

Machine Learning

Lecture 2 Data

Attributes

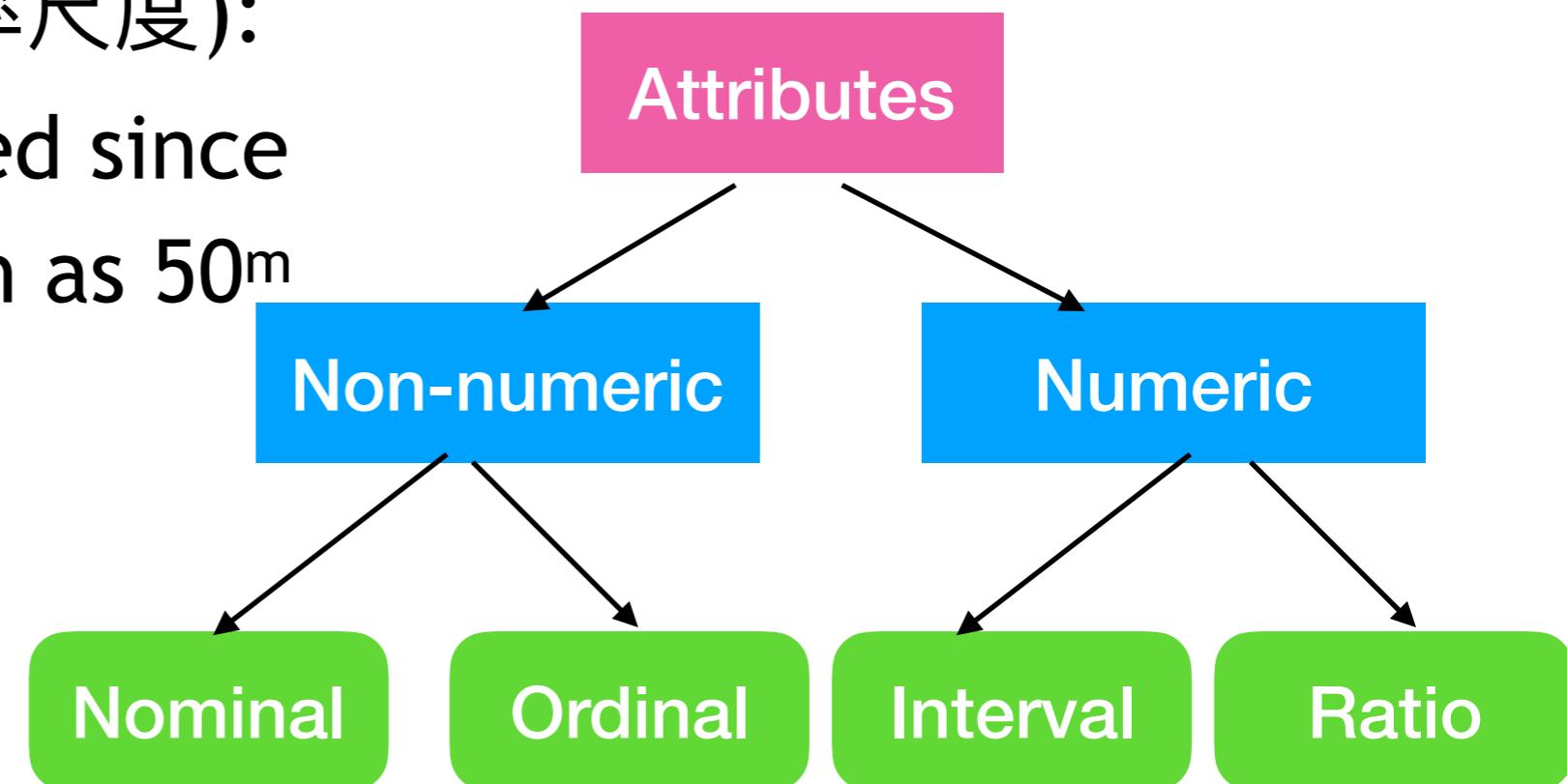
- **Attribute** (or **dimensions, features, variables**)
 - A data field, representing a characteristic or feature of a data object.
 - E.g., *customer _ID, name, address*

人口學家張教授想研究已婚婦女學歷、就業與否、生育意願與小孩數的關係

姓名	小孩數	學歷	是否就業	結婚年齡	意願	身體狀況	忙碌與否
林文月	2	大學	是	27	是	好	忙
張文貞	3	專科	是	25	是	好	很忙
李玲玉	1	研究所	是	32	否	好	非常忙
郭月貞	2	高中	否	24	是	好	正常
陳宜君	1	國中	否	22	否	好	正常
朱昭如	2	高中	是	23	否	好	忙
劉秀貞	0	大學	是	26	是	差不多	很忙
林芳如	1	大學	否	25	是	好	非常忙
許美淑	0	研究所	是	30	否	不好	非常忙
鄭淑玲	2	高中	是	28	是	好	正常

Attributes

- ❑ Types:
 - ❑ Nominal (類別尺度) e.g., red, blue
 - ❑ Ordinal (順序尺度) e.g., freshman, sophomore, junior, senior
 - ❑ Numeric: quantitative
 - ❑ Interval-scaled (區間尺度): 100°C is interval scales
 - ❑ Ratio-scaled (比率尺度):
 100^{m} is ratio scaled since
it is twice as high as 50^{m}



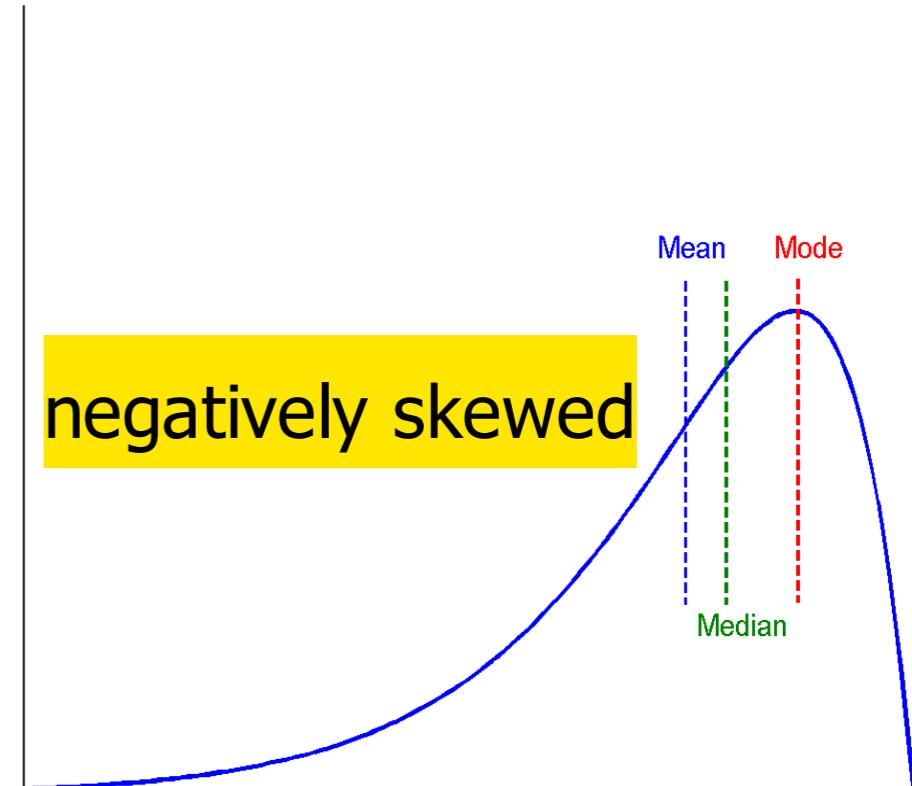
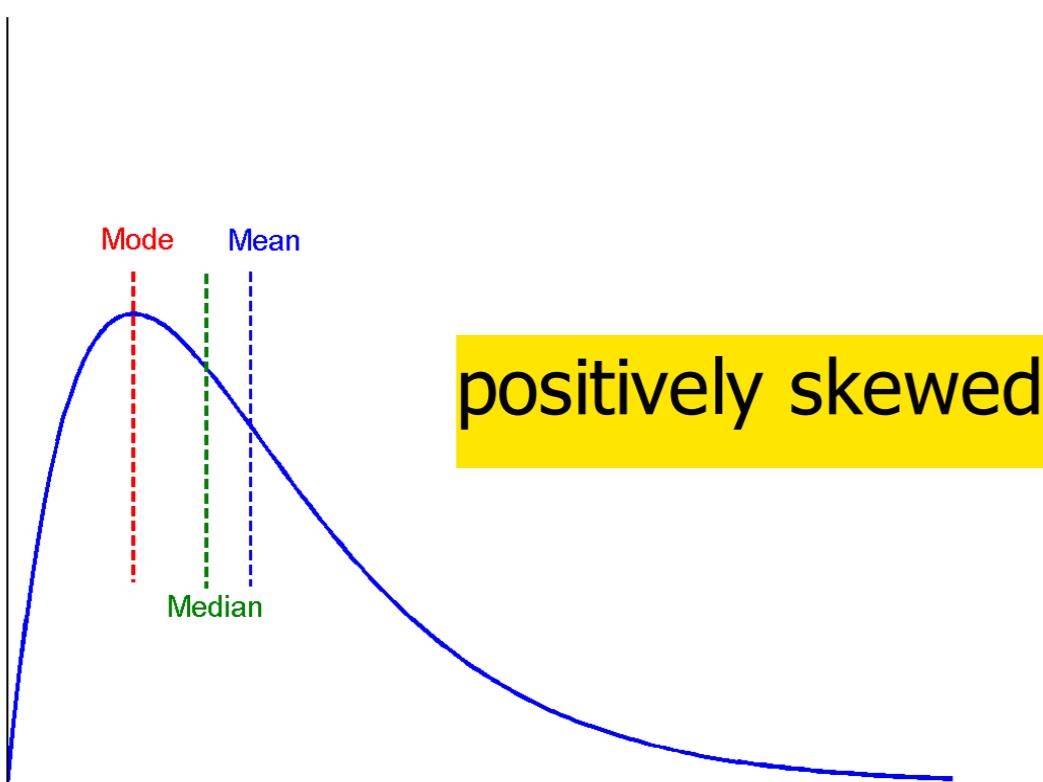
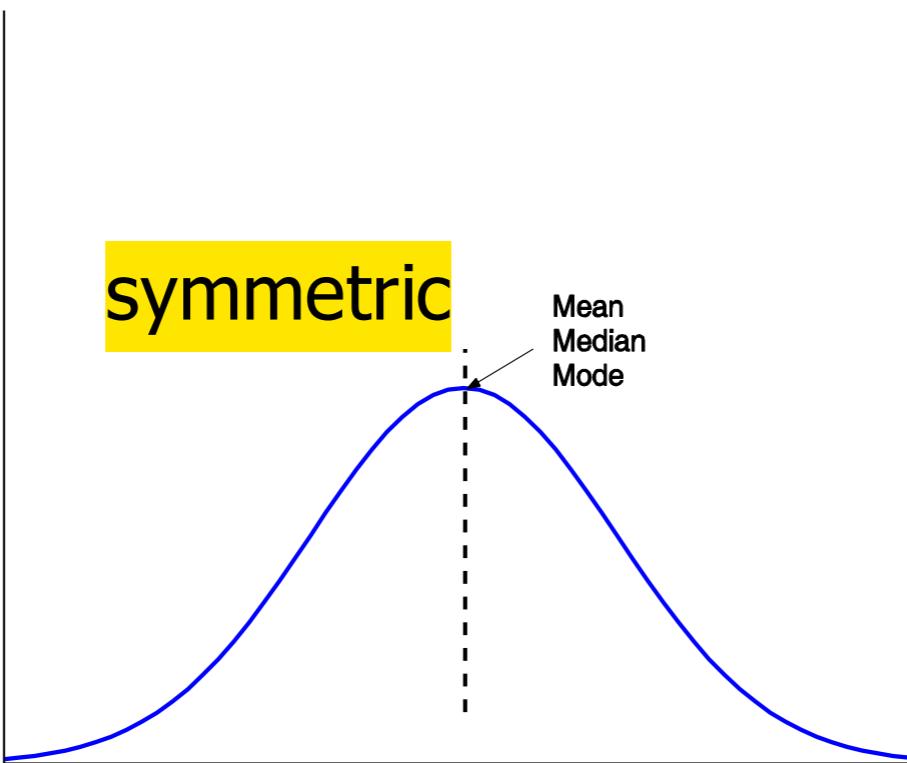
Attribute Types

- ❑ Nominal: categories, states, or “names of things”
 - ❑ *Hair_color* = {*auburn*, *black*, *blond*, *brown*, *grey*, *red*, *white*}
 - ❑ marital status, ID numbers, zip codes
- ❑ Binary (special case of Nomial)
 - ❑ Nominal attribute with only 2 states (0 and 1)
- ❑ Ordinal
 - ❑ Values have a meaningful order (ranking) but magnitude between successive values is not known
 - ❑ *Size* = {*small*, *medium*, *large*}, grades, army rankings

Attribute Types

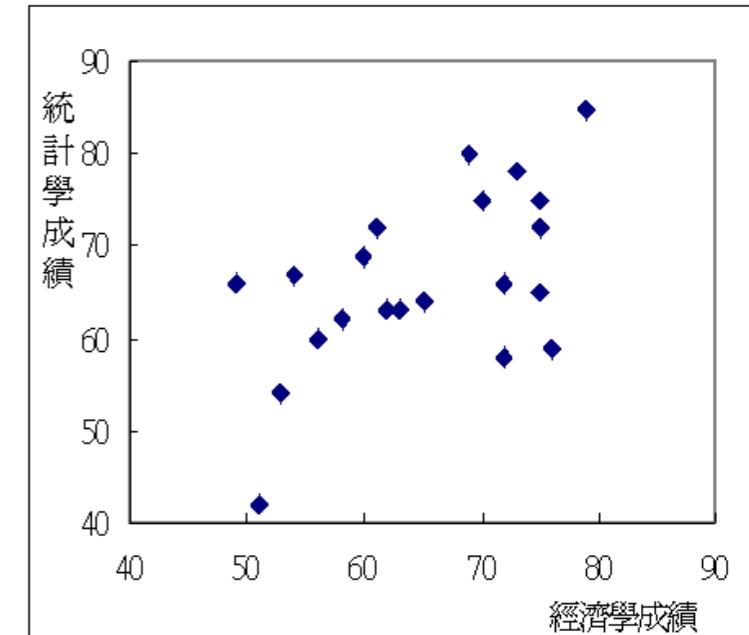
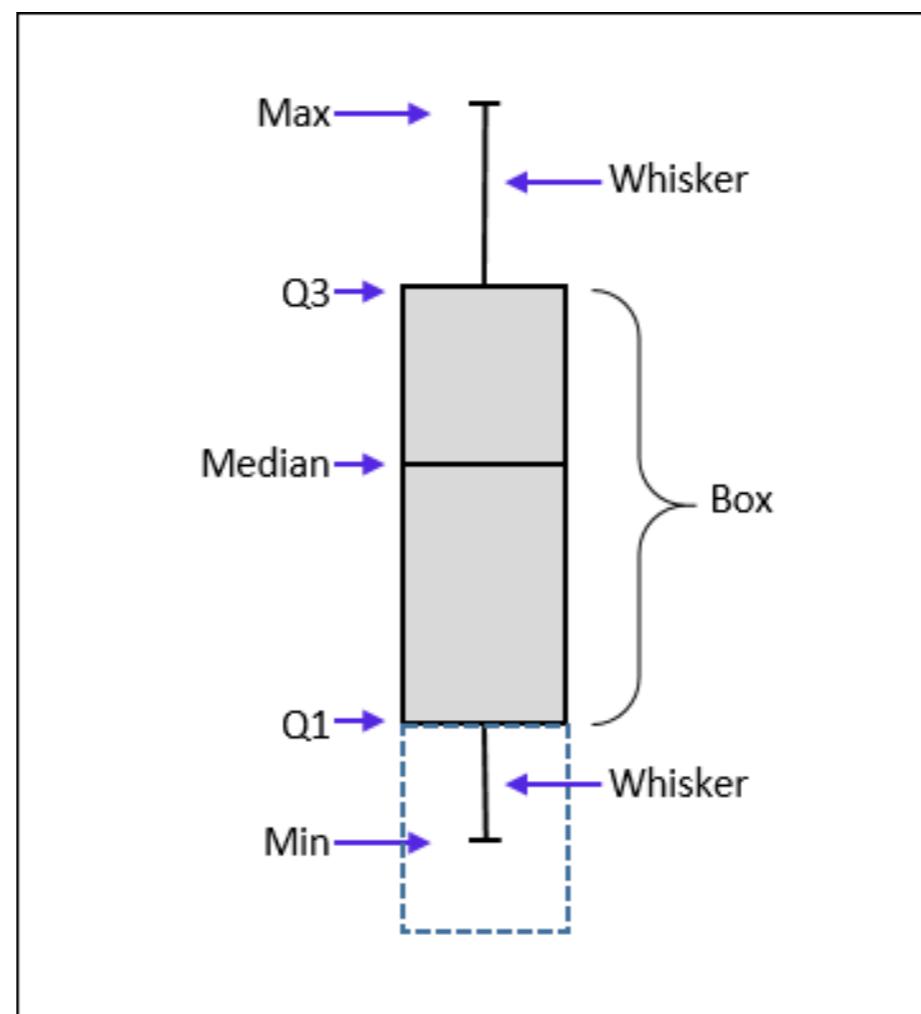
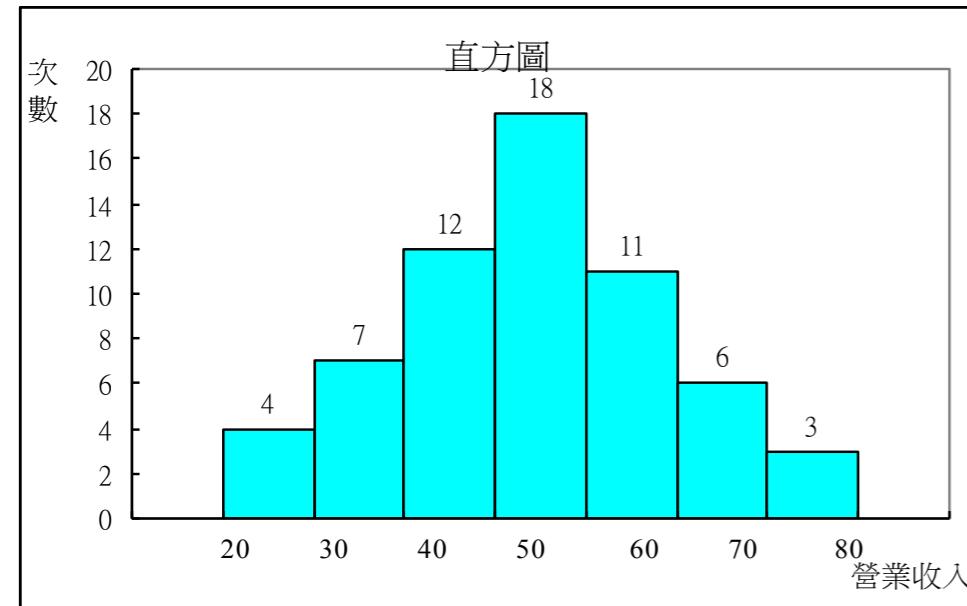
- ❑ Quantity (integer or real-valued)
- ❑ Interval (等距)
 - ❑ Measured on a scale of **equal-sized units**
 - ❑ Values have order
 - ❑ E.g., *temperature in C° or F°, calendar dates*
- ❑ Ratio (等比)
 - ❑ Inherent zero-point
 - ❑ We can speak of values as being an order of magnitude **larger** than the unit of measurement (10 m is twice as high as 5 m).
 - ❑ e.g., *length, counts, monetary quantities*

Symmetric vs. Skewed Data



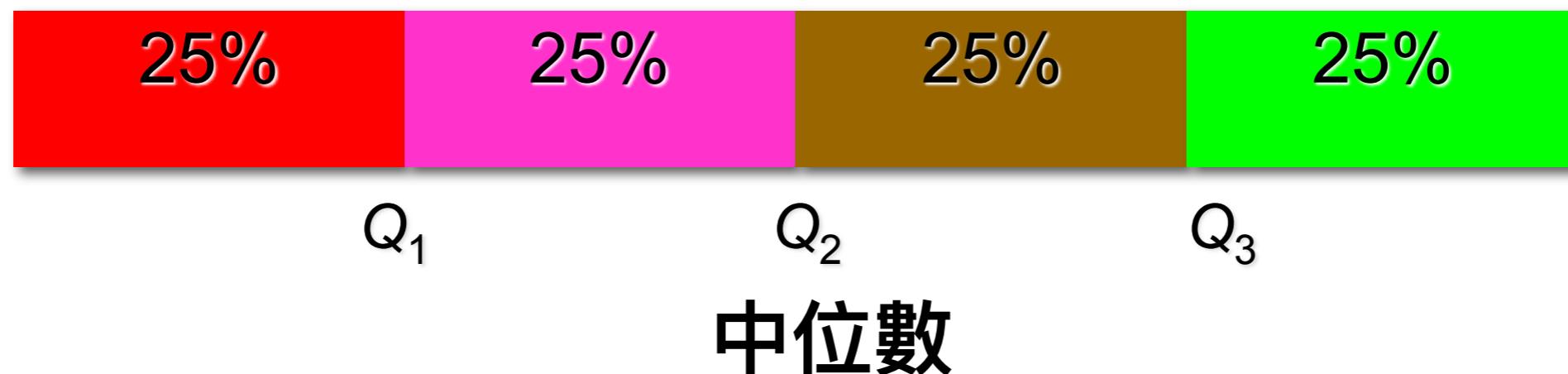
Graphic Displays of Basic Statistical Descriptions

- **Histogram**: x-axis are values, y-axis repres. frequencies
- **Boxplot**: graphic display of five-number summary
- **Scatter plot**: each pair of values is a pair of coordinates and plotted as points in the plane



四分位數

- 四分位數
- 資料由小到大排序



Boxplot

業務員業績

鍾樂水	30	63	66	78	82	96	106	270
朱碧霞	64	82	88	90	96	108	128	166

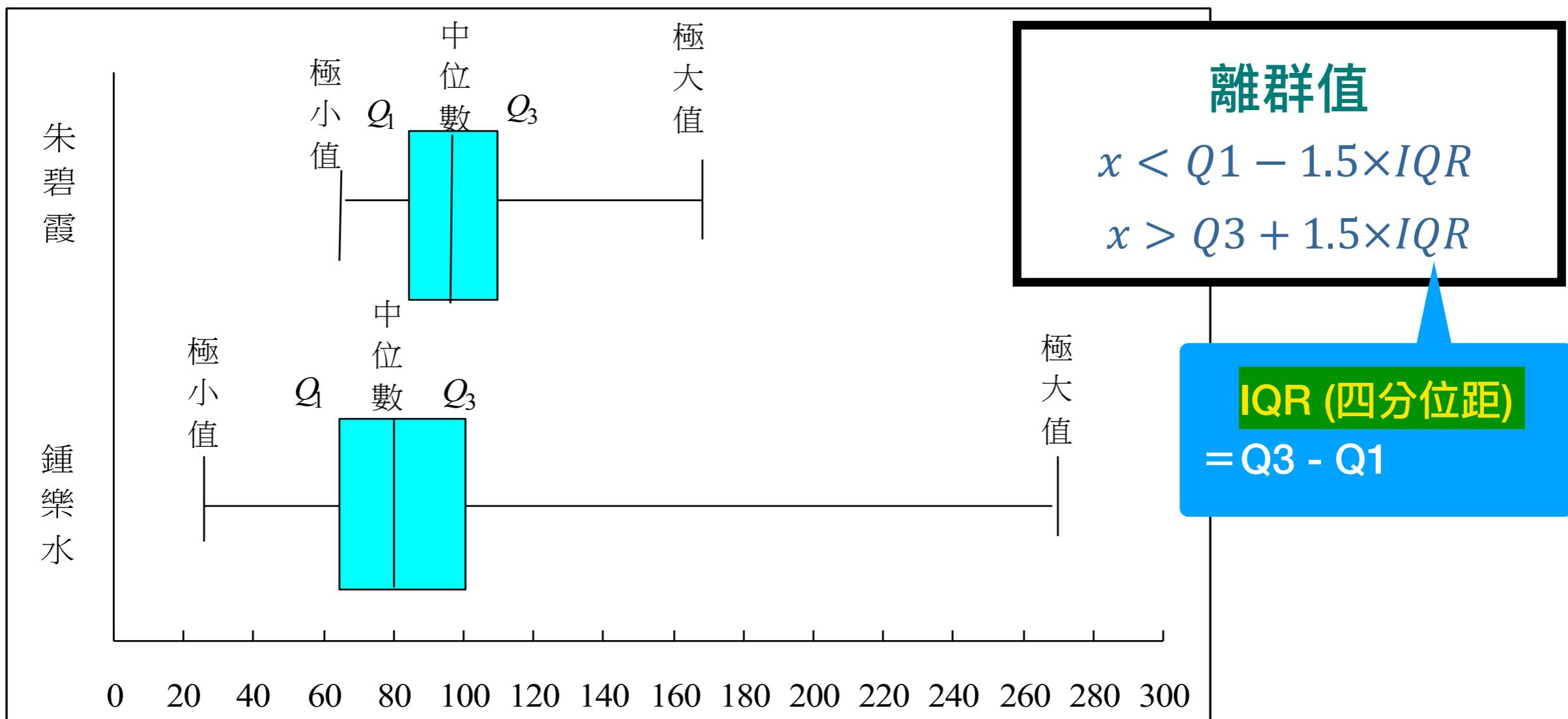
鍾樂水

min = 30, Q1 = 64.5, Median = 80, Q3 = 101, max = 270

Boxplot

業務員業績

鍾樂水	30	63	66	78	82	96	106	270
朱碧霞	64	82	88	90	96	108	128	166



常用距離公式

- $i=(x_{i1}, x_{i2}, \dots, x_{ip})$ 和 $j=(x_{j1}, x_{j2}, \dots, x_{jp})$ 是兩個 p 維資料對象
- 歐幾里得(Euclidean)距離

$$d(i, j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2)}$$

- 曼哈頓(Manhattan)距離

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

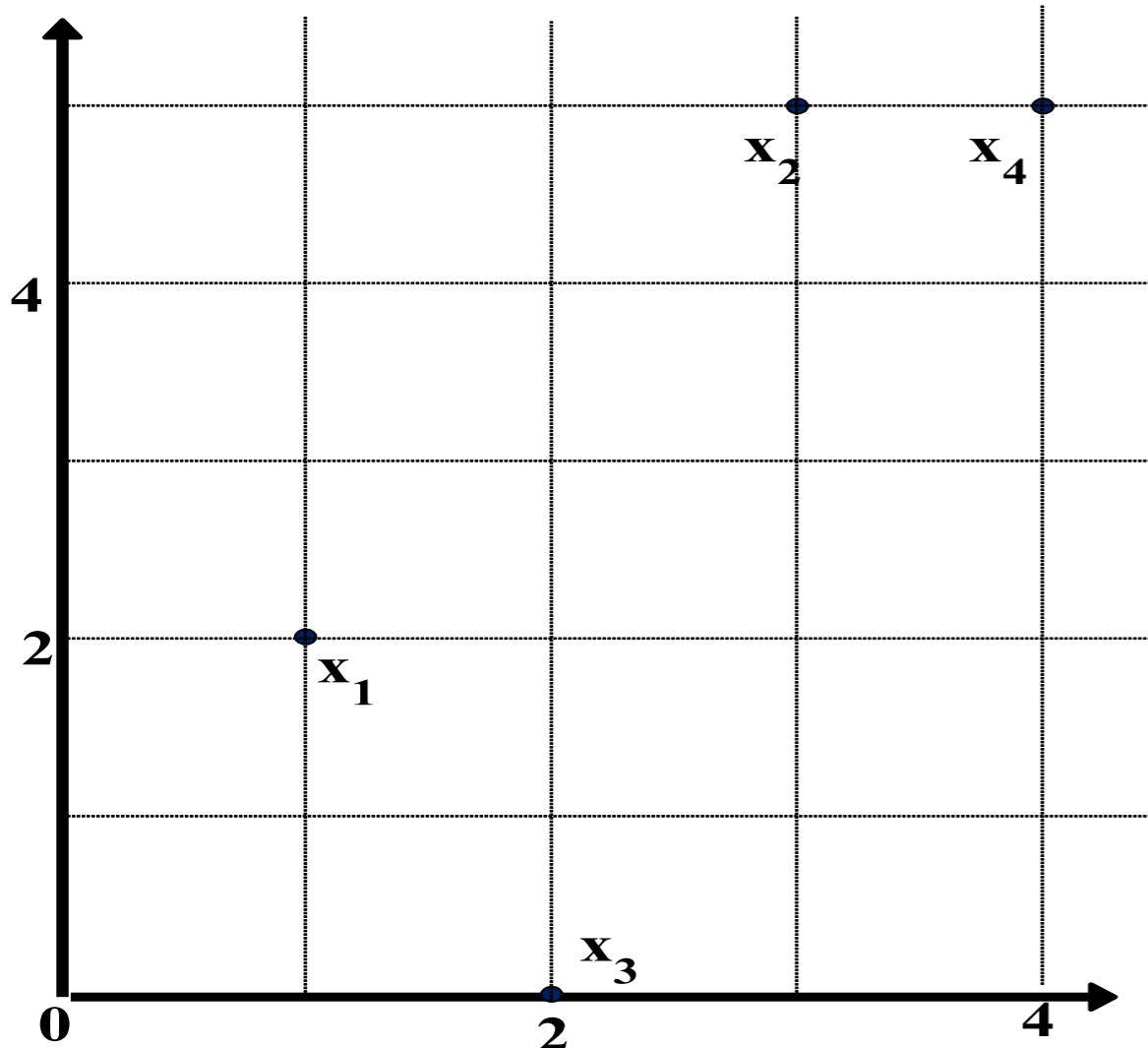
- 明可夫斯基(Minkowski)距離

$$d(i, j) = \sqrt[q]{(|x_{i1} - x_{j1}|^q + |x_{i2} - x_{j2}|^q + \dots + |x_{ip} - x_{jp}|^q)}$$

- 上式中， q 為正整數，如果 $q=1$ 則表示 Manhattan 距離，如果 $q=2$ 則表示 Euclidean 距離

Example

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum (L_∞)

L^∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

Machine Learning Process

Machine Learning Process

- 準備資料（包含資料預處理）
- 選擇演算法
- 調整參數
- 評估結果

Machine Learning Process

■ Iris Data Set (鳶尾花卉數據集)

<https://archive.ics.uci.edu/ml/datasets/Iris>

<https://www.kaggle.com/uciml/iris>

- 150個樣本，都屬於鳶尾屬下的三個亞屬，分別是山鳶尾、變色鳶尾和維吉尼亞鳶尾。

versicolor virginica

- 四個特徵: 花萼和花瓣的長度和寬度

Sepal Petal



準備資料

Attribute Information:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class:
 - Iris Setosa
 - Iris Versicolour
 - Iris Virginica

	A	B	C	D	E
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3	1.4	0.1	Iris-setosa
14	4.3	3	1.1	0.1	Iris-setosa
15	5.8	4	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa
22	5.1	3.7	1.5	0.4	Iris-setosa
23	4.6	3.6	1	0.2	Iris-setosa
24	5.1	3.3	1.7	0.5	Iris-setosa

準備資料

■ Loading the Data

```
# loading libraries
import pandas as pd

# define column names
names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

# loading training data
df = pd.read_csv('iris.data.txt', header=None, names=names)
```

準備資料

■ Loading the Data

```
# loading libraries
import pandas as pd

# define column names
names = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']

# loading training data
df = pd.read_csv('iris.data.txt', header=None, names=names)
```

■ Observing the data

```
# Observing the data
df.head()

df.info()
df.describe()
```

準備資料

■ Plotting graph

```
import matplotlib.pyplot as plt
import seaborn as sns

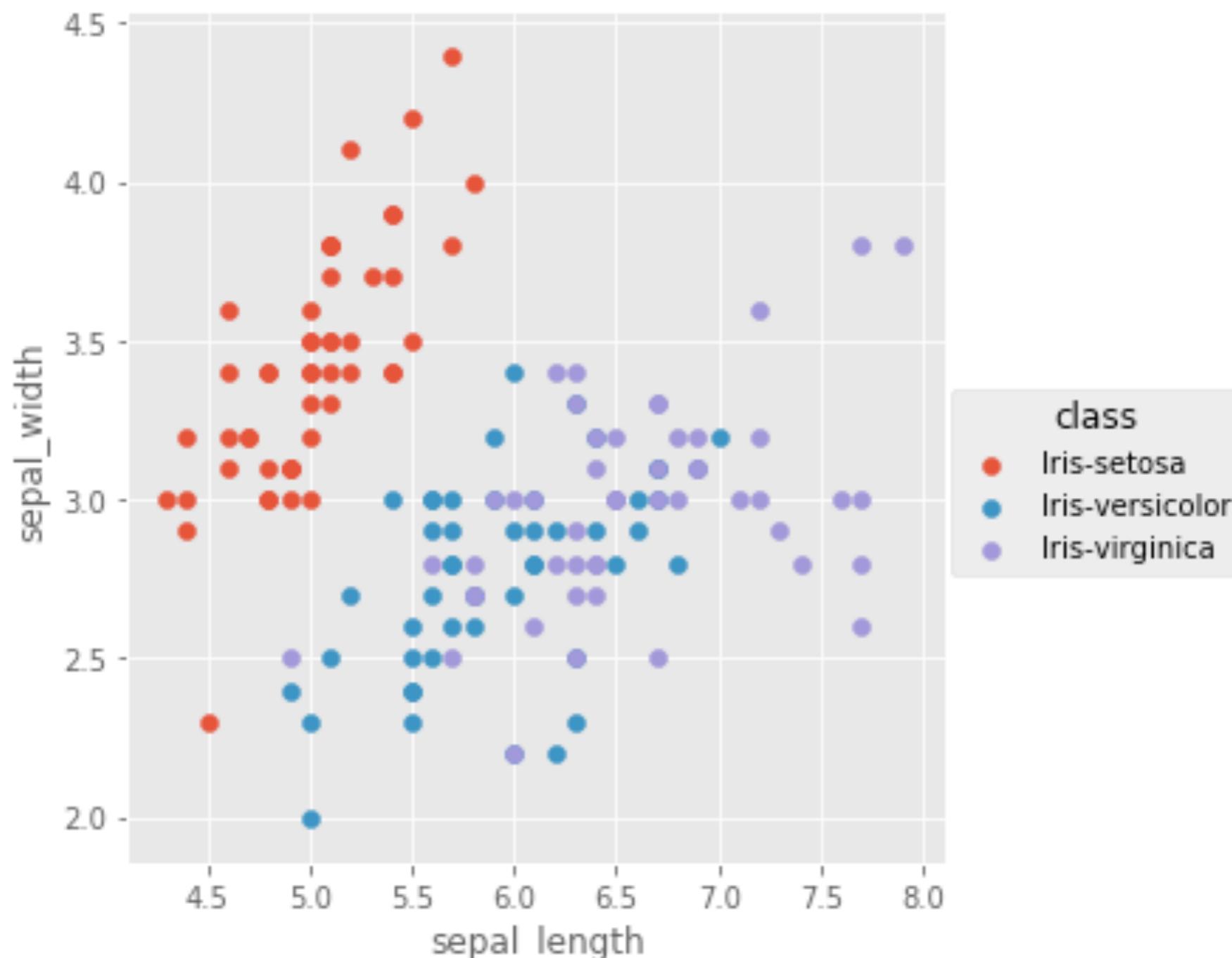
%matplotlib inline

plt.style.use('ggplot')
sns.lmplot("sepal_length", "sepal_width", data = df, fit_reg = False,
           hue = 'class')
```

準備資料

藉由圖形來輔助我們判斷
class 及其他特徵的關係

■ Plotting graph - 花萼長 v.s. 寬



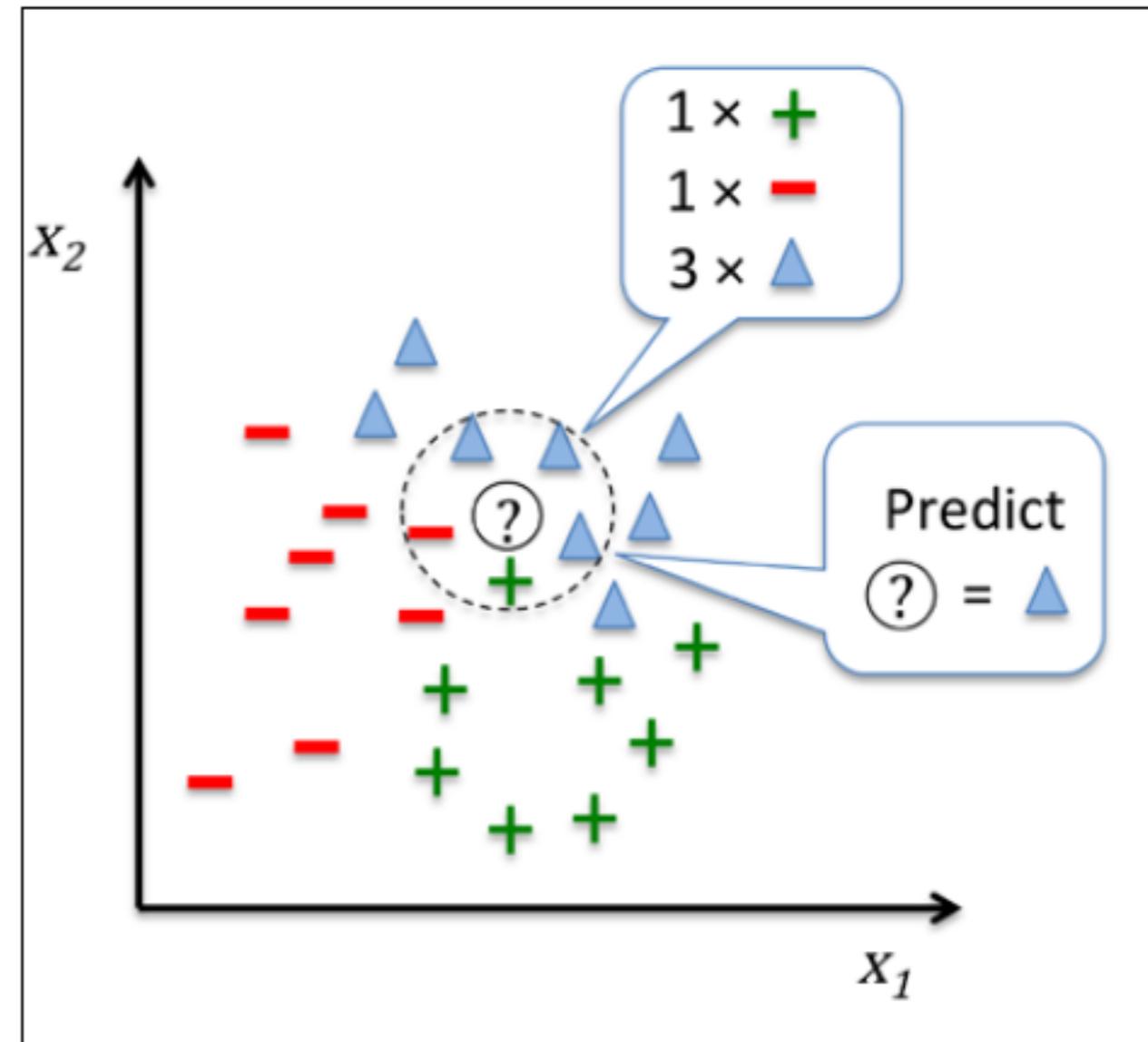
Machine Learning Process

- 準備資料（包含資料預處理）
- 選擇演算法
- 調整參數
- 評估結果

K-NN (K-nearest neighbor classifier)

$$d(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}$$

1. 確定k大小和距離度量($k = 5$)。
2. 對於測試集中的一個樣本，找到訓練集中和它最近的k個樣本。
3. 將這k個樣本的投票結果作為測試樣本的類別 (多數決)。



<https://ljalphabeta.gitbooks.io/python-/content/knn.html>

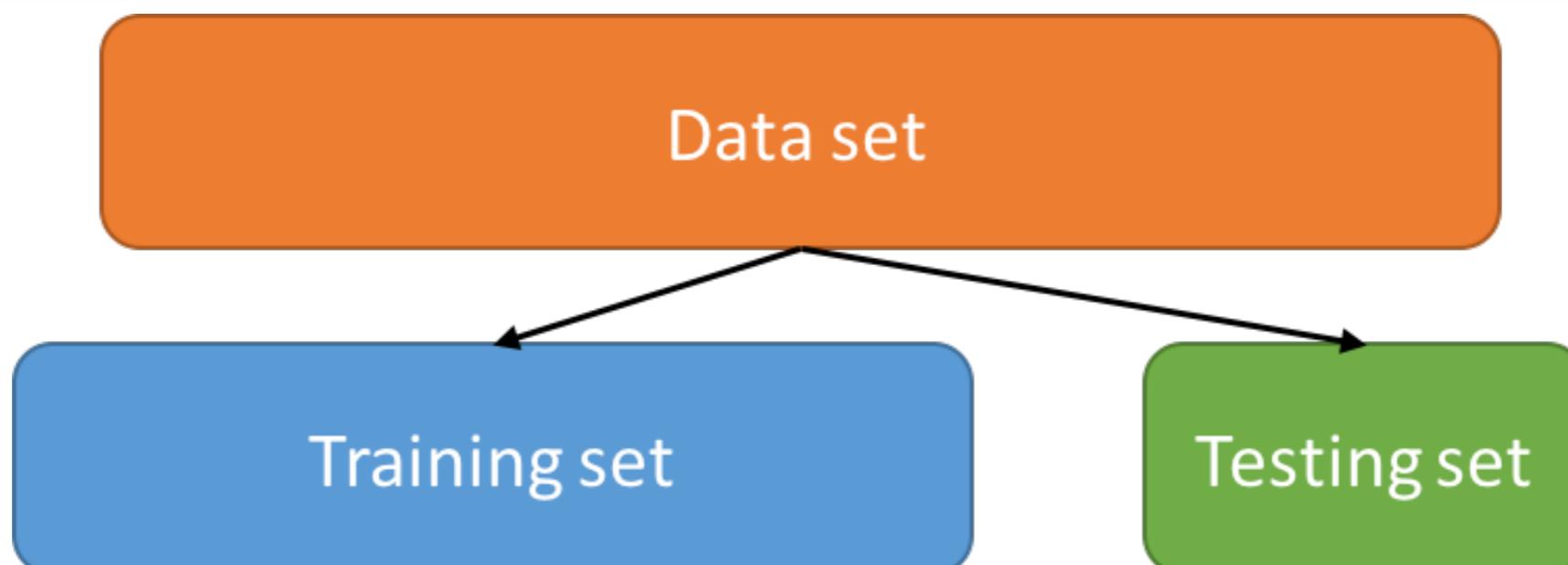
選擇演算法

■ Split into train and test

```
# loading libraries
import numpy as np
from sklearn.cross_validation import train_test_split

# create design matrix X and target vector y
X = df.iloc[:, :-1].values
y = df.iloc[:, 4].values

# split into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```



選擇演算法

■ K-NN

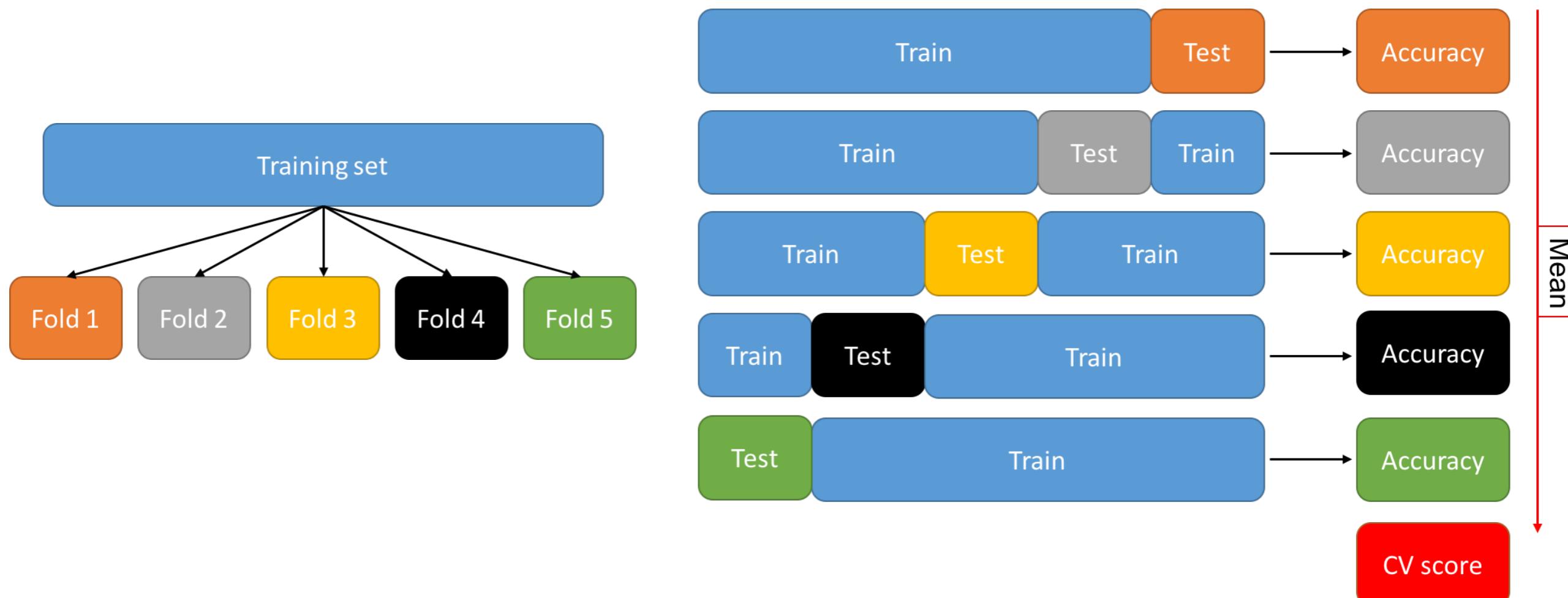
```
from sklearn.neighbors import KNeighborsClassifier  
  
# instantiate learning model ( $k = 3$ )  
knn = KNeighborsClassifier(n_neighbors=3)  
  
# fitting the model  
knn.fit(X_train, y_train)  
  
# predict the response  
pred = knn.predict(X_test)  
  
# evaluate accuracy  
from sklearn.metrics import accuracy_score  
accuracy_score(y_test, pred)
```

Machine Learning Process

- 準備資料（包含資料預處理）
- 選擇演算法
- 調整參數
- 評估結果

調整參數 & 評估結果

■ cross-validation



Source: <https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/>

調整參數 & 評估結果

```
from sklearn.model_selection import cross_val_score  
  
scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
```

資料預處理

Data pre-processing

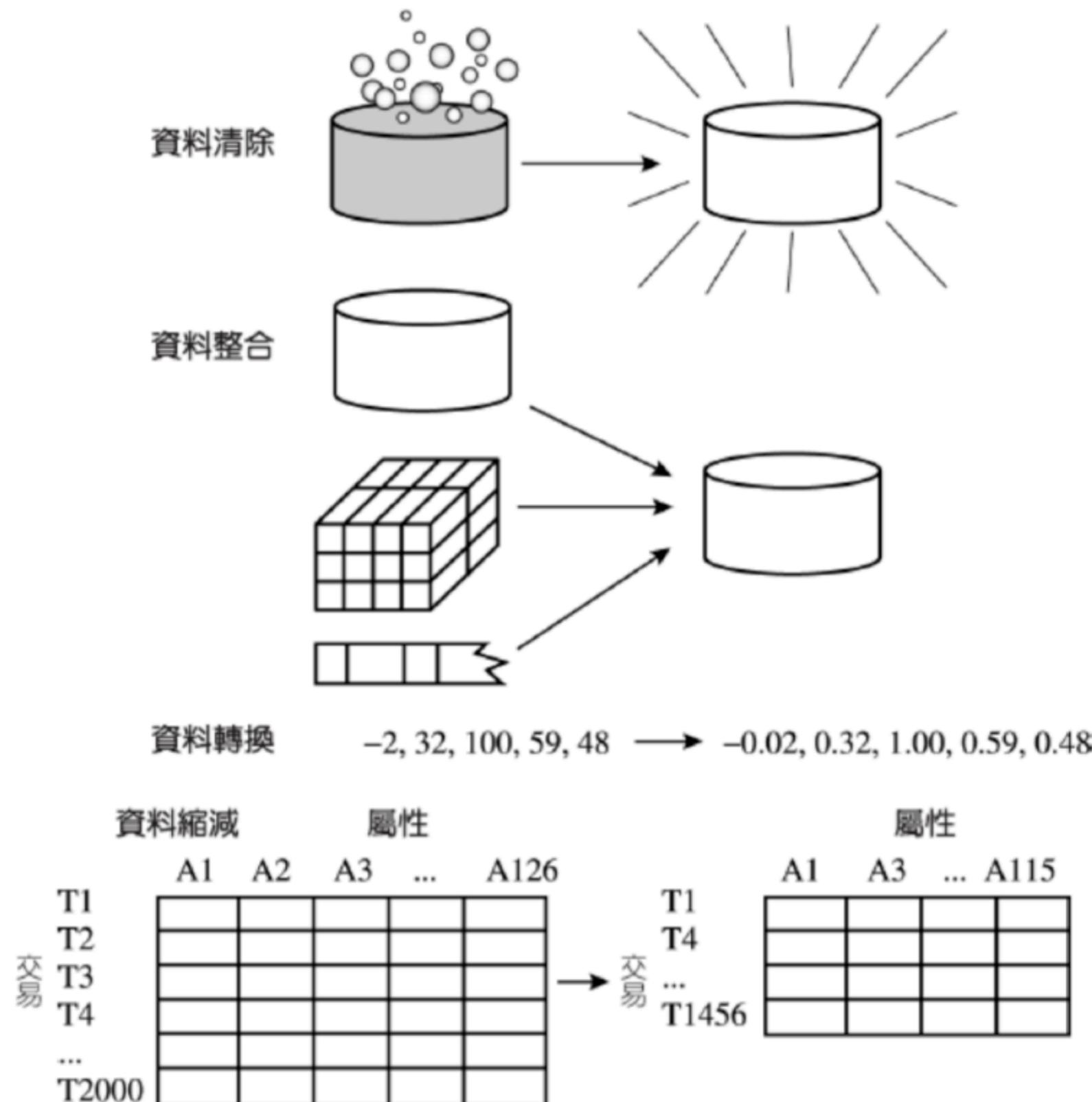
Data preprocessing

- 真實世界的資料是不乾淨(Dirty) – 資料多了，什麼問題都會出現...
 - 不完整(Incomplete):缺少屬性值或某些有興趣的屬性，或僅包含聚集資料。例如：
 - 職業=“ ”，公司人數=20 (無公司人員資料)
 - 有雜訊(Noise):包含錯誤或離異值。例如：
 - 薪資="-10"
 - 不一致(Inconsistent):在資料本身或命名上不一致。例如：
 - 年齡="15" 生日="03/07/1997"
 - 過去打成績 "0~100" 百分制；現在打成績 "A, B, C, D, E" 五等第
 - 重複性的紀錄所造成，如: Customer_ID, Customer_Num
- 沒有高品質的資料，就沒有高品質的探勘結果。

Data preprocessing

- 資料清理 (Data Cleaning)
 - 填入**遺失值**、找出且移除**離異值(雜訊)**、解決**不一致**
- 資料整合 (Data Integration)
 - 整合不同來源的資料庫或檔案
- 資料轉換 (Data Transformation)
 - **標準化與聚合**
- 資料縮減 (Data Reduction)
 - 得到**資料集的壓縮表示**，降低資料量的大小，並預期能獲得相同或近似的分析結果
- 上述方法不一定是個別使用，也可以結合一起來使用。

Data preprocessing



Data description

- 資料預處理如果要成功，對於資料要有全盤的概念是很重要的。而敘述資料彙總 (Descriptive Data Summarization) 用於說明資料整體特性，並指出哪些資料值為雜訊或離異值。
- 使用者想要知道的資料整體特性：
 - 主要傾向
 - 平均值 (Mean)
 - 中位數 (Median)
 - 眇數 (Mode)
 - 資料散佈
 - 四分位數 (Quartiles)
 - 變異數 (Variance)

Data cleaning

■ 資料清理工作

- 填補**遺失值**
- 找出**Outliers**並淡化(平滑)**雜訊**
- 修正**資料的不一致**
- 解決**資料整合所造成重複**

How to handle missing value?

■ 忽略這些值組：

- 通常用於進行分類時，值組的類別資料是遺失的狀況。
- 這個方法不是很有效，除非許多值組的屬性包含遺失值。
- 若具有遺失值的屬性，其所存放資料對分析結果有重大影響時，效果會很差。
- 例如，從事“客戶是否購買房車”的分類工作時：
 - 某客戶的類別資料未被記錄到。
 - 某一筆客戶僅存客戶編號、姓名，其它皆沒有。

■ 利用人工方式填入遺失值：

- 非常費時並不實際。

編號	性別	年齡	婚姻	家庭所得	購買RV 房車
A0001	Male	<35	未婚	高所得	否
A0002	Male	<35	未婚	小康	?
A0003	Female	≥35	已婚	高所得	是
A0004	Male	≥35	未婚	低所得	是
A0005	Female	≥35	已婚	高所得	否
A0006	Male	≥35	已婚	低所得	否
A0007	Female	?	?	?	否
A0008	Male	≥35	已婚	高所得	是
A0009	Male	<35	已婚	低所得	是

How to handle missing value?

■ 自動填入：

- 利用**全域常數 (Global Constant)** 填入遺失值：例., “未知”, “ $\pm\infty$ ”
- 使用**屬性平均值**來填入遺失值
- 使用**相同類別值組的屬性均值**
- 使用**最有可能的值**來填入遺失值
 - 可以透過迴歸、利用貝氏理論的推論式工具或決策樹推論來決定

pandas

■ csv (comma-separated values) 檔

```
In [1]: import pandas as pd  
  
df = pd.read_csv("data.csv")  
df
```

Out[1]:

	A	B	C	D
0	1.0	2.0	3.0	4.0
1	5.0	6.0	NaN	8.0
2	0.0	11.0	12.0	NaN

遺漏值處理

■ 是否有遺漏值

```
In [2]: df.isnull()
```

Out[2]:

	A	B	C	D
0	False	False	False	False
1	False	False	True	False
2	False	False	False	True

```
In [3]: df.isnull().sum()
```

Out[3]:

A 0
B 0
C 1
D 1
dtype: int64

遺漏值處理

■ 刪除具有遺漏值的樣本或特徵

```
In [4]: df.dropna(axis=0) # 刪除有遺漏值的列
```

Out[4]:

	A	B	C	D
0	1.0	2.0	3.0	4.0

```
In [5]: df.dropna(axis=1) # 刪除有遺漏值的行
```

Out[5]:

	A	B
0	1.0	2.0
1	5.0	6.0
2	0.0	11.0

遺漏值處理

- 只刪除某特定特徵有遺漏值的樣本

```
In [6]: df.dropna(subset=['C'])
```

Out[6]:

	A	B	C	D
0	1.0	2.0	3.0	4.0
2	0.0	11.0	12.0	NaN

填補遺漏值

■ 用平均值

median: 中位數
most_frequent: 眇數

```
In [8]: from sklearn.preprocessing import Imputer  
  
imr = Imputer(missing_values='NaN', strategy='mean', axis=0)  
imr = imr.fit(df.values)  
imputed_data = imr.transform(df.values)  
imputed_data
```

```
Out[8]: array([[ 1. ,  2. ,  3. ,  4. ],  
   [ 5. ,  6. ,  7.5,  8. ],  
   [ 0. , 11. , 12. ,  6. ]])
```

help(Imputer): 查說明文件

K-Nearest Neighbor example

	EP1	EP2	EP3	EP4	EP5
Gene1	1.0	2.0	-1.0	-2.0	1.5
Gene2	-1.0	-1.0	2.0	1.5	-1.0
Gene3	?	-2.0	1.5	2.0	2.0
Gene4	-2.0	2.0	2.0	-1.0	2.0

Diagram illustrating the calculation of distances from a query vector (EP1) to four data points (EP2, EP3, EP4, EP5). The distances are:

- $d_1 = 6.18$ (red line)
- $d_2 = 3.24$ (blue line)
- $d_3 = 5.02$ (green line)

$$\text{weight(Gene1)} : \frac{1}{6.18}$$

$$\text{weight(Gene2)} : \frac{1}{3.24}$$

$$\text{weight(Gene3)} : \frac{1}{5.02}$$

Choose K=3

K-Nearest Neighbor example

	EP1	EP2	EP3	EP4	EP5
Gene1	1.0	2.0	-1.0	-2.0	1.5
Gene2	-1.0	-1.0	2.0	1.5	-1.0
Gene3	?	-2.0	1.5	2.0	2.0
Gene4	-2.0	2.0	2.0	-1.0	2.0

d1 = 6.18
d2 = 3.24
d3 = 5.02

The weight between Gene A and Gene X : $Weight(Gene X) = \frac{1/D_{AX}}{1/D_{AB} + 1/D_{AC} + \dots + 1/D_{AK}}$

$$weight(Gene1) = \frac{\frac{1}{6.18}}{\frac{1}{6.18} + \frac{1}{3.24} + \frac{1}{5.02}} = 0.24$$

Choose K=3

0.24

0.46

0.3

$$\text{Gene3_EP1} = 1.0 * weight(Gene1) + (-1.0) * weight(Gene2) + (-2.0) * weight(Gene3) = -1$$

How to handle noisy data?

- 箱狀法
- 迴歸
- 聚類
- 整合電腦與人檢驗
 - 利用人檢測有疑問的值 (例., 處理可能離異值)

How to handle noisy data? binning

- 將資料進行排序，再利用其鄰居的值進行平滑化。
 - ① 排序後的資料先分成數個頻率相同**箱子**(Buckets, Bins)
 - ② 然後對箱子內的資料進行**平滑化**
- 由於箱狀法是考量箱內資料之“鄰居”(Neighborhood)的值，因此它是屬於**區域平滑化**(Local Smoothing)的方法。
- 箱子的寬度愈大，結果越平滑。

Binning

儲存價格資料(\$) : 4, 8, 15, 21, 21, 24, 25, 28, 34

步驟①：分割成(等頻率)箱子

箱子 1 : 4, 8, 15

箱子 2 : 21, 21, 24

箱子 3 : 25, 28, 34

步驟②：用箱子均值平滑化 **利用箱子的平均值/中間值平滑化**

箱子 1 : 9, 9, 9

箱子 2 : 22, 22, 22

箱子 3 : 29, 29, 29

用箱子邊界值平滑化

箱子 1 : 4, 4, 15

箱子 2 : 21, 21, 24

箱子 3 : 25, 25, 34

利用箱子的邊界值 (Bin Boundaries) 平滑化

⇒ 以每個箱子中的最大/最小值(邊界值)為基準，將箱子中的其它資料以最接近的邊界值來加以取代。

Binning

■ 將資料分箱的作法有兩種：

□ 等寬(距離)分割

- 切割成N個等寬範圍的箱子
- 易受離異值主導
- 對偏斜資料處理較差

□ 等深(頻率)分割

- 切割成有大約相同樣本數的箱子
- 具好資料量度性

前述範例資料之等寬分割：

箱號	箱內元素	箱邊界
箱子 1	4, 8	[-, 10)
箱子 2	15	[10, 20)
箱子 3	21, 21, 24, 25, 28	[20, 30)
箱子 4	34	[30, +)

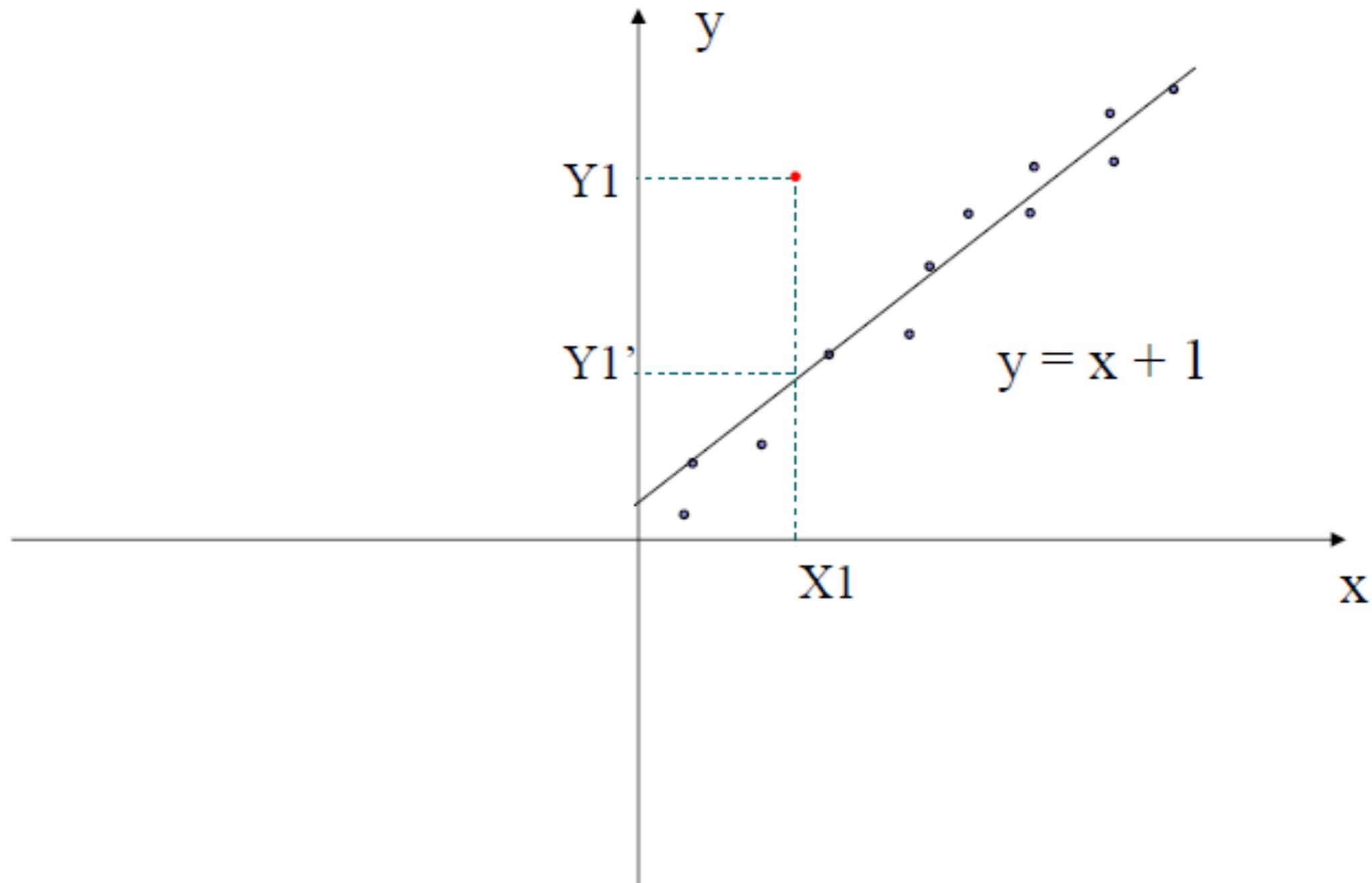
前述範例資料之等深分割：

箱號	箱內元素	箱邊界
箱子 1	4, 8, 15	[-, 20)
箱子 2	21, 21, 24	[20, 25)
箱子 3	25, 28, 34	[25, 35)

Regression

■ 迴歸

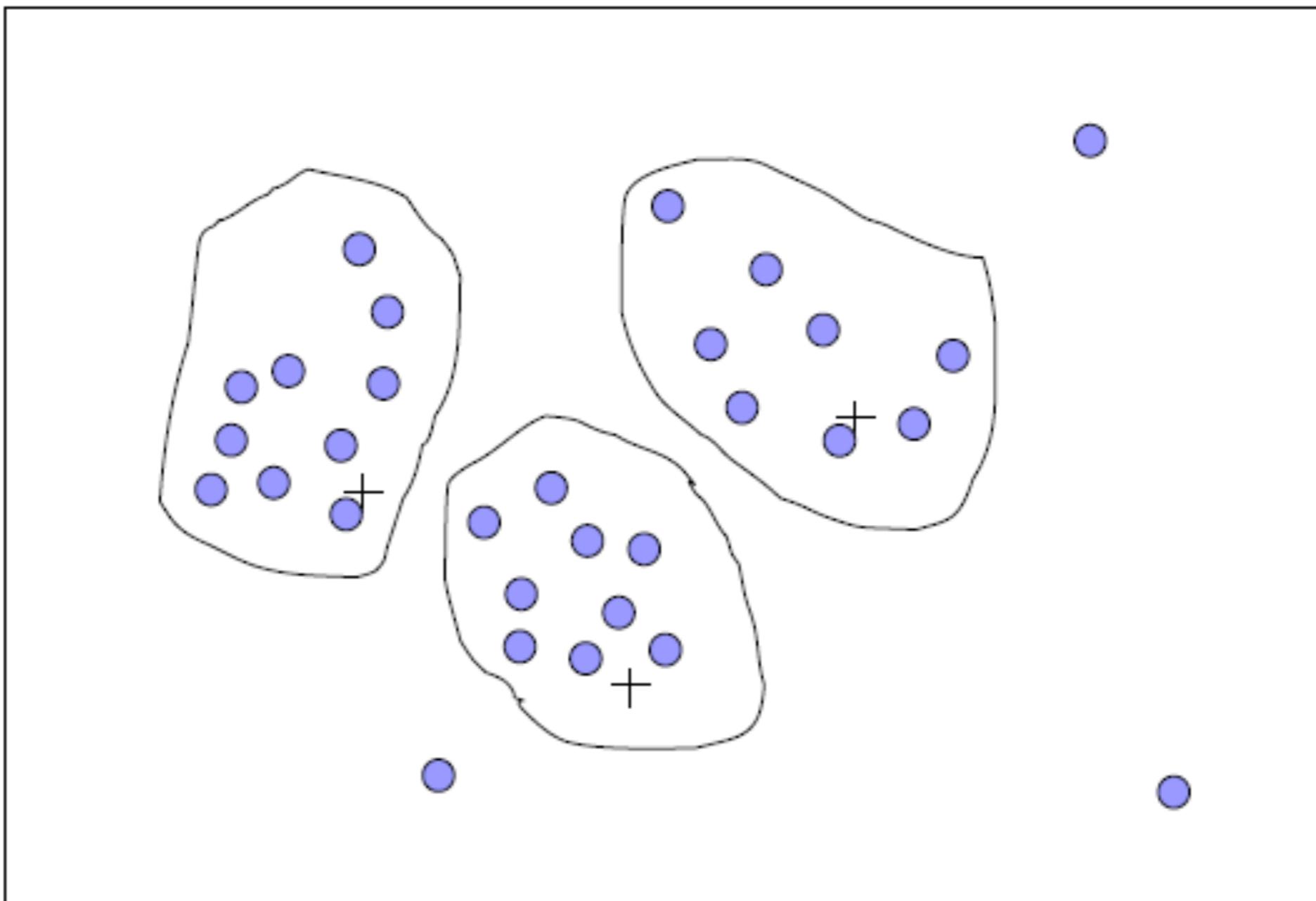
- 透過將資料分佈情況對應至函數來進行平滑化



Clustering

■ 聚類

□ 偵測並移除離異值



Data preprocessing

- 資料清理
- 資料整合
- 資料轉換
- 資料縮減

資料整合

■ 資料整合：

- 將許多來源的資料整合成一個具連貫性的資料

■ 多餘資料

- 一個屬性可由其它單一屬性或一組屬性導出，則此屬性是多餘的。
- 屬性不一致或屬性命名不一致也會造成資料的多餘。

資料整合

- 資料整合：
 - 將許多來源的資料整合成一個具連貫性的資料
- 多餘資料
 - 一個屬性可由其它單一屬性或一組屬性導出，則此屬性是多餘的。
 - 屬性不一致或屬性命名不一致也會造成資料的多餘。
- 發掘並解決資料值衝突問題
 - 真實世界的個體，屬性值會來自不同來源
 - 可能原因：不同表示，不同量化，例.，公制與英制

Data preprocessing

- 資料清理
- 資料整合
- 資料轉換
- 資料縮減

■

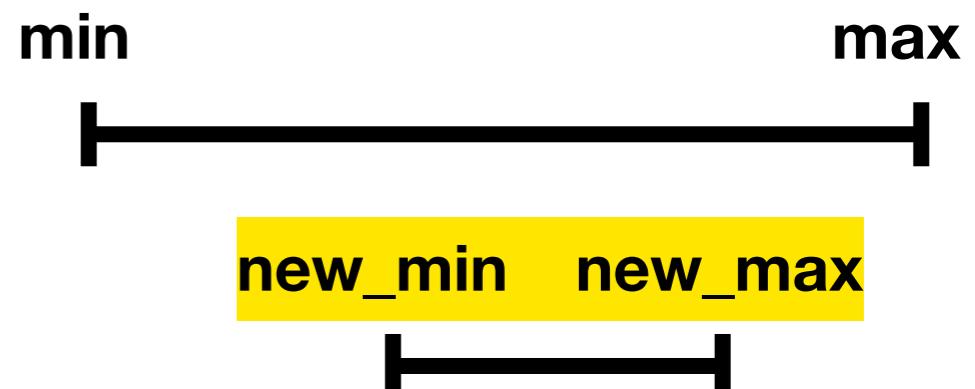
資料轉換：標準化

- min-max標準化: 轉換至 $[new_min_A, new_max_A]$

$$v' = \frac{v - min_A}{max_A - min_A} (new_max_A - new_min_A) + new_min_A$$

- 例. 將收入範圍 \$12,000 至 \$98,000 標準化至 [0, 1]，則
\$73,600：

$$\frac{73,600 - 12,000}{98,000 - 12,000} (1.0 - 0) + 0 = 0.716$$



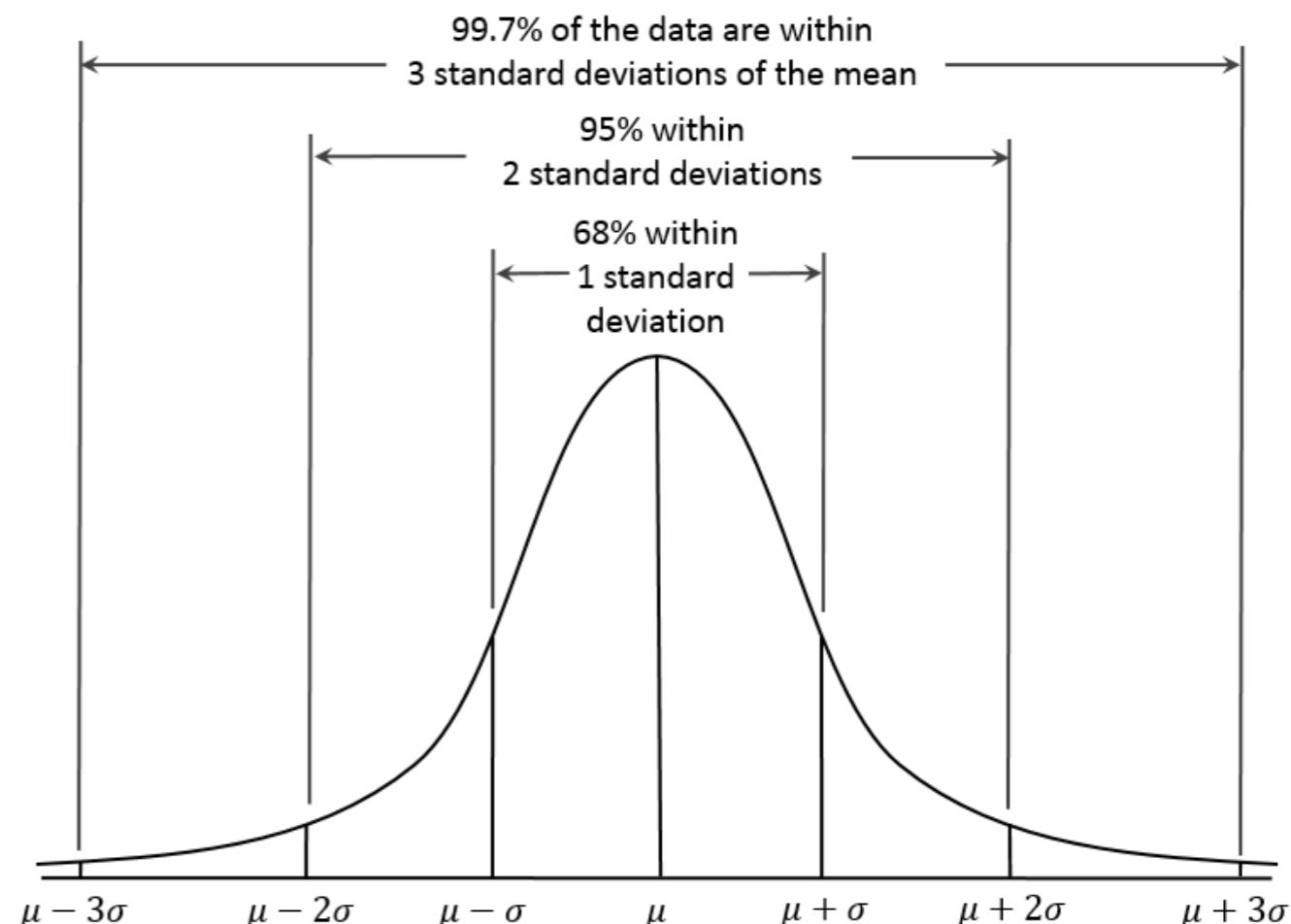
Data normalization

- Z-score標準化 (μ : 均值, σ : 標準差):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

- 此標準化做法可使所有標準化後的值，其平均值為0。
- 例. 當 $\mu = \$54,000$, $\sigma = \$16,000$. 則透過Z-score標準化，
輸入值\$73600會轉換為：

$$\frac{73,600 - 54,000}{16,000} = 1.225$$



Data normalization

■ 十進位標準化

$$v' = \frac{v}{10^j} \quad , \quad j \text{ 為可使 } \text{Max}(|v'|) < 1 \text{ 之最小整數}$$

- 例：假設屬性A的資料值範圍為-986到917，屬性A中的最大絕對值為986。現要進行十進位標準化，我們將每個值除以1000 (也就是 $j = 3$)，所以-986會標準化為-0.986，917會標準化為0.917。

處理非數值型數據

```
In [1]: import pandas as pd  
  
df = pd.DataFrame([['green', 'M', 10.1, 'class1'],  
                   ['red', 'L', 13.5, 'class2'],  
                   ['blue', 'XL', 15.3, 'class1']])  
  
df.columns = ['color', 'size', 'price', 'classlabel']  
df
```

Out[1]:

	color	size	price	classlabel
0	green	M	10.1	class1
1	red	L	13.5	class2
2	blue	XL	15.3	class1

順序特徵

■ 定義對應字典 (mapping dictionary)

```
In [2]: size_mapping = {'XL': 3,  
                      'L': 2,  
                      'M': 1}  
  
df['size'] = df['size'].map(size_mapping)  
df
```

Out[2]:

	color	size	price	classlabel
0	green	1	10.1	class1
1	red	2	13.5	class2
2	blue	3	15.3	class1

順序特徵

■ 將整數值轉回原始字串

```
In [3]: inv_size_mapping = {v: k for k, v in size_mapping.items()}
df['size'].map(inv_size_mapping)
```

```
Out[3]: 0    M
         1    L
         2   XL
Name: size, dtype: object
```

類別標籤編碼

- 將類別標籤編碼為整數值
 - 方法1：定義對應字典

enumerate: 產生許多 (key, value) 的 tuple

In [5]: `import numpy as np`

```
class_mapping = {label: idx for idx, label in enumerate(np.unique(df['classlabel']))}  
class_mapping
```

Out[5]: `{'class1': 0, 'class2': 1}`

In [6]: `df['classlabel'] = df['classlabel'].map(class_mapping)`
`df`

Out[6]:

	color	size	price	classlabel
0	green	1	10.1	0
1	red	2	13.5	1
2	blue	3	15.3	0

類別標籤編碼

- 將類別標籤編碼為整數值
 - 方法2：使用 LabelEncoder 類別

```
In [9]: from sklearn.preprocessing import LabelEncoder  
  
class_le = LabelEncoder()  
y = class_le.fit_transform(df['classlabel'].values)  
y  
  
Out[9]: array([0, 1, 0])
```

類別標籤編碼

- 將類別標籤編碼為整數值
 - 方法2：使用 LabelEncoder 類別

```
In [9]: from sklearn.preprocessing import LabelEncoder
```

```
class_le = LabelEncoder()  
y = class_le.fit_transform(df['classlabel'].values)  
y
```

```
Out[9]: array([0, 1, 0])
```

轉回原始類別標籤：

```
class_le.inverse_transform(y)
```

對名目特徵進行編碼

■ 方法1：使用 LabelEncoder

```
In [11]: X = df[['color', 'size', 'price']].values  
  
color_le = LabelEncoder()  
X[:, 0] = color_le.fit_transform(X[:, 0])  
X
```

```
Out[11]: array([[1, 1, 10.1],  
                 [2, 2, 13.5],  
                 [0, 3, 15.3]], dtype=object)
```

blue = 0, green = 1, red = 2

對類別特徵進行編碼

- 方法2：one-hot encoding: 將每個值轉為一個新的虛擬特徵
 - 方法 2-1：利用OneHotEncoder

```
In [12]: from sklearn.preprocessing import OneHotEncoder
```

```
ohe = OneHotEncoder(categorical_features=[0])
ohe.fit_transform(X).toarray()
```

```
Out[12]: array([[ 0. ,  1. ,  0. ,  1. , 10.1],
   [ 0. ,  0. ,  1. ,  2. , 13.5],
   [ 1. ,  0. ,  0. ,  3. , 15.3]])
```

對類別特徵進行編碼

- 方法2：one-hot encoding: 將每個值轉為一個新的虛擬特徵
- 方法 2-2：利用 pandas 套件

```
In [14]: pd.get_dummies(df[['price', 'color', 'size']])
```

Out[14]:

	price	size	color_blue	color_green	color_red
0	10.1	1	0	1	0
1	13.5	2	0	0	1
2	15.3	3	1	0	0

對類別特徵進行編碼

- 方法 2-3：利用 pandas 套件，並刪除相關的特徵行

```
In [15]: pd.get_dummies(df[['price', 'color', 'size']], drop_first=True)
```

Out[15]:

	price	size	color_green	color_red
0	10.1	1	1	0
1	13.5	2	0	1
2	15.3	3	0	0

Data preprocessing

- 資料清理
- 資料整合
- 資料轉換
- 資料縮減

Data reduction

■ 為何要資料縮減？

- 資料倉儲中選擇進行分析的資料量極為龐大
- 複雜的資料分析與探勘龐大的資料會花費很長的時間

■ 資料縮減

- 希望獲得一個**比原始資料小很多的縮減資料集**，並且它幾乎能保持原始資料的完整性

■ 資料縮減策略

- 資料聚合：對資料進行聚合運算。
- 子屬性集選擇：偵測並移除不相關、低相關、不重要的屬性。
- 維度縮減：利用編碼方式來降低資料集大小。
- 數值縮減：利用其它較小資料表示法來表示資料。例：將資料以模型表示、以分群、取樣方式進行取代。
- 離散化或概念階層建立：將原始資料以較高層次的資料取代。

資料聚合

- 如：將每個分店的每日銷售額彙總出來，如此一來資料量就可以大幅降低。
- 進行資料聚合的原因或優點：
 - 愈小的資料集，其所需的記憶體及處理時間也會比較少
 - 聚合可以看到更高層次的變化情形，如：各分店的月銷售額變化。
 - 群體行為或屬性的表現要比個體來的穩定。
- 缺點：
 - 看不到一些細節變化，如：無法看出哪一天的銷售額較高的情形。

Data reduction

■ 為何要資料縮減？

- 資料倉儲中選擇進行分析的資料量極為龐大
- 複雜的資料分析與探勘龐大的資料會花費很長的時間

■ 資料縮減

- 希望獲得一個**比原始資料小很多的縮減資料集**，並且它幾乎能保持原始資料的完整性

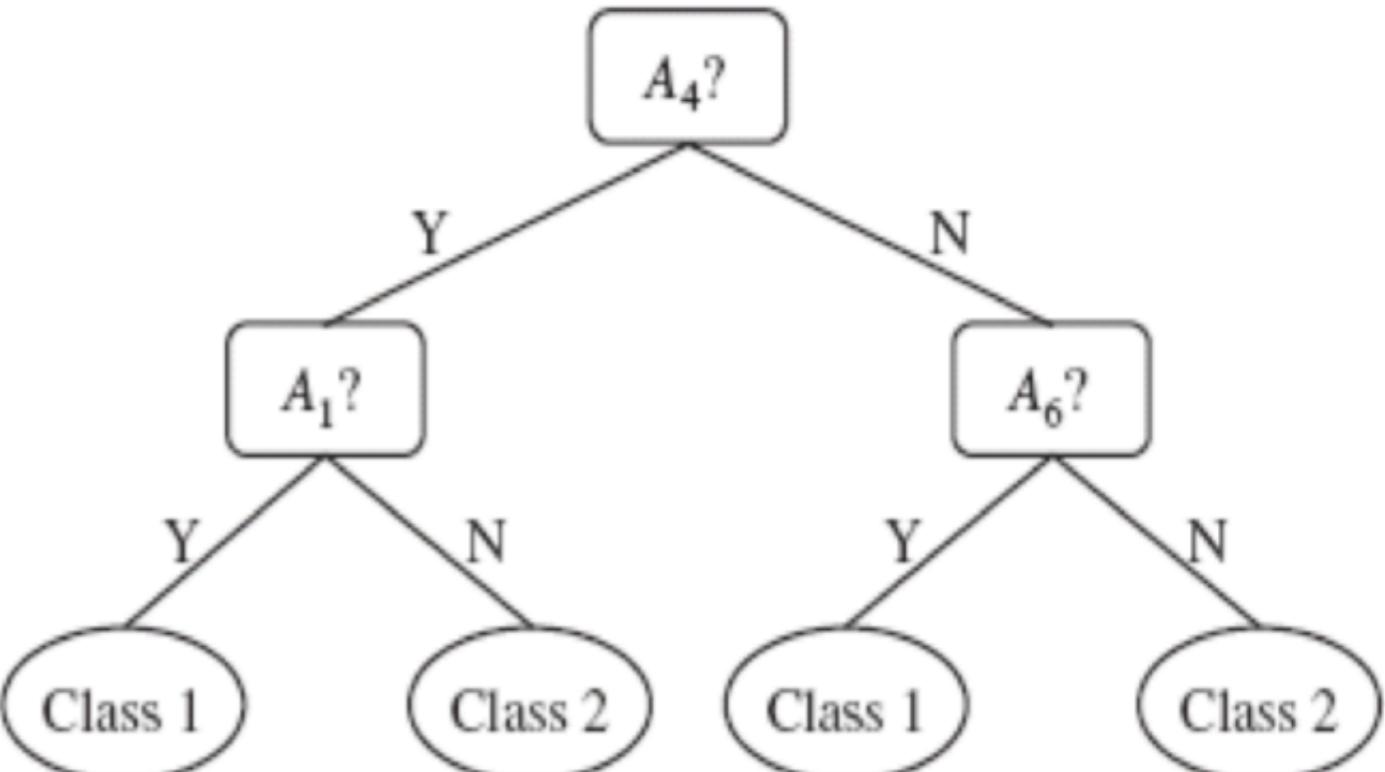
■ 資料縮減策略

- 資料聚合：對資料進行聚合運算。
- 子屬性集選擇：偵測並移除不相關、低相關、不重要的屬性。
- 維度縮減：利用編碼方式來降低資料集大小。
- 數值縮減：利用其它較小資料表示法來表示資料。例：將資料以模型表示、以分群、取樣方式進行取代。
- 離散化或概念階層建立：將原始資料以較高層次的資料取代。

Feature selection

- 進行分析的資料集合可能包含數百個屬性，其中可能有多個屬性與要進行的探勘任務無關或是多餘的。
- 屬性選擇 (例., 屬性子集合選擇):
 - 希望找到**最小的屬性集**，並且這個最小的屬性集的結果要與使用全部屬性集的結果盡量接近
 - 降低探勘屬性的數目，這樣會讓探勘的模式更容易被了解
- 啟發式 (n 個屬性可能有 2^n 數目的選擇):
 - 逐步向前選擇
 - 逐步向後刪除
 - 向前選擇與向後刪除的組合
 - 決策樹歸納

Feature selection

Forward selection	Backward elimination	Decision tree induction
Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_3, A_4, A_5, A_6\}$ => $\{A_1, A_4, A_5, A_6\}$ => Reduced attribute set: $\{A_1, A_4, A_6\}$	Initial attribute set: $\{A_1, A_2, A_3, A_4, A_5, A_6\}$  <pre>graph TD; A4["A4?"] -- Y --> A1["A1?"]; A4 -- N --> A6["A6?"]; A1 -- Y --> C1_1("Class 1"); A1 -- N --> C1_2("Class 2"); A6 -- Y --> C2_1("Class 1"); A6 -- N --> C2_2("Class 2")</pre>
		=> Reduced attribute set: $\{A_1, A_4, A_6\}$

Data reduction

■ 為何要資料縮減？

- 資料倉儲中選擇進行分析的資料量極為龐大
- 複雜的資料分析與探勘龐大的資料會花費很長的時間

■ 資料縮減

- 希望獲得一個**比原始資料小很多的縮減資料集**，並且它幾乎能保持原始資料的完整性

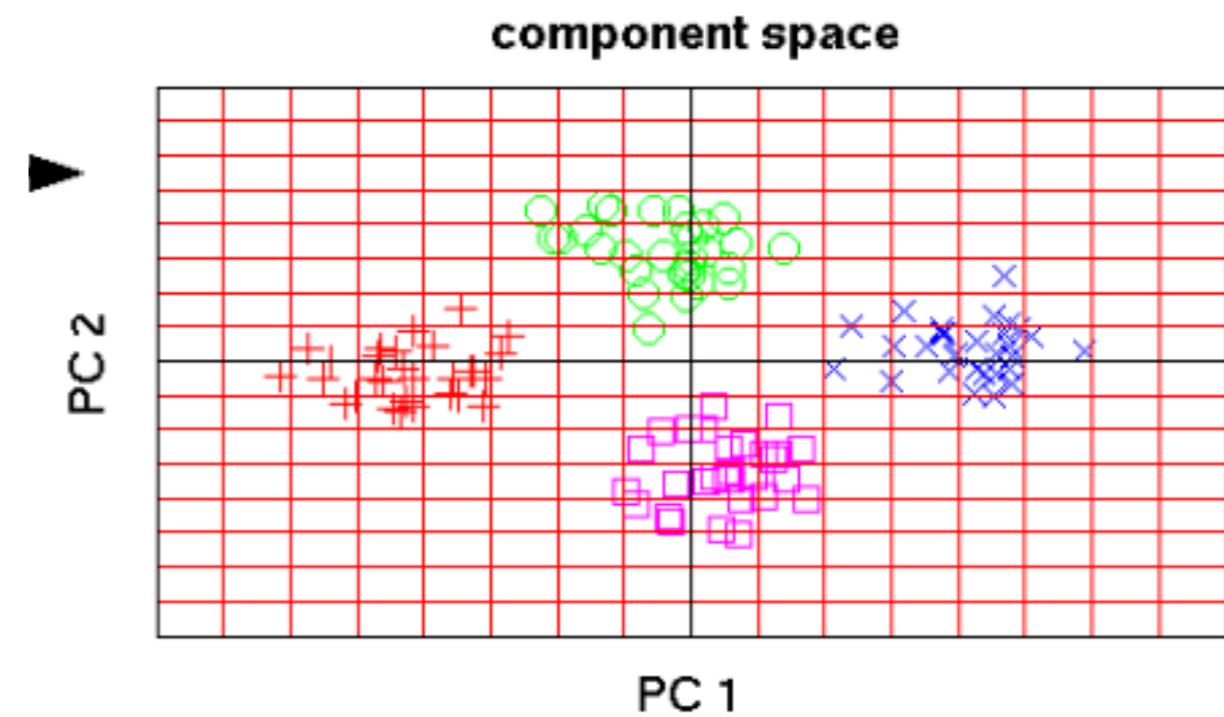
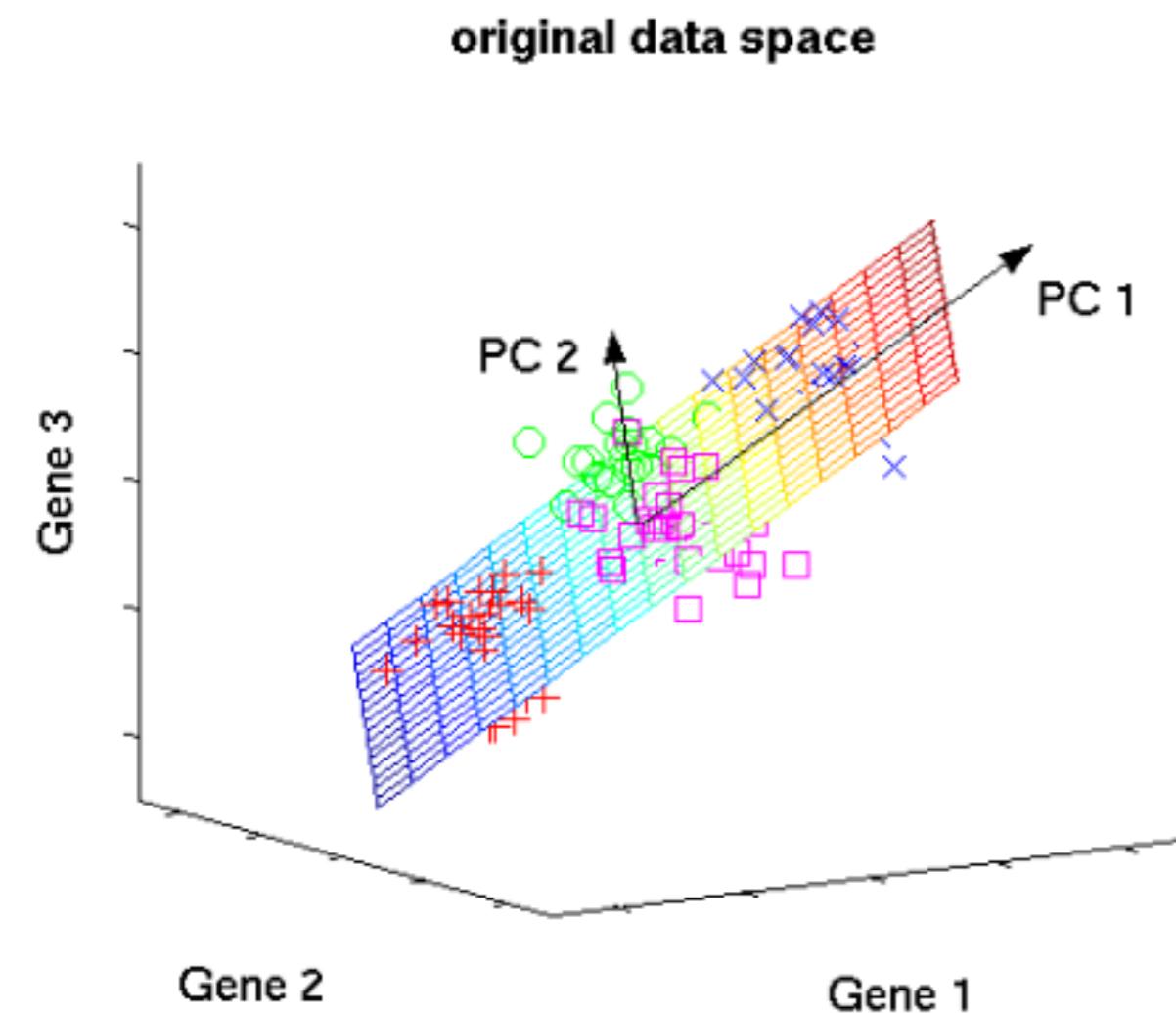
■ 資料縮減策略

- 資料聚合：對資料進行聚合運算。
- 子屬性集選擇：偵測並移除不相關、低相關、不重要的屬性。
- 維度縮減：利用編碼方式來降低資料集大小。
- 數值縮減：利用其它較小資料表示法來表示資料。例：將資料以模型表示、以分群、取樣方式進行取代。
- 離散化或概念階層建立：將原始資料以較高層次的資料取代。

Data dimension reduction

- 維度縮減通常是將一些舊有的屬性結合成一個新的屬性，以達到降低維度的目的。
- 好處：
 - 會刪除掉一些無關的特徵或雜訊值，可提昇資料探勘的品質。
 - 讓資料更易於用視覺化的方式呈現。
 - 演算法的時間及記憶體消耗也會大幅降低。
- 使用編碼或轉換來對原始資料進行縮減或壓縮。
 - **Lossless (無損壓縮)**：原始資料重新建構成一種壓縮的資料，且沒有遺失任何資訊的資料縮減稱之。
 - **Lossy (有損壓縮)**：如果僅能建立接近原始資料的壓縮資料，則此資料縮減稱為有損壓縮。

Dimension Reduction



Data reduction

■ 為何要資料縮減？

- 資料倉儲中選擇進行分析的資料量極為龐大
- 複雜的資料分析與探勘龐大的資料會花費很長的時間

■ 資料縮減

- 希望獲得一個**比原始資料小很多的縮減資料集**，並且它幾乎能保持原始資料的完整性

■ 資料縮減策略

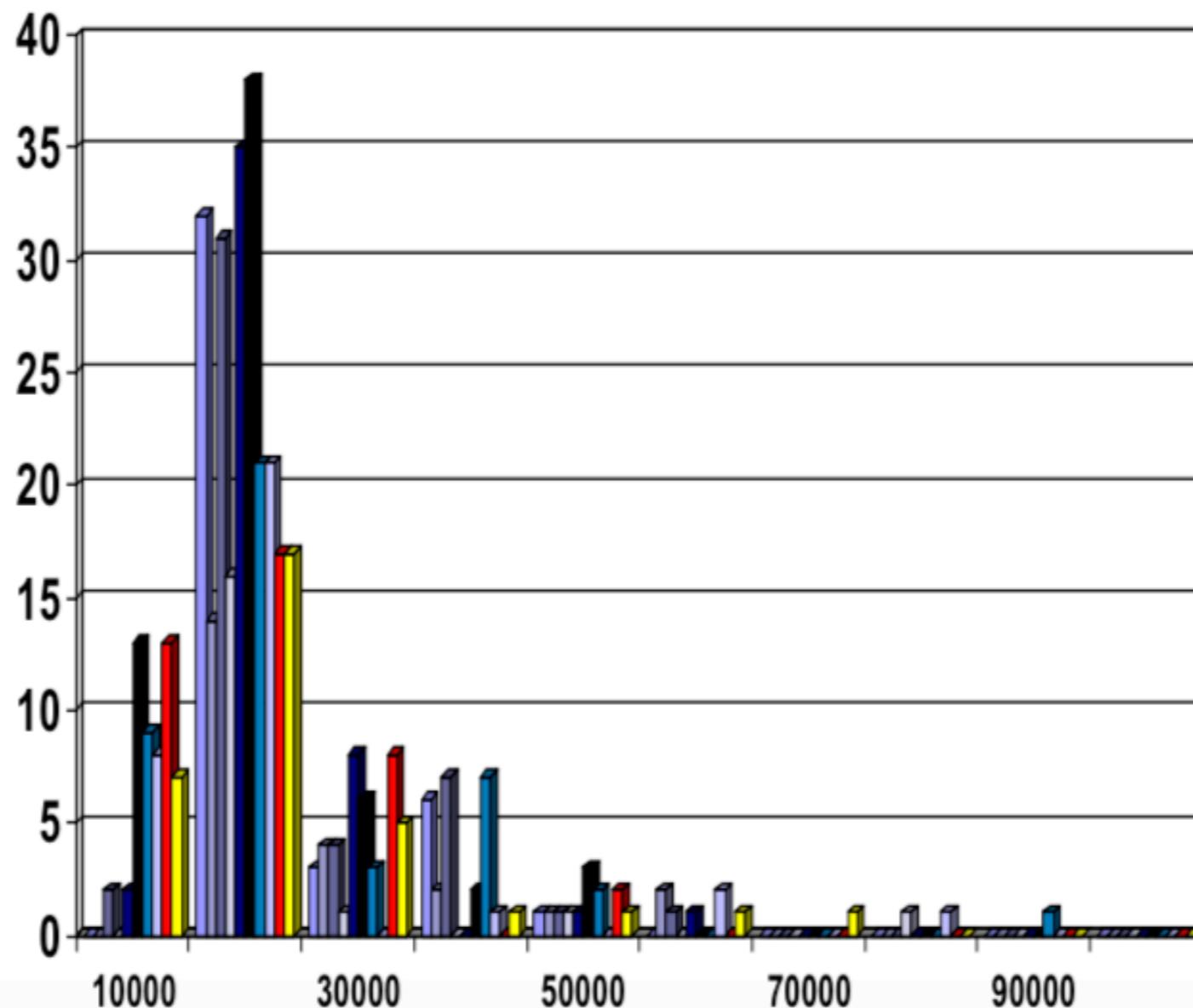
- 資料聚合：對資料進行聚合運算。
- 子屬性集選擇：偵測並移除不相關、低相關、不重要的屬性。
- 維度縮減：利用編碼方式來降低資料集大小。
- 數值縮減：利用其它較小資料表示法來表示資料。例：將資料以模型表示、以分群、取樣方式進行取代。
- 離散化或概念階層建立：將原始資料以較高層次的資料取代。

數值縮減

- 將資料分佈分割成箱子，每個箱子儲存資料的個數、平均或總合

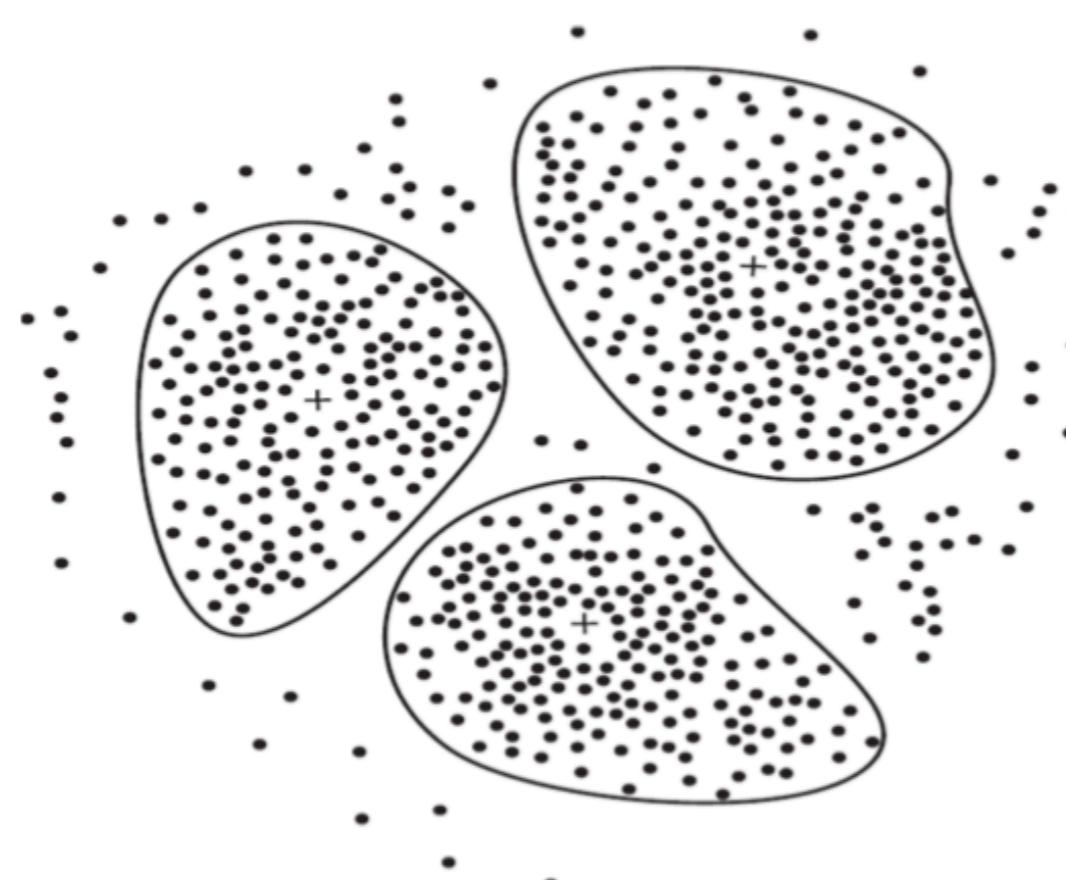
直方圖

$$k = 1 + 3.32 \log_{10} n$$



數值縮減

- 根據相似度進行分群並儲存群組資訊 (例., 中心與直徑)
- 如果資料是可群組會非常有效, 但是資料為不乾淨則否
- 可執行階層式分群並將結果存成多維度樹狀架構
- 有許多分群定義與演算法則的選擇



數值縮減

- 用比較小的隨機樣本 n 來代表原始資料 N
- 選取代表資料的子集合
 - 當資料包含偏斜時，取樣的效能會非常差

T38	youth
T256	youth
T307	youth
T391	youth
T96	middle_aged
T117	middle_aged
T138	middle_aged
T263	middle_aged
T290	middle_aged
T308	middle_aged
T326	middle_aged
T387	middle_aged
T69	senior
T284	senior

T38	youth
T391	youth
T117	middle_aged
T138	middle_aged
T290	middle_aged
T326	middle_aged
T69	senior



Data reduction

■ 為何要資料縮減？

- 資料倉儲中選擇進行分析的資料量極為龐大
- 複雜的資料分析與探勘龐大的資料會花費很長的時間

■ 資料縮減

- 希望獲得一個**比原始資料小很多的縮減資料集**，並且它幾乎能保持原始資料的完整性

■ 資料縮減策略

- 資料聚合：對資料進行聚合運算。
- 子屬性集選擇：偵測並移除不相關、低相關、不重要的屬性。
- 維度縮減：利用編碼方式來降低資料集大小。
- 數值縮減：利用其它較小資料表示法來表示資料。例：將資料以模型表示、以分群、取樣方式進行取代。
- 離散化或概念階層建立：將原始資料以較高層次的資料取代。

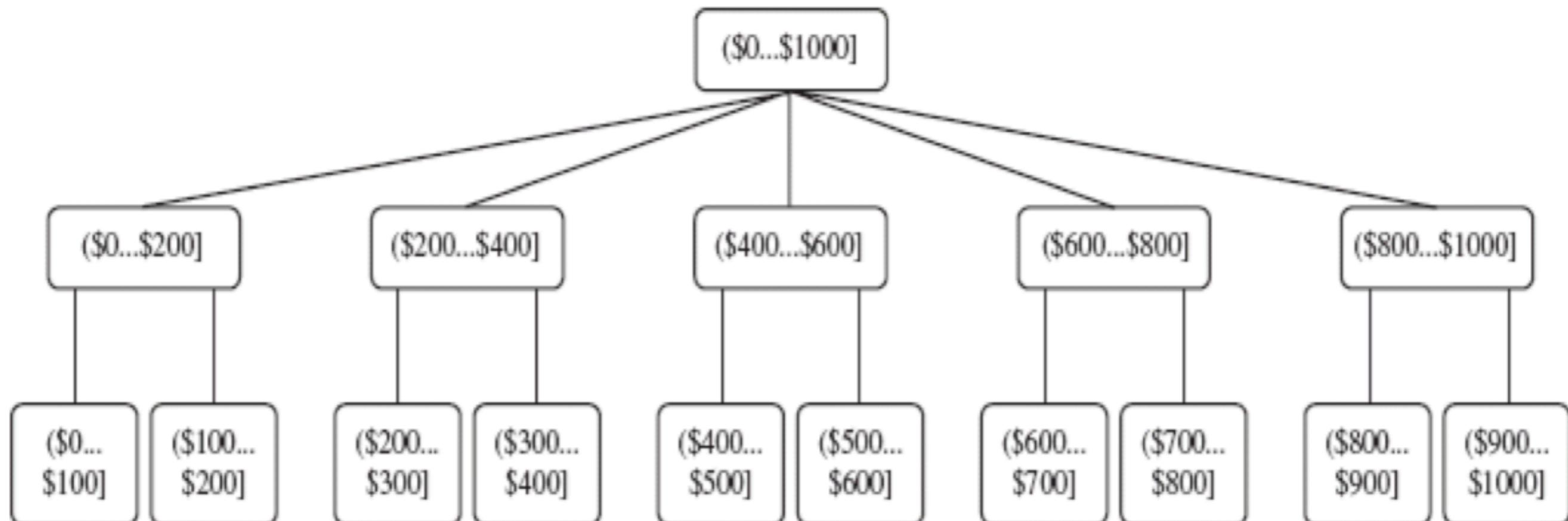
資料離散化

- 有些分類演算法僅能處理類別型資料的問題，所以必須將一些連續值進行離散化的動作。

資料離散化

- 要將連續值屬性離散化包含兩個動作：

- 先將連續值進行排序，然後再定義 $n-1$ 個分割點，將資料分析 n 等份。
 - 將所有值對映至類別中。



資料離散化

- 有些分類演算法僅能處理類別型資料的問題，所以必須將一些連續值進行離散化的動作。
- 再者，如果類別很多，但是有些較不常發生，那麼就可以將部份類別進行結合，以降低類別個數。
- 概念階層形成
 - 概念階層可以透過用較高層次的概念 (如青年、中年、老年) 取代較低層次的概念 (年齡屬性的值) 來進行資料縮減

總結

- 資料預處理主要包含資料整合(data integration)、資料清理(data cleaning)、資料轉換(data transformation)、資料縮減(data reduction)等四項工作
- Garbage in, garbage out，縮寫：GIGO