Machine learning

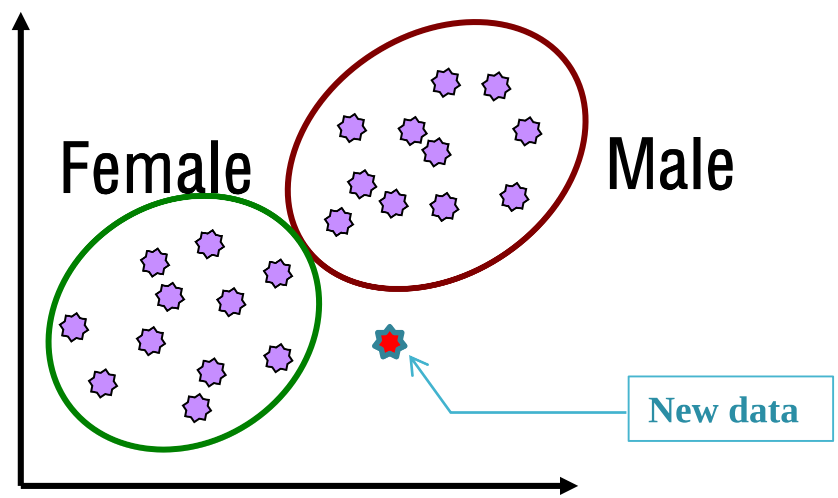
Chapter 3

# K-Nearest Neighbor

Sejong Oh
Bio Information Technology Lab.

# Classification

- Classification
  - Classify new data into one of known category.
  - The category has “label”
  - Supervised learning



# Classification

- Classification example

Feature (attribute, variable) class

Disease A

| No | Height | Weight | running hour | working hour | |
|---|---|---|---|---|---|
| 1 | 0.41 | 0.36 | 0.27 | 0.65 | Patient |
| 2 | 0.23 | 0.37 | 0.34 | 0.68 | patient |
| 3 | 0.38 | 0.38 | 0.46 | 0.95 | patient |
| 4 | 0.45 | 0.31 | 0.37 | 0.75 | patient |
| 5 | 0.37 | 0.45 | 0.48 | 0.75 | patient |
| 6 | 0.28 | 0.26 | 0.36 | 0.86 | patient |
| 7 | 0.66 | 0.44 | 0.51 | 0.98 | patient |
| 8 | 0.55 | 0.43 | 0.43 | 0.91 | patient |
| 9 | 0.23 | 0.44 | 0.28 | 0.78 | patient |
| 10 | 0.41 | 0.53 | 0.46 | 0.86 | patient |
| 11 | 0.65 | 0.38 | 0.74 | 0.51 | normal |
| 12 | 0.89 | 0.53 | 0.67 | 0.46 | normal |
| 13 | 0.58 | 0.54 | 0.56 | 0.43 | normal |
| 14 | 0.78 | 0.55 | 0.67 | 0.34 | normal |
| 15 | 0.89 | 0.56 | 0.81 | 0.56 | normal |
| 16 | 0.65 | 0.57 | 0.81 | 0.43 | normal |
| 17 | 0.75 | 0.67 | 0.76 | 0.35 | normal |
| 18 | 0.46 | 0.48 | 0.65 | 0.42 | normal |
| 19 | 0.89 | 0.69 | 0.78 | 0.23 | normal |
| 20 | 0.78 | 0.81 | 0.88 | 0.26 | normal |

Training data

| Height | Weight | running hour | working hour |
|---|---|---|---|
| 0.5 | 0.44 | 0.45 | 0.61 |

Patient or Normal ?

Test data

# Classification

- Classification example


Apple iPhone

(1)take a picture by phone camera

(2) Search similar image and shows detail information about it

# Classification

- Binary vs. multiple classification

  - Binary classification
    - # of class is two

| | |
|---|---|
| Male | Female |
| Patient | Normal |
| Yes | No |

  - multiple classification
    - # of class over two

| | | |
|---|---|---|
| Well-done | medium | rare |

| | |
|---|---|
| university | High school |
| Middle school | Elementary school |

# Classification analysis procedure

1. Prepare target dataset
2. Divide target dataset into training data and test data

   - assume we don't know class labels of test data

3. Training model using training data
4. Predict class labels of test data using learning model
5. Evaluate prediction performance

- $$\text{accuracy} = \frac{\text{\# of instances that are correctly predicted}}{\text{\# of total instances in test data}}$$

# Classification

- Binary Classification Error

**Fact**

**Predict**

| | Fact is Positive | Fact is Negative |
|---|---|---|
| Predict as Positive | TP | FP |
| Predict as Negative | FN | TN |

**TP : true positive    FP : false positive**
**FN : false negative TN : true negative**

# Classification

- Performance criteria for binary classification

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

$$Sensitivity = \frac{TP}{(TP + FN)}, \quad Specificity = \frac{TN}{(TN + FP)}$$

# KNN

# Problem Definition

| No | running hour | working hour | Category |
|---|---|---|---|
| 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 |

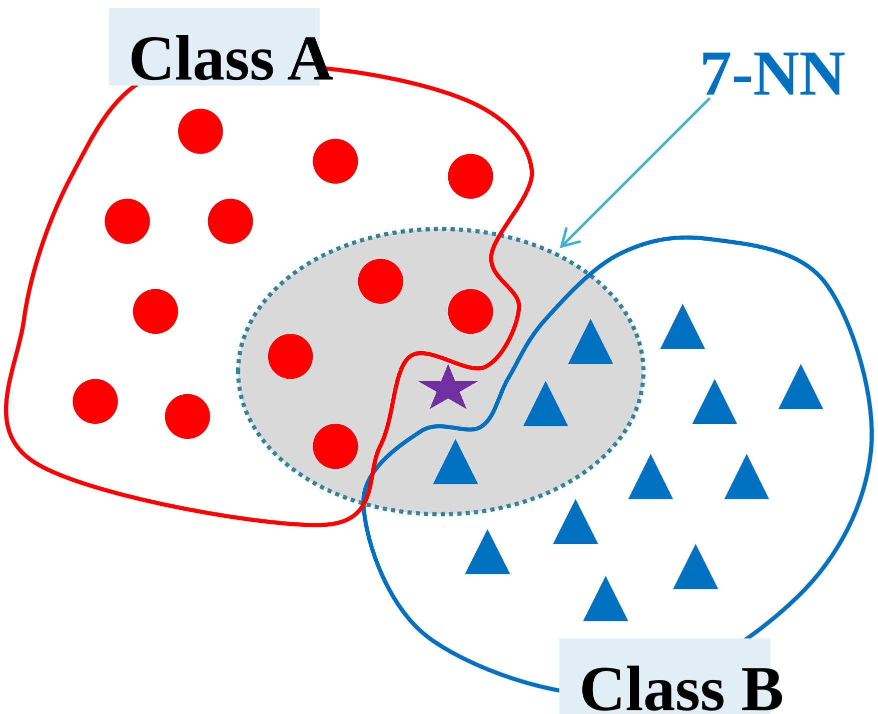
Given Classified Data

Patient or Normal ?

| running hour | working hour |
|---|---|
| 0.45 | 0.61 |

# Idea of KNN

- Find K nearest neighbor for new point ( ★ )
- Decide new point belongs to major class (class A)
  - # of neighbor of Class A > # of neighbor of Class B

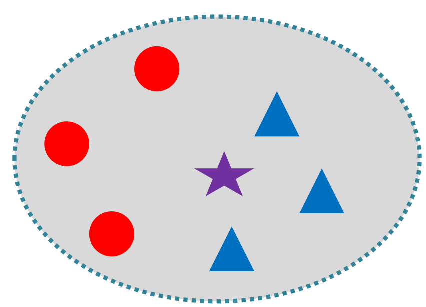


# Idea of KNN

- Algorithm
  - Calculate distance between new point and every point of given classes
  - Choose K nearest points by the distance
  - Choose major class from K points

    (the class is for the new point)

**6-NN**



**???**

# Idea of KNN

- How to calculate the distance between two element ?
  - Using Euclidean distance

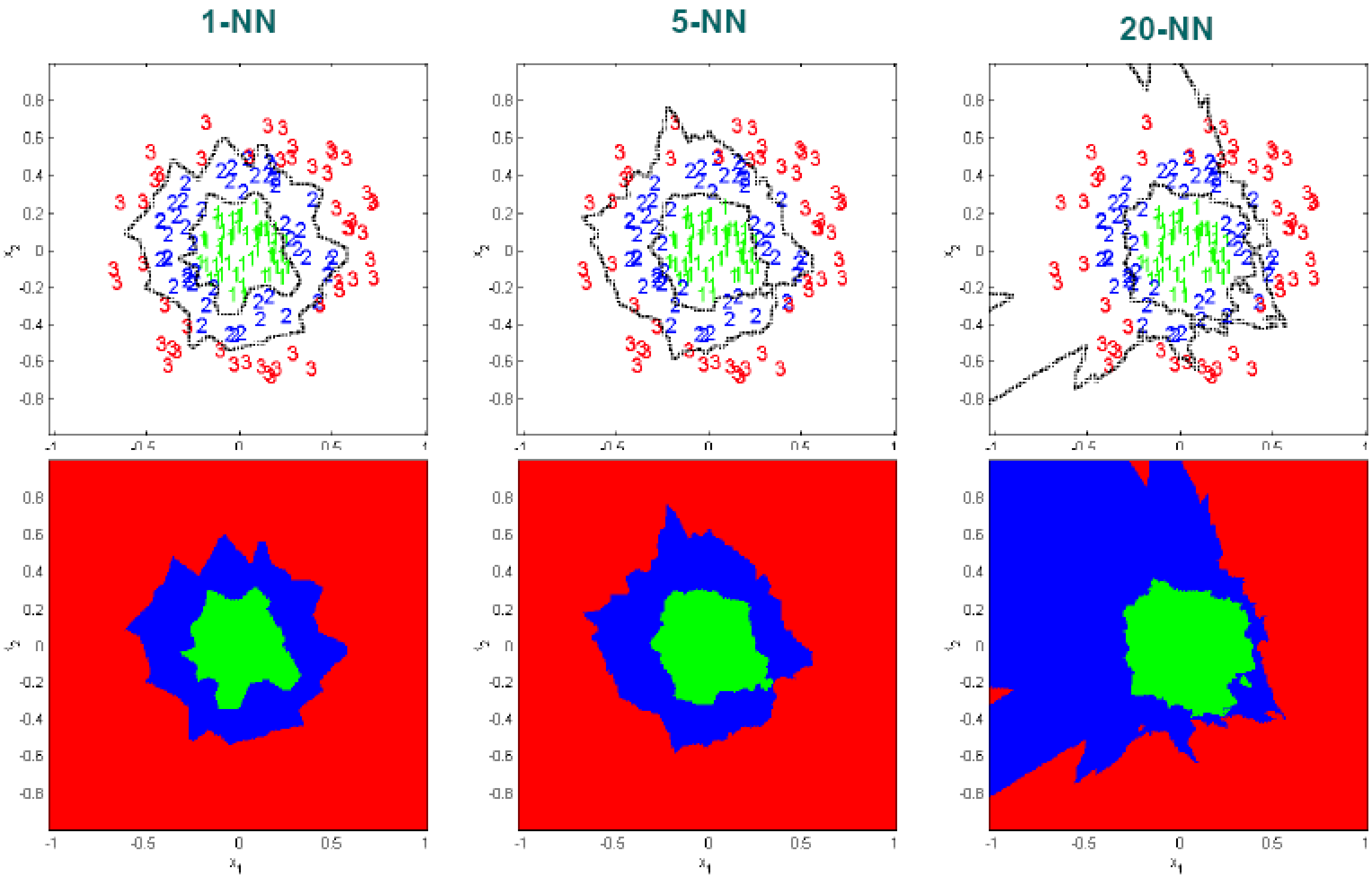
$$\mathbf{p} = (p_1, p_2, \ldots, p_n)$$
$$\mathbf{q} = (q_1, q_2, \ldots, q_n)$$

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

# 1-NN vs. k-NN

- The use of large values of k has two main advantages
  - Yields smoother decision regions
  - Provides probabilistic information
    - The ratio of examples for each class gives information about the ambiguity of the decision
- However, too large a value of k is detrimental
  - It destroys the locality of the estimation since farther examples are taken into account
  - In addition, it increases the computational burden
- A good rule-of-thumb numbers is k should be less than the square root of the total number of training patterns.

# 1NN versus kNN



# KNN in R

- Usage

```
knn(train, test, cl, k = 1, l = 0,
    prob = FALSE, use.all = TRUE)
```

- Parameters
  - **train** : matrix or data frame of training set cases.
  - **test** : matrix or data frame of test set cases.
  - **cl**:  factor of true classifications of training set
  - **k** : number of neighbours considered.
  - **l** : minimum vote for definite decision, otherwise doubt.
  - **prob**:  If this is true, the proportion of the votes for the winning class are returned as attribute prob.
  - **use.all:** controls handling of ties. If true, all distances equal to the kth largest are included. If false, a random selection of distances equal to the kth is chosen to use exactly k neighbours

# Example

- Dataset
  - Liver disorder (간질환)

Feature (attribute) information

1. Class label
2. mcv mean corpuscular volume
3. alkphos alkaline phosphotase
4. sgpt alamine aminotransferase
5. sgot aspartate aminotransferase
6. gammagt gamma-glutamyl transpeptidase
7. drinks number of half-pint equivalents of alcoholic beverages drunk per day

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 85 | 64 | 59 | 32 | 23 | 0 |
| 2 | 0 | 86 | 54 | 33 | 16 | 54 | 0 |
| 3 | 0 | 91 | 78 | 34 | 24 | 36 | 0 |
| 4 | 0 | 87 | 70 | 12 | 28 | 10 | 0 |
| 5 | 0 | 98 | 55 | 13 | 17 | 17 | 0 |
| 6 | 0 | 91 | 72 | 155 | 68 | 82 | 0.5 |
| 7 | 0 | 85 | 54 | 47 | 33 | 22 | 0.5 |
| 8 | 0 | 79 | 39 | 14 | 19 | 9 | 0.5 |
| 9 | 0 | 85 | 85 | 25 | 26 | 30 | 0.5 |
| 10 | 0 | 89 | 63 | 24 | 20 | 38 | 0.5 |

# Example

```r
require("class")    # same as library(class)
setwd("c:/work/data")
ds = read.csv("liver.csv", header = FALSE)
head(ds)
# prepare train/test data
train <- rbind(ds[1:100,], ds[201:270,])
test <- rbind(ds[101:200,], ds[271:345,])
head(train)
head(test)
# run classification test
result <- knn(train[,-1], test[,-1], cl=train[,1], k=1)
result
# performance evaluation 1
acc <- mean(result==test[,1])
acc
```

# Example

```
# more performance evaluation
library(gmodels)    # for CrossTable
tab <- CrossTable(x = test[,1],
                  y = result,
                  prop.chisq=FALSE)
tab

acc2 <- (tab$t[1,1]+tab$t[2,2])/sum(tab$t)
sens <- tab$t[1,1]/(tab$t[1,1]+tab$t[1,2])
spec <- tab$t[2,2]/(tab$t[2,1]+tab$t[2,2])

acc2
sens
spec
```

# Example

Predict

Fact

```
             | result
    test[, 1] |          0 |          1 | Row Total |
-------------|------------|------------|-----------|
           0 |         68 |         32 |       100 |
             |      0.680 |      0.320 |     0.571 |
             |      0.680 |      0.427 |           |
             |      0.389 |      0.183 |           |
-------------|------------|------------|-----------|
           1 |         32 |         43 |        75 |
             |      0.427 |      0.573 |     0.429 |
             |      0.320 |      0.573 |           |
             |      0.183 |      0.246 |           |
-------------|------------|------------|-----------|
Column Total |        100 |         75 |       175 |
             |      0.571 |      0.429 |           |
-------------|------------|------------|-----------|
```

추천 : **CrossTable(x = test[,1],**
**y = result,**
**prop.chisq=F, prop.r=F, prop.t=F)**

# 장단점

- Advantage
  - Simple, powerful
  - 데이터의 분산에 대한 추정을 할 필요가 없다. (non-parametric model )
  - Quick training time(KNN has no training)

- Disadvantage
  - Support no learning . Feature들 사이의 관계에 대한 통찰력을 발견할 수 없다.
  - Low classification speed (모델이 없기 때문)
  - Memory intensive
  - Nominal value, missing value => require preprocessing

# 고려사항 1 : 거리 계산

- 이웃에 있는 case 들을 찾기 위해서는 case 간 거리 계산이 필요
- 일반적으로 Euclidean distance 사용

$\mathbf{p} = (p_1, p_2, \ldots, p_n)$ and $\mathbf{q} = (q_1, q_2, \ldots, q_n)$

$$\mathrm{d}(\mathbf{p},\mathbf{q}) = \mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

- 데이터셋의 속성들의 value range 가 다를 때, instance 간의 거리는 value range가 큰 속성에 의해 결정된다

➡ normalization 필요

# 고려사항 1 : 거리 계산

- Example
  - P2 와 가장 가까운 이웃은 ?

| ID | 키 | 시력 |
|---|---|---|
| P1 | 164 | 0.1 |
| P2 | 169 | 0.7 |
| P3 | 178 | 1.5 |
| P4 | 175 | 0.8 |

```
> ds = matrix(c(164,169,178,175,0.1,0.7,1.5,0.8), ncol=2)
> ds
     [,1] [,2]
[1,]  164  0.1
[2,]  169  0.7
[3,]  178  1.5
[4,]  175  0.8
> dist(ds)
          1         2         3
2  5.035871
3 14.069826  9.035486
4 11.022250  6.000833  3.080584
```

# 고려사항 1 : 거리 계산

- Normalization
  - 각 속성을 0 ~ 1 사이의 값으로 정규화

$$z_i = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

```
> ds.n = ds
> ds.n[,1] = (ds.n[,1]-min(ds.n[,1]))/(max(ds.n[,1])-min(ds.n[,1]))
> ds.n[,2] = (ds.n[,2]-min(ds.n[,2]))/(max(ds.n[,2])-min(ds.n[,2]))
> ds.n
          [,1]      [,2]
[1,] 0.0000000 0.0000000
[2,] 0.3571429 0.4285714
[3,] 1.0000000 1.0000000
[4,] 0.7857143 0.5000000
> dist(ds.n)
          1         2         3
2 0.5578750
3 1.4142136 0.8601139
4 0.9313146 0.4344830 0.5439838
```

# 고려사항 1 : 거리 계산

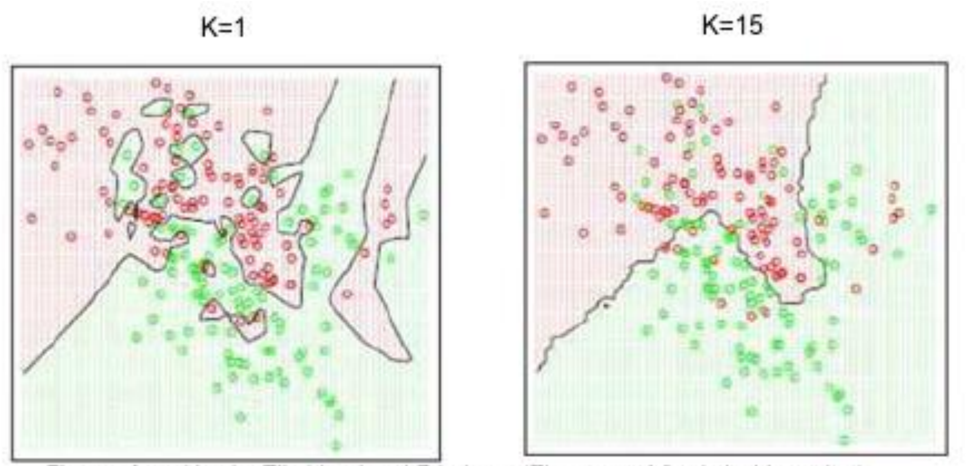
- 데이터셋의 각 속성들을 normalize 하는 것은 거리 계산에 있어서 속성들의 영향력을 동일하게 만드는 과정이다

- normalize 시 max, min 값이 outlier 이면 왜곡이 발생할 수 있으니 주의한다

# 고려사항 2 : k 값의 결정

- K 값은 bias-variance tradeoff 와 관련됨
- K 값을 크게 하면 variance 는 줄어들지만 데이터가 가진 중요한 패턴을 무시할 위험성이 있다
- K 값을 작게 하면 noise 데이터나 outlier 의 영향을 많이 받게 된다.

- 적절한 K 값을 찾아야 하는데 정해진 방법은 없음.
  - 1) instance 수의 제곱근을 사용
  - 2) 여러 개의 K 값을 테스트 해 보고 최적의 분류 성능을 내는 K 를 선택




Figures from Hastie, Tibshirani and Friedman (Elements of Statistical Learning)

Larger k produces smoother boundary effect and can reduce the impact of class label noise.

But when k is too large, say k=N, we always predict the majority class

# 고려사항 3. 명목형 속성

- 명목형 속성 (nominal attribute)

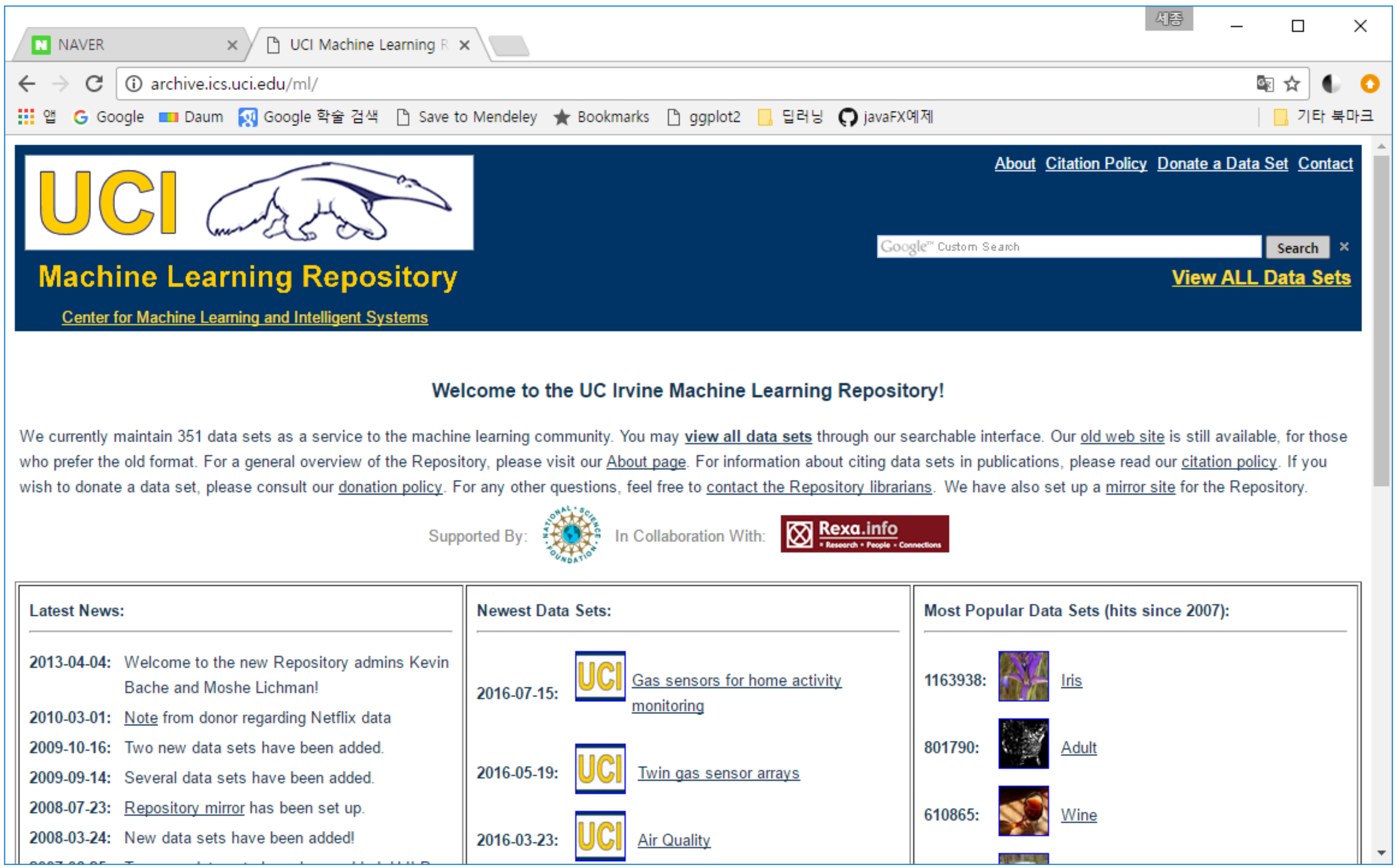
- 색깔 (color), 선호 정당 등의 값들은 크기를 정할 수 없기 때문에 거리 계산을 할 수 없다. → KNN 적용 전에 데이터에서 제외 해야
- 학력(중졸,고졸, 대졸), 평점 (A,B,C,D,F) 은 명목형 속성이나 순서 개념이 존재 하므로 숫자로 변환한 다음 KNN 적용

  예1) 중졸-1, 고졸-2, 대졸-3

  예2) 중졸-1, 고졸-2, 대졸-5 (등간격으로 숫자를 부여해야 한다)

# Task

- Dataset : 위스콘신 유방암 센터 자료
  - http://archive.ics.uci.edu/ml/
  - Breast cancer Wisconsin diagnostic

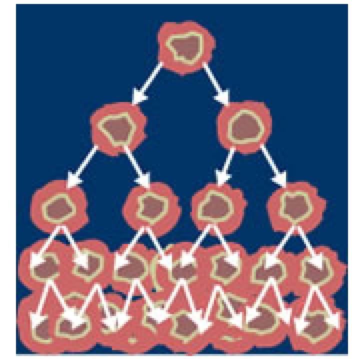


# Task

## Breast Cancer Wisconsin (Diagnostic) Data Set

*Download*: Data Folder, Data Set Description

**Abstract**: Diagnostic Wisconsin Breast Cancer Database



| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate | **Number of Instances:** | 569 | **Area:** | Life |
| **Attribute Characteristics:** | Real | **Number of Attributes:** | 32 | **Date Donated** | 1995-11-01 |
| **Associated Tasks:** | Classification | **Missing Values?** | No | **Number of Web Hits:** | 469904 |

- 암세포 이미지에서 32 feature 추출
- Class
  - M : 악성
  - B : 양성

# Task

- 실습목표 : wbcd 데이터셋에 대해 KNN 을 적용한 예측 모델을 만들고, 모델의 성능을 평가 한다

- Dataset & source code : material “chapter 3” 참조
- 설치해야 하는 package : “class”, “gmodels”

```
install.packages("class")
install.packages("gmodels")
```

---

Source code & data
http://www.acornpub.co.kr/book/machine-learning-r