# Data Science Toolchain

presented by Jie-Han Chen

slide: https://goo.gl/1hXBGk

# Language & Software

- Python
- R
- Java
- Matlab
- Octave
- Jupyter Notebook

# Python

- Open Source Community
- Package
- Web Service
- Good Readability
- Machine Learning

# R

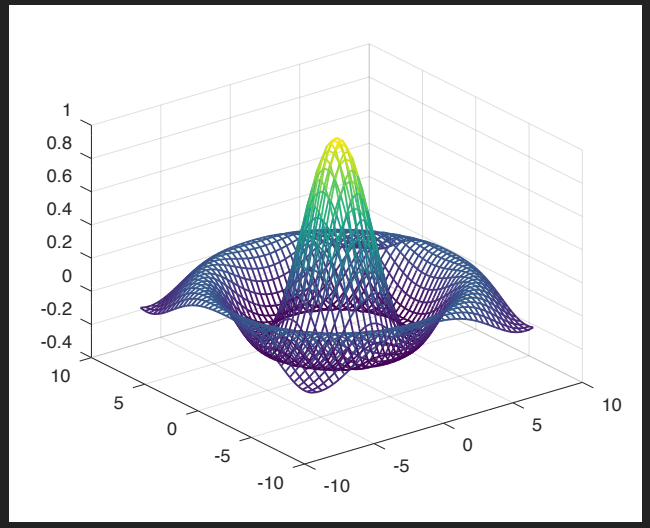
- Open Source Community
- Built-in Statistics Package
- Standalone computing & data analysis
- Slower than Python

# Java

- High Performance
- Big Data
- Poor Visualization, Modeling

# Matlab & Octave

- Powerful built-in math functions
- Simple Data Visualization tool
- Prototyping



# Jupyter Notebook

- Support 40+ programming language.
  eg: Python, R, Scala...
- Excellent for sharing your experiments
- Markdown, Latex
- example1
- example2

# Language & Software

- Python
- R
- Java
- Matlab
- Octave
- Jupyter Notebook

# Data Science Roadmap

# Data Science Toolchains

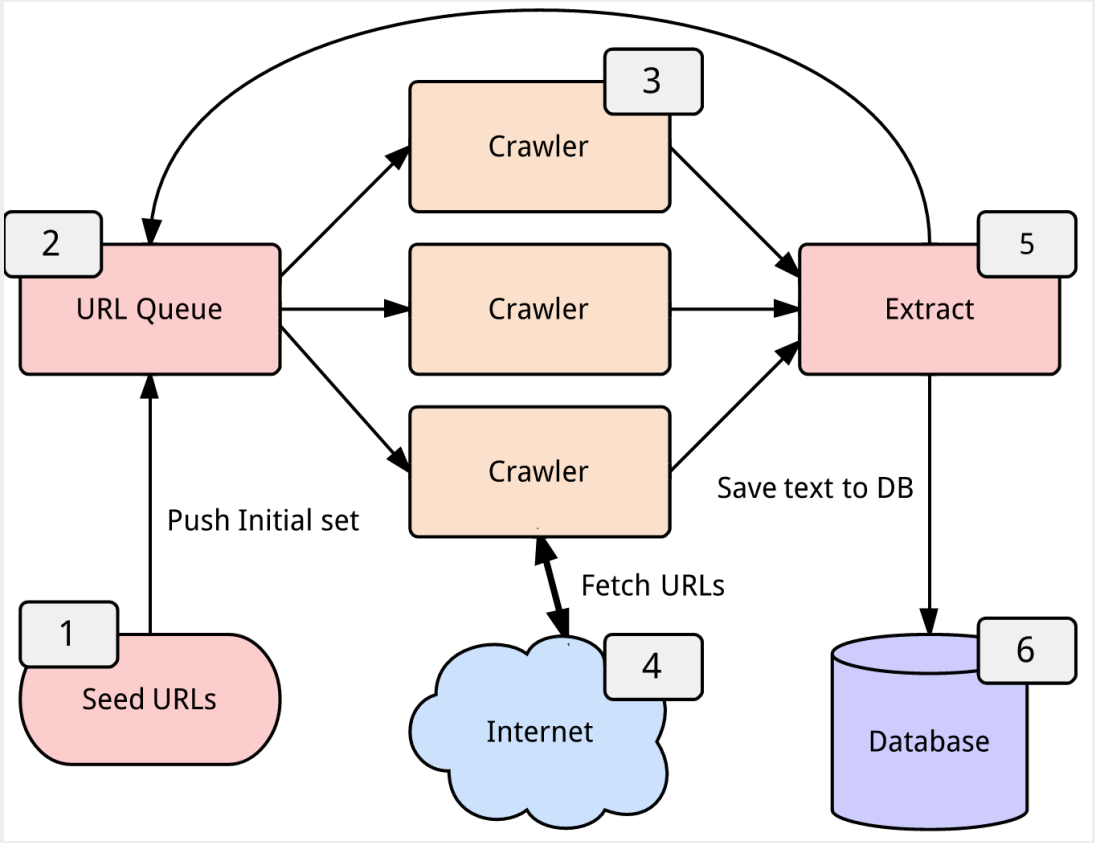
- Data Collection
- Data Visualization
- Data Storage
- Algorithm & Modeling


Where is
your data?

# Data Collection

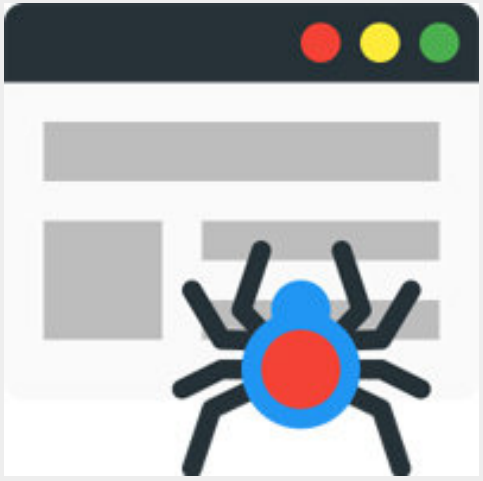
- Using API: Facebook, Wikipedia
- Web Scraper

# Web Scraper


3
Crawler
2
URL Queue
Crawler
5
Extract
Crawler
Push Initial set
Save text to DB
Fetch URLs
1
Seed URLs
4
Internet
6
Database

# Web Scraper

## HTTP request + HTML Parser



# HTTP: python-requests

- Better than built-in urllib
- Sessions with Cookie Persistence
- Thread-safety

# HTTP: python-requests

```
>>> r = requests.get('https://api.github.com/user', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json; charset=utf8'
>>> r.encoding
'utf-8'
>>> r.text
u'{"type":"User"...'
>>> r.json()
{u'private_gists': 419, u'total_private_repos': 77, ...}
```

# HTTP: python-requests

```
>>> payload = {'key1': 'value1', 'key2': 'value2'}

>>> r = requests.post("http://httpbin.org/post", data=payload)
>>> print(r.text)
{
  ...
  "form": {
    "key2": "value2",
    "key1": "value1"
  },
  ...
}
```

# Web page

```
<!DOCTYPE html>
<html lang="en">
  <head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/fb# 2490221586: http://ogp.me/ns/fb/2490221586#">
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1, user-scalable=no" />
    <meta name="include_mode" content="async">

    <!-- SL:start:notranslate -->
    <title>用十分鐘理解 《神經網路發展史》</title>
    <meta name="description" content="十分鐘系列： http://ccc.nqu.edu.tw/wd.html#ccc/slide.wd">
    <!-- SL:end:notranslate -->

    <meta name="robots" content="index">
    <meta id='globalTrackingUrl' content="https://www.linkedin.com/li/track">

<script type="text/javascript">window.NREUM||(NREUM={});NREUM.info=
{"transactionName":"cV9ZRUZWXQ1XEU1FDVtUUkJcVkZORAoHQQ==","applicationID":"22199103","applicationTime":1165,"queue
","errorBeacon":"bam.nr-data.net","beacon":"bam.nr-data.net"}</script>
<script type="text/javascript">(window.NREUM||(NREUM={})).loader_config={xpid:"XQ4PQlRRCQoJVVFR"};window.NREUM||(N
{function r(n){if(!e[n]){var o=e[n]={exports:{}};t[n][0].call(o.exports,function(e){var o=t[n][1][e];return r(o||e
e[n].exports}if("function"==typeof __nr_require)return __nr_require;for(var o=0;o<n.length;o++)r(n[o]);return r}({
{try{c.console&&console.log(t)}catch(e){}}var o,i=t("ee"),a=t(15),c={};try{o=localStorage.getItem("__nr_flags").sp
console.log&&(c.console=!0,o.indexOf("dev")!==-1&&(c.dev=!0),o.indexOf("nr_dev")!==-1&&(c.nrDev=!0))}catch(s){}c.n
{r(t.stack)}),c.dev&&i.on("fn-err",function(t,e,n){r(n.stack)}),c.dev&&(r("NR AGENT IN DEVELOPMENT MODE"),r("flags
")))},{}],2:[function(t,e,n){function r(t,e,n,r,o){try{d?d-=1:i("err",[o||new UncaughtException(t,e,n)])}catch(c){
Date).getTime(),!0])}catch(s){}}return"function"==typeof f&&f.apply(this,a(arguments))}function UncaughtException(
```

# Parser!

Regular Expression?

# BeautifulSoup

HTML/XML parser

# 用 BeautifulSoup 抓取 Ptt 標題

# HTML parser

| Parser | Typical usage | Advantages | Disadvantages |
|---|---|---|---|
| Python’s html.parser | `BeautifulSoup(markup, "html.parser")` | • Batteries included<br>• Decent speed<br>• Lenient (as of Python 2.7.3 and 3.2.) | • Not very (before Pyt or 3.2.2) |
| lxml’s HTML parser | `BeautifulSoup(markup, "lxml")` | • Very fast<br>• Lenient | • External dependenc |
| lxml’s XML parser | `BeautifulSoup(markup, "lxml-xml")`<br>`BeautifulSoup(markup, "xml")` | • Very fast<br>• The only currently supported XML parser | • External dependenc |
| html5lib | `BeautifulSoup(markup, "html5lib")` | • Extremely lenient<br>• Parses pages the same way a web browser does<br>• Creates valid HTML5 | • Very slow<br>• External dependenc |

# More Powerful Tool?

# Scrapy

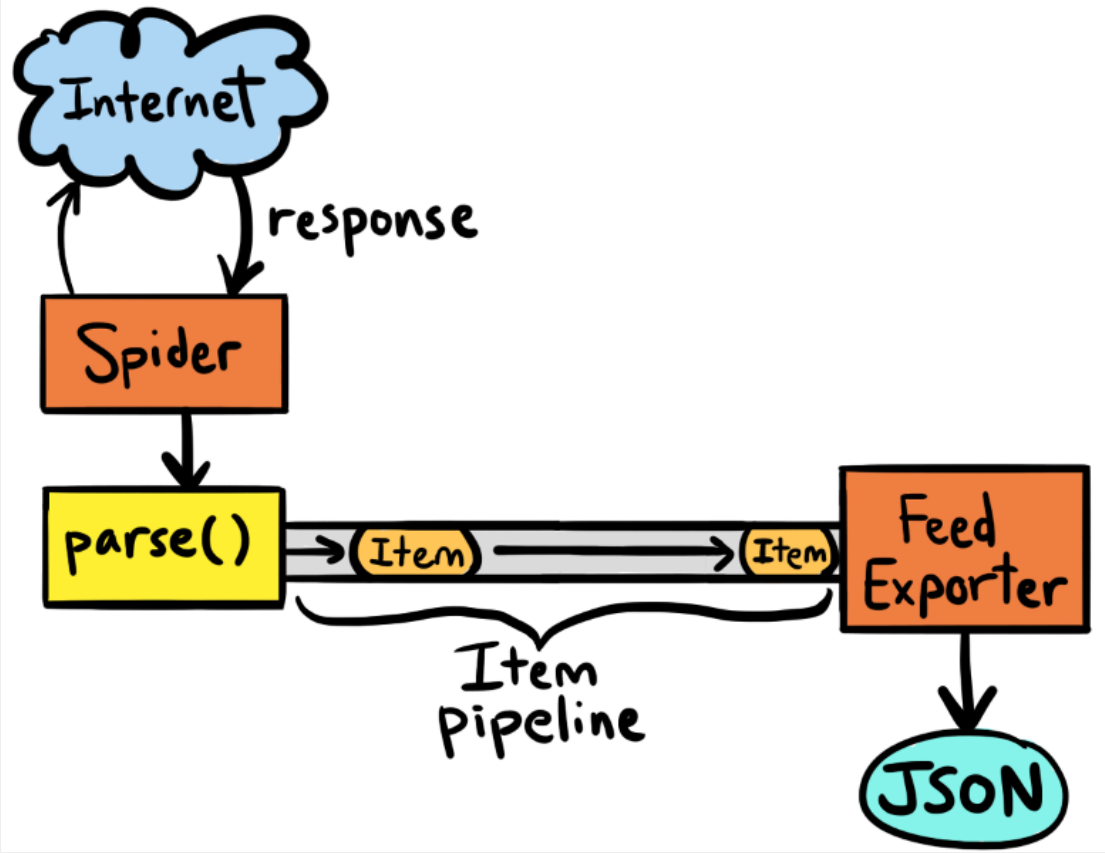
An open source and collaborative **framework** for extracting the data you need from websites.In a **fast**, simple, yet extensible way.

# Scrapy

$ scrapy startproject tutorial

```
tutorial/
    scrapy.cfg            # deploy configuration file

    tutorial/             # project's Python module, you'll import your code from here
        __init__.py

        items.py          # project items definition file

        pipelines.py      # project pipelines file

        settings.py       # project settings file

        spiders/          # a directory where you'll later put your spiders
            __init__.py
```

# Scrapy


Internet
response
Spider
parse()
Item
Item
Feed Exporter
Item pipeline
JSON

# Scrapy

path: /scrapy/dmoz.py
crawler name: dmoz

```
1  from scrapy.spiders import Spider
2  from scrapy.selector import Selector
3
4  from dirbot.items import Website
5
6
7  class DmozSpider(Spider):
8      name = "dmoz"
9      allowed_domains = ["dmoz.org"]
10     start_urls = [
11         "http://www.dmoz.org/Computers/Programming/Languages/Python/Books/",
12         "http://www.dmoz.org/Computers/Programming/Languages/Python/Resources/",
13     ]
14
```

# Scrapy

```
def parse(self, response):
    """
    The lines below is a spider contract. For more info see:
    http://doc.scrapy.org/en/latest/topics/contracts.html

    @url http://www.dmoz.org/Computers/Programming/Languages/Python/Resources/
    @scrapes name
    """
    sites = response.css('#site-list-content > div.site-item > div.title-and-desc')
    items = []

    for site in sites:
        item = Website()
        item['name'] = site.css(
            'a > div.site-title::text').extract_first().strip()
        item['url'] = site.xpath(
            'a/@href').extract_first().strip()
        item['description'] = site.css(
            'div.site-descr::text').extract_first().strip()
        items.append(item)

    return items
```

# Scrapy

$ scrapy crawl dmoz

# Selenium

**Selenium is a portable software testing framework for web applications.**

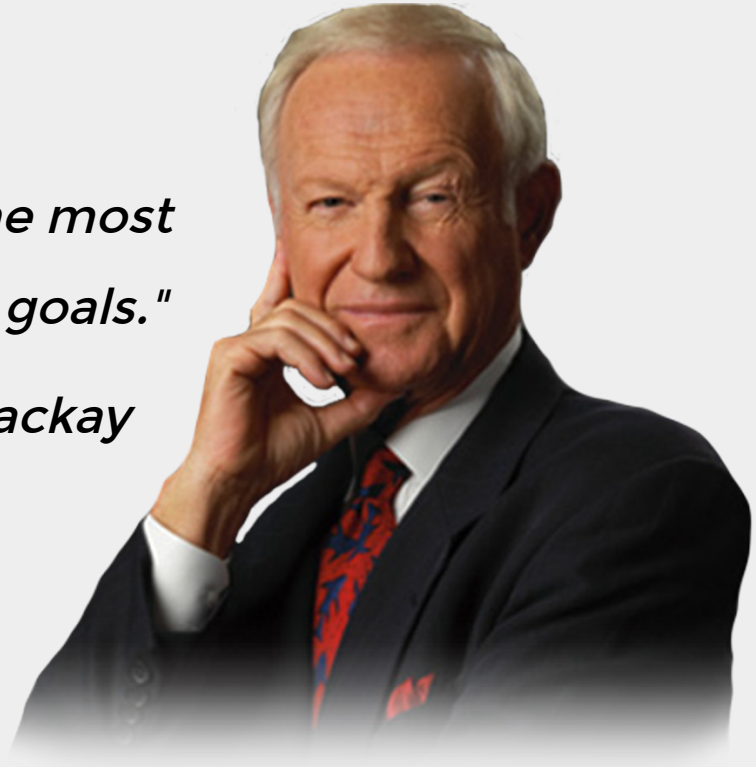
# robots.txt

# youtube.com/robots.txt

```
# robots.txt file for YouTube
# Created in the distant future (the year 2000) after
# the robotic uprising of the mid 90's which wiped out all humans.

User-agent: Mediapartners-Google*
Disallow:

User-agent: *
Disallow: /channel/*/community
Disallow: /comment
Disallow: /get_video
Disallow: /get_video_info
Disallow: /login
Disallow: /results
Disallow: /signup
Disallow: /t/terms
Disallow: /user/*/community
Disallow: /verify_age
Disallow: /watch_ajax
Disallow: /watch_fragments_ajax
Disallow: /watch_popup
Disallow: /watch_queue_ajax
```

*"I believe that **visualization** is one of the most powerful means of achieving personal goals."*

*—Harvey Mackay*

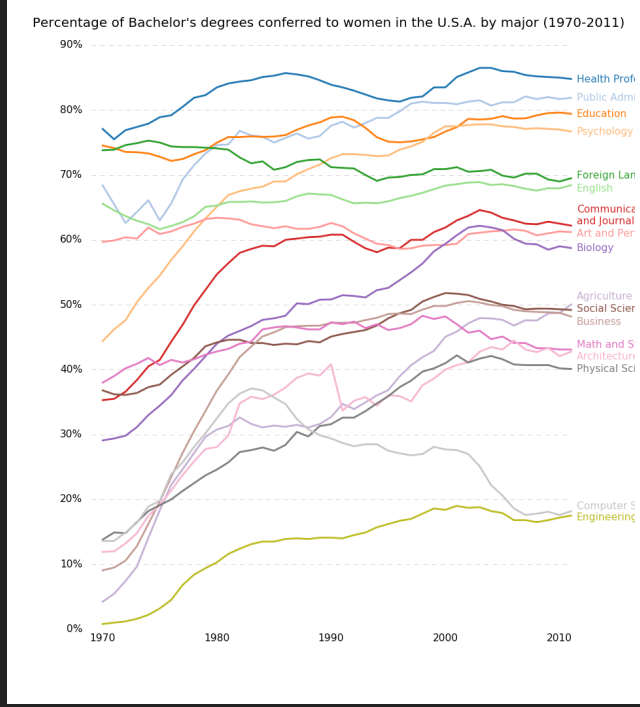


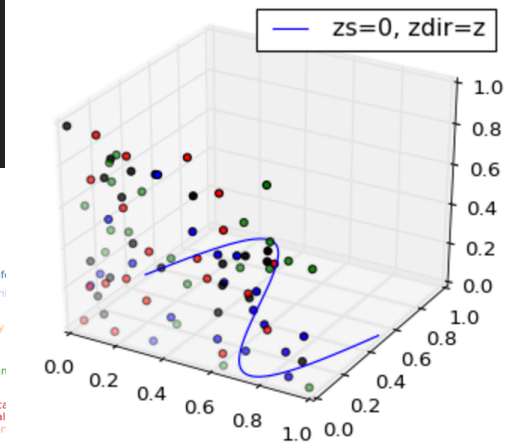

Data Visualization

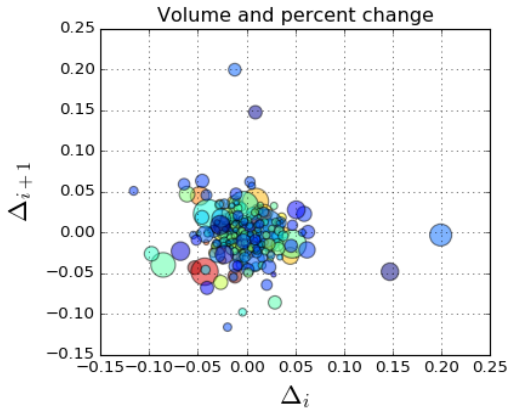
# Data Visualization

- Matplotlib, ggplot2
- D3.js
- Bokeh
- Tableau
- PlotDB
- Leaflet

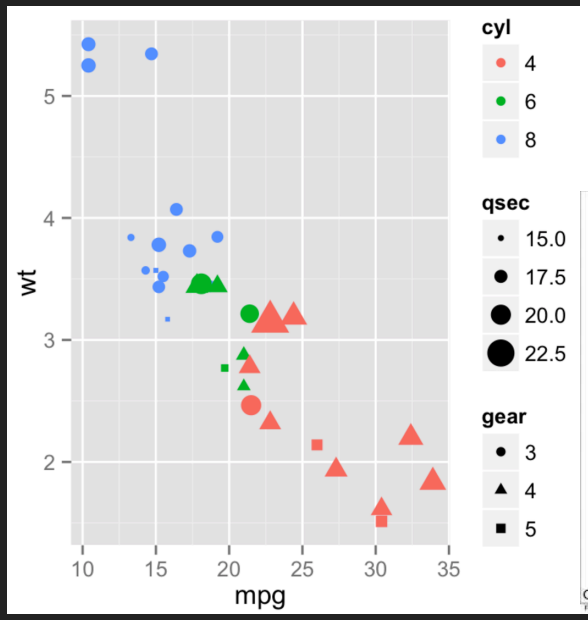
# Matplotlib

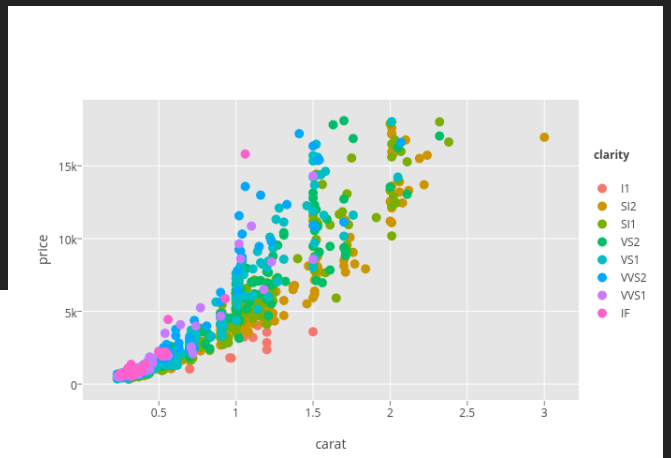

Percentage of Bachelor's degrees conferred to women in the U.S.A. by major (1970-2011)
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
1970
1980
1990
2000
2010
Health Prof
Public Admi
Education
Psychology
Foreign Lan
English
Communica
and Journal
Art and Per
Biology
Agriculture
Social Scier
Business
Math and S
Architectur
Physical Sci
Computer S
Engineering

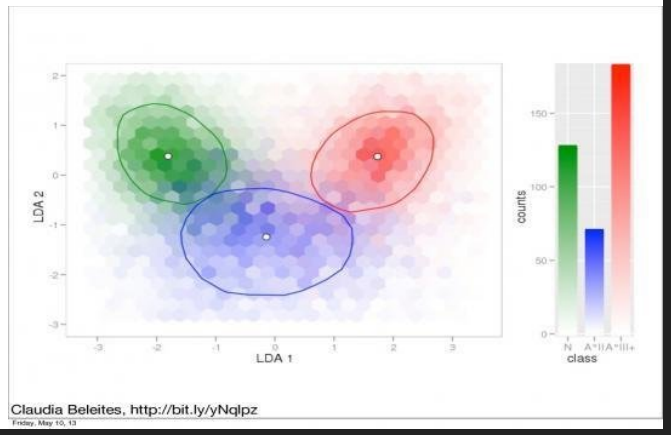

zs=0, zdir=z
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Volume and percent change
$\Delta_{i+1}$
$\Delta_i$
0.25
0.20
0.15
0.10
0.05
0.00
−0.05
−0.10
−0.15
−0.15 −0.10 −0.05 0.00 0.05 0.10 0.15 0.20 0.25

# ggplot2

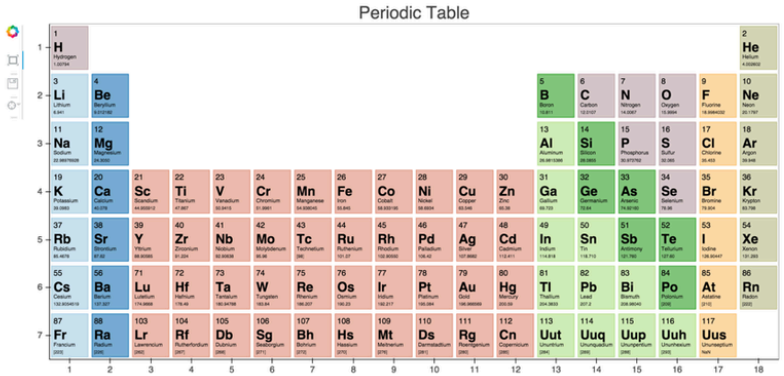

wt
mpg
cyl
4
6
8
qsec
15.0
17.5
20.0
22.5
gear
3
4
5


price
carat
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF

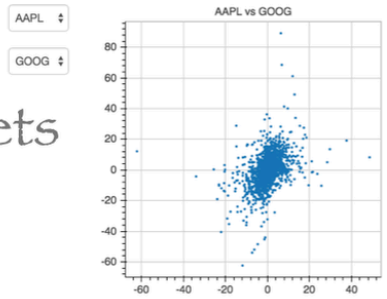

LDA 1
LDA 2
counts
class
Claudia Beleites, http://bit.ly/yNqlpz


- Data Visualization Project
- Interacive
- Web frontend
- example1
- example2

# Custom visualizations

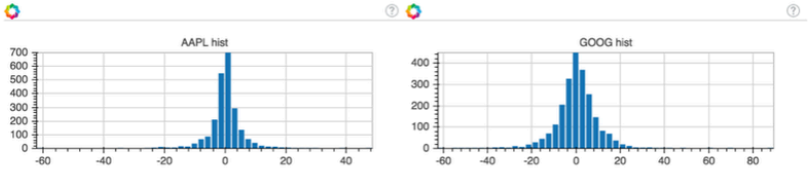


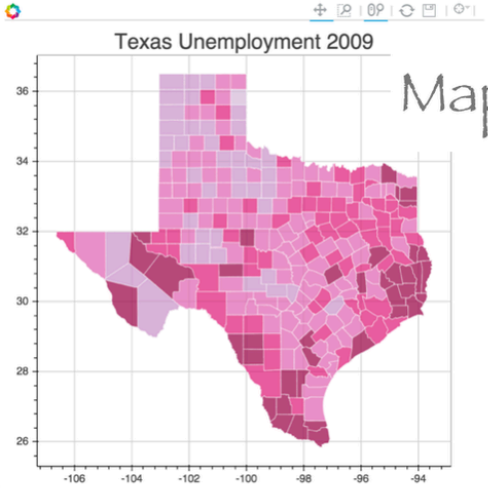

## Widgets




| | AAPL | AAPL_returns | GOOG | GOOG_returns |
|---|---|---|---|---|
| count | 2258.000000 | 2258.000000 | 2258.000000 | 2258.000000 |
| mean | 218.713048 | 0.194594 | 494.682700 | 0.346306 |
| std | 171.674172 | 5.239248 | 164.446873 | 9.411532 |
| min | 15.108500 | -62.237000 | 99.950000 | -62.400000 |
| 25% | 80.502825 | -1.631675 | 393.127500 | -4.095000 |
| 50% | 163.155500 | 0.146000 | 499.785000 | 0.240000 |
| 75% | 338.082500 | 2.155750 | 595.465000 | 4.957500 |
| max | 685.662000 | 48.618000 | 923.670000 | 89.310000 |

## Dashboards

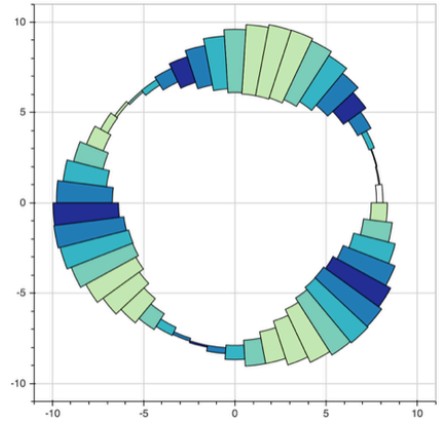


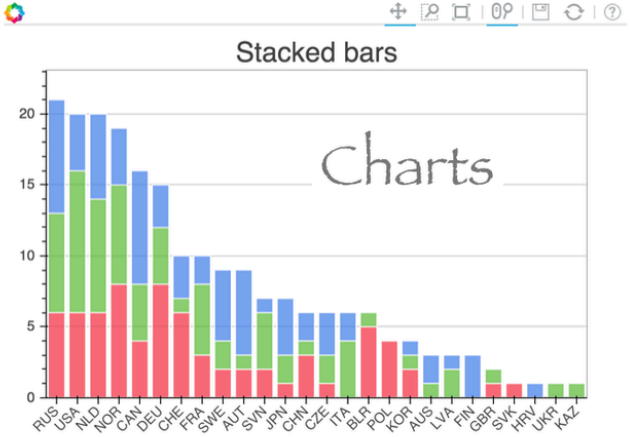




## Maps

## Bokeh

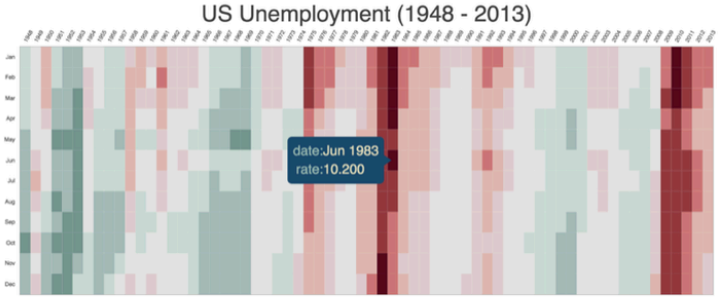
## Tools






## Charts

## Hover




## Streaming/ Animations

# Bokeh

- Python, R, Scala, Julia
- Interactive
- Jupyter Notebook

# Tableau

資料視覺化的商業軟體

# Tableau

- 商業化分析軟體 (有試用期)
- 不需要撰寫 code，人性化的操作介面
- 可處理多種 Data Source
- 圖表種類較少
- 慢
- 雲端社群，分享資料


PLOTDB
100+ Visualizations for Your Data
enough with basic charts? need full control of your viswork?


- 利用 Data Visulization 呈現故事
- 操作：選樣板，上傳資料，拖曳
- 可以改 Code
- 台灣人做的!


an open-source JavaScript library
for mobile-friendly interactive maps


+
−
A4200
OLD ST
Whitechapel
City of London
NEW RD
CABLE ST
London
River Thames
St. James's Park
LONG LN
Southwark Park
NEW KENT RD
MILLBANK
Leaflet | Map data © OpenStreetMap contributors, CC-BY-SA, Imagery © Mapbox

# Programming

```
var circle = L.circle([51.508, -0.11]
    color: 'red',
    fillColor: '#f03',
    fillOpacity: 0.5,
    radius: 500
}).addTo(mymap);

 var polygon = L.polygon([
     [51.509, -0.08],
     [51.503, -0.06],
     [51.51, -0.047]
 ]).addTo(mymap);
```

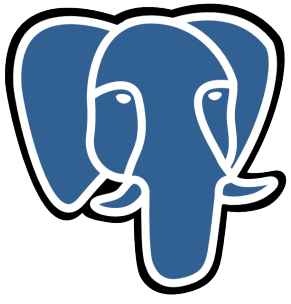
# Using GeoJSON with Leaflet

- 將資訊移出程式碼, Configurable
- 讓資料的操作結構化

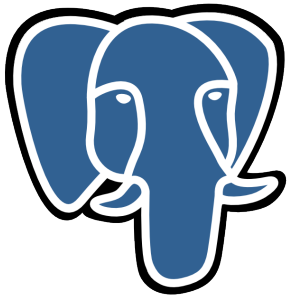
# Using GeoJSON with Leaflet

```
var myLines = [{
    "type": "LineString",
    "coordinates": [[-100, 40], [-105, 45], [-110, 55]]
}, {
    "type": "LineString",
    "coordinates": [[-105, 40], [-110, 45], [-115, 55]]
}];

var myStyle = {
    "color": "#ff7800",
    "weight": 5,
    "opacity": 0.65
};

L.geoJSON(myLines, {
    style: myStyle
}).addTo(map);
```


DATABASE
RELATIONAL
DBMS
SECURITY
DESIGNER
MIRRORING
RELATIONSHIP
LAYOUT
SERVER
PERFORMANSE
TYPE
AVAILABILITY
NETWORK
OPTIMIZATION
QUERY
OBJECT
SYSTEM
MIGRATION
ARCHITECTUR
ENTITY
ATA
STORA
CONT
BUSINE
DEVELOPER
DESIGN
TABLE
SQL
XML
KEY
SOFTWARE
LANGUAGE
COLLECTION
STRUCTURE
TRANSACTION
CONCEPT
RESTORE
ADMINISTRATION
REPLICATION
RECOVERY
BACKUP
DEVELOPMENT
INDEX

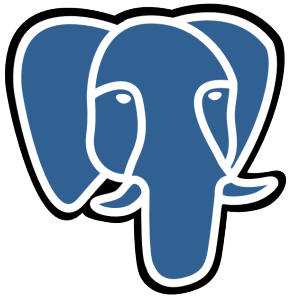


PostgreSQL



PostgreSQL

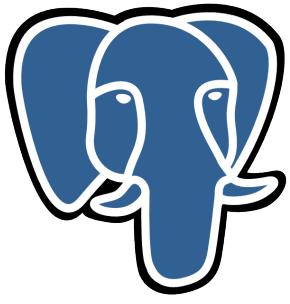




PostgreSQL



CouchDB

PostgreSQL



CouchDB





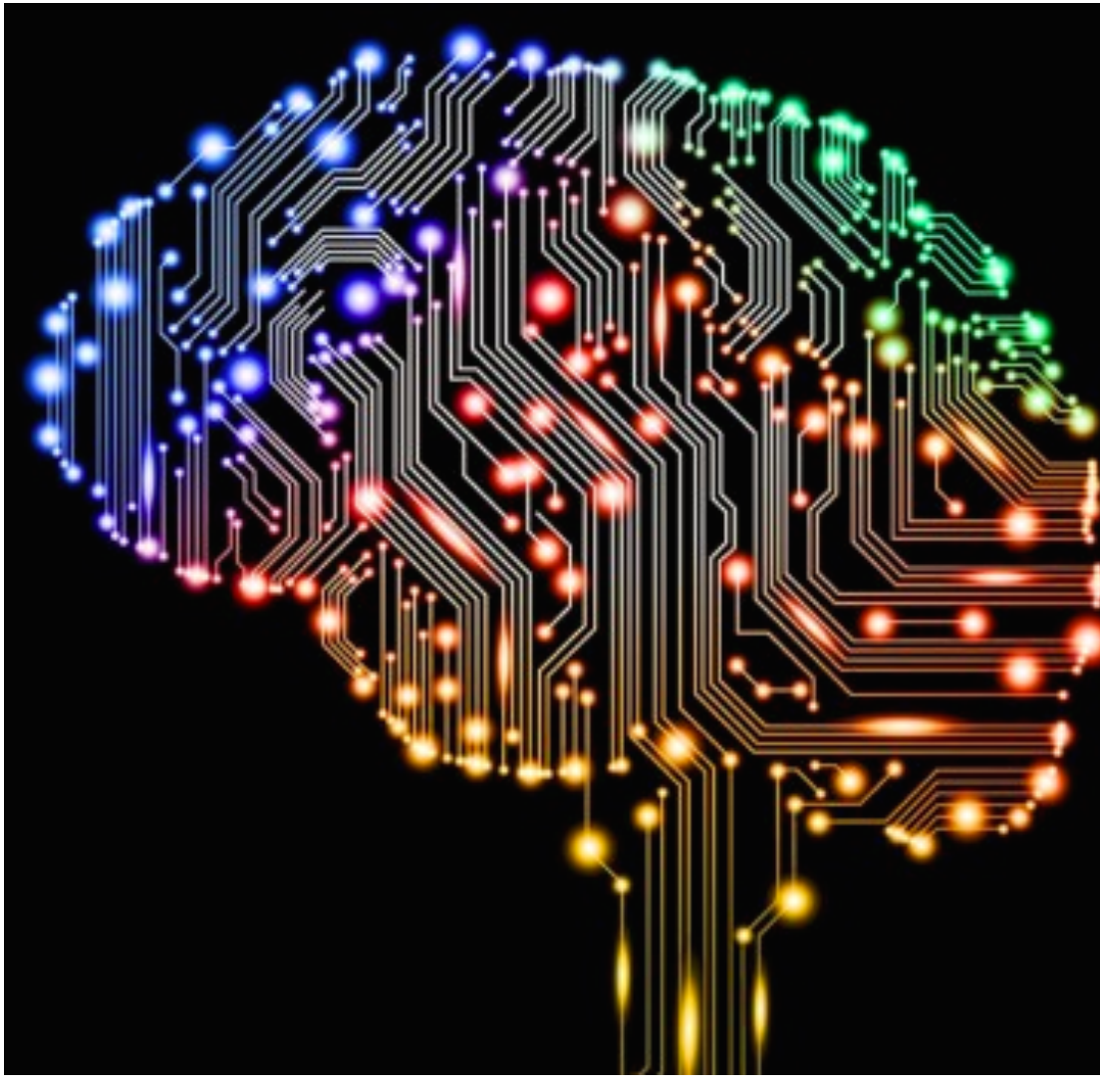
PostgreSQL
mongoDB
redis
neo4j
CouchDB
VOLTDB
MySQL
APACHE HBASE
cassandra


S3 Simple Storage Service


S3

1. Key-value
2. Permission
3. Data Visualization
4. Big Data (Spark)

# Algorithm & Modeling



# Algorithm & Modeling

- python-numpy + python-pandas + scikit-learn
- libsvm
- spark-Mlib
- Weka
- Deep Learning

Numpy + Pandas
+ Scikit-learn

# Numpy

- 能夠處理多維矩陣運算
- 逼近 C 的運算效能
- 提供線性代數常用運算式

# Numpy - data structure

ndarray (n-dim array)

- ndim
- size
- shape
- dtype

```
In [2]: import numpy as np

        matrix = np.array([
            [2, 4, 6, 8],
            [1, 3, 5, 7]
        ])

In [3]: matrix.ndim

Out[3]: 2

In [4]: matrix.size

Out[4]: 8

In [5]: matrix.shape

Out[5]: (2, 4)

In [6]: matrix.dtype

Out[6]: dtype('int64')
```

# Numpy
# generate matrix

```
In [2]: matrix = np.arange(10)

In [3]: matrix
Out[3]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [5]: matrix = np.arange(7, 20, 2)

In [6]: matrix
Out[6]: array([ 7,  9, 11, 13, 15, 17, 19])
```

# Numpy
# generate matrix

```
In [11]: matrix = np.zeros((2, 4))

In [12]: matrix
Out[12]:
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])
```

```
In [15]: matrix = np.ones((3, 2))

In [16]: matrix
Out[16]:
array([[ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]])

In [17]: matrix.dtype
Out[17]: dtype('float64')
```

# Numpy
# generate matrix

```
In [19]: matrix = np.random.random((3, 4))

In [20]: matrix
Out[20]:
array([[ 0.49229397,  0.55287717,  0.19734024,  0.00747744],
       [ 0.82855221,  0.41269592,  0.8615331 ,  0.2248113 ],
       [ 0.00545853,  0.79343231,  0.61711361,  0.59147531]])

In [21]: matrix * 5
Out[21]:
array([[ 2.46146986,  2.76438586,  0.98670119,  0.03738722],
       [ 4.14276106,  2.06347961,  4.3076655 ,  1.12405651],
       [ 0.02729267,  3.96716154,  3.08556806,  2.95737653]])

In [22]: matrix + 4
Out[22]:
array([[ 4.49229397,  4.55287717,  4.19734024,  4.00747744],
       [ 4.82855221,  4.41269592,  4.8615331 ,  4.2248113 ],
       [ 4.00545853,  4.79343231,  4.61711361,  4.59147531]])
```

# Numpy
# generate matrix

```
In [23]: matrix = np.arange(100).reshape(10, 10)

In [24]: matrix
Out[24]:
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24, 25, 26, 27, 28, 29],
       [30, 31, 32, 33, 34, 35, 36, 37, 38, 39],
       [40, 41, 42, 43, 44, 45, 46, 47, 48, 49],
       [50, 51, 52, 53, 54, 55, 56, 57, 58, 59],
       [60, 61, 62, 63, 64, 65, 66, 67, 68, 69],
       [70, 71, 72, 73, 74, 75, 76, 77, 78, 79],
       [80, 81, 82, 83, 84, 85, 86, 87, 88, 89],
       [90, 91, 92, 93, 94, 95, 96, 97, 98, 99]])
```

```
In [25]: matrix[0, 1]
Out[25]: 1

In [26]: matrix[0, 2:]
Out[26]: array([2, 3, 4, 5, 6, 7, 8, 9])

In [27]: matrix[0, 2:].shape
Out[27]: (8,)
```

# Numpy
# generate matrix

```
In [32]: matrix = np.arange(12).reshape(3, 4)

In [33]: matrix
Out[33]:
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

In [34]: index = matrix > 4

In [35]: index
Out[35]:
array([[False, False, False, False],
       [False,  True,  True,  True],
       [ True,  True,  True,  True]], dtype=bool)
```

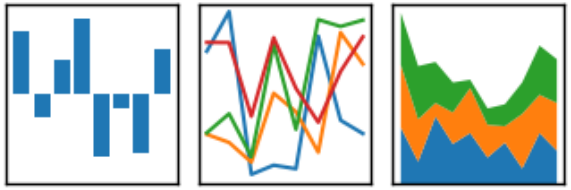
```
In [36]: matrix[index] = 0

In [37]: matrix
Out[37]:
array([[0, 1, 2, 3],
       [4, 0, 0, 0],
       [0, 0, 0, 0]])
```

# Numpy - linalg

```
DESCRIPTION
    Core Linear Algebra Tools
    -------------------------
    Linear algebra basics:

    - norm            Vector or matrix norm
    - inv             Inverse of a square matrix
    - solve           Solve a linear system of equations
    - det             Determinant of a square matrix
    - lstsq           Solve linear least-squares problem
    - pinv            Pseudo-inverse (Moore-Penrose) calculated using a singular
                      value decomposition
    - matrix_power    Integer power of a square matrix

    Eigenvalues and decompositions:

    - eig             Eigenvalues and vectors of a square matrix
    - eigh            Eigenvalues and eigenvectors of a Hermitian matrix
    - eigvals         Eigenvalues of a square matrix
    - eigvalsh        Eigenvalues of a Hermitian matrix
    - qr              QR decomposition of a matrix
    - svd             Singular value decomposition of a matrix
    - cholesky        Cholesky decomposition of a matrix

    Tensor operations:

    - tensorsolve     Solve a linear tensor equation
    - tensorinv       Calculate an inverse of a tensor
```

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



- 基於 numpy 發展而來
- 資料型態 Series, DataFrame
- 善於處理各種形式的資料: csv, json ...
- 缺值 nan 處理
- 數據合併

# Series - 一維資料

```
In [6]: series_data = pandas.Series([n * 10 for n in range(10)])

In [7]: series_data
Out[7]:
0     0
1    10
2    20
3    30
4    40
5    50
6    60
7    70
8    80
9    90
dtype: int64

In [8]: series_data.values
Out[8]: array([ 0, 10, 20, 30, 40, 50, 60, 70, 80, 90])
```

# Series - 一維資料

```
In [9]: series_data[0]
Out[9]: 0

In [10]: series_data[1]
Out[10]: 10

In [11]: series_data[1: 5]
Out[11]:
1    10
2    20
3    30
4    40
dtype: int64

In [12]: series_data[series_data > 70]
Out[12]:
8    80
9    90
dtype: int64
```

# Series - 一維資料

```
In [17]: series_data = pandas.Series([4, -7, -5, 3], index=['a', 'b', 'c', 'd'])

In [18]: series_data
Out[18]:
a    4
b   -7
c   -5
d    3
dtype: int64

In [19]: series_data['a']
Out[19]: 4

In [20]: series_data['a':'c']
Out[20]:
a    4
b   -7
c   -5
dtype: int64
```

# Series - 一維資料

```
In [21]: states_data = {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000, 'Utah': 5000}

In [22]: series_data = pandas.Series(states_data)

In [23]: series_data
Out[23]:
Ohio      35000
Oregon    16000
Texas     71000
Utah       5000
dtype: int64

In [24]: series_data.index
Out[24]: Index(['Ohio', 'Oregon', 'Texas', 'Utah'], dtype='object')
```

# DataFrame - 多維 Series

```
In [27]: data
Out[27]:
{'pop': [1.5, 1.7, 3.6, 2.4, 2.9],
 'state': ['Ohio', 'Ohio', 'Ohio', 'Nevada', 'Nevada'],
 'year': [2000, 2001, 2002, 2001, 2002]}

In [28]: frame_data = pandas.DataFrame(data)

In [29]: frame_data
Out[29]:
   pop   state  year
0  1.5    Ohio  2000
1  1.7    Ohio  2001
2  3.6    Ohio  2002
3  2.4  Nevada  2001
4  2.9  Nevada  2002
```

# Pandas - import

```
In [1]: import pandas

        tainan_air_quality_data = pandas.read_csv("./aircheck.csv")

In [2]: print (tainan_air_quality_data)
```

| | 測站名稱 | 測站編號 | 總懸浮微粒採樣開始時間 | 總懸浮微粒採樣結束時間 | 天候 | TSP | PM10 |
|---|---|---|---|---|---|---|---|
| 0 | 民德國中 | 3532910A0011 | 1040108.0 | 1040109.0 | 晴 | 133 | * | * |
| 1 | 民德國中 | 3532910A0011 | 1040113.0 | 1040114.0 | 陰 | * | 51 | 26 |
| 2 | 民德國中 | 3532910A0011 | 1040120.0 | 1040121.0 | 晴 | 162 | * | * |
| 3 | 協進國小 | 3533110A0010 | 1040108.0 | 1040109.0 | 晴 | 130 | * | * |
| 4 | 協進國小 | 3533110A0010 | 1040113.0 | 1040114.0 | 陰 | * | 52 | 31 |
| 5 | 協進國小 | 3533110A0010 | 1040120.0 | 1040121.0 | 晴 | 169 | * | * |
| 6 | 安平污水廠 | 3532810A0008 | 1040108.0 | 1040109.0 | 晴 | 133 | * | * |
| 7 | 安平污水廠 | 3532810A0008 | 1040113.0 | 1040114.0 | 陰 | * | 52 | 21 |
| 8 | 安平污水廠 | 3532810A0008 | 1040120.0 | 1040121.0 | 晴 | 163 | * | * |
| 9 | 仁德區公所 | 3521010A0011 | 1040108.0 | 1040109.0 | 晴 | 138 | * | * |
| 10 | 仁德區公所 | 3521010A0011 | 1040113.0 | 1040114.0 | 陰 | * | 55 | 30 |
| 11 | 仁德區公所 | 3521010A0011 | 1040120.0 | 1040121.0 | 晴 | 175 | * | * |
| 12 | 永康區衛生所 | 3521410A0016 | 1040105.0 | 1040106.0 | 晴 | 127 | * | * |
| 13 | 永康區衛生所 | 3521410A0016 | 1040113.0 | 1040114.0 | 陰 | * | 42 | 22 |
| 14 | 永康區衛生所 | 3521410A0016 | 1040126.0 | 1040127.0 | 陰 | 112 | * | * |
| 15 | 新化區公所 | 3518910A0005 | 1040105.0 | 1040106.0 | 晴 | 96 | * | * |
| 16 | 新化區公所 | 3518910A0005 | 1040113.0 | 1040114.0 | 陰 | * | 39 | 26 |
| 17 | 新化區公所 | 3518910A0005 | 1040126.0 | 1040127.0 | 陰 | 110 | * | * |

# Pandas - import

```
In [2]: print (tainan_air_quality_data)
```

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 23 | 西港區衛生所 | 3519910A0008 | 1040126.0 | 1040127.0 | 陰 | 107 | * | * | |
| 24 | 麻豆國小 | 3518710A0013 | 1040106.0 | 1040107.0 | 晴 | 120 | * | * | |
| 25 | 麻豆國小 | 3518710A0013 | 1040115.0 | 1040116.0 | 晴 | * | 53 | 10 | |
| 26 | 麻豆國小 | 3518710A0013 | 1040126.0 | 1040127.0 | 陰 | 103 | * | * | |
| 27 | 學甲區公所 | 3519110A0012 | 1040106.0 | 1040107.0 | 晴 | 119 | * | * | |
| 28 | 學甲區公所 | 3519110A0012 | 1040115.0 | 1040116.0 | 晴 | * | 63 | 17 | |
| 29 | 學甲區公所 | 3519110A0012 | 1040126.0 | 1040127.0 | 陰 | 99 | * | * | |
| .. | ... | ... | ... | ... | ... | ... | ... | ... | |
| 40 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 41 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 42 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 43 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 44 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 45 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 46 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 47 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 48 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 49 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 50 | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

# Pandas - import

```
In [3]: tainan_air_quality_data.dropna(how='all')
```

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 國小 | | | | | | | | | | | | | | | | |
| **31** | 新營國小 | 3518410A0009 | 1040115.0 | 1040116.0 | 晴 | * | 66 | 20 | * | * | * | * | * | * | * | NaN | NaN |
| **32** | 新營國小 | 3518410A0009 | 1040126.0 | 1040127.0 | 陰 | 105 | * | * | 2.98 | 9.51 | 18.86 | 0.043 | * | * | * | NaN | NaN |
| **33** | 白河區衛生所 | 3518610A0017 | 1040106.0 | 1040107.0 | 晴 | 114 | * | * | 2.23 | 7.05 | 14.16 | 0.056 | 1031226 | 1040126 | 4.49 | NaN | NaN |
| **34** | 白河區衛生所 | 3518610A0017 | 1040115.0 | 1040116.0 | 晴 | * | 79 | 20 | * | * | * | * | * | * | * | NaN | NaN |
| **35** | 白河區衛生所 | 3518610A0017 | 1040126.0 | 1040127.0 | 陰 | 115 | * | * | 2.94 | 8.56 | 17.19 | 0.051 | * | * | * | NaN | NaN |
| | ... | | | | | | | | | | | | | | | | |

# Pandas - import

- read_csv
- read_excel
- read_hdf
- read_sql
- read_json
- read_msgpack (experimental)
- read_html
- read_gbq (experimental)
- read_stata
- read_sas
- read_clipboard
- read_pickle

# Pandas - NaN

```
In [36]: df2
Out[36]:
        one       two     three four   five  timestamp
a       NaN -0.282863 -1.509059  bar   True        NaT
c       NaN  1.212112 -0.173215  bar  False        NaT
e  0.119209 -1.044236 -0.861849  bar   True 2012-01-01
f -2.104569 -0.494929  1.071804  bar  False 2012-01-01
h       NaN -0.706771 -1.039575  bar   True        NaT

In [37]: df2.fillna(0)
Out[37]:
        one       two     three four   five  timestamp
a  0.000000 -0.282863 -1.509059  bar   True 1970-01-01
c  0.000000  1.212112 -0.173215  bar  False 1970-01-01
e  0.119209 -1.044236 -0.861849  bar   True 2012-01-01
f -2.104569 -0.494929  1.071804  bar  False 2012-01-01
h  0.000000 -0.706771 -1.039575  bar   True 1970-01-01
```

# Pandas - NaN

```
In [47]: dff
Out[47]:
          A         B         C
0  0.271860 -0.424972  0.567020
1  0.276232 -1.087401 -0.673690
2  0.113648 -1.478427  0.524988
3       NaN  0.577046 -1.715002
4       NaN       NaN -1.157892
5 -1.344312       NaN       NaN
6 -0.109050  1.643563       NaN
7  0.357021 -0.674600       NaN
8 -0.968914 -1.294524  0.413738
9  0.276662 -0.472035 -0.013960
```

```
In [48]: dff.fillna(dff.mean())
Out[48]:
          A         B         C
0  0.271860 -0.424972  0.567020
1  0.276232 -1.087401 -0.673690
2  0.113648 -1.478427  0.524988
3 -0.140857  0.577046 -1.715002
4 -0.140857 -0.401419 -1.157892
5 -1.344312 -0.401419 -0.293543
6 -0.109050  1.643563 -0.293543
7  0.357021 -0.674600 -0.293543
8 -0.968914 -1.294524  0.413738
9  0.276662 -0.472035 -0.013960
```

# Pandas - NaN

```
In [47]: dff
Out[47]:
          A         B         C
0  0.271860 -0.424972  0.567020
1  0.276232 -1.087401 -0.673690
2  0.113648 -1.478427  0.524988
3       NaN  0.577046 -1.715002
4       NaN       NaN -1.157892
5 -1.344312       NaN       NaN
6 -0.109050  1.643563       NaN
7  0.357021 -0.674600       NaN
8 -0.968914 -1.294524  0.413738
9  0.276662 -0.472035 -0.013960
```

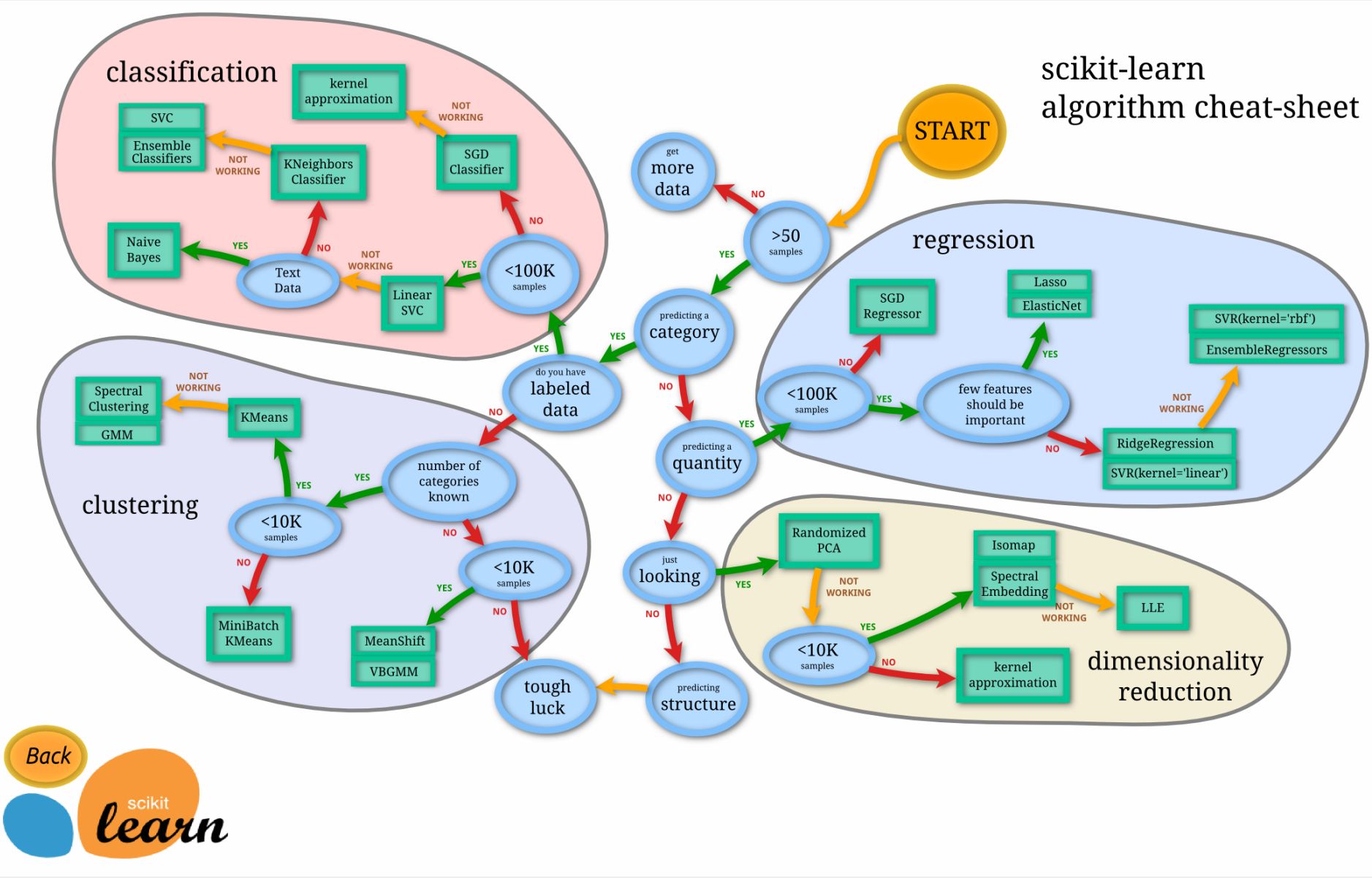
```
In [49]: dff.fillna(dff.mean()['B':'C'])
Out[49]:
          A         B         C
0  0.271860 -0.424972  0.567020
1  0.276232 -1.087401 -0.673690
2  0.113648 -1.478427  0.524988
3       NaN  0.577046 -1.715002
4       NaN -0.401419 -1.157892
5 -1.344312 -0.401419 -0.293543
6 -0.109050  1.643563 -0.293543
7  0.357021 -0.674600 -0.293543
8 -0.968914 -1.294524  0.413738
9  0.276662 -0.472035 -0.013960
```

# Pandas - operation

- Merge
- Grouping
- Reshaping
- . . .


machine learning in Python


- Dataset
- Feature Engineering
- Modeling
- Evaluation

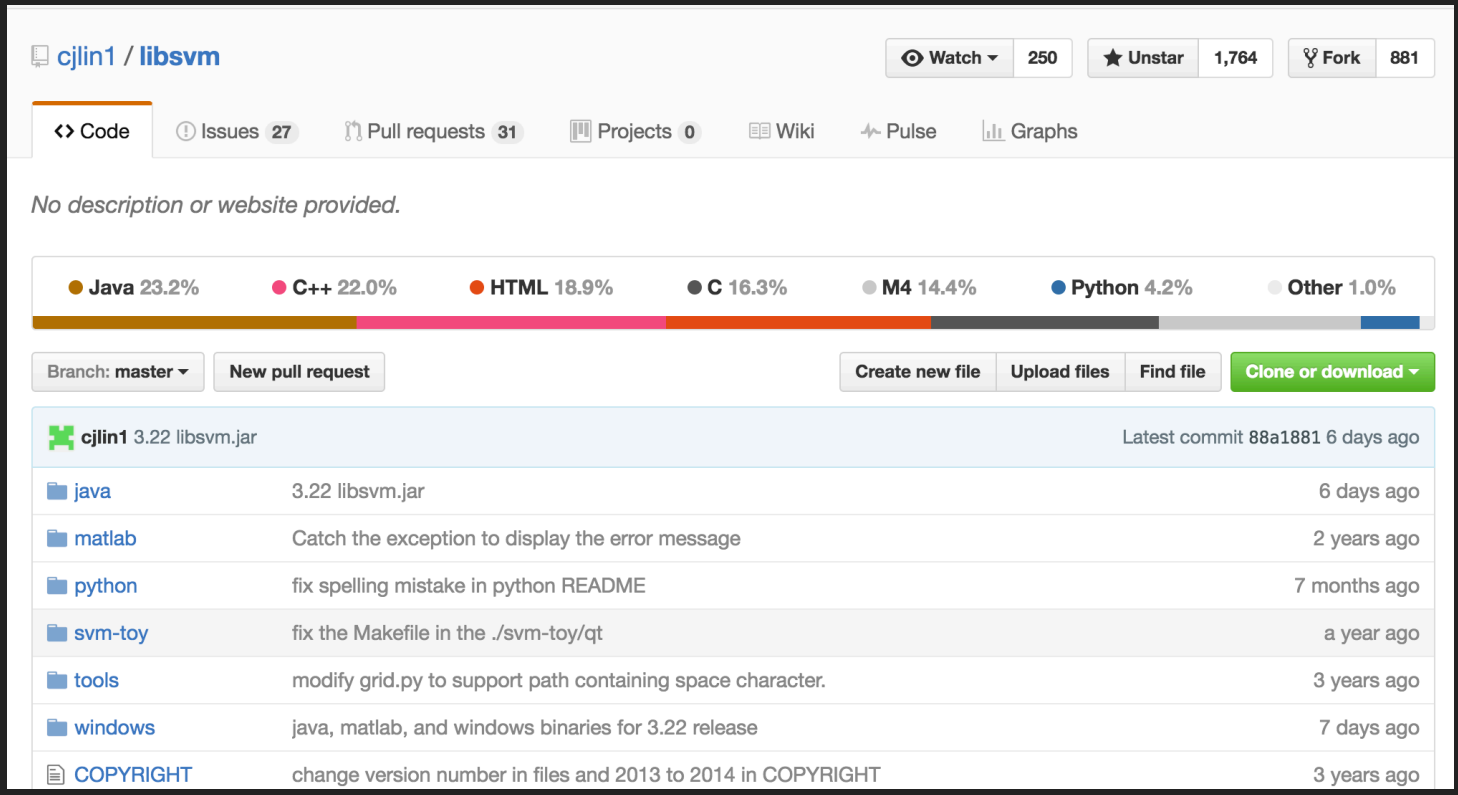

scikit-learn
algorithm cheat-sheet
START
get more data
NO
>50 samples
YES
predicting a category
YES
NO
do you have labeled data
YES
NO
predicting a quantity
YES
NO
just looking
YES
NO
predicting structure
tough luck
classification
kernel approximation
NOT WORKING
SVC
Ensemble Classifiers
NOT WORKING
KNeighbors Classifier
SGD Classifier
NO
Naive Bayes
YES
Text Data
NO
NOT WORKING
Linear SVC
YES
<100K samples
YES
NO
regression
SGD Regressor
NO
Lasso
ElasticNet
YES
<100K samples
YES
few features should be important
NO
NOT WORKING
SVR(kernel='rbf')
EnsembleRegressors
RidgeRegression
SVR(kernel='linear')
clustering
Spectral Clustering
GMM
NOT WORKING
KMeans
YES
number of categories known
YES
NO
<10K samples
NO
MiniBatch KMeans
<10K samples
YES
NO
MeanShift
VBGMM
dimensionality reduction
Randomized PCA
YES
NOT WORKING
Isomap
Spectral Embedding
NOT WORKING
LLE
<10K samples
YES
NO
kernel approximation
Back
scikit learn

# LIBSVM

- C
- Support many programming languages
- Easy to use
- Dataset

# LIBSVM - install

$ git clone

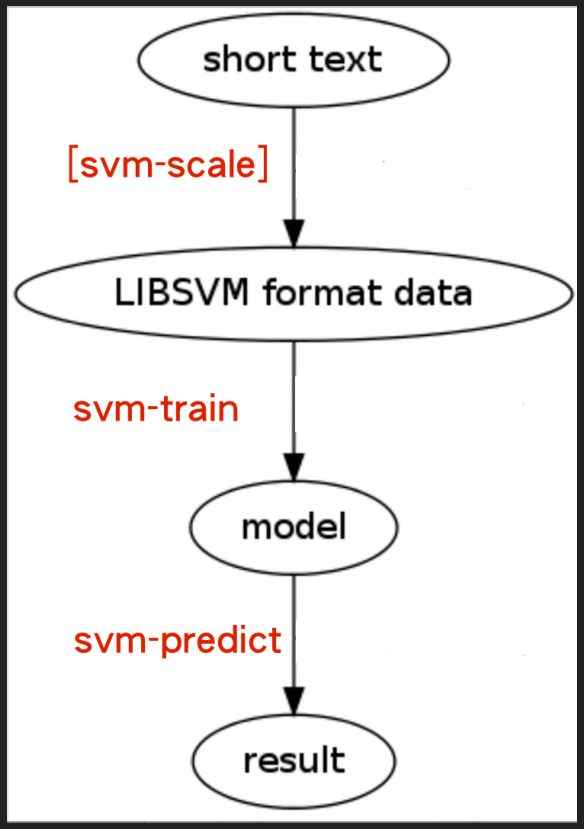


# LIBSVM - install

$ make

```
JIElitedeMacBook-Pro:libsvm-3.22 jielite$ ls
COPYRIGHT         heart_scale.model svm-scale         svm.def
FAQ.html          java              svm-scale.c       svm.h
Makefile          matlab            svm-toy           svm.o
Makefile.win      python            svm-train         tools
README            svm-predict       svm-train.c       windows
heart_scale       svm-predict.c     svm.cpp
JIElitedeMacBook-Pro:libsvm-3.22 jielite$
```

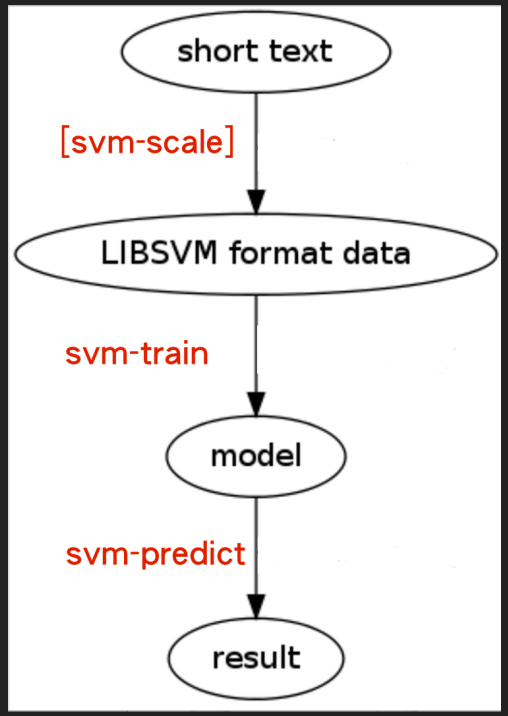
# LIBSVM - workflow



# LIBSVM - data format

```
[label] [index1]:[value1] [index2]:[value2] ...
[label] [index1]:[value1] [index2]:[value2] ...
```

- label 資料的分類
- index 資料欄位, attribute 順序
- value 資料數值, attribute 的數值



# LIBSVM - data format

```
+1 1:0.708333 2:1 3:1 4:-0.320755 5:-0.105023 6:-1 7:1 8:-0.419847 9:-1 1
-1 1:0.583333 2:-1 3:0.333333 4:-0.603774 5:1 6:-1 7:1 8:0.358779 9:-1 10
+1 1:0.166667 2:1 3:-0.333333 4:-0.433962 5:-0.383562 6:-1 7:-1 8:0.06870
-1 1:0.458333 2:1 3:1 4:-0.358491 5:-0.374429 6:-1 7:-1 8:-0.480916 9:1 1
-1 1:0.875 2:-1 3:-0.333333 4:-0.509434 5:-0.347032 6:-1 7:1 8:-0.236641
-1 1:0.5 2:1 3:1 4:-0.509434 5:-0.767123 6:-1 7:-1 8:0.0534351 9:-1 10:-0
+1 1:0.125 2:1 3:0.333333 4:-0.320755 5:-0.406393 6:1 7:1 8:0.0839695 9:1
+1 1:0.25 2:1 3:1 4:-0.698113 5:-0.484018 6:-1 7:1 8:0.0839695 9:1 10:-0.
+1 1:0.291667 2:1 3:1 4:-0.132075 5:-0.237443 6:-1 7:1 8:0.51145 9:-1 10:
+1 1:0.416667 2:-1 3:1 4:0.0566038 5:0.283105 6:-1 7:1 8:0.267176 9:-1 10
-1 1:0.25 2:1 3:1 4:-0.226415 5:-0.506849 6:-1 7:-1 8:0.374046 9:-1 10:-0
-1 2:1 3:1 4:-0.0943396 5:-0.543379 6:-1 7:1 8:-0.389313 9:1 10:-1 11:-1
-1 1:-0.375 2:1 3:0.333333 4:-0.132075 5:-0.502283 6:-1 7:1 8:0.664122 9:
```

# LIBSVM - toy


APACHE
Spark™
MLlib


| Tool | 2016 %Share | 2015 %share | % change |
|---|---|---|---|
| Hadoop | 22.1% | 18.4% | +20.5% |
| Spark | 21.6% | 11.3% | +91% |
| Hive | 12.4% | 10.2% | +21.3% |
| MLlib | 11.6% | 3.3% | +253% |
| SQL on Hadoop tools | 7.3% | 7.2% | +1.6% |
| H2O | 6.7% | 2.0% | +234% |
| HBase | 5.5% | 4.6% | +18.6% |
| Apache Pig | 4.6% | 5.4% | -16.1% |
| Apache Mahout | 2.6% | 2.8% | -7.2% |
| Dato | 2.4% | 0.5% | +338% |
| Datameer | 0.4% | 0.9% | -52.3% |
| Other Hadoop/HDFS-based tools | 4.9% | 4.5% | +7.5% |


- 分散式學習架構, Hadoop
- 支援 Java, Scala, R, Python
- 提供完善的工具鏈
- 高速的運算效能


```
# Every record of this DataFrame contains the label and
# features represented by a vector.
df = sqlContext.createDataFrame(data, ["label", "features"])

# Set parameters for the algorithm.
# Here, we limit the number of iterations to 10.
lr = LogisticRegression(maxIter=10)

# Fit the model to the data.
model = lr.fit(df)

# Given a dataset, predict each point's label, and show the results.
model.transform(df).show()
```


- 分散式學習架構, Hadoop
- 支援 Java, Scala, R, Python
- 提供完善的工具鏈
- 高速的運算效能


- Classification: logistic regression, naive Bayes,...
- Regression: generalized linear regression, survival regression,...
- Decision trees, random forests, and gradient-boosted trees
- Recommendation: alternating least squares (ALS)
- Clustering: K-means, Gaussian mixtures (GMMs),...
- Topic modeling: latent Dirichlet allocation (LDA)
- Frequent itemsets, association rules, and sequential pattern mining

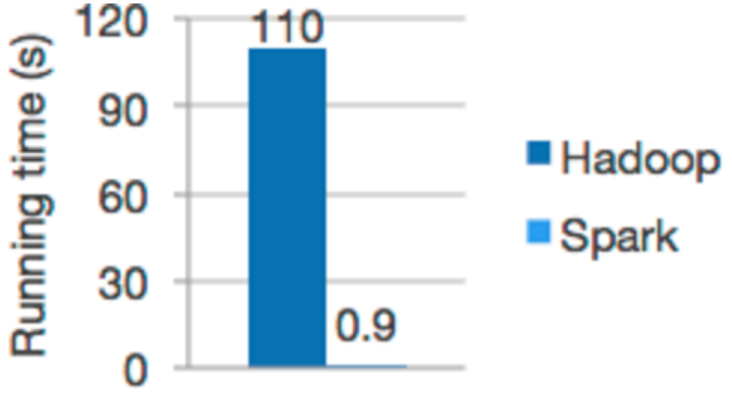

- Feature transformations: standardization, normalization, hashing,...
- ML Pipeline construction
- Model evaluation and hyper-parameter tuning
- ML persistence: saving and loading models and Pipelines


- 分散式學習架構, Hadoop
- 支援 Java, Scala, R, Python
- 提供完善的工具鏈
- 高速的運算效能


## Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Apache Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.

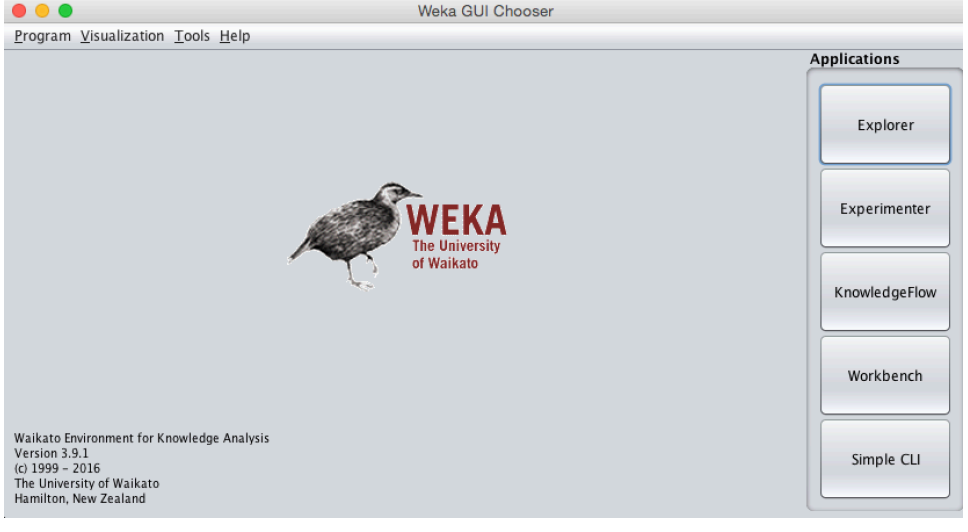



Logistic regression in Hadoop and Spark


APACHE
Spark™
MLlib


# Weka

- Java library
- Big Data
- Support GUI



# Deep Learning

- Theano
- Pylearn2
- Keras
- Tensorflow
- Caffe
- Deeplearning4J
- ...

# Theano

A CPU and GPU Math Compiler in

Python

- Base on Numpy
- Implemented by Cython
- Dynamic C code generation
- GPU & CUDA
- tensor, math expression

Theano tutorial:
http://www.slideshare.net/SergiiGavrylov/theano-tutorial

# Keras

High-level neural networks library

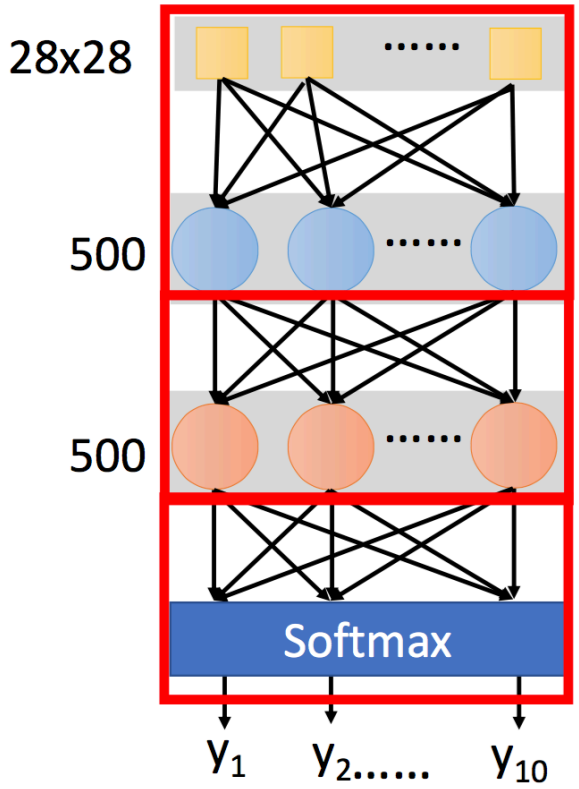
- 底層使用 Theano, Tensorflow
- Support GPU
- 簡單且快速的製作出 prototype

# Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function




```
model = Sequential()
```

```
model.add( Dense( input_dim=28*28,
                  output_dim=500 ))
model.add( Activation('sigmoid') )
```

```
model.add( Dense( output_dim=500 ) )
model.add( Activation('sigmoid') )
```

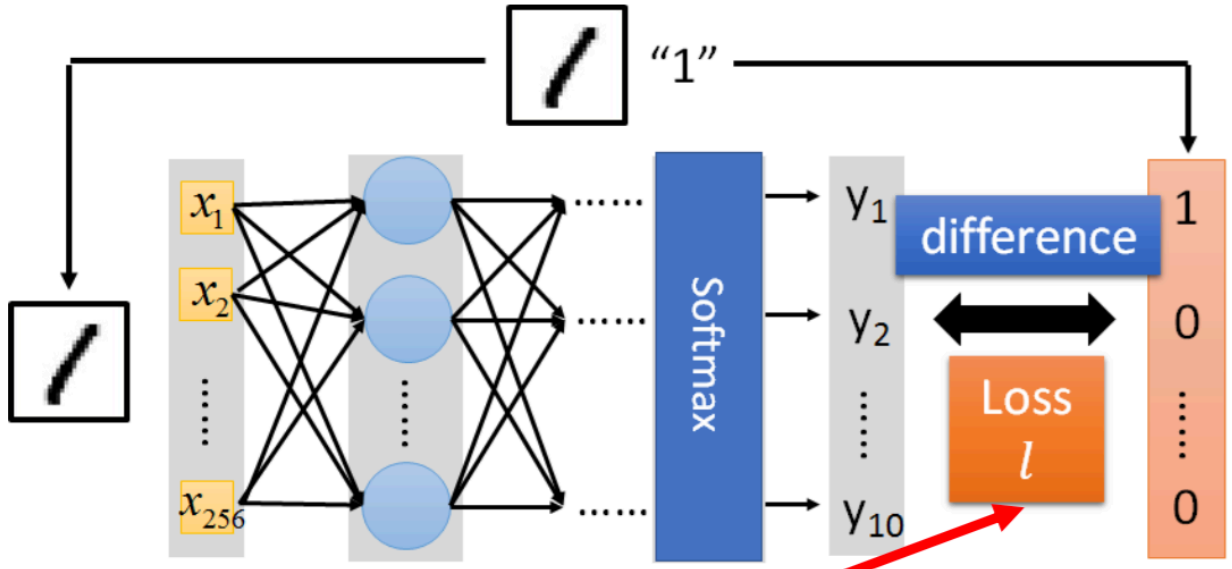
```
model.add( Dense(output_dim=10 ) )
model.add( Activation('softmax') )
```

# Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function




```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

# Keras


Step 3.1: Configuration

```
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

$$w \leftarrow w - \eta \partial L / \partial w$$

0.1

Step 3.2: Find the optimal network parameters

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

Training data (Images)

Labels (digits)

Next lecture

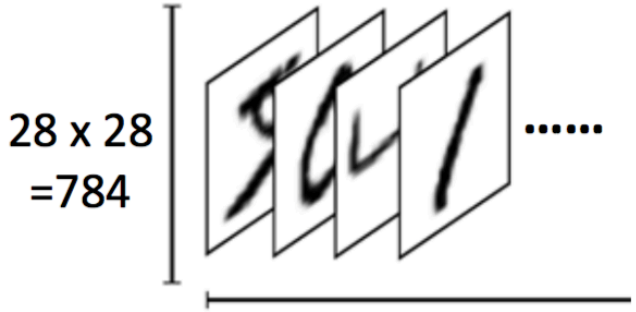
# Keras

Step 1: define a set of function → Step 2: goodness of function → Step 3: pick the best function

Step 3.2: Find the optimal network parameters

```
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

numpy array



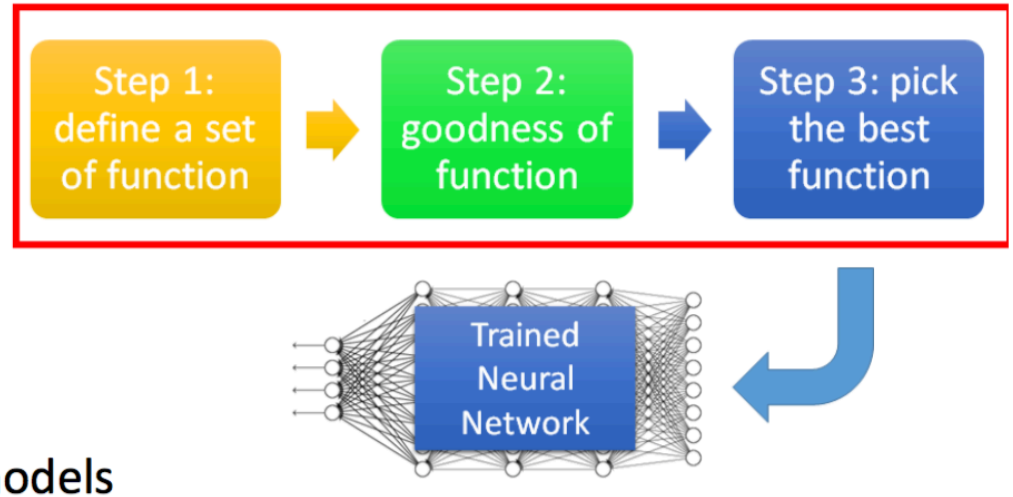

Number of training examples

numpy array


Number of training examples

https://www.tensorflow.org/versions/r0.8/tutorials/mnist/beginners/index.html

# Keras




Save and load models

http://keras.io/getting-started/faq/#how-can-i-save-a-keras-model

How to use the neural network (testing):

case 1:

```
score = model.evaluate(x_test,y_test)
print('Total loss on Testing Set:', score[0])
print('Accuracy of Testing Set:', score[1])
```

case 2:

```
result = model.predict(x_test)
```

# 去哪裡尋找適合的 Tool？


kaggle


KDnuggets


twitter

# Homework

- 在 Github 建立一個 repo 分享你用過 Data science 的相關工具
- Database, Social Network Analytics, ML library, Deep Learning Platform ...
- 上課提過的也沒關係
- 作業內容需包含
  - READM.md: 這個 Repo 的指引
  - Demo Code
  - 工具的說明文件
- 怎麼繳交? Google 表單填寫繳交資料
  https://goo.gl/forms/PQPz8u2glyunQvfM2

作業疑問 email: ita3051@gmail.com 陳杰翰