 400, 100, 280, 750, 390, 60)
TS <- data.frame(Season,store, SalesAmt, stringsAsFactors = F)
```

```
> TS
  Season store SalesAmt
1     s1     a      300
2     s2     b      800
3     s3     c      400
4     s4     a      100
5     s1     a      280
6     s2     b      750
7     s3     c      390
8     s4     c       60
```



## 추가2 - 유의점(dummy 후)

```
library(dummies)
dummy.data.frame(select(TS, -c(SalesAmt)))
```

```
> dummy.data.frame(select(TS, -c(SalesAmt)))
  SeasonS1 SeasonS2 SeasonS3 SeasonS4 storea storeb storec
1         1         0         0         0         1         0         0
2         0         1         0         0         0         1         0
3         0         0         1         0         0         0         1
4         0         0         0         1         1         0         0
5         1         0         0         0         1         0         0
6         0         1         0         0         0         1         0
7         0         0         1         0         0         0         1
8         0         0         0         1         0         0         1
```

```
library(e1071)
dummy.code(TS$Season)
dummy.code(TS$store)
```

```
> dummy.code(TS$Season)
      S1  S2  S3  S4
[1,]    1   0   0   0
[2,]    0   1   0   0
[3,]    0   0   1   0
[4,]    0   0   0   1
[5,]    1   0   0   0
[6,]    0   1   0   0
[7,]    0   0   1   0
[8,]    0   0   0   1
> dummy.code(TS$store)
      a  b  c
[1,]    1   0   0
[2,]    0   1   0
[3,]    0   0   1
[4,]    1   0   0
[5,]    1   0   0
[6,]    0   1   0
[7,]    0   0   1
[8,]    0   0   1
```



# 클러스터링(군집화)

## 클러스터링 (군집화)

: 데이터를 유사한 아이템의 그룹(클러스터)으로 자동 분리하는 **자율 머신 러닝 작업**

- 예측보다는 지식의 발견에 사용됨
- 클러스터 안에 있는 데이터는 동질성이 커야 하고  
클러스터 밖에 있는 데이터는 이질성이 커야 함



# 클러스터링 사용 분야 예시

## 1) 타겟 마케팅 캠페인

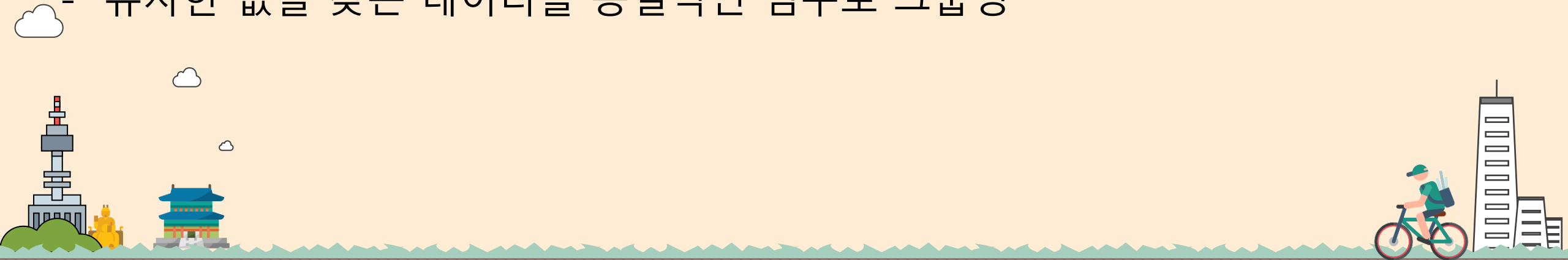
- 유사한 인구 통계나 구매 패턴을 가진 그룹으로 고객 세분화

## 2) 이상 행동 탐지

- 기존 클러스터 밖의 사용 패턴을 찾아 무단 네트워크 침입 등의 이상 감지

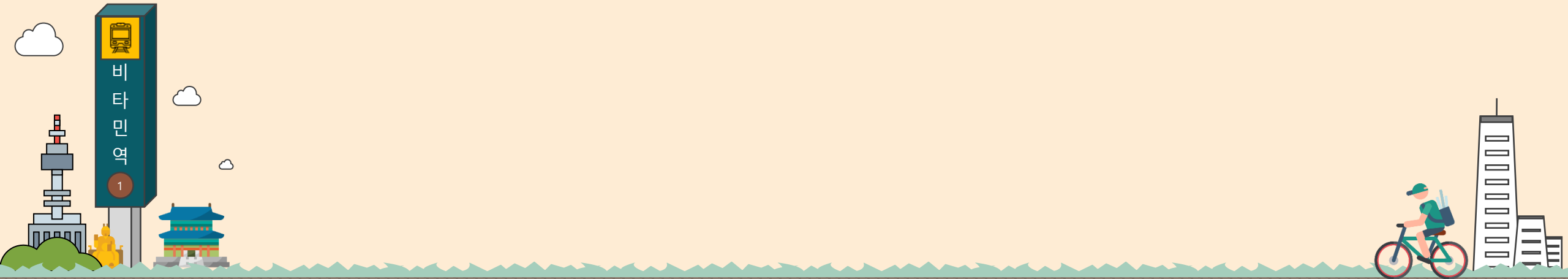
## 3) 초대형 데이터셋의 단순화

- 유사한 값을 갖는 데이터를 동질적인 범주로 그룹핑



# 비지도 학습

- 클러스터링은 레이블이 없는 '비지도 학습'에 해당됨
- 클러스터링은 어떤 예시 그룹이 긴밀하게 연관돼 있는지 말해주지만 실행 가능하고 의미 있는 레이블을 적용하는 것은 사람의 책임
- K-NN & K-means의 가장 큰 차이  
: 지도 학습 vs 비지도 학습



# K-means 클러스터링 방법 요약

1. 데이터를  $k$  개의 초기 군집으로 나눔  
( $k$  값은 사전에 정해 줌)
2. 랜덤으로 군집의 중심을 정함  
각 군집의 중심에 가장 가까운 군집에 데이터 할당 (유클리드 거리 사용)
3. 각 군집에 할당된 데이터들의 평균을 새로운 군집의 중심(centroid)으로 잡고  
새로운 중심에 가장 가까운 군집에 데이터 재할당
4. 3번을 반복
5. 더 이상의 재배치가 생기지 않을 시 중단 → 최종 결과물

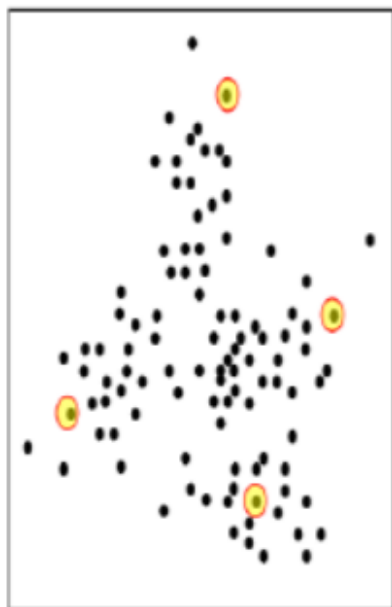


# K-means 클러스터링 방법 요약

## [K-means 군집분석 단계]

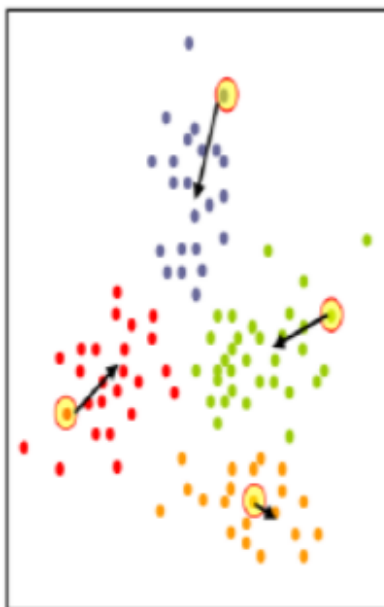
군집의 수 결정 ( $K = 4$ )

초기 Seed 데이터 지정



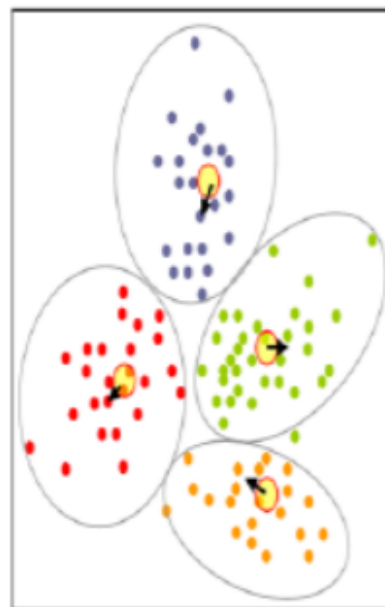
군집화 작업 후

군집중심 산출



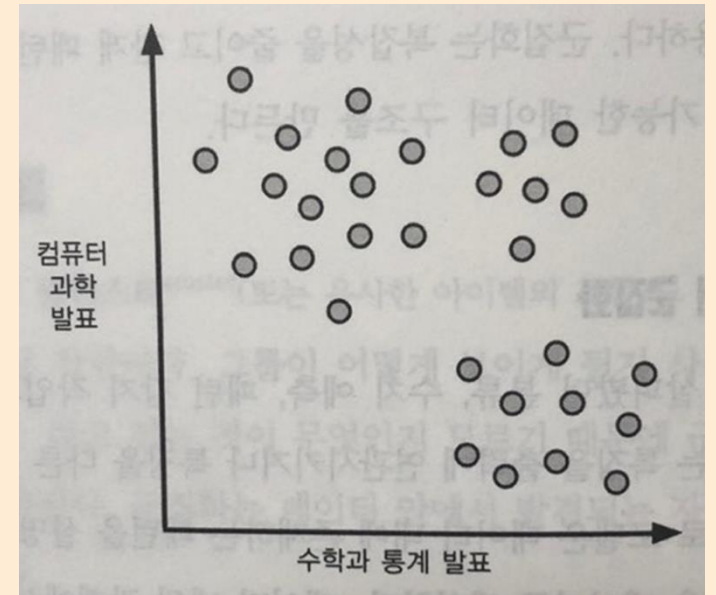
군집화 재작업 및

군집 중심 재산출



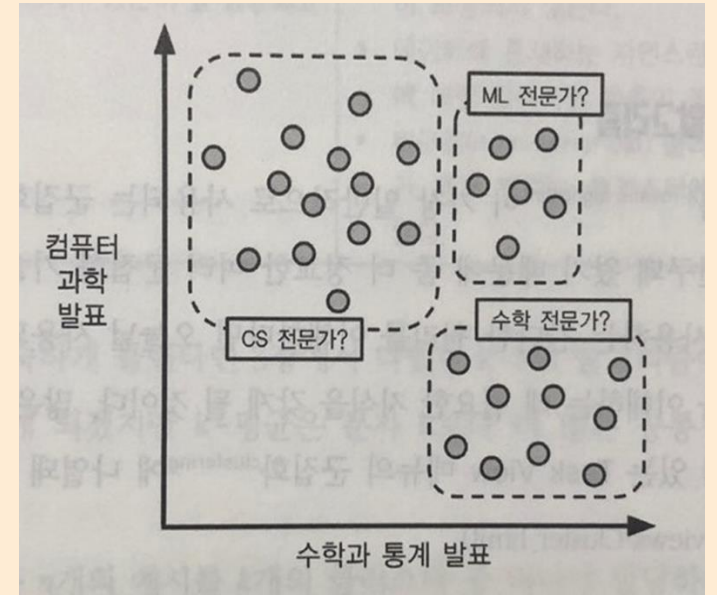
# 클러스터링 예시 - 데이터 과학 주제 컨퍼런스

- 데이터베이스 과학, 수학과 통계, 머신 러닝의 세 전문 분야로 그룹을 나누려고 함
- 컴퓨터 과학 관련 논문 수와 수학과 통계 관련 논문 수에 대한 정보 존재
- 분류 기준
  - 1) DB 과학 : 컴퓨터 과학 ↑ & 수학과 통계 ↓
  - 2) 수학과 통계 : 컴퓨터 과학 ↓ & 수학과 통계 ↑
  - 3) 머신 러닝 : 컴퓨터 과학 ↑ & 수학과 통계 ↑
- 2차원 (기준이 두 개)



# 클러스터링 예시 - 데이터 과학 주제 컨퍼런스

- 시각적으로 패턴이 보이지만 머신 러닝을 통한 객관성을 필요로 함
- 옆의 경우처럼 데이터가 눈으로 확인된 그룹에 완전히 포함되는지 확신하기 어려움
- 클러스터링 활용 시, 데이터들이 얼마나 밀접하게 연관되어 있는지 측정해 그룹 생성 가능



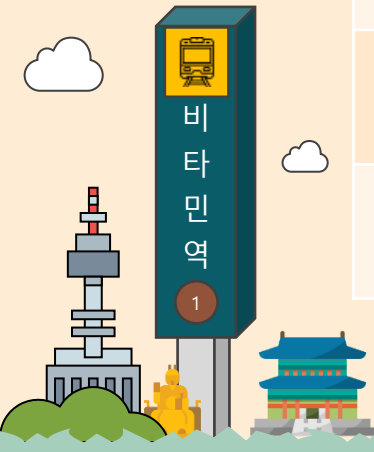
# K-평균 군집화 알고리즘

K-평균 군집화 알고리즘 (K-means algorithm)

: 가장 일반적으로 사용되는 군집화 방법

< 장·단점 >

장점	단점
비통계적 용어로 설명 가능한 간단한 원리	최신 군집화 알고리즘만큼 정교하지 않음
유연하며 간단한 조정으로 대다수의 단점 해결 가능	임의의 요소를 사용하므로 최적의 클러스터 집합을 찾는 것이 보장되지 않음
많은 사례에서 성공적으로 실행됨	클러스터 개수에 대한 합리적인 추측이 필요함
	비구형 클러스터 및 밀도가 크게 변하는 클러스터에는 이상적이지 않음





# K-평균 군집화 알고리즘

K-평균 군집화 알고리즘 (K-means algorithm)

: n개의 예시(데이터)를 k개의 클러스터 중 하나에 할당하는 알고리즘

- **k(seed)**는 사전에 결정된 숫자
- 목표 : 클러스터 내의 차이를 최소화하고 클러스터 간의 차이를 최대화
- **휴리스틱 과정**을 사용함



# 휴리스틱 과정

## 휴리스틱

: 불충분한 시간이나 정보로 인하여 합리적인 판단을 할 수 없거나 필요 없는 경우,  
경험에 기반하여 문제를 해결, 학습 또는 발견해 내는 방법

### - 단계

- 1) k개의 초기 클러스터 집합에 예시(데이터) 할당
- 2) 클러스터 경계를 조정함으로써 할당을 수정
- 3) 1&2 과정 반복



# 거리를 이용한 할당 및 수정

거리를 사용하는 이유

: k-평균은 k-NN과 마찬가지로 데이터를 **다차원 특징 공간의 좌표**로 취급

- 할당 및 수정 단계

1) 클러스터의 중심으로 제공할 k개의 점을 **임의로** 선택

→ k-평균 알고리즘에서 굉장히 민감하게 작용

→ 이후에 데이터가 적절히 배치되도록 자극하는 **촉매제 역할**을 함



# 거리를 이용한 할당 및 수정

## 2) 할당

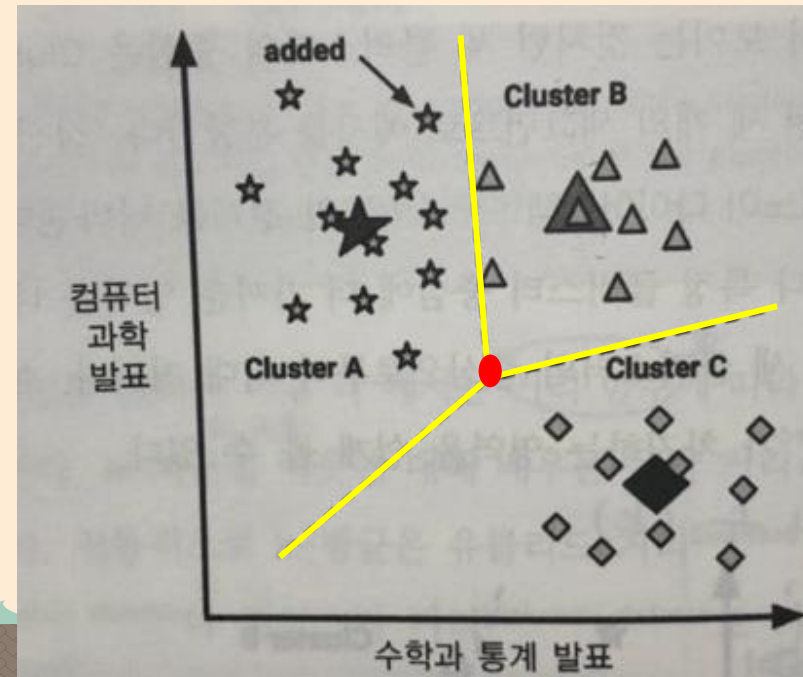
### - 보로노이 다이어그램 (Voronoi diagram)

: 특정 클러스터의 중심에 더 가까운 영역

→ K개로 나누어진 각 클러스터가 차지하는 영역을 쉽게 볼 수 있음

- 세 경계가 만나는 점

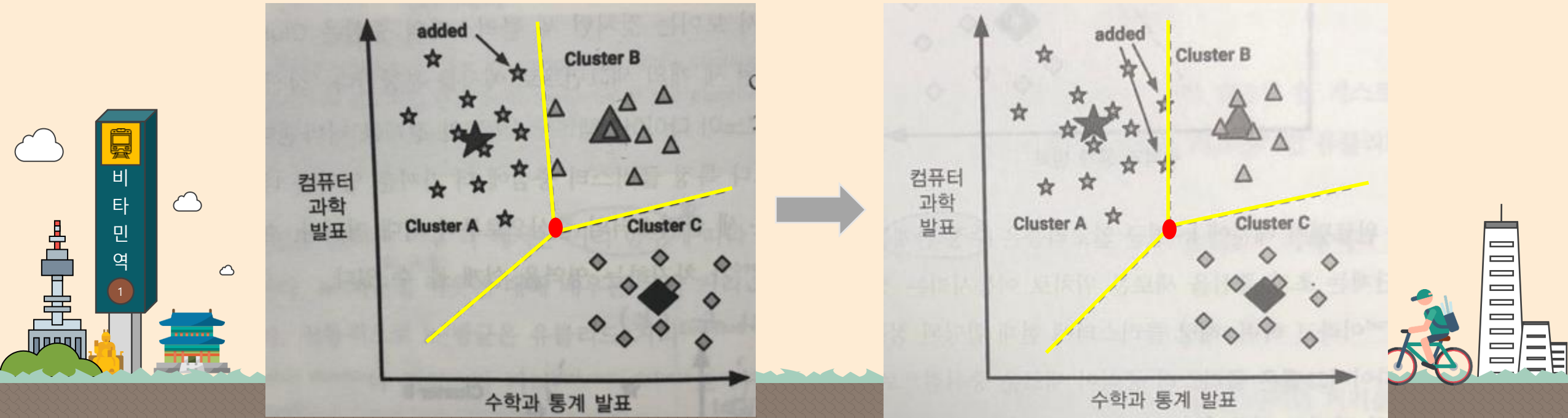
: 세 클러스터의 중심으로부터 **최대 거리**



# 거리를 이용한 할당 및 수정

## 3) 수정

- **중심점** : 초기 중심을 새로운 위치로 이동시킨 부분
  - 처음 할당된 점들의 **평균 위치**로 계산됨
  - 보로노이 다이어그램 경계도 동시 이동



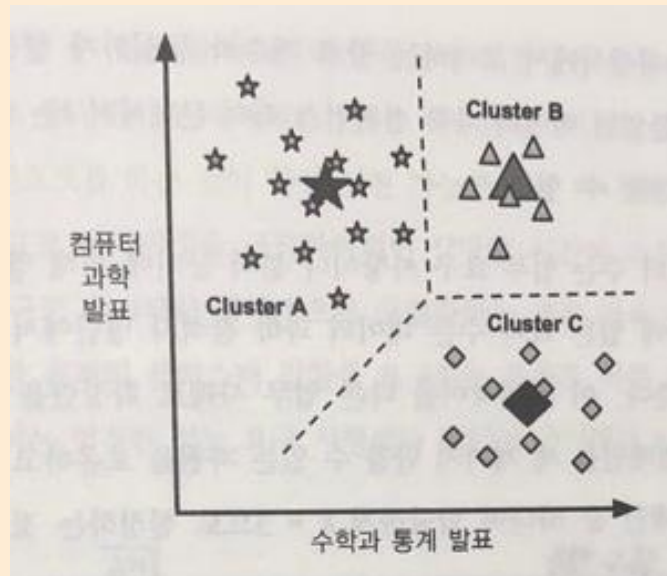
# 거리를 이용한 할당 및 수정

4) 할당 및 수정 반복

5) k-평균 알고리즘 중단

: 할당 및 수정을 반복할 때 재할당이 일어나지 않으면 중단  
(직전의 클러스터와 변화된 부분이 없다면 중단)

- 최종 클러스터 :



# 주의 사항

1) k-평균 알고리즘은 클러스터 **중심의 출발 위치**에 매우 민감함

- 해결 방법

① 특징 공간의 어디든 존재할 수 있는 랜덤 값 선택

② 중심의 출발 위치 선정 자체를 생략

→ 각 데이터를 클러스터에 임의로 할당함으로써 알고리즘 수정 단계로 바로 진행

→ 최종 클러스터 집합에 특정 편향을 더해서 결과를 향상시키기 위해 사용



# 주의 사항

2) k-평균 알고리즘은 클러스터 수(k)에 민감함

- 적합한 클러스터 개수 선택 방법

① 사전 지식이 있는 경우 : 실제 그룹에 대한 선형적 지식을 적용

② 사전 지식이 없는 경우

(1)  $k = \sqrt{n/2}$  (n = 데이터셋의 데이터 수)

→ 대형 데이터셋의 경우 너무 많은 클러스터를 만든다는 단점 존재

(2) 엘보법





# 엘보법 & 엘보 포인트

## 엘보법

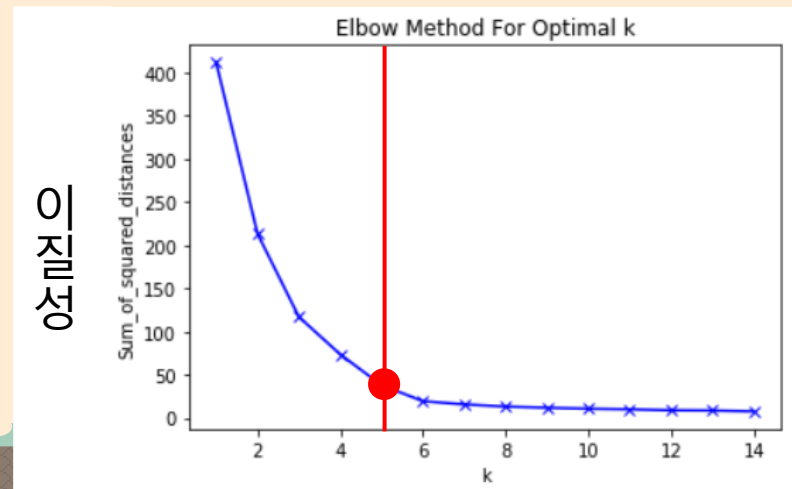
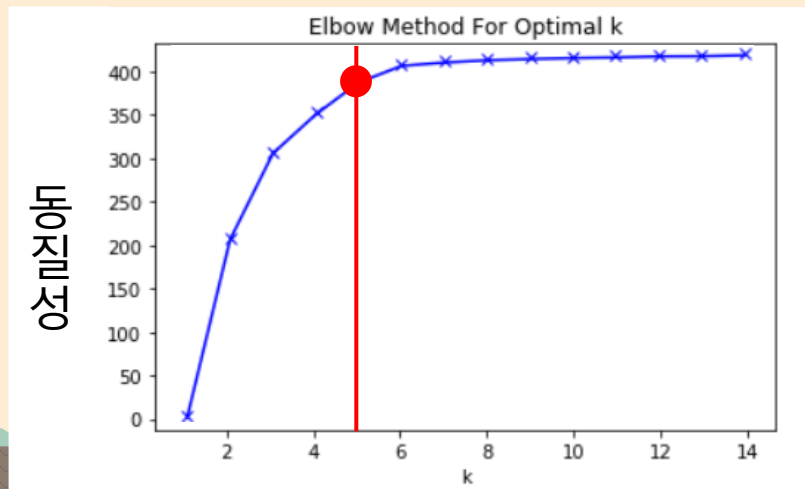
: 클러스터 내의 **동질성**과 **이질성**의 변화정도를 측정해 엘보 포인트를 찾는 방법

→  $k \propto$  클러스터 내의 동질성

→  $k \propto \frac{1}{\text{이질성}}$  클러스터 내의 이질성

## 엘보 포인트

: 특정 포인트를 넘으면 결과가 약화되는 지점(k값)



# 범주형 변수의 dummy 코딩

k-means 알고리즘은 점들의 평균을 계산해서 centroid를 결정함  
따라서 범주형 변수의 경우 더미화를 해줘야 함

## 더미 코딩

: 범주형 변수의 모든 레벨에 대해 이진(1 또는 0)값을 갖는 별도의 더미 변수를 생성하는 방법

Ex) 성별 (여성 = 1, 남성 = 0)  
과일 (사과 = 1, 키위 = 0)



# 범주형 변수의 클러스터링 (k-modes)

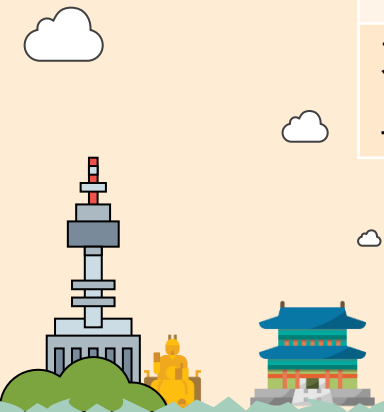
## k-modes

: k-means 알고리즘의 확장된 방법으로 범주형 변수를 클러스터링 할 때 효율적

k-means VS k-modes

:

k-means	k-modes
'거리'를 기준으로 군집화	'차이'를 기준으로 군집화
평균을 사용해 다음 군집의 중심을 계산	불일치하는 데이터의 <b>mode</b> (벡터 간의 차이가 가장 적은 데이터로 이루어진 벡터)
거리가 좁을 수록 서로 연관성이 높음을 의미	불일치하는 데이터 수가 적을수록 연관성이 높음을 의미



# 범주형 변수의 클러스터링 (k-modes)

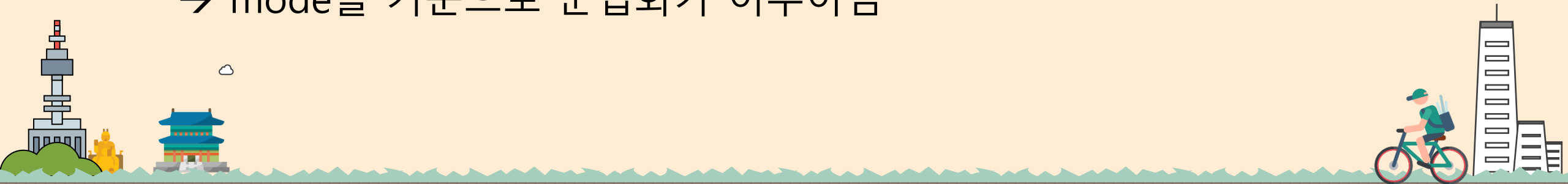
## k-modes의 **dissimilarity**

Dissimilarity를  $d(X,Y)$ 라고 정의

만약  $X = [\text{blue}, \text{cold}, A, \text{seoul}, \text{Kim}]$  이고,  
 $Y = [\text{red}, \text{cold}, B, \text{seoul}, \text{Park}]$  이라면  
불일치 하는 개수가 3개 이므로  $d(X,Y) = 3$ 이라고 표현

Mode는 모든 점(X)들과 군집의 중점(Q) 사이에서  $d(X,Q)$ 를 구했을 때  
그 값이 최소가 되는 값(벡터)를 의미함

→ mode를 기준으로 군집화가 이루어짐



# kmodes ( ) 함수

**kmodes(data, modes, iter.max, weighted=F, fast=T)**

: data = 범주형 자료 (행렬 또는 데이터 프레임 형식)

modes = mode 수 (클러스터 수)

iter.max = 반복 가능한 최대 횟수

- **klaR** 패키지에 들어 있는 함수



## kmodes ( ) 함수 예시

```
install.packages("klaR")
library(klaR)

x <- rbind(matrix(rbinom(250,2,0.25), ncol=5),
            matrix(rbinom(250, 2, 0.75), ncol=5))
```

```
colnames(x) <- c("a","b","c","d","e")
view(x)
```

```
cl <- kmodes(x,2)
cl
plot(jitter(x), col=cl$cluster)
points(cl$modes, col=1:5, pch=8)
```

```
> c1
K-modes clustering with 2 clusters of sizes 50, 50
```

## cluster modes:

	a	b	c	d	e
1	0	0	0	0	0
2	2	2	2	2	2

clustering vector:

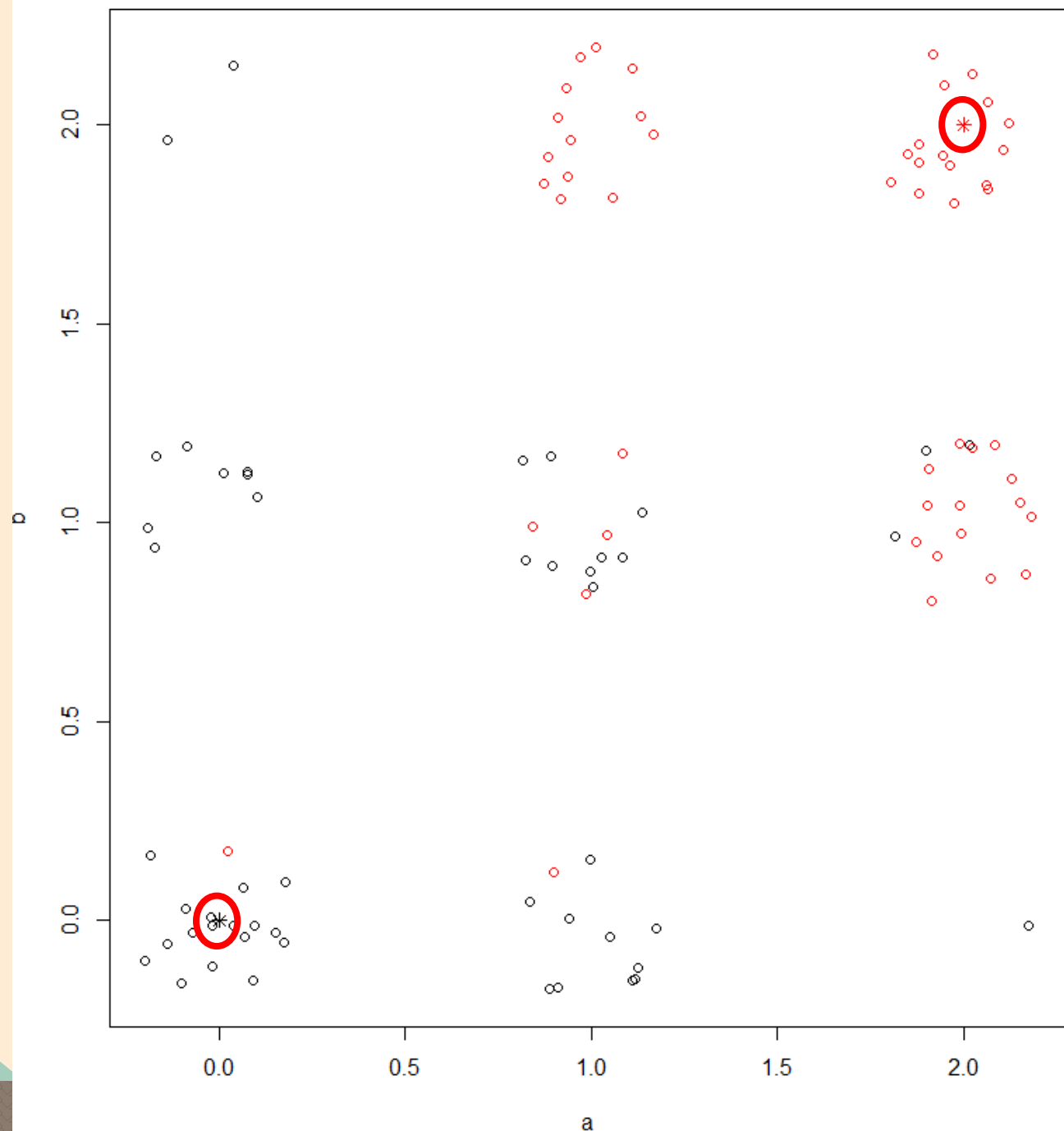
[illegible]

```
within cluster simple-matching distance by cluster:
```

 $[1] \quad 111 \quad 100$ 

Available components:

```
[1] "cluster"      "size"         "modes"        "withindiff"
[5] "iterations"  "weighted"
```



# 함수 설명

## **kmeans(x, centers, iter.max, nstart, ..) 함수**

: x=data, centers=k(클러스터 수)

iter.max=반복 가능한 최대 횟수, nstart=군집화 시행 횟수

## **createFolds(features, num\_folds) 함수 – caret package**

: 교차 검증 기능 중 K-fold 방법 구현 (데이터 셋을 K개의 sub-set으로 분리)

features = 분리하려는 데이터 셋 , num\_folds = k

## **createDataPartition(x, times, p, groups, list ) 함수**

: 데이터를 트레이닝 데이터 세트와 테스트 데이터 세트로 분리함

x = 데이터 셋, times = sub-set 수, p = 트레이닝 세트로 들어가는 데이터 비율,

list = list 형식으로의 표현 여부(TRUE,FALSE)

# 함수 설명

## scale(x, center=T, scale=T) 함수

: 정규화된 값 ( $z=(x-\text{평균})/\text{표준편차}$ )을 구하는 함수

x = 행렬 데이터

## lapply(x, FUN) 함수

: 리스트나 벡터에 특정 함수를 적용하는 함수

## aggregate( ) 함수

: 반복적인 코드를 최소화 하는 함수

## ave(a,b,FUN=function(x)mean(x,na.rm=TRUE)) 함수

: 벡터 b와 동일한 크기로 벡터 a의 평균을 반복해서 출력하는 함수

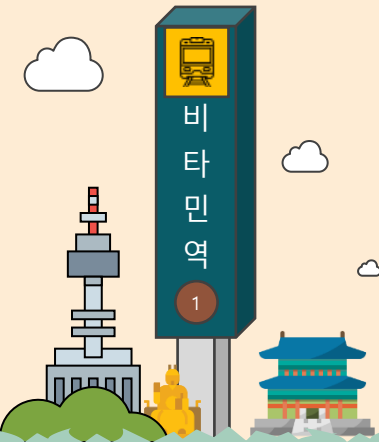


# k-평균 군집화 사용 예제 - 십대 시장 세분화

## < 데이터 정보 >

- 미국 고등학생(1~4학년)의 SNS 페이지 텍스트를 갖는 데이터셋
- 학생들의 성별, 연령, SNS 친구 수, 다양한 관심 분야 등의 정보가 담겨 있음
- SNS를 많이 활용하는 십대를 관심 분야 별로 나누어 그들이 관심 없어 하는 광고가 타깃팅 되지 않도록 하는 것을 목표로 크롤링한 자료

→ 함께 첨부한 html 파일로 예제 설명하겠습니다!  
(html에 있는 코드 쳐서 예습과제로 제출하시면 됩니다~)



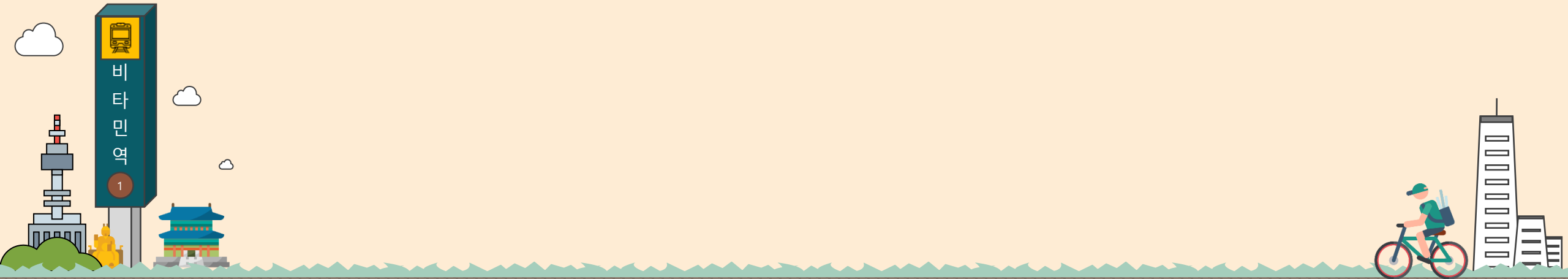


# K-NN & K-means 실습

문혜현, 오민석, 이지선

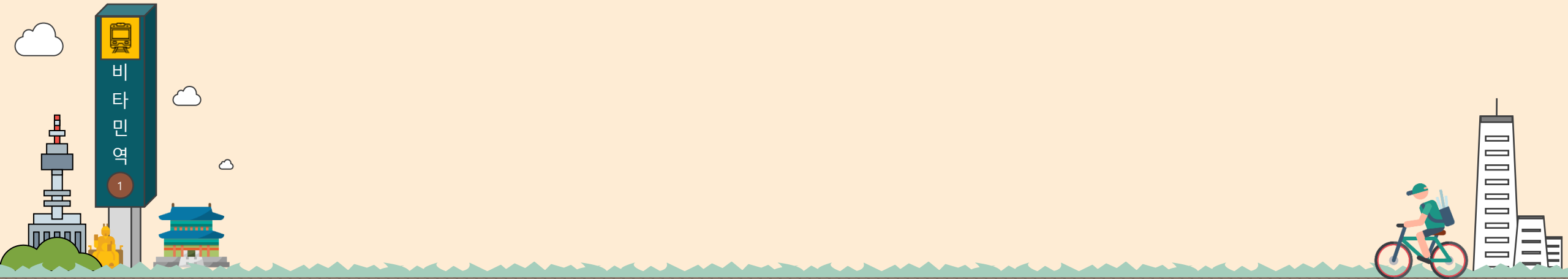
# 데이터 소개

- 서울시 공기 관련 데이터
- 2018년 3월 25일의 각 구의 시간별 공기질에 대한 수치가 들어 있음
- So<sub>2</sub>, co, o<sub>3</sub>, pm<sub>10</sub>(미세먼지) 등등의 정보가 담겨있음
- 공기질이 가장 나쁨, 중간, 좋음인 세 구를 뽑기
- 각 원소들의 상관관계를 확인함



# Knn 실습

1. train, test set으로 분류하기(sample을 이용해)
2. K가 1일 때의 knn의 정확성 알아보기(수치-confusionMatrix, 그래프 두 가지로)
3. 적당한 k 값 찾기(for 문을 이용해 보기)



# kmeans

1. Kmeans에 이용할 데이터 scale 해보기
2. Kmeans 함수 이용해 clustering 하기
3. 그래프, table 함수를 이용해 정확성 확인
4. Elbow 기법 실습하기

