

# **Data Analysis, A tutorial**

by LIN, WAN-CHING

# About Me

R&D at III, 資安所 智慧技術中心 智慧邊際運算組

GitHub: <https://github.com/wanching>. (<https://github.com/wanching>)

# Overview

## 1. Introduction

- 介紹資料分析相關套件

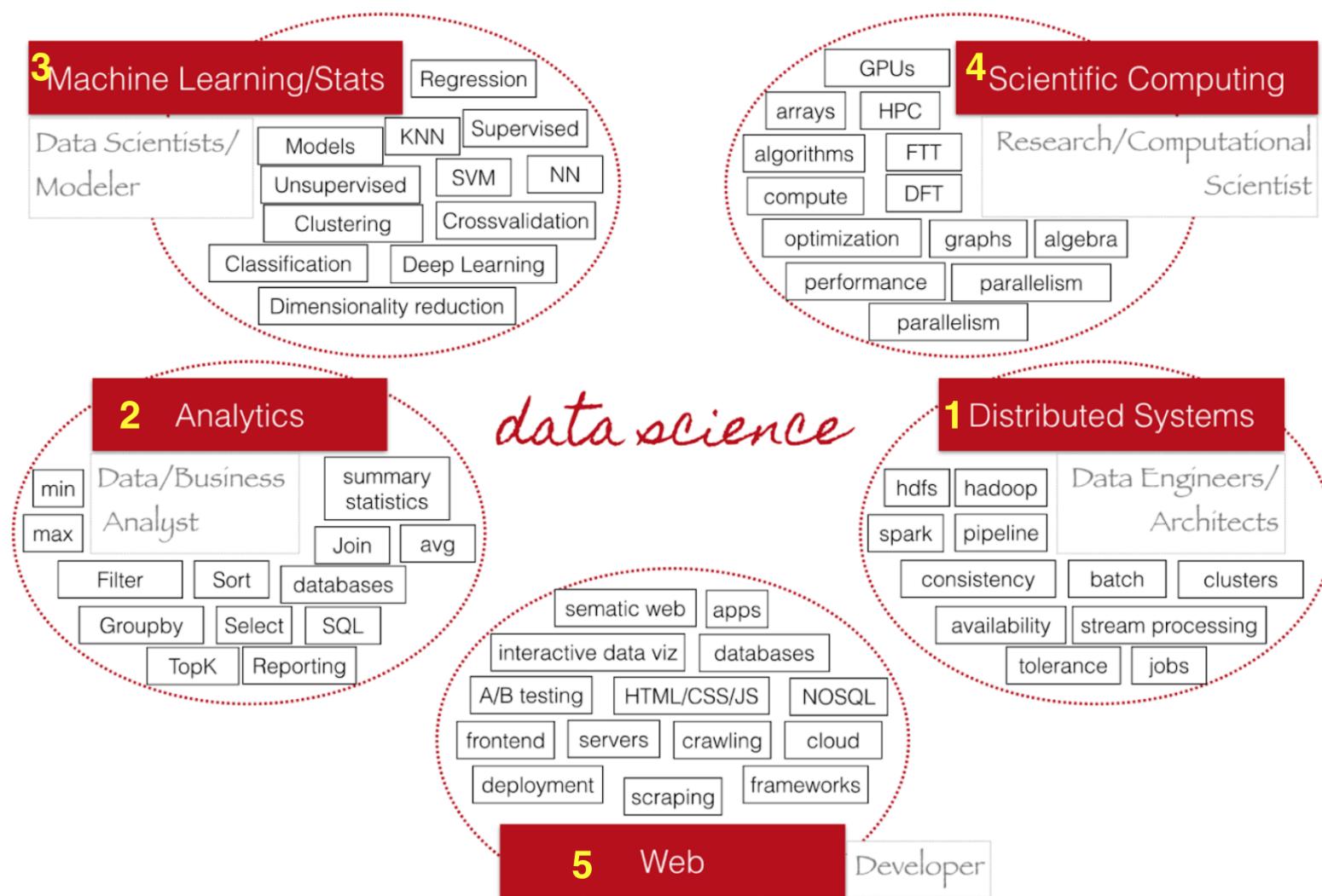
## 2. Implementation

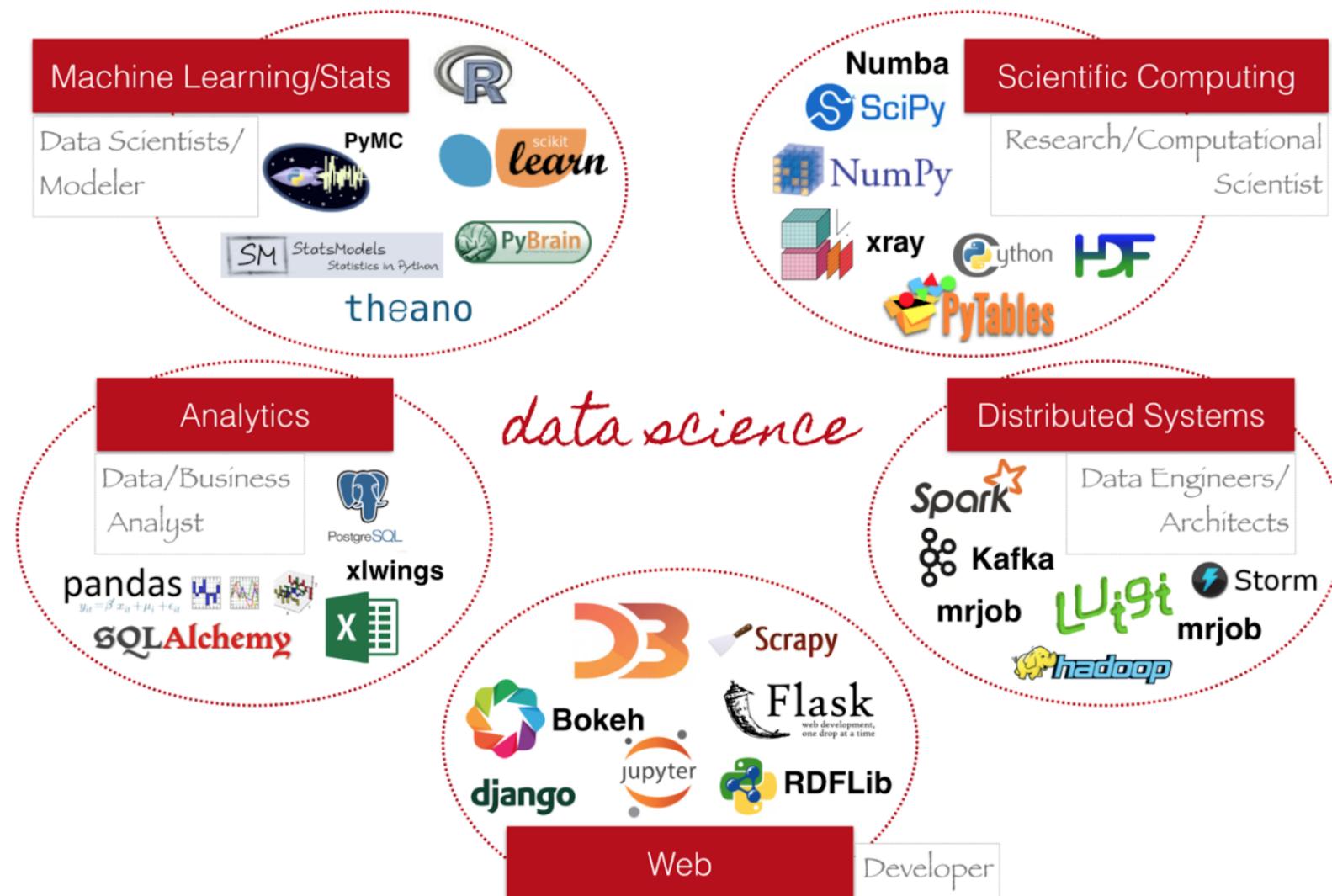
- 一些套件的用法

## 3. Case

# Introduction

介紹資料分析相關套件





# Python 資料分析相關套件

| Library      | Type             | Commits | Contributors | Releases | Watch | Star  | Fork  | Commits / Contributors | Commits / Releases | Star/Contributors |
|--------------|------------------|---------|--------------|----------|-------|-------|-------|------------------------|--------------------|-------------------|
| NumPy        | Data wrangling   | 15980   | 522          | 125      | 280   | 4286  | 2012  | 31                     | 128                | 8                 |
| SciPy        | Data wrangling   | 17213   | 489          | 91       | 244   | 3043  | 1775  | 35                     | 189                | 6                 |
| Pandas       | Data wrangling   | 15089   | 762          | 76       | 626   | 9394  | 3709  | 20                     | 199                | 12                |
| Matplotlib   | Visualization    | 21754   | 588          | 60       | 413   | 5190  | 2517  | 37                     | 363                | 9                 |
| Seaborn      | Visualization    | 1699    | 71           | 11       | 176   | 3878  | 580   | 24                     | 154                | 55                |
| Bokeh        | Visualization    | 15724   | 223          | 40       | 322   | 5720  | 1401  | 71                     | 393                | 26                |
| Plotly       | Visualization    | 2486    | 33           | 7        | 149   | 2044  | 512   | 75                     | 355                | 62                |
| SciKit-Learn | Machine learning | 21793   | 842          | 80       | 1650  | 18246 | 9997  | 26                     | 272                | 22                |
| Keras        | Machine learning | 3519    | 428          | 28       | 1025  | 15043 | 5227  | 8                      | 126                | 35                |
| TensorFlow   | Machine learning | 16785   | 795          | 29       | 5002  | 55486 | 26433 | 21                     | 579                | 70                |
| Theano       | Machine learning | 25870   | 300          | 23       | 520   | 6171  | 2116  | 86                     | 1125               | 21                |
| Scrapy       | Data scraping    | 6325    | 243          | 78       | 1427  | 20124 | 5353  | 26                     | 81                 | 83                |
| NLTK         | NLP              | 12449   | 196          | 20       | 376   | 4649  | 1358  | 64                     | 622                | 24                |
| Gensim       | NLP              | 2878    | 179          | 43       | 300   | 4182  | 1595  | 16                     | 67                 | 23                |
| Statsmodels  | Statistics       | 8960    | 119          | 19       | 194   | 2019  | 977   | 75                     | 472                | 17                |

ActiveWizards.com

28.04.2017

Ref: <https://activewizards.com/blog/top-15-libraries-for-data-science-in-python/>  
[\(https://activewizards.com/blog/top-15-libraries-for-data-science-in-python/\)](https://activewizards.com/blog/top-15-libraries-for-data-science-in-python/)

# Implementation

一些套件的用法

# Numpy

支援高階大量的維度陣列與矩陣運算，此外也針對陣列運算提供大量的數學函式函式庫。

# Numpy 屬性

## ndarray (N-dimensional array)

- ndim: 維度
- shape: 行數和列數
- size: 元素個數

### 1 D ARRAY:

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| C | O | D | I | N | G | E | E | K |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

← single row of elements

### 2 D ARRAY:

|            | col 0  | col 1 | col 2 |   |
|------------|--------|-------|-------|---|
| i<br>row 0 | j<br>0 | 0     | 1     | 2 |
| row 1      | 0      | A     | A     | A |
| row 2      | 1      | B     | B     | B |
| row 3      | 2      | C     | C     | C |

↑  
ROWS

← column

} array elements

# Numpy屬性

```
In [1]: # 引入 numpy 模組
import numpy as np
np1 = np.array([[1,2,3],[2,3,4]]) # list轉成矩陣
print(np1)

# 顯示相關訊息
print('np1 number of dim:', np1.ndim) # 維度
print('np1 shape :', np1.shape) # 行數和列數
print('np1 size:', np1.size) # 元素個數
```

```
[[1 2 3]
 [2 3 4]]
np1 number of dim: 2
np1 shape : (2, 3)
np1 size: 6
```

# Numpy 基本運算(1)

```
In [2]: import numpy as np  
a = np.array([10,20,30,40])  
b = np.arange(4)  
  
print('a= ', a)  
print('b= ', b)  
  
print('a+b= ', a+b)  
print('a-b= ', a-b)  
print('a*b= ', a*b)  
print('b**2= ', b**2)  
print('10*np.sin(a)= ', 10*np.sin(a))
```

```
a= [10 20 30 40]  
b= [0 1 2 3]  
a+b= [10 21 32 43]  
a-b= [10 19 28 37]  
a*b= [ 0 20 60 120]  
b**2= [0 1 4 9]  
10*np.sin(a)= [-5.44021111 9.12945251 -9.88031624 7.4511316 ]
```

Ref: <http://www.numpy.org/> (<http://www.numpy.org/>).

## Numpy 基本運算(2)

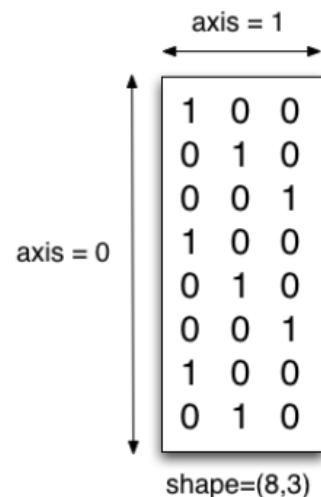
```
In [3]: import numpy as np  
A = np.arange(3,14)  
print('A=', A)
```

```
A= [ 3  4  5  6  7  8  9 10 11 12 13]
```

```
In [4]: A = np.arange(3,15).reshape((3,4))  
print('A=', A)  
print('A[1][1]= ', A[1][1])  
print(A.sum(axis=1))
```

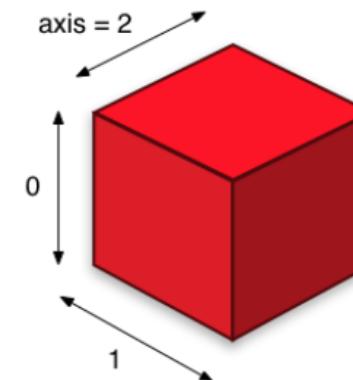
```
A= [[ 3  4  5  6]  
 [ 7  8  9 10]  
 [11 12 13 14]]  
A[1][1]=  8  
[18 34 50]
```

## Anatomy of an array



The **axes** of an array describe the order of indexing into the array, e.g., `axis=0` refers to the first index coordinate, `axis=1` the second, etc.

The **shape** of an array is a tuple indicating the number of elements along each axis. An existing array `a` has an attribute `a.shape` which contains this tuple.



- all elements must be of the same dtype (datatype)
- the default dtype is float
- arrays constructed from list of mixed dtype will be upcast to the "greatest" common type

Ref:

<http://pages.physics.cornell.edu/~myers/teaching/ComputationalMethods/python/arrays.html>  
<http://pages.physics.cornell.edu/~myers/teaching/ComputationalMethods/python/arrays.html>

# Pandas

基於Numpy的一個套件，不論在讀取或處理資料都很簡單且方便

# Pandas基本介紹

- Numpy 和 Pandas有什麼不同？
  - Numpy 是array形式的，沒有欄位名稱或標籤
  - Pandas 基於Numpy構建的，有列表的標籤
- 主要有兩個資料結構
  - Series
  - DataFrame

# Series

```
In [5]: import pandas as pd  
import numpy as np  
a = pd.Series([1,2,4,np.nan,1])  
print(a)
```

```
0    1.0  
1    2.0  
2    4.0  
3    NaN  
4    1.0  
dtype: float64
```

Series的表現形式為：左邊索引，右邊為值

因為我們沒指定索引，所以它會自己建立0到N-1(N為長度)的整數型索引

# DataFrame

```
In [6]: import pandas as pd  
import numpy as np  
  
dates = pd.date_range('20160101', periods=6)  
df = pd.DataFrame(np.random.randn(6,4), index=dates, columns=['a','b','c','d'])  
  
print(df)
```

|            | a         | b         | c         | d         |
|------------|-----------|-----------|-----------|-----------|
| 2016-01-01 | 0.197324  | -0.632418 | 0.085139  | -0.512581 |
| 2016-01-02 | -2.255365 | -0.782680 | -0.533920 | -0.731298 |
| 2016-01-03 | -0.884892 | 0.116170  | -2.288559 | -1.321955 |
| 2016-01-04 | 1.757448  | 0.599724  | 0.725383  | 0.918065  |
| 2016-01-05 | 1.389660  | 0.962131  | -0.618456 | 0.189330  |
| 2016-01-06 | 0.009656  | 3.819625  | -0.353362 | 0.713873  |

Dataframe為一個表格型的資料結構，既有行索引也有列索引，可被看成由Series組成的大集合

<https://www.jianshu.com/p/214798dd8f93> (<https://www.jianshu.com/p/214798dd8f93>)

```
In [7]: import pandas as pd
import numpy as np

df2 = pd.DataFrame({'A' : np.random.rand(4),
                    'B' : pd.Timestamp('20130102'),
                    'C' : pd.Series(1,index=list(range(4)),dtype='float32'),
                    'D' : np.array([3] * 4,dtype='int32'),
                    'E' : pd.Categorical(["test","train","test","train"]),
                    'F' : 'foo'})

print(df2)
# print(df2.dtypes) # 資料類型
# print(df2.columns) # 資料名稱
# print(df2.describe()) # 資料的總結
# print(df2.T) # transpose
# print(df2.sort_values(by='A'))
```

|   | A        | B          | C   | D | E     | F   |
|---|----------|------------|-----|---|-------|-----|
| 0 | 0.590392 | 2013-01-02 | 1.0 | 3 | test  | foo |
| 1 | 0.392421 | 2013-01-02 | 1.0 | 3 | train | foo |
| 2 | 0.223525 | 2013-01-02 | 1.0 | 3 | test  | foo |
| 3 | 0.546504 | 2013-01-02 | 1.0 | 3 | train | foo |

# Dataframe的loc(根據標籤)

# Dataframe的iloc(根據序列)

```
In [9]: dates = pd.date_range('20130101', periods=6)
df = pd.DataFrame(np.arange(24).reshape((6,4)),index=dates, columns=['A','B','C','D'])

print(df)
print(df.iloc[3,1])
print(df.iloc[3:5,1:3])
print(df.iloc[[1,3,5],1:3])
```

|            | A  | B  | C  | D  |
|------------|----|----|----|----|
| 2013-01-01 | 0  | 1  | 2  | 3  |
| 2013-01-02 | 4  | 5  | 6  | 7  |
| 2013-01-03 | 8  | 9  | 10 | 11 |
| 2013-01-04 | 12 | 13 | 14 | 15 |
| 2013-01-05 | 16 | 17 | 18 | 19 |
| 2013-01-06 | 20 | 21 | 22 | 23 |
| 13         |    |    |    |    |
|            |    | B  | C  |    |
| 2013-01-04 | 13 | 14 |    |    |
| 2013-01-05 | 17 | 18 |    |    |
|            |    | B  | C  |    |
| 2013-01-02 | 5  | 6  |    |    |
| 2013-01-04 | 13 | 14 |    |    |
| 2013-01-06 | 21 | 22 |    |    |

# Dataframe的Boolean indexing

```
In [10]: dates = pd.date_range('20130101', periods=6)
df = pd.DataFrame(np.arange(24).reshape((6,4)),index=dates, columns=['A','B','C','D'])

print(df)
print(df[df.A>8])
```

|            | A  | B  | C  | D  |
|------------|----|----|----|----|
| 2013-01-01 | 0  | 1  | 2  | 3  |
| 2013-01-02 | 4  | 5  | 6  | 7  |
| 2013-01-03 | 8  | 9  | 10 | 11 |
| 2013-01-04 | 12 | 13 | 14 | 15 |
| 2013-01-05 | 16 | 17 | 18 | 19 |
| 2013-01-06 | 20 | 21 | 22 | 23 |

|            | A  | B  | C  | D  |
|------------|----|----|----|----|
| 2013-01-04 | 12 | 13 | 14 | 15 |
| 2013-01-05 | 16 | 17 | 18 | 19 |
| 2013-01-06 | 20 | 21 | 22 | 23 |

# Pandas I/O

| Format Type | Data Description     | Reader         | Writer       |
|-------------|----------------------|----------------|--------------|
| text        | CSV                  | read_csv       | to_csv       |
| text        | JSON                 | read_json      | to_json      |
| text        | HTML                 | read_html      | to_html      |
| text        | Local clipboard      | read_clipboard | to_clipboard |
| binary      | MS Excel             | read_excel     | to_excel     |
| binary      | HDF5 Format          | read_hdf       | to_hdf       |
| binary      | Feather Format       | read_feather   | to_feather   |
| binary      | Parquet Format       | read_parquet   | to_parquet   |
| binary      | Msgpack              | read_msgpack   | to_msgpack   |
| binary      | Stata                | read_stata     | to_stata     |
| binary      | SAS                  | read_sas       |              |
| binary      | Python Pickle Format | read_pickle    | to_pickle    |
| SQL         | SQL                  | read_sql       | to_sql       |
| SQL         | Google Big Query     | read_gbq       | to_gbq       |

Ref: <https://pandas.pydata.org/pandas-docs/stable/io.html>

(<https://pandas.pydata.org/pandas-docs/stable/io.html>)

# Pandas load csv

```
In [11]: import pandas as pd  
  
data = pd.read_csv('student.csv')  
print(data)  
  
data.to_pickle('student.pickle')
```

|    | Student ID | name  | age | gender |
|----|------------|-------|-----|--------|
| 0  | 1100       | Kelly | 22  | Female |
| 1  | 1101       | Clo   | 21  | Female |
| 2  | 1102       | Tilly | 22  | Female |
| 3  | 1103       | Tony  | 24  | Male   |
| 4  | 1104       | David | 20  | Male   |
| 5  | 1105       | Catty | 22  | Female |
| 6  | 1106       | M     | 3   | Female |
| 7  | 1107       | N     | 43  | Male   |
| 8  | 1108       | A     | 13  | Male   |
| 9  | 1109       | S     | 12  | Male   |
| 10 | 1110       | David | 33  | Male   |
| 11 | 1111       | Dw    | 3   | Female |
| 12 | 1112       | Q     | 23  | Male   |
| 13 | 1113       | W     | 21  | Female |

# Scikit-learn

適合想要使用Python實作機器學習的初學者，提供許多演算法，包含監督式、非監督式...等

# Machine Learning ≈ Looking for a Function

- Speech Recognition

$$f( \text{[sound波形图]} ) = \text{"How are you"}$$

- Image Recognition

$$f( \text{[小猫图片]} ) = \text{"Cat"}$$

- Playing Go

$$f( \text{[围棋棋盘]} ) = \text{"5-5" (next move)}$$

- Dialogue System

$$f( \text{“Hi”} ) = \text{“Hello”}$$

(what the user said)      (system response)

Ref: [http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML\\_2017\\_2/Lecture/policy.pdf](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017_2/Lecture/policy.pdf)  
[\(http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML\\_2017\\_2/Lecture/policy.pdf\)](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017_2/Lecture/policy.pdf)

# Framework

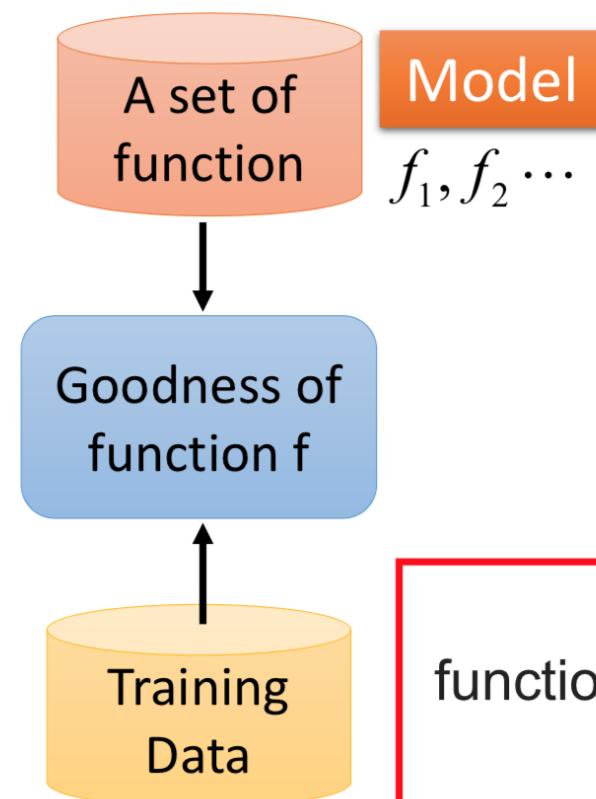


Image Recognition:

$$f(\text{cat image}) = \text{"cat"}$$


|   |   |
|---|---|
| $f_1(\text{cat image}) = \text{"cat"}$  | $f_2(\text{cat image}) = \text{"money"}$  |
|  |  |
| <b>Better!</b>  |   |
| $f_1(\text{dog image}) = \text{"dog"}$  | $f_2(\text{dog image}) = \text{"snake"}$  |
|  |  |

function input:



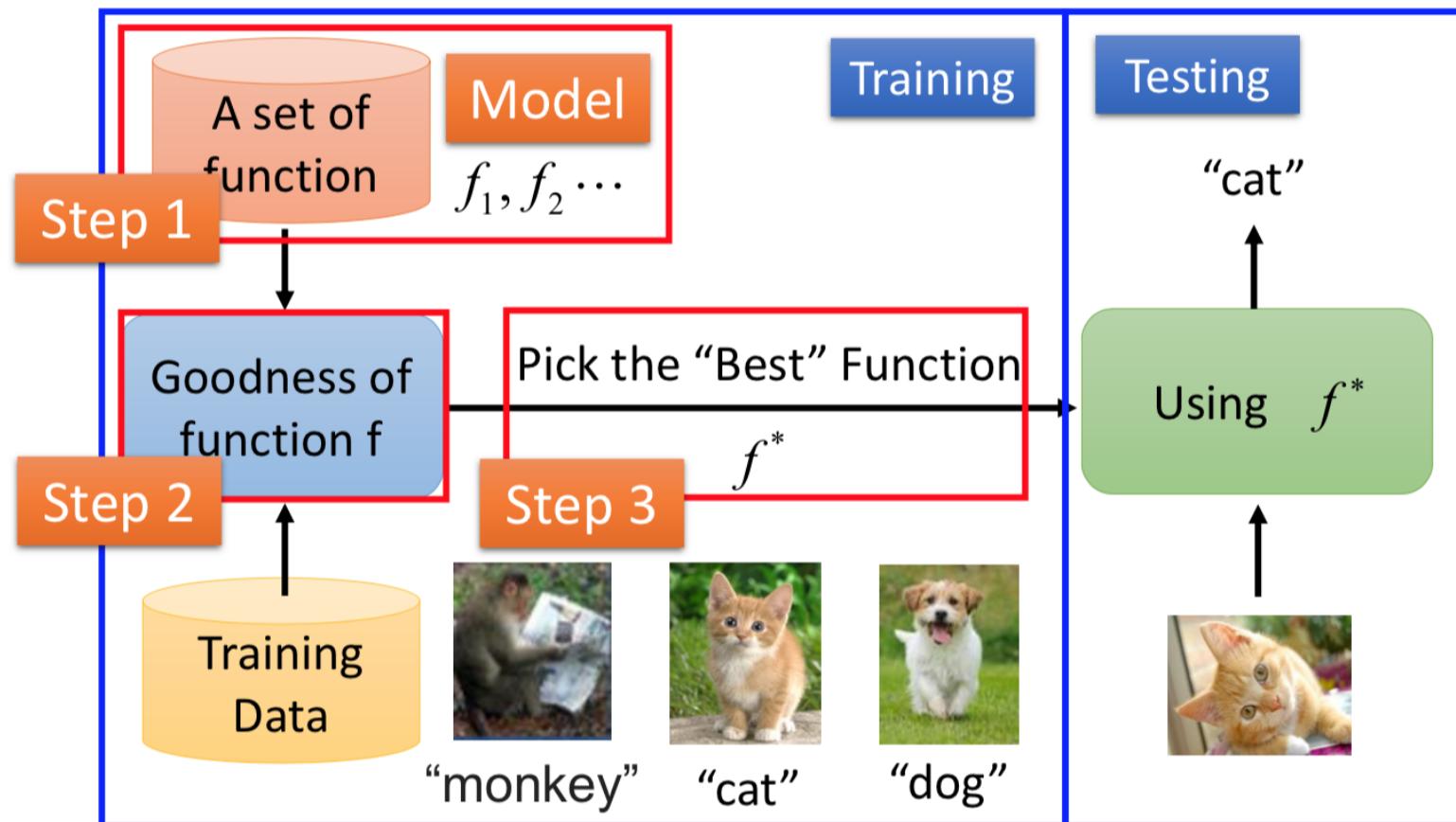
function output: "monkey"    "cat"    "dog"

Ref: [http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML\\_2017\\_2/Lecture/policy.pdf](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017_2/Lecture/policy.pdf)  
[\(http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML\\_2017\\_2/Lecture/policy.pdf\)](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017_2/Lecture/policy.pdf)

# Image Recognition: Framework

Image Recognition:

$$f(\text{cat}) = \text{"cat"}$$



Ref: [http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML\\_2017\\_2/Lecture/policy.pdf](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017_2/Lecture/policy.pdf)  
[\(http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML\\_2017\\_2/Lecture/policy.pdf\)](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017_2/Lecture/policy.pdf)

# 範例 : The digits dataset

- [http://scikit-learn.org/stable/auto examples/datasets/plot digits last image.html](http://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html)  
[\(http://scikit-learn.org/stable/auto examples/datasets/plot digits last image.html\)](http://scikit-learn.org/stable/auto_examples/datasets/plot_digits_last_image.html)

# 1. 載入手寫數字資料

In [12]:

```
# 這行是在ipython notebook的介面裏專用，如果在其他介面則可以拿掉
%matplotlib inline
from sklearn import datasets, svm, metrics

import matplotlib.pyplot as plt

# 載入數字資料集
digits = datasets.load_digits()

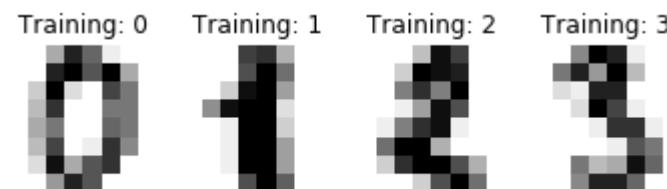
for key, value in digits.items():
    try:
        print(key, value.shape)
    except:
        print(key)

data (1797, 64)
target (1797,)
target_names (10,)
images (1797, 8, 8)
DESCR
```

| 顯示                          | 說明  |
|-----------------------------|---|
| ('images', (1797L, 8L, 8L)) | 共有 1797 張影像，影像大小為 8x8                     |
| ('data', (1797L, 64L))      | data 則是將8x8的矩陣攤平成64個元素之一維向量               |
| ('target_names', (10L,))    | 說明10種分類之對應 [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] |
| DESCR                       | 資料之描述                                     |
| ('target', (1797L,))        | 記錄1797張影像各自代表那一個數字                        |

每張影像所對照的實際數字存在`digits.target`變數中

```
In [13]: images_and_labels = list(zip(digits.images, digits.target))
for index, (image, label) in enumerate(images_and_labels[:4]):
    plt.subplot(2, 4, index + 1)
    plt.axis('off')
    plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    plt.title('Training: %i' % label)
```



## 2. 訓練以及分類

```
In [14]: n_samples = len(digits.images)

# 資料攤平: 1797 x 8 x 8 -> 1797 x 64
# 這裏的-1代表自動計算，相當於 (n_samples, 64)
data = digits.images.reshape((n_samples, -1))

# 產生SVC分類器
classifier = svm.SVC(gamma=0.001)

# 用前半部份的資料來訓練
classifier.fit(data[:n_samples // 2], digits.target[:n_samples // 2])

expected = digits.target[n_samples // 2:]

# 利用後半部份的資料來測試分類器，共 899 筆資料
predicted = classifier.predict(data[n_samples // 2:])

print("expected前10個值= ", expected[:10])
print("predicted前10個值= ", predicted[:10])
```

```
expected前10個值=  [8 8 4 9 0 8 9 8 1 2]
predicted前10個值=  [8 8 4 9 0 8 9 8 1 2]
```

### 3. 分類準確度統計

- 使用Confusion matrix(混淆矩陣)來統計
- `metrics.confusion_matrix(真實資料:899, 預測資料:899)`

```
In [15]: print("Confusion matrix:\n%s"
    % metrics.confusion_matrix(expected, predicted))
```

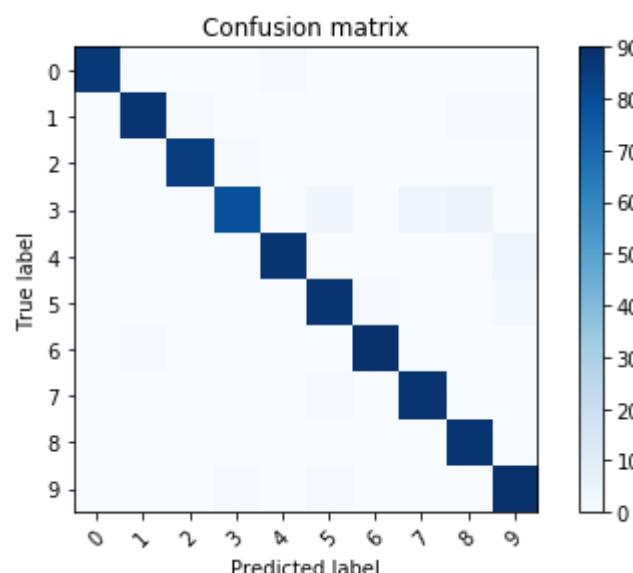
```
Confusion matrix:
[[ 87   0   0   0   1   0   0   0   0   0]
 [  0  88   1   0   0   0   0   0   1   1]
 [  0   0  85   1   0   0   0   0   0   0]
 [  0   0   0  79   0   3   0   4   5   0]
 [  0   0   0   0  88   0   0   0   0   4]
 [  0   0   0   0   0  88   1   0   0   2]
 [  0   1   0   0   0   0  90   0   0   0]
 [  0   0   0   0   0   1   0  88   0   0]
 [  0   0   0   0   0   0   0   0  88   0]
 [  0   0   0   1   0   1   0   0   0  90]]
```

由矩陣可以看出，實際為3時，有數次誤判為5,7,8。

## Confusion matrix (圖示)

```
In [16]: def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):
    import numpy as np
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(digits.target_names))
    plt.xticks(tick_marks, digits.target_names, rotation=45)
    plt.yticks(tick_marks, digits.target_names)
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')

plt.figure()
plot_confusion_matrix(metrics.confusion_matrix(expected, predicted))
```



# Confusion matrix (以手寫3為例)

|                    | True Positive | True Negative |
|--------------------|---------------|---------------|
| Predicted Positive | 79            | 2             |
| Predicted Negative | 12            | 806           |

| Measure                                 | Value  | Derivations  |
|---|--------|--|
| <b>Sensitivity</b>                      | 0.8681 | $TPR = TP / (TP + FN)$   |
| <b>Specificity</b>                      | 0.9975 | $SPC = TN / (FP + TN)$   |
| <b>Precision</b>                        | 0.9753 | $PPV = TP / (TP + FP)$   |
| <b>Negative Predictive Value</b>        | 0.9853 | $NPV = TN / (TN + FN)$   |
| <b>False Positive Rate</b>              | 0.0025 | $FPR = FP / (FP + TN)$   |
| <b>False Discovery Rate</b>             | 0.0247 | $FDR = FP / (FP + TP)$   |
| <b>False Negative Rate</b>              | 0.1319 | $FNR = FN / (FN + TP)$   |
| <b>Accuracy</b>                         | 0.9844 | $ACC = (TP + TN) / (P + N)$  |
| <b>F1 Score</b>                         | 0.9186 | $F1 = 2TP / (2TP + FP + FN)$   |
| <b>Matthews Correlation Coefficient</b> | 0.9119 | $TP \cdot TN - FP \cdot FN / \sqrt{((TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN))}$ |

## Confusion Matrix 統計數據

- `metrics.classification_report(expected, predicted)`

```
In [17]: print("Classification report for classifier %s:\n%s\n"
      % (classifier, metrics.classification_report(expected, predicted)))
```

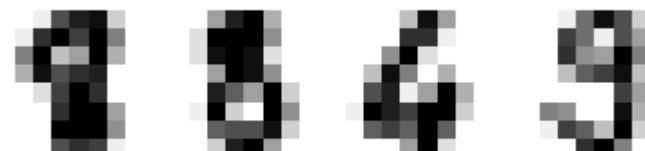
Classification report for classifier SVC(C=1.0, cache\_size=200, class\_weight=None, coef0=0.0,
decision\_function\_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
max\_iter=-1, probability=False, random\_state=None, shrinking=True,
tol=0.001, verbose=False):

|             | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0           | 1.00      | 0.99   | 0.99     | 88      |
| 1           | 0.99      | 0.97   | 0.98     | 91      |
| 2           | 0.99      | 0.99   | 0.99     | 86      |
| 3           | 0.98      | 0.87   | 0.92     | 91      |
| 4           | 0.99      | 0.96   | 0.97     | 92      |
| 5           | 0.95      | 0.97   | 0.96     | 91      |
| 6           | 0.99      | 0.99   | 0.99     | 91      |
| 7           | 0.96      | 0.99   | 0.97     | 89      |
| 8           | 0.94      | 1.00   | 0.97     | 88      |
| 9           | 0.93      | 0.98   | 0.95     | 92      |
| avg / total | 0.97      | 0.97   | 0.97     | 899     |

## 觀察測試影像以及預測(分類)結果得對應關係

```
In [18]: images_and_predictions = list(  
    zip(digits.images[n_samples // 2:], predicted))  
for index, (image, prediction) in enumerate(images_and_predictions[:4]):  
    plt.subplot(2, 4, index + 5)  
    plt.axis('off')  
    plt.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')  
    plt.title('Prediction: %i' % prediction)
```

Prediction: 8 Prediction: 8 Prediction: 4 Prediction: 9



手寫字辨識完整程式碼：[http://scikit-learn.org/stable/\\_downloads/plot\\_digits\\_classification.py](http://scikit-learn.org/stable/_downloads/plot_digits_classification.py)([http://scikit-learn.org/stable/\\_downloads/plot\\_digits\\_classification.py](http://scikit-learn.org/stable/_downloads/plot_digits_classification.py)).

## 學習推薦：

- 莫須Python: <https://morvanzhou.github.io/> (<https://morvanzhou.github.io/>)
- 李宏毅教授:  
<https://www.youtube.com/channel/UC2ggjtuuWvxrHHHiaDH1dlQ/playlists>  
(<https://www.youtube.com/channel/UC2ggjtuuWvxrHHHiaDH1dlQ/playlists>).

Thanks for listening!