

# PYTHON機器學習入門

## UNIT 3 : MACHINE LEARNING BASIC

授課教師：江尚瑀

# OUTLINE

- 資料特徵
  - 特徵選擇
  - 特徵距離的計算
  - 資料標準化
- **Scikit-Learn**的簡介
  - 基本程式架構
- 模型評估

A decorative graphic on the left side of the slide consisting of two parallel, wavy vertical lines. The inner line is a vibrant lime green, and the outer line is a light cream color. They extend from the top to the bottom of the frame.

# FEATURE

PART 1

# 特徵選擇

- 資料的特徵 (或稱為屬性) 是否都是有用?!
  - 能減少不必要或無用的特徵進入模型, 可以提升準確度並且可以降低模型的計算量以提升預測反應速度.
- 如何檢驗特徵與目標(target)或特徵與特徵之間的關聯性?

# 皮爾森相關係數(PEARSON CORRELATION COEFFICIENT)

- 可以衡量兩變數**x**和**y**的「線性」相依程度

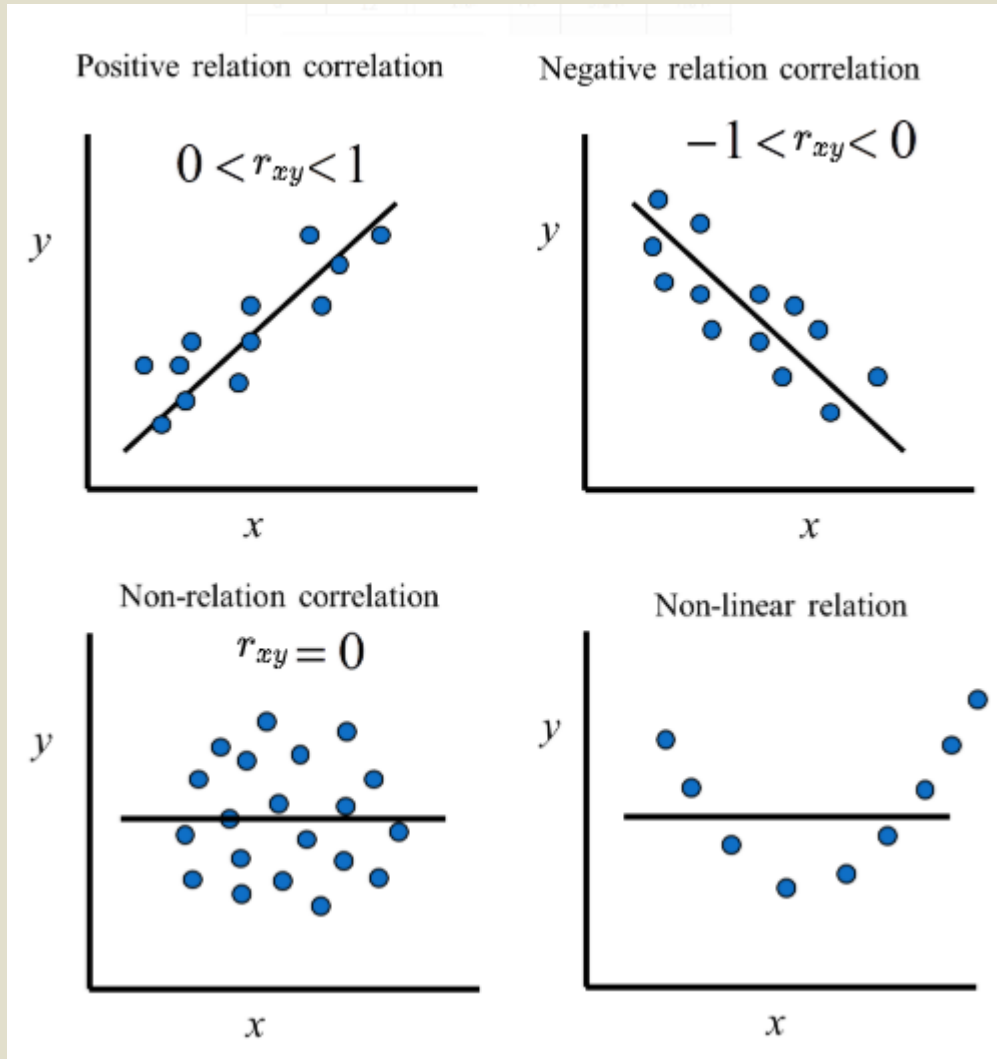
$$r_{xy} = \frac{\text{cov}(x, y)}{\text{std}(x) \times \text{std}(y)} \quad -1 \leq r_{xy} \leq 1$$

$$r_{xy} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}} \quad (\text{Eq.3})$$

where:

- $n$  is sample size
- $x_i, y_i$  are the individual sample points indexed with  $i$
- $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$  (the sample mean); and analogously for  $\bar{y}$

# 皮爾森相關係數(PEARSON CORRELATION COEFFICIENT)



1.) 完全正相關為  $r_{xy} = 1$   
即  $x = y$ , 所有  $(x, y)$  點會全部落在  
45度的直線上

2.) 某特徵與目標的  $|r_{xy}|$  愈高,  
表示此特徵對於目標有較大影響力

# 皮爾森相關係數範例

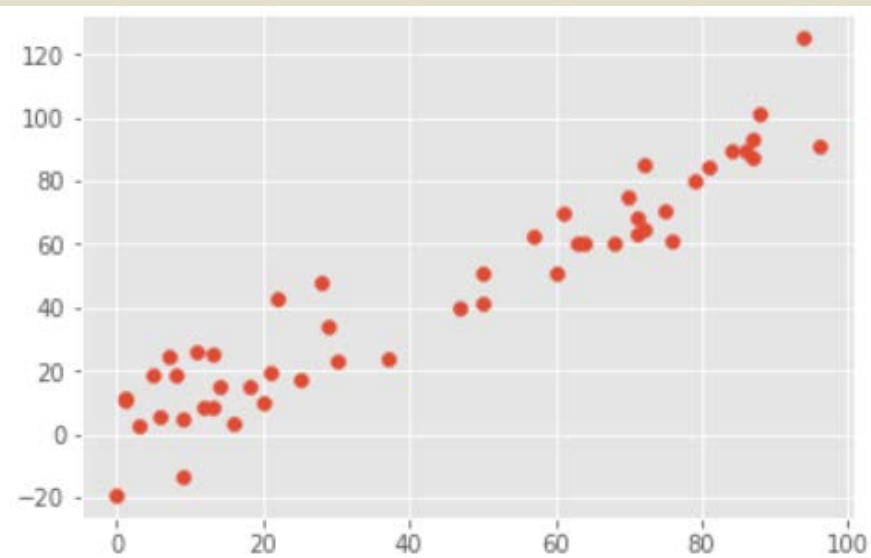
```
np.random.seed(1)

# 50 random integers between 0 and 100
x = np.random.randint(0, 100, 50)

# Positive Correlation with some noise
y = x + np.random.normal(0, 10, 50)

coef=np.corrcoef(x, y)
print(coef)
plt.scatter(x, y)
plt.show()
```

```
[[1.          0.94798835]
 [0.94798835 1.          ]]
```



# 皮爾遜相關係數實例

- 購物問卷調查實例

詢問消費者**2019**年對於整個購物的滿意度，以及針對調查對象詢問今年再次繼續購買商品的次數。

平均滿意度為**8**

平均再次購買次數**12**

問卷編號	滿意度	再次購買次數
1	8	12
2	9	15
3	10	16
4	7	18
5	8	6
6	9	11
7	5	3
8	7	12
9	9	11
10	8	16



# 皮爾遜相關係數實例

- 將數據代入皮爾遜係數公式

問卷編號	滿意度 – 平均滿意度	再次購買次數 – 平均再次購買次數
1	0	0
2	1	3
3	2	4
4	-1	6
5	0	-6
6	1	-1
7	-3	-9
8	-1	0
9	1	-1
10	0	4

- $r = 30 / \sqrt{18} \sqrt{196} = 0.505$

問卷編號	$(X_i - \bar{X})(Y_i - \bar{Y})$	$(X_i - \bar{X})^2$	$(Y_i - \bar{Y})^2$
1	0	0	0
2	3	1	9
3	8	4	16
4	-1	1	36
5	0	0	36
6	-1	1	1
7	27	9	81
8	0	1	0
9	-1	1	1
10	0	0	16
總計	30	18	196

# 皮爾遜相關係數實例

- 消費滿意度與下次會再購買是有中度正相關

相關係數絕對值	相關程度
約 = 1	完全相關 (Perfect correlated)
0.7 ~ 0.99	高度相關 (Highly correlated)
0.4 ~ 0.69	中度相關 (Moderately correlated)
0.1 ~ 0.39	低度相關 (Modestly correlated)
0.01 ~ 0.09	接近無相關 (Weakly correlated)
約 = 0	無相關

# 皮爾遜相關係數實例

- Python 程式計算上述過程

```
x = np.array([8, 9, 10, 7, 8, 9, 5, 7, 9, 8])
```

```
y = np.array([12, 15, 16, 18, 6, 11, 3, 12, 11, 16])
```

np.mean計算平均值

問卷編號	滿意度	再次購買次數
1	8	12
2	9	15
3	10	16
4	7	18
5	8	6
6	9	11
7	5	3
8	7	12
9	9	11
10	8	16

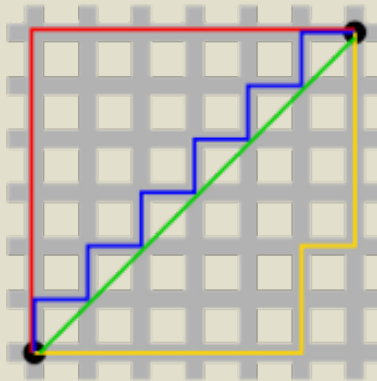
# 特徵距離的計算

- 資料間的相似程度 (**Similarity**) 即計算它們的特徵距離

- Distance Metrics:**

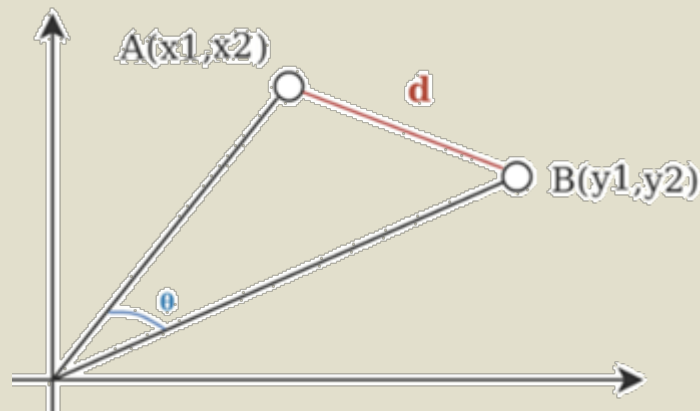
$n$ -number of features  $x_i$  and  $y_i$  are the features of vectors  $x$  and  $y$  respectively, in the two dimensional vector space.

$$\mathbf{x} = (x_1, x_2, x_3, \dots, x_n) \text{ and } \mathbf{y} = (y_1, y_2, y_3, \dots, y_n)$$



1.) **Manhattan Distance:** 
$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

2.) **Euclidean Distance:** 
$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$



3.) **Cosine Distance:** 
$$\cos \theta = \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|}$$

# 分別計算 EUCLIDEAN DISTANCE

	x1	x2	x3	x4	x5	x6
D1	2	3	4	2	1	15
D2	1	2	2	4	3	51
D3	1	4	5	2	2	35

$$d(D1, D2) = \sqrt{|15 - 51|^2}$$

$$d(D2, D3) = \sqrt{|51 - 35|^2}$$

$$d(D1, D3) = \sqrt{|15 - 35|^2}$$

距離會被值域較大的特徵所決定

# FEATURE RE-SCALING

將每個特徵值的尺度轉成一致

將不同變化範圍的值映射到相同的固定範圍

- **Min-Max Normalization**

- Re-scaling the range of a vector to make all elements lie between 0 and 1

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

Min-Max Normalization使用時機，資料的上下界通常是已知的固定值

- **Z-score Standardization**

- Subtract the mean and divide by the standard deviation

$$x' = \frac{x - \mu}{\sigma} \quad x' \sim (0, 1)$$

# EXAMPLE

	x1	x2	x3	x4	x5	x6
D1	2	3	4	2	1	15
D2	1	2	2	4	3	51
D3	1	4	5	2	2	35

After Min-Max Normalization

	x1	x2	x3	x4	x5	x6
D1	1					0
D2	0					1
D3	0					0.5556

距離不會被值域特別大的特徵影響太多

# COSINE SIMILARITY

Angle between two vectors times their lengths

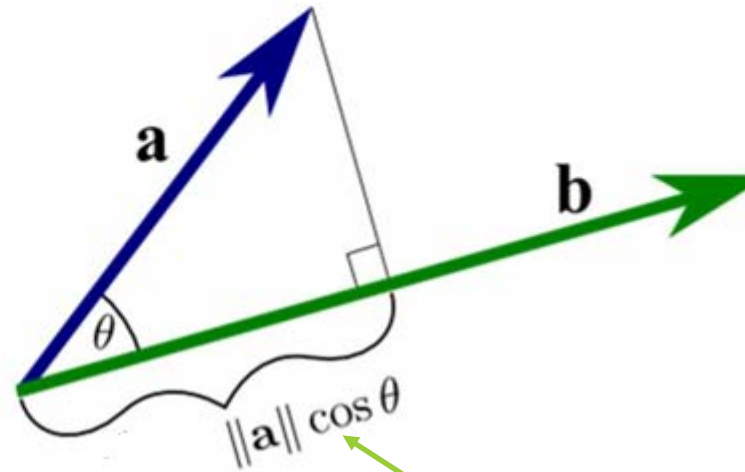
$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i$$

(standard inner product in Cartesian coordinates)

Many uses:

- Project vector onto another vector,  
project into basis,  
project into tangent plane,  
...

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$$



n 維歐幾里得空間  $\mathbb{R}^n$  上的向量

$$\vec{v} = (v_1, v_2, \dots, v_n)$$

，其模長或範數為：

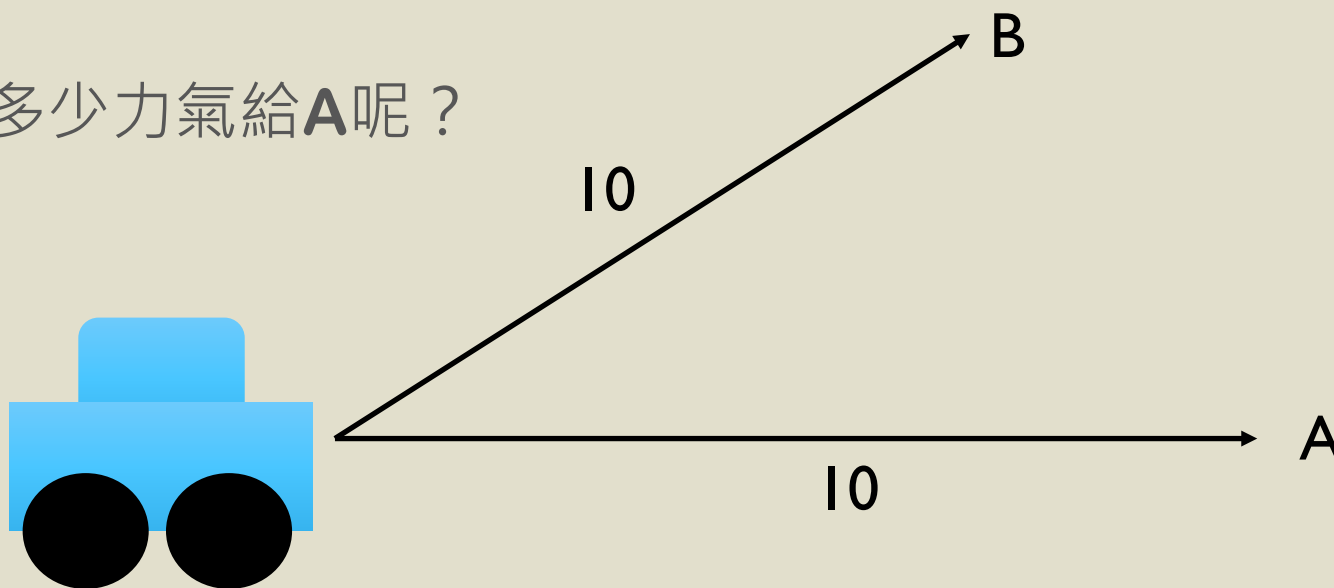
$$|\vec{v}| = \|\vec{v}\| = \sqrt{v_1^2 + v_2^2 + \dots + v_n^2}$$

$\mathbf{a} \cdot \mathbf{b}$  is  $\|\mathbf{b}\|$  times this length

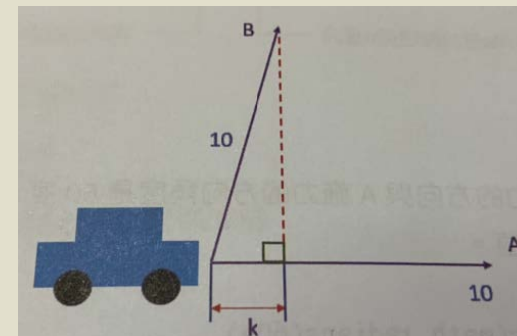
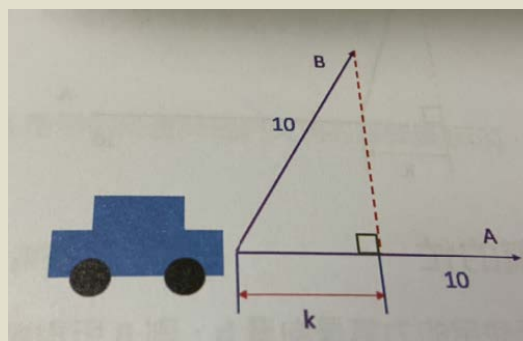
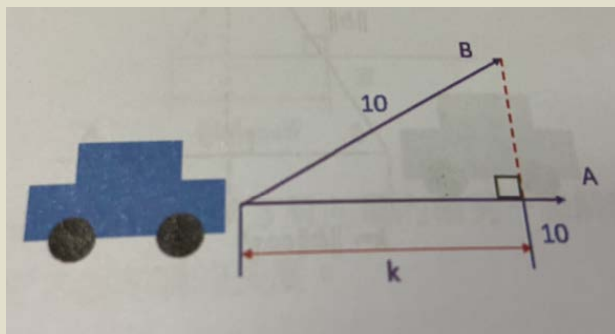


# 向量內積

- 究竟**B**貢獻多少力氣給**A**呢？

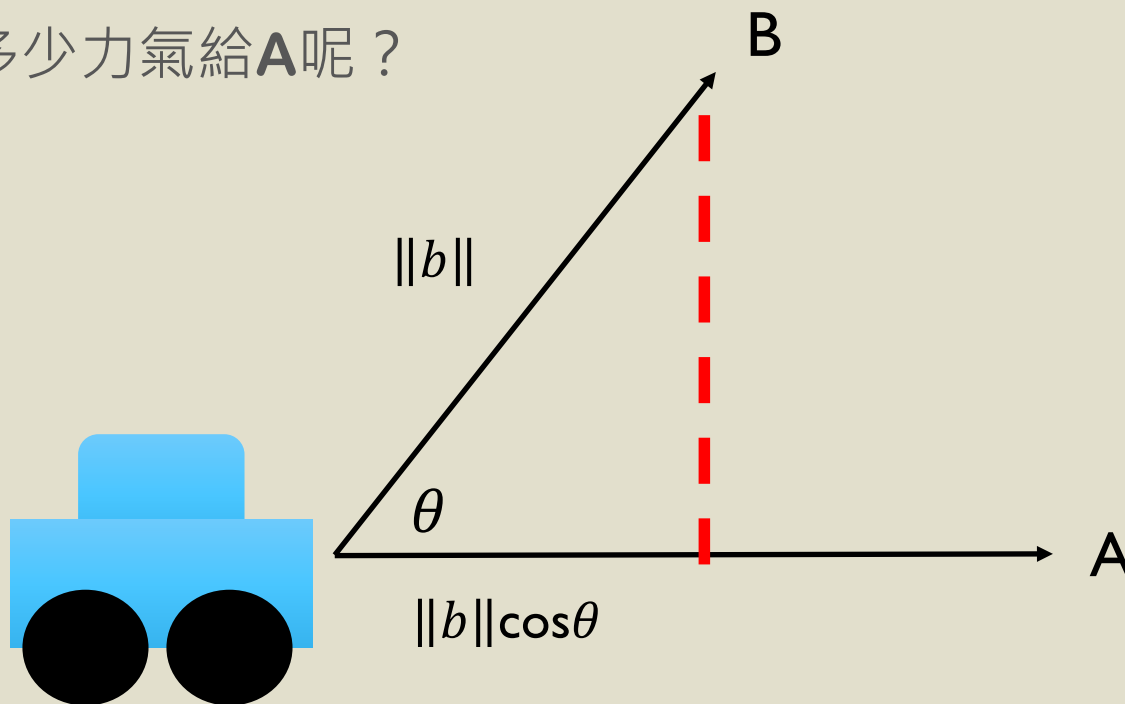


- 角度越大，值越小，對**A**的幫助越小



# 向量內積

- 究竟**B**貢獻多少力氣給**A**呢？



- 內積定義是兩個向量與他們夾腳的餘弦值(**cos**)  
 $ab = \|a\| \|b\| \cos\theta$       **numpy**模組的**dot()**方法

向量**a**的長度

向量**b**映射到向量**a**的長度

# 向量內積的計算

- 兩條直線的夾角公式

$$\cos\theta = \frac{a \cdot b}{\|a\| \|b\|} \quad \cos\theta = \frac{a_1b_1 + a_2b_2}{\|a\| \|b\|}$$

**Q:** 假設座標平面有A,B,C,D四點(AB組成向量ab、CD組成向量cd)  
請計算這兩個向量的cos值與夾角度數

**\*Hint**

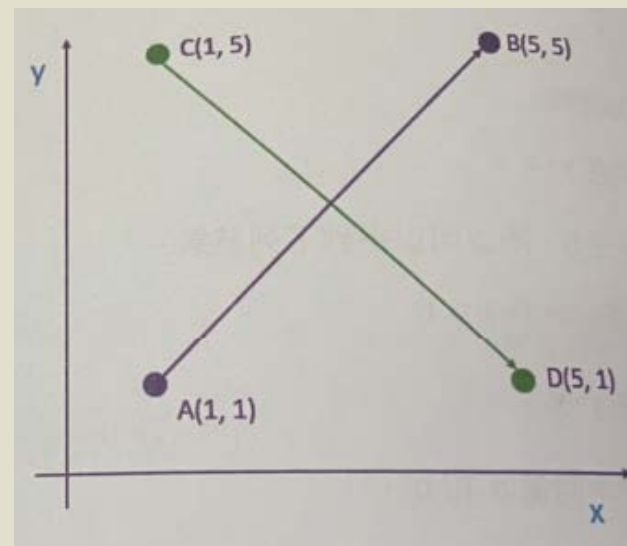
向量的長度 -> `numpy.linalg.norm()`方法

內積 -> `numpy.dot()`方法

Cos值要轉成弧度再轉成角度

弧度 = `math.acos(cos值)`

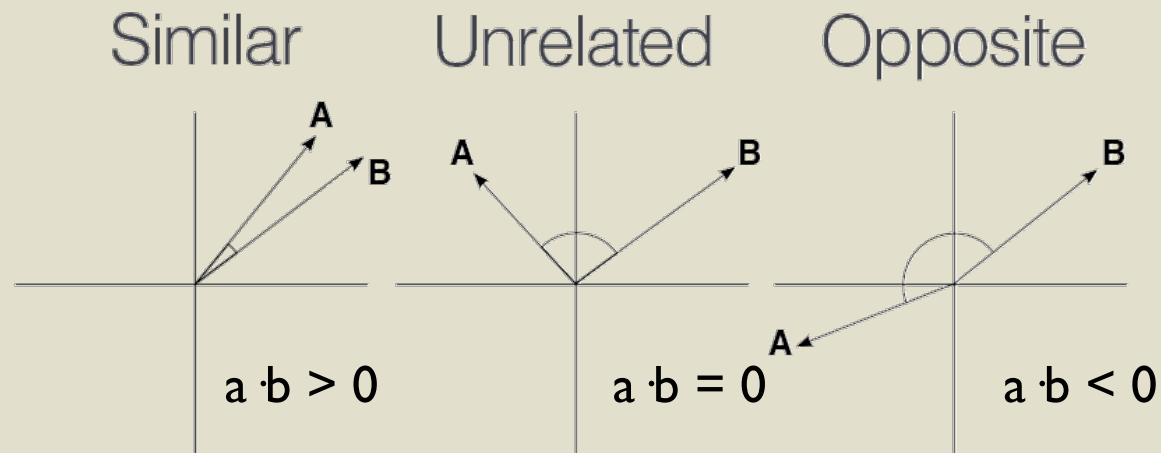
角度 = `math.degrees()`



# 向量內積的性質

餘弦相似程度  $\cos\theta = \frac{a_1b_1+a_2b_2}{\|a\| \|b\|}$

- 分母是向量長度一定大於**0**，上述公式可推導：  
向量內積是**正值**，兩向量的夾角**小於90度**  
向量內積是**0**，兩向量的夾角**等於90度**  
向量內積是**負值**，兩向量的夾角**大於90度**



# 機器學習實例

- 題目：判斷下列句子的相似程度

1. 機器與機械
2. 學習機器碼
3. 機器人學習

編號	句子	機	器	與	械	學	習	碼	人
1	機器與機械	2	1	1	1	0	0	0	0
2	學習機器碼	1	1	0	0	1	1	1	0
3	機器人學習	1	1	0	0	1	1	0	1

$A = (2, 1, 1, 1, 0, 0, 0, 0)$

$B = (1, 1, 0, 0, 1, 1, 1, 0)$

$C = (1, 1, 0, 0, 1, 1, 0, 1)$

計算 **AB**、**AC**、**BC** 的相似程度

# USE EUCLIDEAN DISTANCE OR COSINE SIMILARITY ?

- The Cosine metric is a measurement of orientation and not magnitude
  - Euclidean是要是相同向量空間，而且magnitude會影響計算的距離
  - Cosine不看magnitude(強度)，只在乎2個向量是否具有相同方向(不一定要有相同向量空間)。

## Euclidean

-> 體現個體數值特徵的絕對差異，使用者行為指標分析或價值的相似度

## Cosine Similarity

-> 使用者對內容的評分來區分，興趣的相似程度與差異

# 相關性實作

```
def euclidean_distance(x, y):  
    return np.sqrt(np.sum((x - y) ** 2))  
  
def cosine_similarity(x, y):  
    return np.dot(x, y) / (np.sqrt(np.dot(x, x)) * np.sqrt(np.dot(y, y)))
```

值愈大, 角度愈小, 表示很有相關

值愈小, 角度愈大, 表示愈不相關

# 相關性實作

- 二維[2D]

$x=[0,5], y=[1,5]$  , 歐式距離( $x1,y1$ )= ? ,  $\cos$ 距離( $x1,y1$ )= ?

$x1 = [0,5] \ y1 = [1,5]$  、  $x2 = [0,5] \ y2 = [100,5]$

$x1y1, x2y2$  哪個較不相關?

- 三維[3D]

$A = (99,1,1)$  ,  $B = (0,1,1)$  ,  $C = (100,0,0)$

歐式距離( $A,B$ )= ? , 歐式距離( $A,C$ )= ?

$\cos$ 距離( $A,B$ )還( $A,C$ )相關性比較高?

- 多維[nD]

$a=[1 \ 0 \ 0 \ 0], b=[1 \ 0 \ 1 \ 1], c=[1 \ 1 \ 1 \ 1], d=[10 \ 0 \ 0 \ 0]$

$ab \ ac$  哪個比較近( $\cos$ 距離)?  $\cos$ 距離現象( $ad$ )



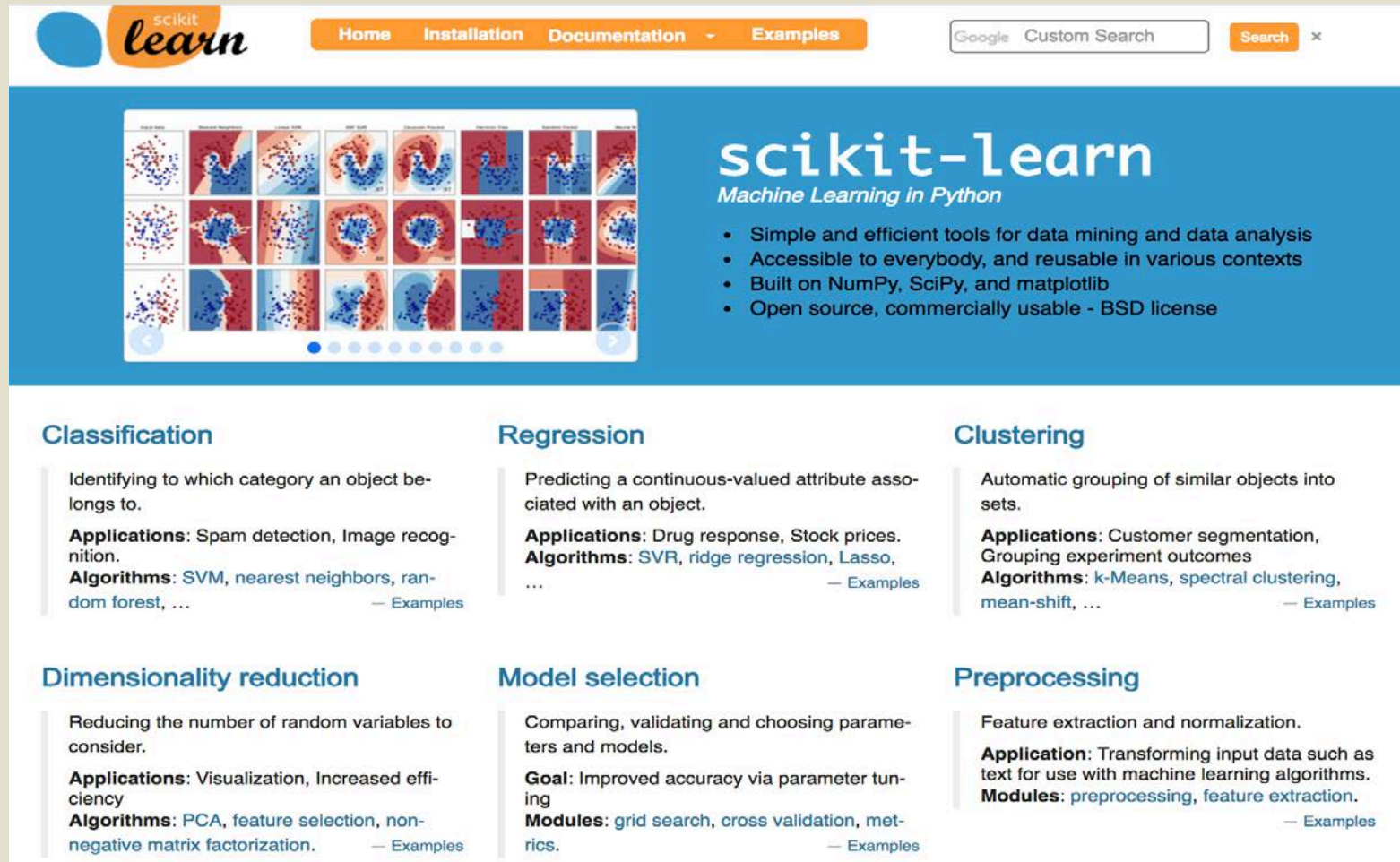
A decorative wavy line in a light green color runs vertically along the left side of the slide, separating a light beige area from the dark teal background.

# THE INTRO OF SCIKIT-LEARN

PART 2

# ABOUT SCIKIT-LEARN

- A python library that provides various implementation of machine learning/data mining algorithms



The screenshot shows the scikit-learn website homepage. At the top, there is a navigation bar with links for Home, Installation, Documentation, and Examples. A search bar is also present. The main header features the scikit-learn logo and the tagline "Machine Learning in Python". Below this, a grid of 12 small plots illustrates various machine learning concepts. To the right of the grid, a list of key features is provided. The main content area is divided into six sections, each representing a different machine learning task: Classification, Regression, Clustering, Dimensionality reduction, Model selection, and Preprocessing. Each section includes a brief description, applications, algorithms, and a link to examples.

**scikit-learn**  
Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

### Classification

Identifying to which category an object belongs to.

**Applications:** Spam detection, Image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, ... — Examples

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, ridge regression, Lasso, ... — Examples

### Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, ... — Examples

### Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** PCA, feature selection, non-negative matrix factorization. — Examples

### Model selection

Comparing, validating and choosing parameters and models.

**Goal:** Improved accuracy via parameter tuning

**Modules:** grid search, cross validation, metrics. — Examples

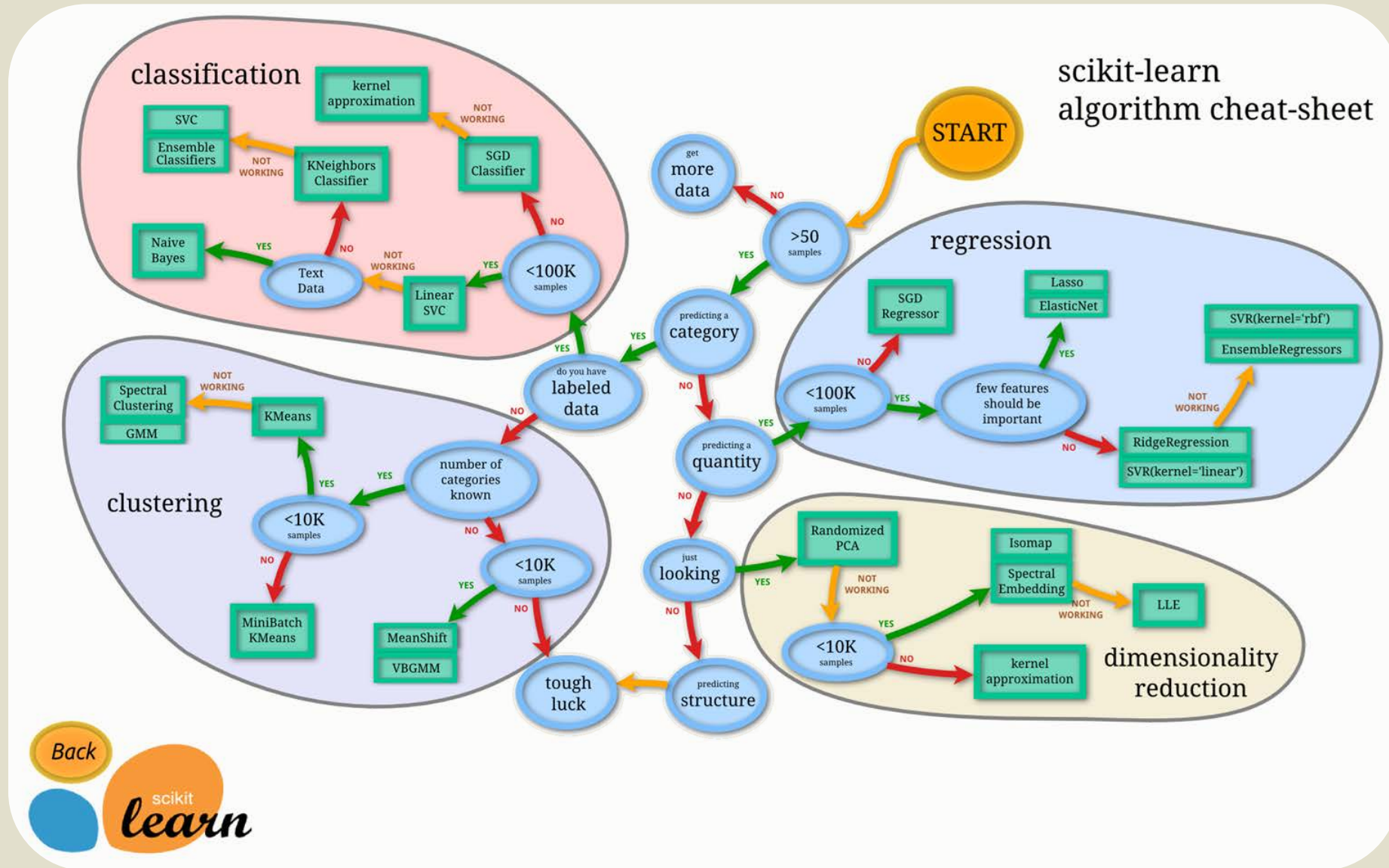
### Preprocessing

Feature extraction and normalization.

**Application:** Transforming input data such as text for use with machine learning algorithms.

**Modules:** preprocessing, feature extraction. — Examples

# ABOUT SCIKIT-LEARN



# SCIKIT-LEARN五大功能

1. 資料前處理(**Preprocessing**)
2. 迴歸(**Regression**)
3. 分類(**Classification**)
4. 維度縮減(**Dimensionality reduction**)
5. 分群(**Clustering**)

# INSTALL SCIKIT LEARN

- Anaconda 已內建 Scikit learn

```
Anaconda Prompt (anaconda3) - python
(base) C:\Users\User>python
Python 3.8.8 (default, Apr 13 2021, 15:08:03) [MSC v.1916 64 bit (AMD
64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information
>>> from sklearn.datasets import load_iris
>>> iris_data = load_iris()
>>> print(iris_data.keys())
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'featu
re_names', 'filename'])
>>> print(iris_data.data)
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]]
```

from sklearn.datasets import load\_iris  
iris\_data = load\_iris()  
print(iris\_data.keys())  
print(iris\_data.data)

- 加州大學 **Irvine** 分校的機器學習資料集、**Kaggle** 網站與 **KD Nuggets** 整理的資料集資源

# SCIKIT-LEARN 的基本程式架構

## 1. 讀取資料&pre-processing

```
pd.read_csv('data.csv')
```

## 2. 切分訓練集與測試集

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
```

## 3. 模型配適

```
model = svm.SVC(...)  
model.fit(X_train, y_train)
```

## 4. 預測

```
pred_Y = model.predict(X_train)  
pred_Y = model.predict(X_test)
```

## 5. 評估(計算成績可能是誤差值或正確率或..)

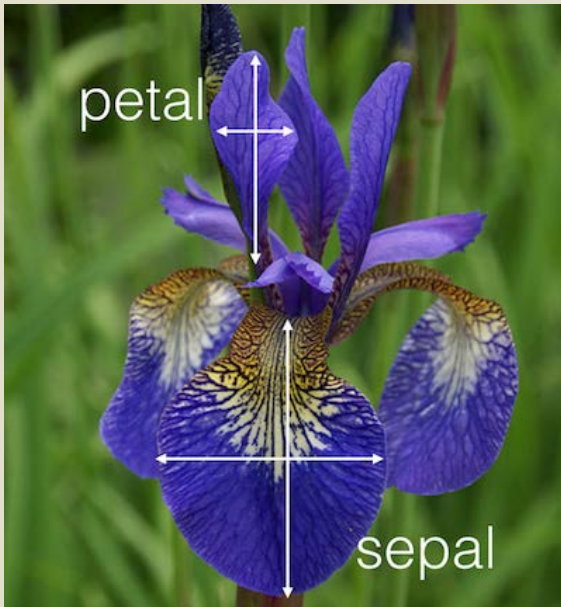
```
model.score(X_train, y_train)  
  
model.score(X_test, y_test)
```



# LOAD IRIS DATASET FROM SKLEARN

```
# load dataset
iris = datasets.load_iris()
print(iris['feature_names'])
X = pd.DataFrame(iris['data'], columns=iris['feature_names'])
X = X[['sepal length (cm)', 'petal length (cm)']]
print("target_names: "+str(iris['target_names']))
Y = pd.DataFrame(iris['target'], columns=['target'])
```

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
target_names: ['setosa' 'versicolor' 'virginica']
```



Example: `scikit_learn_Basic.ipynb`

# 切分訓練集與測試集

- To randomly divide the data, sklearn provides a function called **train\_test\_split()**

## Cross validation: train\_test\_split()

```
: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    iris_data[['sepal length (cm)', 'petal length (cm)']], iris_data[['target']], test_size=0.3)
```



# 資料前處理

- 運算的資料都只能是數值(**numeric**)
- 如何處理非數值資料？
  - Ordinal features(or label)
  - Nominal/Categorical features (or label)
- 不同**features** 的數值範圍差異會有影響嗎？
  - 溫度: 最低**0** 度、最高**40** 度
  - 距離: 最近**0** 公尺、最遠**10000** 公尺

# 非結構化資料



圖片

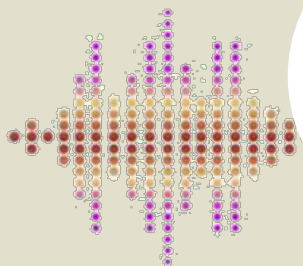
非結構化  
資料

文章

影片



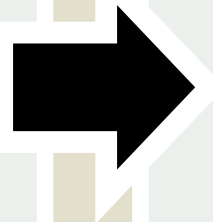
音軌



- 旗艦級手機是各個品牌所推出搭載了最新作業系統，例如蘋果的iPhone 15 Pro，三星的Galaxy S24 Ultra，以及華為的Mate 60 Pro。這些手機通常支援最新的應用程式，但舊款手機可能無法支援。因此，用戶在選擇手機時，應考慮其是否支援最新的應用程式，以確保其能拍攝微距、廣角等多角度，品質可媲美單眼相機的款式，當然也不用擔心手機支援不了最新推出的應用程式。

# 非結構化資料轉換

- 非結構化資料



- 結構化資料

- 照片類別: 園藝植物
- 種類: 球根花卉植物
- 圖片尺寸: **4280 x 2368**
- 解析度**DPI: 1080p**
- 拍照日期: **2021/3/1**
- 拍攝地點: 阿里山
- . . .

# 非結構化資料轉換

## • 非結構化資料

旗艦級手機是各個品牌所推出搭載了最新功能、最高規格的型號，例如配備有最新處理器，而擁有超越其它機型的使用速度；又或者裝置了多顆鏡頭而能拍攝微距、廣角等多角度，品質可媲美單眼相機的款式，當然也不用擔心手機支援不了最新推出的應用程式。不過，旗艦級手機的定價大多為**25000**元以上，較適合薪資有一定水準的商務人士。



## • 結構化資料

- 字數: **150**字
- 單詞數: **20**個
- 標點符號數: **11**個
  - 逗點: **5**個
  - 句點: **2**個
  - . . .
- 正向詞: **5**個
- 負向詞: **2**個
- . . .

# 非結構化資料轉換

來源IP

造訪時間

192.168.11.99 - root [06/Mar/2017:22:25:25 +0800] "PROPFIND  
/owncloud/remote.php/webdav/GoogleDrive/proxmox/template/ HTTP/1.1" 207 2551 "-"  
"davfs2/1.4.6 neon/0.29.6"

造訪網址

# 如何定義對方是否會答應交往？



 非結構化資料

理系が恋に落ちたので証明してみた



# 如何定義對方是否會答應交往？

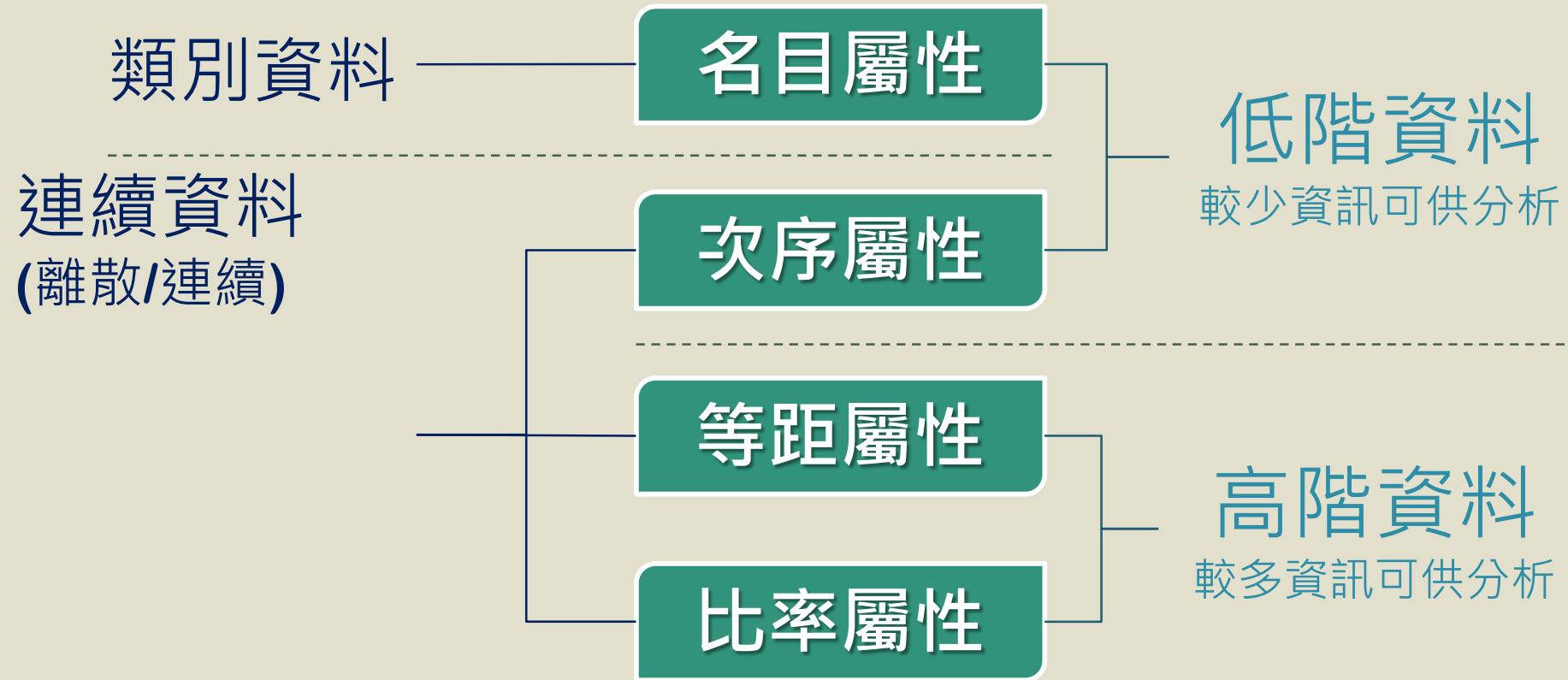


結構化資料

日誌法  
Diary Methods

理系が恋に落ちたので証明してみた

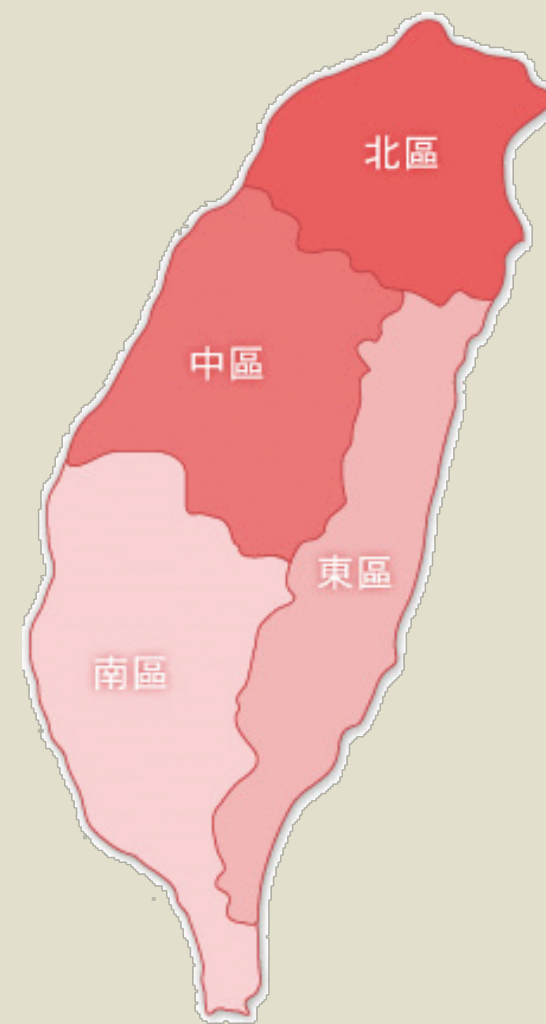
# 結構化資料類別





# 名目屬性

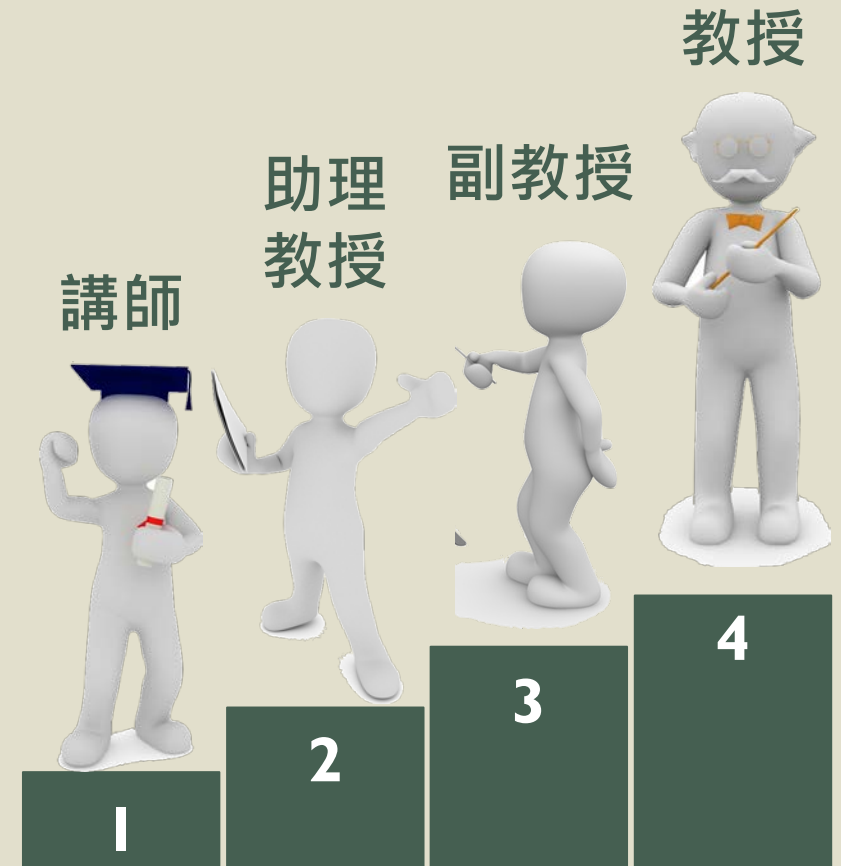
- 名目屬性(Categorical Attribute) :
  - 資料處理時有時會以數值表示
    - 北區(1) / 中區(2) / 南區(3) / 東區(4)
  - 經常具有互斥的特性
    - 男、女；年輕、老年



Q：還有甚麼資料為名目屬性的資料？

# 次序屬性

- 次序屬性(Ordinal Attribute) :
  - 不同組值間可比較大小
    - 教授(4) / 副教授(3) / 助理教授(2) / 講師(1)
  - 不同大小組值間非比例關係
    - 教授(4)  $\neq$  助理教授(2)  $\times 2$

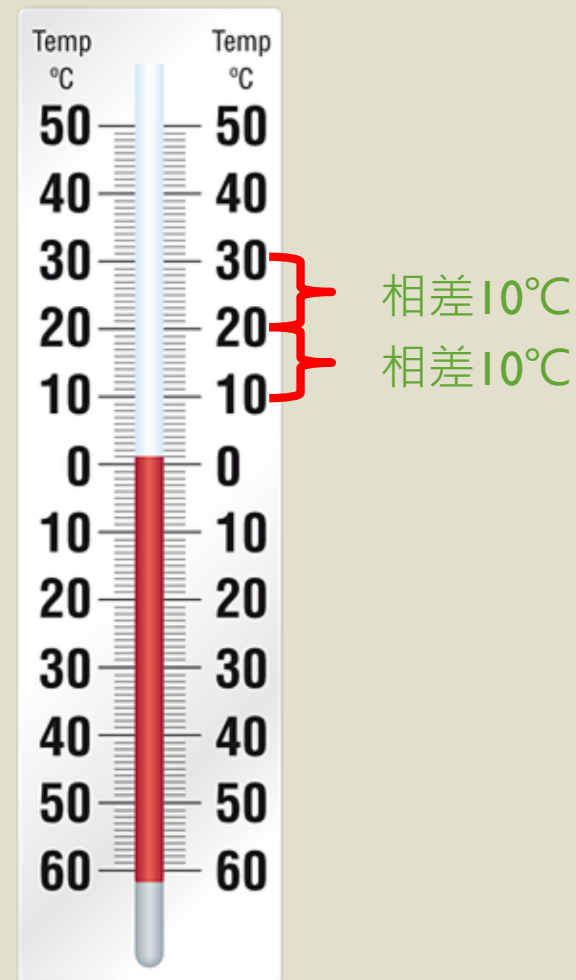


Q：還有甚麼資料為次序屬性的資料？

# 等距屬性

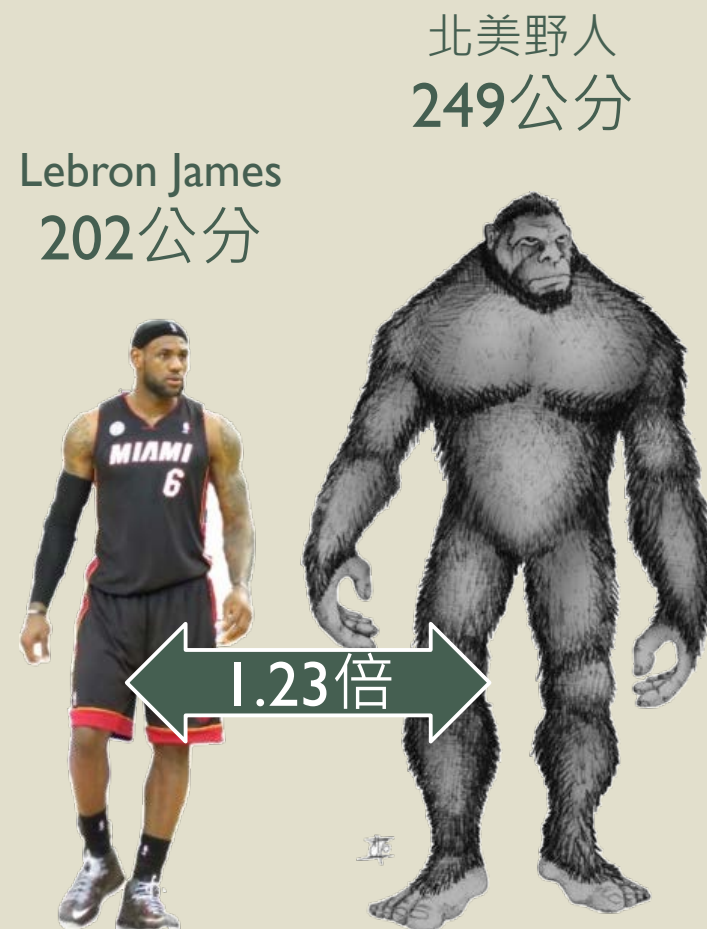
- 等距屬性(Interval Attribute) :
  - 各數值間擁有固定距離
    - $20^{\circ}\text{C}$ 與 $10^{\circ}\text{C}$ 差值 =  $30^{\circ}\text{C}$ 與 $20^{\circ}\text{C}$ 差值
  - 可進行數學加減計算
    - $30^{\circ}\text{C} - 20^{\circ}\text{C} = 10^{\circ}\text{C}$
  - 無法進行乘除運算，因無絕對零點
    - 滿意度評分

Q：還有甚麼資料為等距屬性的資料？



# 比率屬性

- 比率屬性(Ratio Attribute) :
  - 各數值間具有一定意義的比例
    - 身高200cm為100cm的2倍
  - 可進行數學乘除計算
    - 分數40分乘以2為80分
  - 比率變項資料分布情況可用眾數、中位數、算術平均數和幾何平均數來描述



Q：還有甚麼資料為比率屬性的資料？

# 結構化資料類別比較

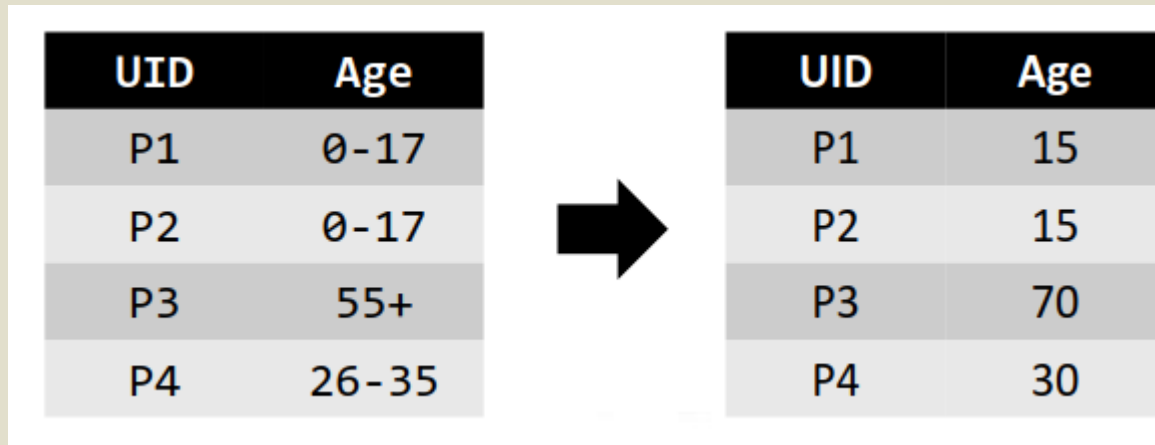
資料類別	別名	可分類 資料	有順序關係 可比較大小	固定距離& 可加減運算	絕對原點 可乘除運算	舉例
名目屬性	類別屬性	○	×	×	×	性別、種族、血型
次序屬性	順序屬性	○	○	×	×	社經地位、 教育程度、滿意度
等距屬性	區間屬性	○	○	○	×	溫度、智商、年份、 緯度
比率屬性	比例屬性	○	○	○	○	身高、體重、考試分 數、時間

# ENCODING ORDINAL FEATURES

- Ordinal features

For example: {Low, Medium, High}  $\rightarrow$  {0,1,2} or {5, 10, 15}

- Create a new feature using mean or median



The diagram illustrates the process of encoding an ordinal feature. It shows two tables connected by a large black arrow pointing from left to right. The left table has columns 'UID' and 'Age'. The 'Age' column contains categorical values: '0-17', '0-17', '55+', and '26-35'. The right table has the same 'UID' column, but the 'Age' column contains numerical values: 15, 15, 70, and 30. This represents the replacement of each categorical age group with its median value.

UID	Age
P1	0-17
P2	0-17
P3	55+
P4	26-35

UID	Age
P1	15
P2	15
P3	70
P4	30

# ENCODING CATEGORICAL FEATURES

- Categorical features → 沒有次序或大小之分
- Blood types: {"A", "B", "O", "AB"}
- **One-hot encoding**

A [1,0,0,0]

B [0,1,0,0]

O [0,0,1,0]

AB [0,0,0,1]

A [1 0 0 0] → RMSE =  $(1-1)^2 + (0-0)^2 + (0-0)^2 + (0-0)^2 = 0$

B [0 1 0 0] → RMSE =  $(0-0)^2 + (1-1)^2 + (0-0)^2 + (0-0)^2 = 0$

O [0 0 1 0] → RMSE =  $(0-0)^2 + (0-0)^2 + (1-1)^2 + (0-0)^2 = 0$

# ENCODING CATEGORICAL FEATURES

```
from keras.utils import np_utils
from sklearn.preprocessing import LabelEncoder
from sklearn import datasets
from sklearn.model_selection import train_test_split
```

```
encoder = LabelEncoder()  
encoded_Y = encoder.fit_transform(Y)  
print(encoded_Y)  
# convert integers to dummy variables (one hot encoding)  
dummy_y = np_utils.to_categorical(encoded_Y) # one-hot encoding [0. 1. 0.]  
print(dummy_y)
```

[illegible]

[[1. 0. 0.]  
[1. 0. 0.]  
[1. 0. 0.]  
[1. 0. 0.]  
[1. 0. 0.] ←  
[1. 0. 0.]  
[1. 0. 0.]  
[1. 0. 0.]  
[1. 0. 0.]  
[1. 0. 0.]

## ← One-hot encoding

## LabelEncoder()

## to\_categorical()

```
['setosa' 'versicolor' 'virginica']
```

0

1

2



# DATA SCALING-STANDARDSCALER()

```
from sklearn.preprocessing import StandardScaler  
sc = StandardScaler().fit(X_train)  
print(sc.mean_) #mean  
print(sc.scale_) #standard deviation
```

```
[5.49 2.87]  
[0.60642983 1.44275035]
```

```
#transform: (x-u)/std.  
X_train_std = sc.transform(X_train)  
print(X_train_std)
```

The scaler instance can then be used on new data to transform it the same way it did on the training set:

```
X_test_std = sc.transform(X_test)  
print(X_test_std)
```