Chapter 9

社群資料分析

/ 胡筱薇

現代幾乎人人都有自己喜愛的社群媒體，社群媒體中累積的大量資料，是許多資料科學家熱衷分析的標的物之一。本章將帶領讀者從無到有完整地體驗 Facebook 社群資料的分析流程，搭配資料分析演算法和相關套件（如 seaborn、jieba 和 sklearn），進行資料分析與視覺化。本章共有五個小節，包括：環境安裝、認識 Facebook API、EDA 探索式資料分析、文字探勘和資料建模。學習完本章後，讀者將學會抓取 Facebook 社群資料並以演算法分析、以圖表呈現分析結果、以及運用線性迴歸模型預測。

9-1 環境安裝：Python 程式語言

經過前面章節的練習，大家應該對 Python 有相當認識，也完成不少程式練習吧！！甚麼，你覺得前面使用 IDLE 的介面寫程式很麻煩？安裝 numpy 或 pandas 等套件實在很頭昏？既然大家都花不少力氣終於入門 Python，那我們就來介紹進階的工具：Anaconda，協助大家開發複雜的程式工具，也能夠玩玩 Python 提供的各種神奇功能。

本章將以 Python 的進階應用為介紹主題，基本上是互相獨立的單元，大家可以照順序慢慢閱讀慢慢練習，也可以挑自己有興趣的主題跳著嘗試，萬一碰到有連續性的程式範例而看不懂時，再回頭慢慢研讀整個主題就可以，總之就是依照自己喜歡的步調練習更深入的 Python 程式吧。

Anaconda

首先，我們需要先安裝 Anaconda 軟體來啟動 Python 環境！ Anaconda 為一款集合許多工具的軟件包，同時也包含了不同的編譯器 (Jupyter notebook、Spyder、RStudio 等)，編譯器為撰寫程式語言的地方，而我們在以下將會選用 Jupyter notebook 編譯器作為撰寫 Python 的環境。

官方下載網址：https://www.anaconda.com/download/

圖 9-1 Anaconda Navigator

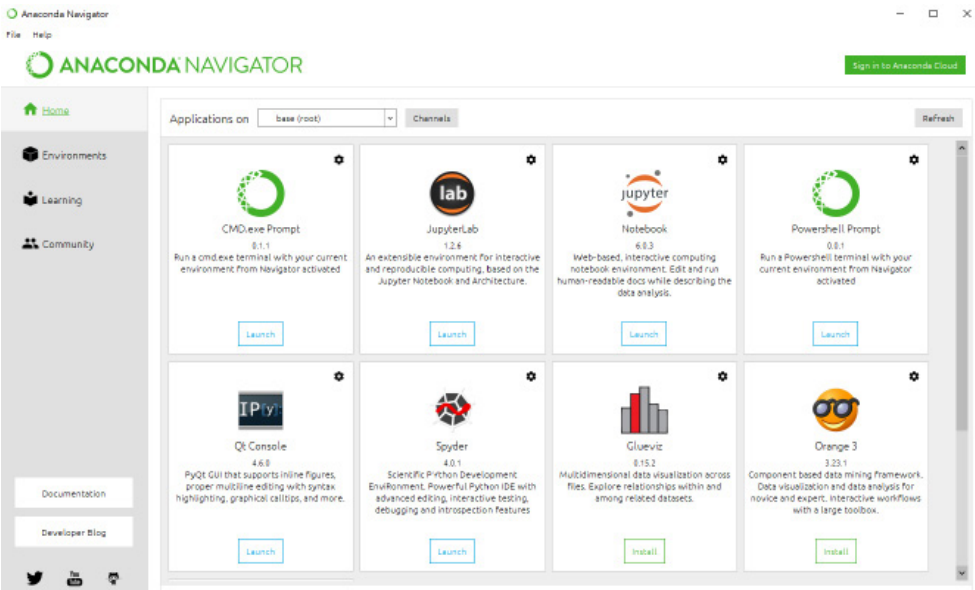


圖 9-2 Anaconda Navigator 開啟後畫面

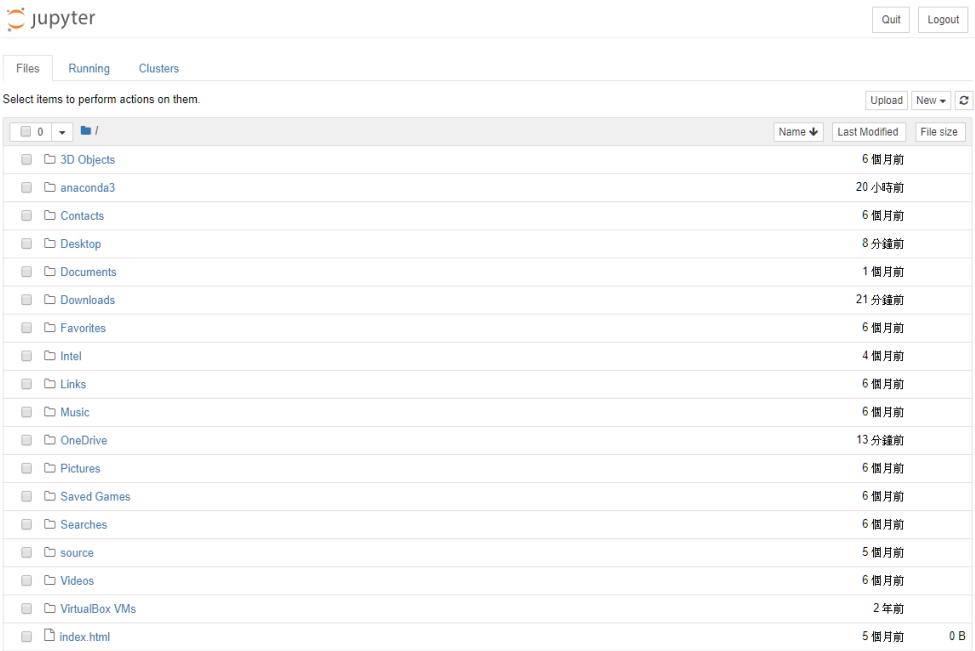


圖 9-3 Jupyter Notebook

安裝完成後點選 Anaconda-Navigator 開啟軟體 (圖 9-1)，接著在界面中找尋 Jupyter Notebook 點擊下方 Launch(圖 9-2)，最後在網頁瀏覽器中跳出此畫面 (圖 9-3) 即代表安裝成功，下一小節將一一介紹如何使用 Jupyter notebook 撰寫 Python 程式。

Jupyter Notebook

新增一個 Notebook

圖 9-4 新增一個 Notebook

Notebook 也就是撰寫 Python 的檔案，而畫面 (圖 9-3) 中的所有資料夾或檔案是真實存在於此台電腦中的資料，因此我們可以自由選擇今天寫的程式要存放於哪個資料夾當中。而新增一個 Notebook 的方式為點擊右上角的 New 選單，接著點選 Python 3 即可 (圖 9-4)。

重新命名

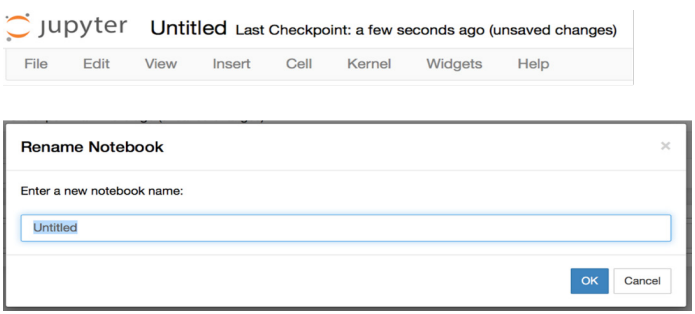


圖 9-5 重新命名

成功新增一個 Notebook 檔案後記得先幫檔案重新命名，方便日後找尋。重新命名的方式可以直接從電腦中的檔案位置按右鍵重新命名，也可以在 Notebook 當中點選最上方的 Untitled 字眼進行更改 (圖 9-5)。

撰寫程式 & 程式執行

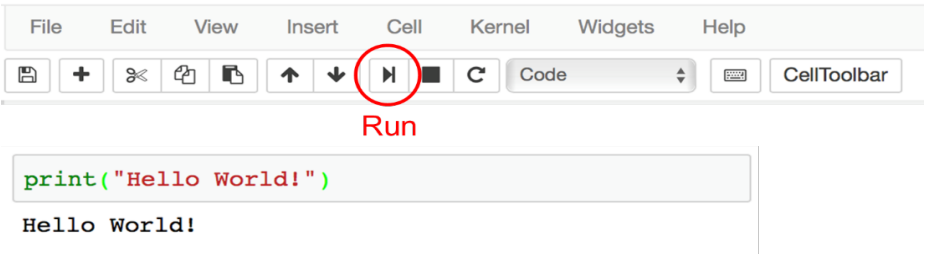


圖 9-6

接著我們在 Notebook 中間會看到有一個方框的區塊，此區塊即用來撰寫 Python 程式的地方，我們可以在此位置試著寫入 print(“Hello World!”) 字眼。撰寫完畢後會發現什麼事都還沒發生，這是因為我們尚未給予執行的指令，執行程式的方式共有三種，其一為點擊 (圖 9-6) 中紅色圈選的區塊，或是可以使用兩種快捷鍵 (Ctrl + Enter or Shift + Enter) 也可以成功執行，執行完成後會看到方框底下成功打印出 Hello World! 的字眼！

Cell 介紹

```
In [2]: # 這裡是第2個cell
        print("Hello Jurassic Park!")

        Hello Jurassic Park!
```

圖 9-7 Cell 介紹

剛剛我們寫入程式的方框又可稱之為 Cell，在 Notebook 檔案中我們可以自由新增很多個 Cell 來撰寫不同的程式 (點選上排 + 號即可)，而每一個 Cell 會獨立儲存各自執行的結果，這也是 Jupyter notebook 獨特的功能之一。我們可以試著新增第二個 Cell 執行 (圖 9-7) 當中的內容。

註解介紹

```
# 這是註解
# 我的第一支程式
print("Hello World!")
```

```
Hello World!
```

圖 9-8 註解測試

接著來介紹註解的功能，註解的特性為並非是程式語言，因此不論撰寫的內容為何皆不會被執行。而註解的目的在於方便使用者記錄此段內容為何，或是供他人閱讀時不用花太多時間即可快速掌握我們撰寫程式的內容，也就是寫筆記的概念。而註解撰寫的方式如 (圖 9-8)，在撰寫的內容前面加上 # 字號，可以發現字體顏色統一改變為一致的顏色，同時執行後可以看到下方只打印出 Hello World! 的字眼而沒有把註解的內容也打印出來。

Markdown 介紹

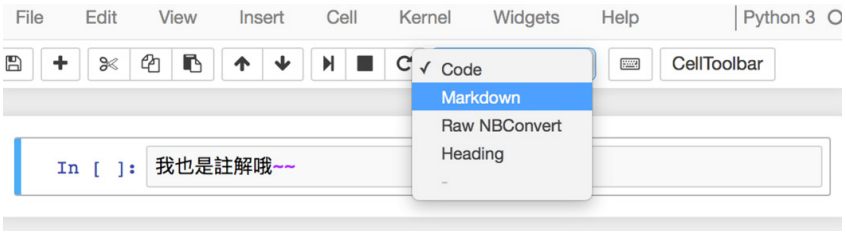


圖 9-9 Markdown

圖 9-10 Markdown

在這邊我們再來介紹另外兩種的註解方式，同時也是 Jupyter notebook 的獨有功能之一，也就是 Markdown 與 Heading。兩者的特色為不需要在撰寫程式的 Cell 中即可完成註解的撰寫，整體排版更加簡潔，類似於 Word 文件的排版格式。使用方式為 (圖 9-9) 中點擊上方 Code 的下拉式選單 (圖 9-11)，改選為 Markdown 或 Heading，接著於 Cell 當中打印想要撰寫的文字內容在執行，即可成功 (圖 9-10 & 9-12)。

Heading 介紹

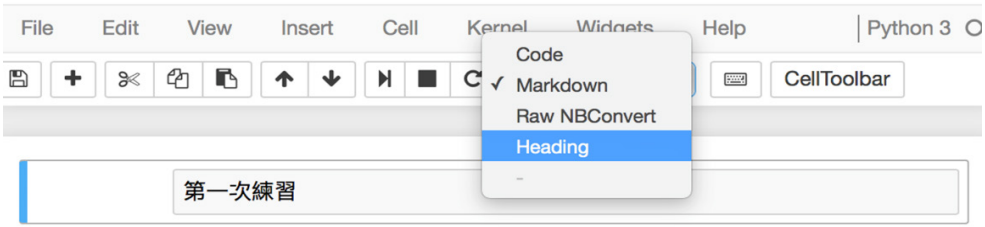


圖 9-11 Heading

圖 9-12 Heading

儲存檔案

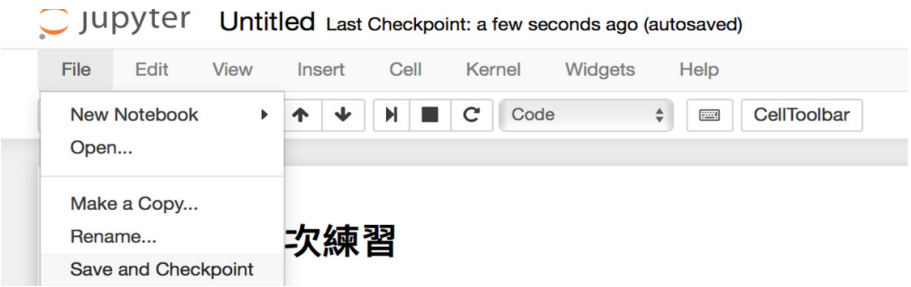


圖 9-13 儲存檔案

最後來教學如何儲存檔案，可以點選左上方 Flie 下拉式選單當中的 Save and Checkpoint(圖 9-13)，或是使用快捷鍵 (Ctrl + s) 即可成功儲存檔案。在這邊建議各位讀者要有一邊撰寫程式一邊儲存檔案的好習慣，減少全部都撰寫完成後才儲存檔案，以免中間 Anaconda 或是電腦當機，而造成撰寫的內容都消失的情況發生。

檔案位置

```
In [3]: import os
        os.getcwd()

Out[3]: '/Users/andy/Documents/python'
```

圖 9-14

另外如果真的找尋不到 Notebook 在電腦中的存放位置時，可以執行圖 9-14 的程式碼，此段程式會告訴我們 Notebook 的電腦存放路徑。

下載 Python 第三方套件

我們在前面有提到之所以選用 Python 有一原因為其具有豐富的函式庫套件，而這些額外的函式庫套件可以協助我們進行網路爬蟲、資料分析、機器學習、人工智慧、網站架設、遊戲開發等等的實務應用，接下來將會教學如何在 OS X 以及 Windows 的環境中下載其他的函式庫。

OS X 環境 (Mac 用戶)

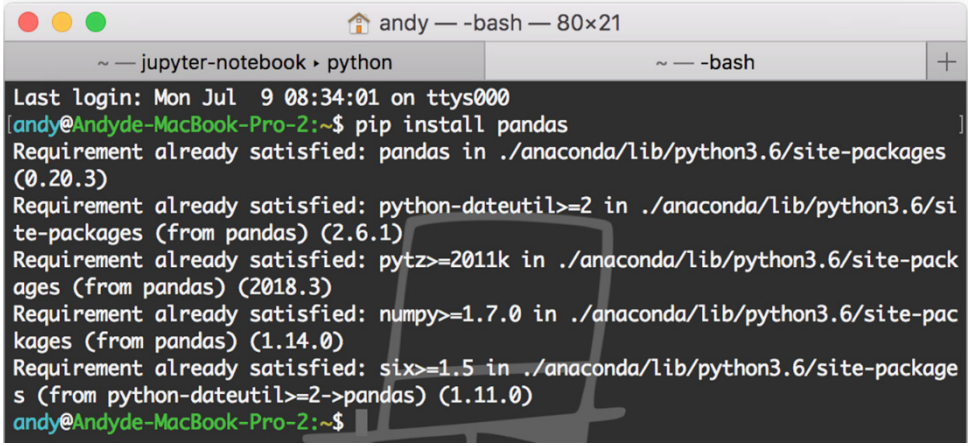


圖 9-15 pip install (OS X 環境)

Mac 用戶的話可以直接打開應用程式當中的終端機軟體，接著輸入 pip install 套件名稱，出現如圖 9-15 當中的畫面即為安裝完成。

Windows 環境

圖 9-16 pip install (Windows 環境)

Windows 用戶則在應用程式中搜尋 Anaconda Prompt，點選右鍵以系統管理員身份執行，接著輸入 pip install 套件名稱，出現如 (圖 9-16) 當中的畫面即為安裝完成。

OS X + Windows 環境

```
!pip install pandas
```

```
Requirement already satisfied: pandas in /Users/andy/anacon
da/lib/python3.6/site-packages (0.20.3)
Requirement already satisfied: python-dateutil>=2 in /Users
/andy/anaconda/lib/python3.6/site-packages (from pandas) (2
.6.1)
Requirement already satisfied: pytz>=2011k in /Users/andy/a
naconda/lib/python3.6/site-packages (from pandas) (2018.3)
Requirement already satisfied: numpy>=1.7.0 in /Users/andy/
anaconda/lib/python3.6/site-packages (from pandas) (1.14.0)
Requirement already satisfied: six>=1.5 in /Users/andy/anac
onda/lib/python3.6/site-packages (from python-dateutil>=2->
pandas) (1.11.0)
```

圖 9-17 pip install (OS X + Windows 環境)

除了使用各自的終端機安裝第三方套件的方法外，也可以直接在 Notebook 當中進行下載，在 Cell 當中首先打 ! 符號，後面輸入 pip install 套件名稱在執行，同樣可以下載成功 (圖 9-17)。範例中安裝的為 pandas 套件，此一套件廣泛用於資料分析當中，協助清整資料以及進行統計性分析，後續我們也會使用此一套件，因此建議讀者可以照著上述的方法先進行安裝。

小結

在此一章節中，藉由我們的帶領，相信各位都已成功安裝好 Anaconda，以及開啟 Jupyter notebook 試著執行看看 Python 程式了。在後面的章節中我們將會帶領各位進行大量的實作，中間不但會使用到大量的第三方函式庫，同時也會使用到許多較複雜的 Python 程式 (如 Json)，因此建議各位讀者若是之前從來沒有程式語言基礎，可以先稍微了解一下的基礎 Python 教學，這樣在後續的章節中才不會覺得太吃力呦！

9-2 資料取得：認識 Facebook API

本章節將會為各位介紹什麼是 API 以及如何透過 API 取得 Facebook 粉絲專頁的資料，中間資料爬取的過程皆會透過 Python 程式來完成。Facebook 作為社群網站的龍頭之一想必大家都已經耳熟能詳、且每天都必定會點開來看些消息吧！因此我們認為選用 Facebook 粉絲專頁的資料作為後續分析的範例是再適合不過的了。

API 介紹

API 全名為 Application Programming Interface，中文名稱又可稱為 ” 應用程式介面 ”；維基百科給予 API 的定義為「與網際網路相連的端系統提供的一個應用程式介面是軟體系統不同組成部分銜接的約定。」如果單看此定義有點太過於學術，並不好理解，因此我們用一個自動販賣機的例子來舉例說明。假設今天我們口渴了想要一瓶可樂，而我們需要透過前方面板上的按鈕來告訴販賣機我們今天想要的飲料是可樂，當點選完成以及付完款後可樂就會從下方跑出來給我們了。在取得可樂的過程中我們可以把 Facebook 的資料當作是販賣機當中的飲料，而中間的面板按鈕就是 API，最後付完款就可以取得 Facebook 的資料了，因此整體流程如下：

1. 想要一瓶可樂 (想要的資料)

2. 投幣按下按鈕 (送出請求)

3. 販賣機掉出可樂 (取得資料)

想必大家一定會很好奇，欸奇怪為什麼我只是想要 Facebook 的資料而已需要這麼麻煩？原因在於任何的社群媒體 (如：FB, IG, Twitter...) 都掌握了相當多關於個人隱私的資料，當這些資料隨意流出時會產生相當多的麻煩與風暴 (如 :2018 年 Facebook 劍橋流出資料事件)。這些社群媒體公司有必要控管哪些資料是可以開放給大眾使用，而又有哪些資料是需要保護使用者不會造成個資外流的。因此 API 就是我們與社群媒體公司中間接洽的窗口，透過 API 才可以取得我們想要的資料。

當然在使用 API 時有可能發生一種情況，以自動販賣機的例子來說：明明知道超商有賣水蜜桃口味的可樂，可是販賣機裡卻沒有看到，所以我們就知道販賣機裡沒有賣的飲料等同於 API 不開放、不讓你取得的資料。甚至有些公司的 API 可能也會有每月取得資料多寡的上限，就像販賣機的飲料賣完一樣，或是超過此上限就需要付費才能取得。雖然不開放的資料無法透過 API 來取得，但其實還是有其他透過網路爬蟲的技巧可以取得，不過這就是另外一個議題了，有興趣的讀者可以在去搜尋相關文章內容。

Facebook API

今天我們想要取得的是 Facebook 粉絲專頁的資料，因此就要透過 Facebook API 這個管道來取得資料；目前僅開放粉絲專頁、公開社團以及些許好友的公開資料，非好友的使用者資料因涉及隱私權問題並無開放。此外因為 2018 年的劍橋事件造成現階段只有該粉絲專頁的管理者能夠取得自己名下管理的粉專資料而已，因此若讀者並非是任何粉專的管理者，建議可以先隨意建立一個粉絲專頁，並且簡單發些文和留言，再試著使用以下的方法看能不能成功取得這些資料呦。

步驟 1：進入 **Facebook API** 官網：**https://developers.facebook.com**

圖 9-18 Facebook API 官網

步驟 2：點擊右上角更多的下拉式選單，找到 ” 工具 ” 後點擊進入頁面。

圖 9-19 找到 ” 工具 ” 選項

步驟 3：在開發人員工具頁面中，找尋圖形 **API** 測試工具，點擊進入頁面。

圖 9-20 找尋圖形 API 測試工具

步驟 4：進入圖形 **API** 測試工具後，點擊右側的 **Generate Access Token** 。

圖 9-21 Generate Access Token

步驟 5：點擊取得用戶存取權杖。

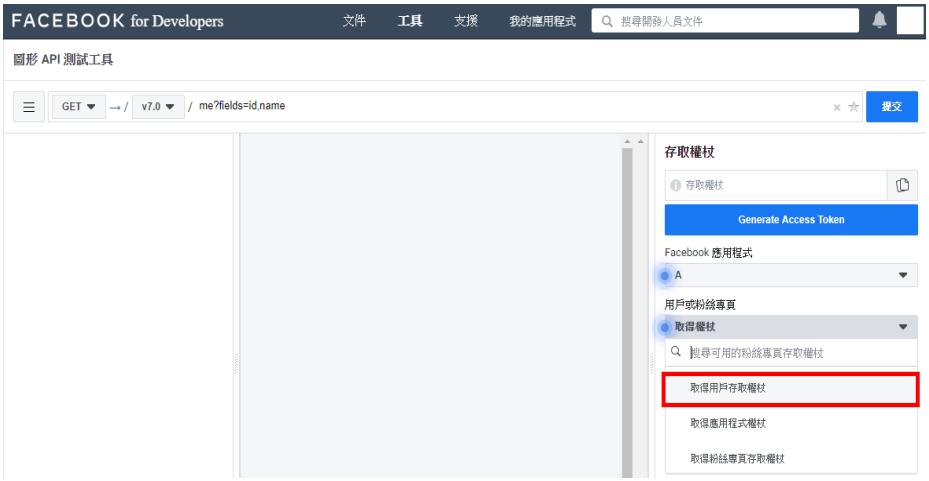


圖 9-22 取得用戶權杖

步驟 6：此頁面能設定我們能夠取得什麼樣資料的權限，現在我們要抓取的是粉絲專頁的資料，因此將中間「活動、社團和粉絲專頁」的權限全部打開。

圖 9-23 設定權限

步驟 7：設定完成權限後，回到 **Facebook** 的粉絲專頁網頁，並且複製上方的網址列。

圖 9-24 複製網址列

步驟 8：照著圖 **9-25** 中的指示貼上複製好的網址列，按下右方的提交，此時可以看見畫面下方出現了粉絲專頁的名稱以及該粉絲專頁的專屬 **id** 。

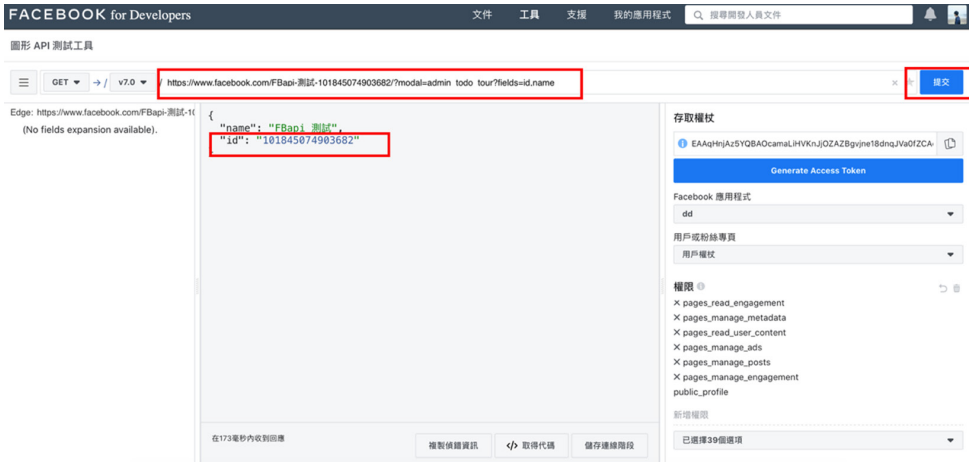


圖 9-25 取得粉絲專頁的 id

步驟 9：將出現的粉絲專頁 **id** 貼回剛剛貼上網址列的地方，重新按下提交後可以發現左邊的白框出現了搜尋欄位。

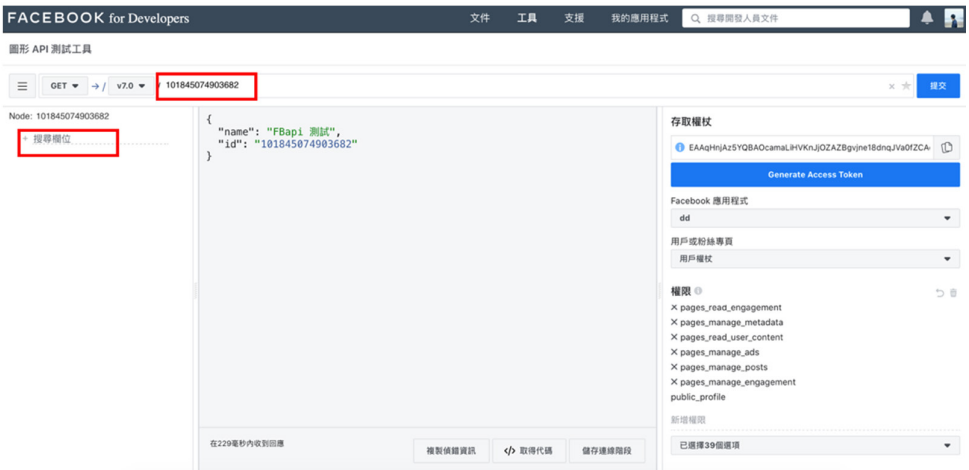


圖 9-26 出現搜尋欄位

步驟 10：在左方的搜尋欄位中找尋 **posts** 並點擊。

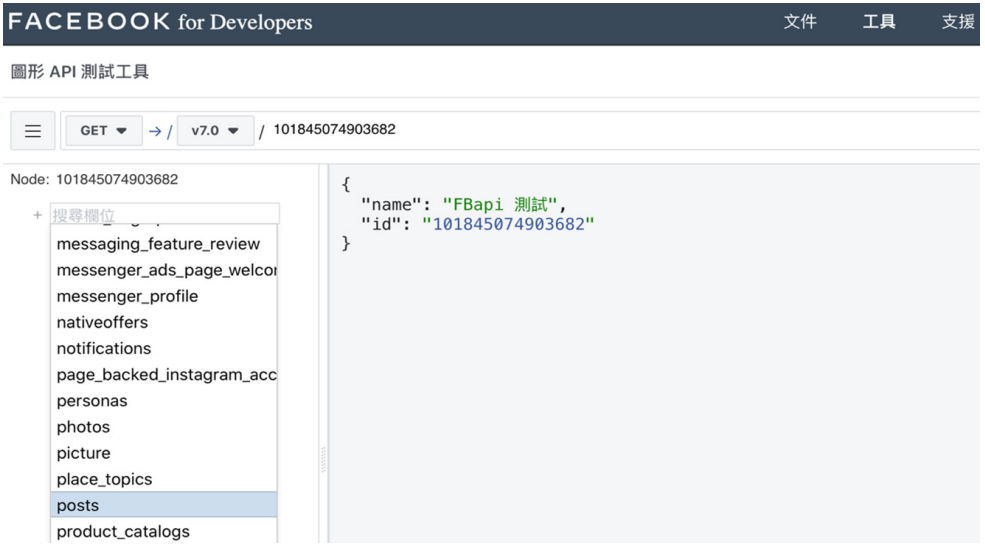


圖 9-27 點選

步驟 11：點擊提交後可以發現中間畫面出現了該粉絲專頁最新的貼文內容！

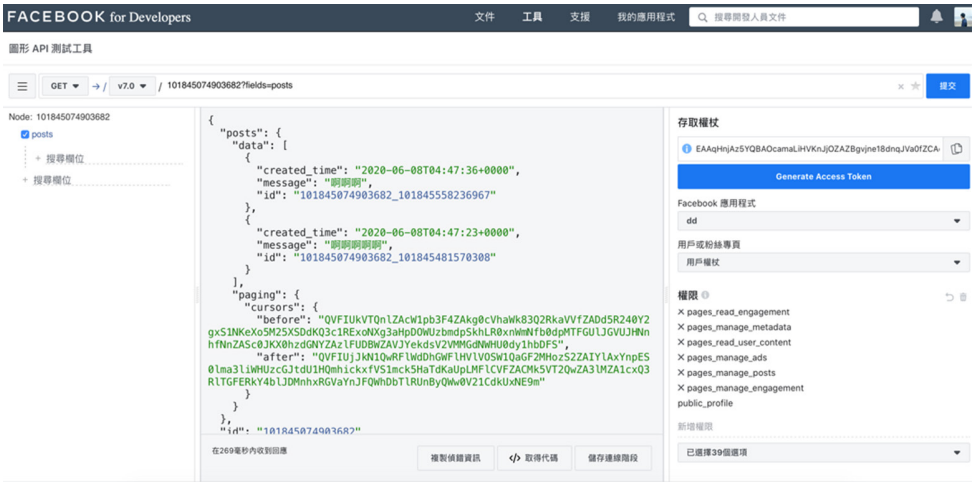


圖 9-28 貼文內容

步驟 12：剛剛我們所做的動作也就是透過 **API** 請求取得 **posts** 貼文內容的資料，而我們可以繼續新增其他想要的資訊 **(** 如 **: Likes** 按讚數量等 **)**。

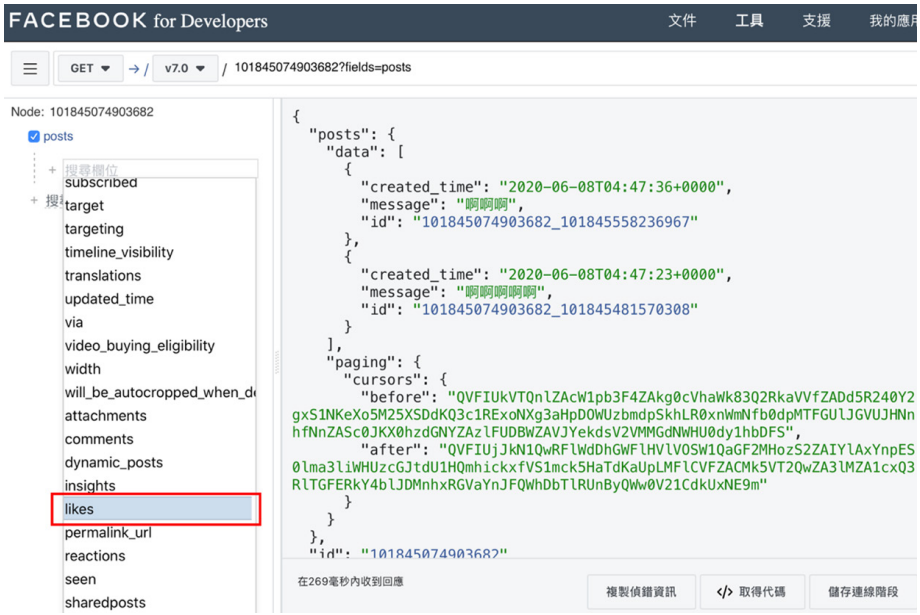


圖 9-29 可以繼續新增其他的資訊

步驟 13：點擊圖 **9-30** 畫面中的小箭頭將整個網址展開。

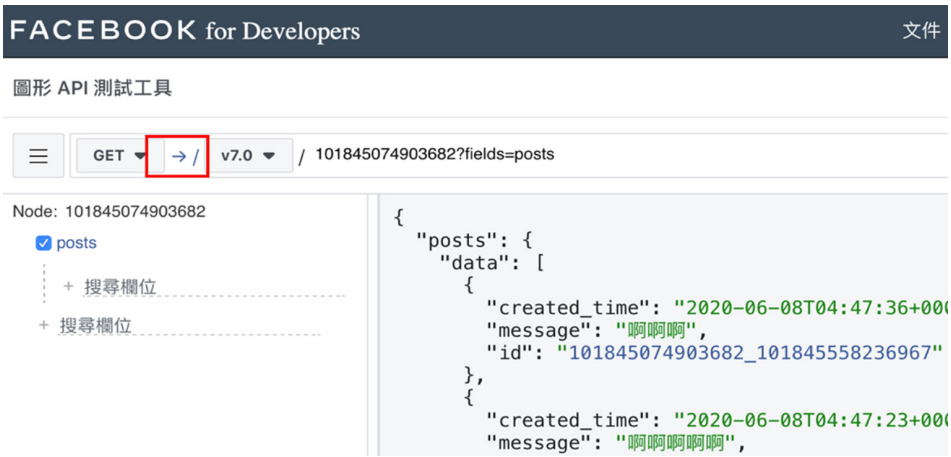


圖 9-30 網址展開

步驟 14：最後我們將此網址列複製起來，後續將透過 **Python** 程式來抓取資料。

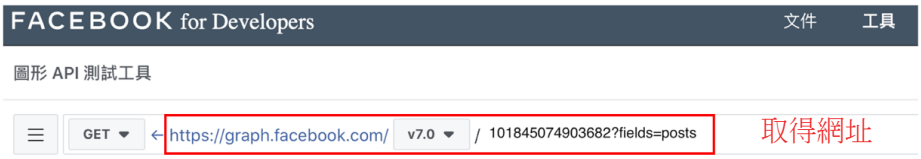


圖 9-31 複製網址

Python 串接 API

接下來在此小節中，我們將教各位如何透過 Python 程式串接 Facebook API 以及成功將粉絲專頁的資料存取成 Excel 檔案輸出。

匯入套件

首先先匯入相關套件，尚未安裝過的記得安裝 Json 與 requests 套件

```
01 import json
02 import requests
```

串接 Facebook API

此時我們需要剛剛圖形 API 測試工具頁面中的存取權杖、網址以及粉專 ID (圖 9-32)。

```
token = 存取權杖
res = requests.get( 搜尋欄位的網址 )
```

圖 9-32

依照相關位置將資訊複製至各字串當中，token 存放的是存取權杖，res 存放的則是請求過後取得的資料。

```
token =
'EAAVBPxZBVDb8BAIk78UfZBuID04j8VrrPT29IOOVUB8Dmo5
WZCa7uFIRlhpGjiW1ZBmiKLwS769xukxNUZAzoNri57UrL5Eld
9ZBkY9xlre6WgYXWl49QHK0qiBZBAG5e35bUTWVGhEPDhoxv5t
behZBeDybBZBO6Qr9K3C28yfQrQtHnmuFv8QQZCeDws22ZBB1N
cZD'

res = requests.get('https://graph.facebook.com/v2.9/{}/
posts&access_token={}'.format(325767030937170, token))
```

實際上因為需要取得文章按讚數、分享數、留言等相關資料，因此 requests 的網址會是非常長的一段，建議各位讀者可以直接複製後，按下執行送出請求。

https://graph.facebook.com/v2.9/{}/posts?fields=id,message,type,created_time,link,shares,comments.limit(0).summary(true),likes.limit(0).summary(total_count).as(reaction_like),reactions.type(LOVE).limit(0).summary(total_count).as(reactions_love),reactions.type(WOW).limit(0).summary(total_count).as(reactions_wow),reactions.type(HAHA).limit(0).summary(total_count).as(reactions_haha),reactions.type(SAD).limit(0).summary(total_count).as(reactions_sad),reactions.type(ANGRY).limit(0).summary(total_count).as(reactions_angry)&access_token={}

使用 json 讀取其格式

透過 API 抓回來的資料是使用 Json 的方式進行存取，因此我們使用 `json` 套件進行解碼，並存入 `fanpage` 變數當中。

```
12  fanpage = json.loads(res.text)
13  print(fanpage)
```

執行結果：

```
{'data': [{'id': '325767030937170_1050616578452208' ,
'message': '【徵才】\n 原友蘭教授 誠徵 # 專任全職助理 \n\n➡ 工作
地點：東海大學 \n 工作時間：週一到週五，# 沒有打卡，但是每週會需要固
定一天報告進度 \n➡ 工作條件：英文能書信溝通、統計分析、報告撰寫，
會處理 #GoogleAnalysis 資料或是 R 語言的優先考慮 \n➡ 工作福利：
\n* 有 3-4 次出國機會（印尼與菲律賓）\n* 包含住宿與機票 \n➡ 薪資
待遇：\n* 32000-34000 之間 + 勞健保 + 一個月年終 \n\n➡ 其他事項：
\n* 有碩士學會就可申請在東海兼課 \n* 會幫忙申請教師證 \n\n➡ 聯繫
方式：\n* yoyoyuan@go.thu.edu.tw\n* 0910-659134', 'type':
'status', 'created_time': '2019-11-06T02:55:08+0000',
'shares': {'count': 1}, 'comments': {'data':
```

文章相關資料位在鍵值為 `data` 的名稱內。

```
14  fanpage['data']
```

執行結果：

```
[{'comments': {'data': [],
    'summary': {'can_comment': True, 'order': 'ranked',
'total_count': 0}},
  'created_time': '2019-11-06T02:55:08+0000',
  'id': '325767030937170_1050616578452208',
  'message': '【徵才】\n 原友蘭教授 誠徵 # 專任全職助理 \n\n➡ 工作地點：東海大學 \n 工作時間：週一到週五，# 沒有打卡，但是每週會需要固定一天報告進度 \n➡ 工作條件：英文能書信溝通、統計分析、報告撰寫，會處理 #GoogleAnalysis 資料或是 R 語言的優先考慮 \n➡ 工作福利：\n* 有 3-4 次出國機會（印尼與菲律賓）\n* 包含住宿與機票 \n➡ 薪資待遇：\n* 32000-34000 之間 + 勞健保 + 一個月年終 \n\n➡ 其他事項：\n* 有碩士學會就可申請在東海兼課 \n* 會幫忙申請教師證 \n\n➡ 聯繫方式：\n* yoyoyuan@go.thu.edu.tw\n* 0910-659134',
```

我們可以發現透過一次的 API 會抓回最新的 25 筆文章內容。

```
15 len(fanpage['data'])
```

執行結果：

```
25
```

查看第一筆資料

可以發現裡面存放文章內容、貼文時間、按讚數量等等資訊。

```
16 fanpage['data'][0]
```

執行結果：

```
{'comments': {'data': [],
  'summary': {'can_comment': True, 'order': 'ranked',
'total_count': 0}},
 'created_time': '2019-11-06T02:55:08+0000',
 'id': '325767030937170_1050616578452208',
 'message': '【徵才】\n 原友蘭教授 誠徵 # 專任全職助理 \n\n➡ 工作地點：東海大學 \n 工作時間：週一到週五，# 沒有打卡，但是每週會需要固定一天報告進度 \n➡ 工作條件：英文能書信溝通、統計分析、報告撰寫，會處理 #GoogleAnalysis 資料或是 R 語言的優先考慮 \n➡ 工作福利：\n* 有 3-4 次出國機會（印尼與菲律賓）\n* 包含住宿與機票 \n➡ 薪資待遇：\n* 32000-34000 之間 + 勞健保 + 一個月年終 \n\n➡ 其他事項：\n* 有碩士學會就可申請在東海兼課 \n* 會幫忙申請教師證 \n\n➡ 聯繫方式：\n* yoyoyuan@go.thu.edu.tw\n* 0910-659134',
```

第一筆資料發文時間：

```
17 | fanpage['data'][0]['created_time']
```

執行結果：

```
'2019-11-06T02:55:08+0000'
```

第一筆資料文章 ID：

```
18 | fanpage['data'][0]['id']
```

執行結果：

```
'325767030937170_1050616578452208'
```

第一筆資料文章內容：

```
19 | fanpage['data'][0]['message']
```

執行結果：

```
'【徵才】\n 原友蘭教授 誠徵 # 專任全職助理 \n\n➡ 工作地點：東海大學 \n 工作時間：週一到週五，# 沒有打卡，但是每週會需要固定一天報告進度 \n➡ 工作條件：英文能書信溝通、統計分析、報告撰寫，會處理 #GoogleAnalysis 資料或是 R 語言的優先考慮 \n➡ 工作福利：\n* 有 3-4 次出國機會（印尼與菲律賓）\n* 包含住宿與機票 \n➡ 薪資待遇：\n* 32000-34000 之間 + 勞健保 + 一個月年終 \n\n➡ 其他事項：\n* 有碩士學會就可申請在東海兼課 \n* 會幫忙申請教師證 \n\n➡ 聯繫方式：\n* yoyoyuan@go.thu.edu.tw\n* 0910-659134'
```

第一筆資料不同的按讚數量 (例 : 讚 , 驚訝 , 愛心 , 笑 , 生氣 , 難過)

```
20 print(fanpage['data'][0]['reaction_like']['summary']['total_count'])
21 print(fanpage['data'][0]['reactions_wow']['summary']['total_count'])
22 print(fanpage['data'][0]['reactions_love']['summary']['total_count'])
23 print(fanpage['data'][0]['reactions_haha']['summary']['total_count'])
24 print(fanpage['data'][0]['reactions_angry']['summary']['total_count'])
25 print(fanpage['data'][0]['reactions_sad']['summary']['total_count'])
```

執行結果：

```
12
0
0
0
0
0
```

第一筆資料分享數量 :

```
26 fanpage['data'][0]['shares']['count']
```

執行結果：

```
1
```

第一筆資料留言數量 :

```
27 fanpage['data'][0]['comments']['summary']['total_count']
```

執行結果：

```
2
```

現在我們得知了所有與文章相關資訊的存放位置了，而我們一共有 25 筆文章資料，因此我們需要透過迴圈的方式將資料一筆筆取出，再存入不同的串列當中。

把資訊存入 list 裡

首先建立各個資訊的空串列變數，在透過 for 迴圈將每一筆資料相對應的資訊放入串列當中。第 43 行那邊加入了例外處理，原因是當分享數量是 0 時它並不會出現在 Json 檔裡，因此需要我們手動給予否則程式會出錯。

```
time, ID, context, like, wow, love,
 haha, angry, sad, share, comment = [], [],
 [], [], [], [], [], [], [], [], []

for i in fanpage['data']:
    time.append(i['created_time'])
    ID.append(i['id'])
    context.append(i['message'])

    like.append(i['reaction_like']['summary']['total_count'])
    wow.append(i['reactions_wow']['summary']['total_count'])
    love.append(i['reactions_love']['summary']['total_count'])
    haha.append(i['reactions_haha']['summary']['total_count'])
    angry.append(i['reactions_angry']['summary']['total_count'])
    sad.append(i['reactions_sad']['summary']['total_count'])
    try:
        share.append(i['shares']['count'])
    except:
        share.append(0)
    comment.append(i['comments']['summary']['total_count'])
```


25 筆資料以外的資料呢？

我們透過一次的 API 請求只會回傳 25 筆資料而已，原因是若一次就想把上千上萬篇的資料取得容易造成伺服器塞車或當機現象。而若要抓取存有下一個 25 筆資料的 Json 需要再次透過 API 請求，而換頁資訊位在鍵值為 paging 和 next 的欄位裡。

```
fanpage['paging']['next']
```

執行結果：

```
'https://graph.facebook.com/v2.9/325767030937170/posts?access_
token=EAAVBPxZBVDb8BAJib0SAxWnOtZCPBjXVnuPtUx0iVMYdq0GlcRrh
xwU5sZCGv3idx0ObuWgW3UVWzHXTpk3ZBB4gi3aIldxrgUdyyRYoZCQIvl9
ZCAoZB7jXnZA8cT74gQZAXEMHj3XpK7rZCZAUeSLySdOpZCHIQNi2B
ab8Dh4HZAr0zqkYWARIYo0LpDzNms3dIQGUY746LfRfDsgZD
ZD&fields=id%2Cmessage%2Ctype%2Ccreated_time%2Clink%2Cshares%2Ccomments.
limit%280%29.summary%28true%29%2Clikes.limit%280%29.summary%28to
tal_count%29.as%28reaction_like%29%2Creactions.type%28LOVE%29.limit%280%29.
summary%28total_count%29.as%28reactions_love%29%2Creactions.type%28WOW%29.
limit%280%29.summary%28total_count%29.as%28reactions_wow%29%2Creactions.
type%28HAHA%29.limit%280%29.summary%28total_count%29.
```

抓取 25 筆以外的資料

上述的執行結果已經列出了接下來需要請求的網址列了，因此可以透過前面同樣的方法再次取得資料。不過一次一次的抓取 25 筆資料的速度實在是太慢了，在這邊我們可以透過 while 迴圈幫助我們預先設定好總共想抓取的文章數量有多少，再一次全部抓取回來。如下面程式所述，page 為存放爬取頁數的變數，當每次抓取後 page 會自動 +1，而當 page 大於等於 5 時抓取會自動停止，也就是共抓取 100 筆文章資料。

```
page=2
url = fanpage['paging']['next']
while page < 5:
    res = requests.get(url)
    fanpage2 = json.loads(res.text)
    for i in fanpage2['data']:
        time.append(i['created_time'])
        ID.append(i['id'])
        context.append(i['message'])

        like.append(i['reaction_like']['summary']['total_count'])
        wow.append(i['reactions_wow']['summary']['total_count'])
        love.append(i['reactions_love']['summary']['total_count'])
        haha.append(i['reactions_haha']['summary']['total_count'])
        angry.append(i['reactions_angry']['summary']['total_count'])
        sad.append(i['reactions_sad']['summary']['total_count'])
        try:
            share.append(i['shares']['count'])
        except:
            share.append(0)
        comment.append(i['comments']['summary']['total_count'])
    url = fanpage2['paging']['next']
    page += 1
```

轉成 DataFrame 格式

成功抓取 100 筆的文章資料後，使用 zip 函數將不同的 list 合併。

```
information = list(zip(time, ID, context, like,
wow, love, haha, angry, sad, share, comment))
```

並且匯入 pandas 套件將其轉成 DataFrame 格式。

```
import pandas as pd
df = pd.DataFrame(information,
columns=['time','id','context','like','wow',
'love','haha','angry','sad','share','comment'])
```

透過 DataFrame 的操作我們可以輕易將抓取的資訊轉成表格內容存放。

```
df.head()
```

	time	id	context	like	wow	love	haha	angry	sad	share	comment
0	2018-11-06T02:55:08+0000	325767030937170_1050616578452208	【徵才】\n原友蘭教授 誠徵 #專任全職助理\n\n➡ 工作地點：東海大學\n工作時間 週一...	12	0	0	0	0	0	1	0
1	2018-10-26T15:30:42+0000	325767030937170_1045065325674000	【東吳巨資碩士甄試報名】\n網路蓬勃發展的時代，KOL的人數也逐漸上升，\n但究竟要如何經營...	15	0	0	0	0	0	0	0
2	2018-10-24T04:50:17+0000	325767030937170_1043766419137224	【東吳巨資碩士甄試報名】\n想要了解時下深度學習、AI究竟是什麼嗎？\n就是現在！東吳巨資碩...	36	0	2	3	0	0	8	9
3	2018-10-19T16:07:39+0000	325767030937170_1041456549368211	【東吳巨資碩士甄試報名】\n還在煩惱報名步驟繁瑣嗎？\n還在擔心是不是少準備了什麼東西嗎？\...	13	0	0	0	0	0	2	0
4	2018-10-18T11:00:01+0000	325767030937170_1040710716109461	【東吳巨資碩士甄試報名】\n現在是數據化的時代，你卻還一片徬徨嗎？\n想要學以致用，習得一些...	50	0	1	0	0	0	11	0

圖 9-33

連同第一次換頁前爬的共爬取了 4 頁，100 筆資料。

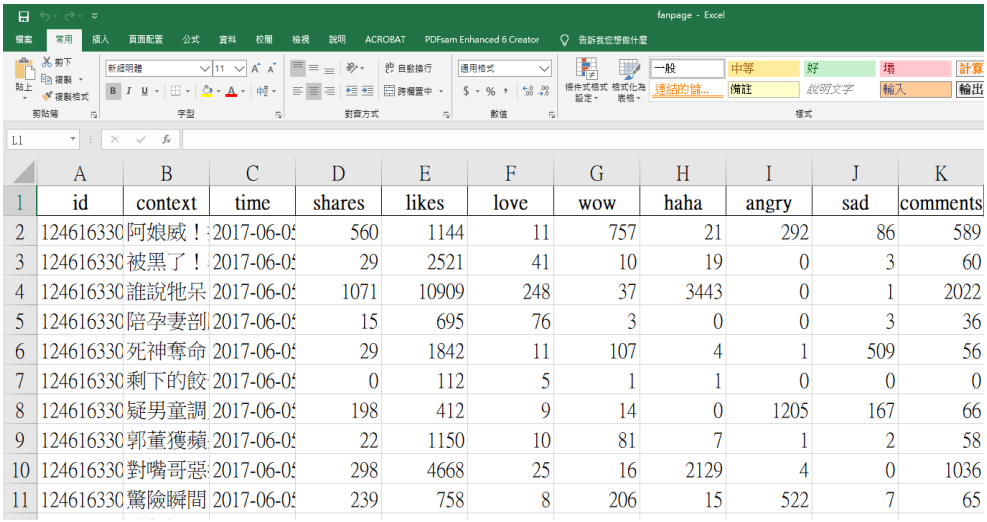
```
df.info()
```

執行結果：

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 11 columns):
time        100 non-null object
id          100 non-null object
context     100 non-null object
like        100 non-null int64
wow         100 non-null int64
love        100 non-null int64
haha        100 non-null int64
angry       100 non-null int64
sad         100 non-null int64
share       100 non-null int64
comment     100 non-null int64
dtypes: int64(8), object(3)
memory usage: 8.7+ KB
```

最後將此 DataFrame 匯出成 Excel 檔案，完成粉絲專頁的資料搜尋步驟。

```
80 df.to_excel('粉絲專頁資料.xlsx', index=False)
```



	A	B	C	D	E	F	G	H	I	J	K
1	id	context	time	shares	likes	love	wow	haha	angry	sad	comments
2	124616330	阿娘威！	2017-06-0	560	1144	11	757	21	292	86	589
3	124616330	被黑了！	2017-06-0	29	2521	41	10	19	0	3	60
4	124616330	誰說牠呆	2017-06-0	1071	10909	248	37	3443	0	1	2022
5	124616330	陪孕妻剖	2017-06-0	15	695	76	3	0	0	3	36
6	124616330	死神奪命	2017-06-0	29	1842	11	107	4	1	509	56
7	124616330	剩下的餃	2017-06-0	0	112	5	1	1	0	0	0
8	124616330	疑男童調	2017-06-0	198	412	9	14	0	1205	167	66
9	124616330	郭董獲蘋	2017-06-0	22	1150	10	81	7	1	2	58
10	124616330	對嘴哥惡	2017-06-0	298	4668	25	16	2129	4	0	1036
11	124616330	驚險瞬間	2017-06-0	239	758	8	206	15	522	7	65

圖 9-34

小結

在此一節中我們向各位介紹了何謂 API、如何使用 Facebook API 以及透過 Python 程式串接取得資料，中間使用到了大量的 Python 語法進行資料的清理以及邏輯運算，當然這只是其中一種取得 FB 粉專資料的寫法而已，網路上搜尋也可以找到不同的程式寫法，也鼓勵大家可以多瀏覽參考不同的寫法，或許可以得到不同的經驗值呢。若看完此節覺得有點吃力實屬正常，這已經是相當實用以及高階的應用範圍了，甚至很多公司會專門請人幫忙爬取甚至分析相關的粉絲專頁網站，因此還是建議新手可以多多練習 Python 程式的基本功 (如 : 迴圈、串列的處理、Json 的處理等等)，倘若看完此章節還跟得上的讀者相信對於 Python 已經有一定程度的了解及經驗了呢！那就趕緊跟著我們進入下一章節吧。

9-3 簡單料理：EDA 探索式資料分析

本章節將會為各位讀者示範一些使用 Python 清理資料的方式，包含處理遺失值、加減乘除計算、轉換時間格式等，所有過程皆會使用到 pandas 這個第三方的函式庫。清理完資料後接著會進行一些簡單的描述性統計以及使用視覺化圖表的方式呈現數據，過程中會使用到另外兩款函式庫 matplotlib 以及 seaborn，還沒安裝的讀者建議可以先行進行安裝。在這部分使用到的範例資料是東森新聞 FB 粉絲專頁在 2017 年左右的 10,000 篇文章資料，建議各位可以先與我們用同一份資料實作，之後再改用自己的其他資料試試看。

資料前處理

匯入套件與資料

首先匯入 pandas 套件以及匯入粉專資料。

```
01 import pandas as pd
02 df = pd.read_excel('fanpage.xlsx')
```

查看資料 1

此份資料的欄位也就是上一章節使用 API 所抓取的欄位資訊。

```
03 df.head(2)
```

	id	context	time	shares	likes	love	wow	haha	angry	sad	comments
0	1246163309068800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😩😩\n\n影片來源...	2017-06-05T03:09:40+0000	560	1144	11	757	21	292	86	589
1	1246163309068800_1560454417322977	被黑了！李毓芬演唱「大落拍」 網友卻意外發現「亮點」\n#條紋編：這一段應該是昨天的亮點表演...	2017-06-05T03:00:00+0000	29	2521	41	10	19	0	3	60

圖 9-35

查看資料 2

整份資料共有 9,975 筆資料，且發現 context 欄位有遺失值。

```
04  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9975 entries, 0 to 9974
Data columns (total 11 columns):
id          9975 non-null object
context     9968 non-null object
time        9975 non-null object
shares      9975 non-null int64
likes       9975 non-null int64
love        9975 non-null int64
wow         9975 non-null int64
haha        9975 non-null int64
angry       9975 non-null int64
sad         9975 non-null int64
comments    9975 non-null int64
dtypes: int64(8), object(3)
memory usage: 857.3+ KB
```

圖 9-36 查看資料 info()

處理遺失值

context 內容為空值的填入字串 " 無 "，若不先處理遺失值後續的統計分析將會出錯。

```
05  df['context'] = df['context'].fillna('無')
```

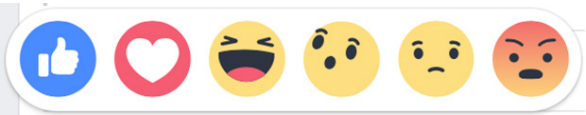
再次查看資料可以發現成功去除空值，每個欄位都有 9,975 筆資料。

```
06  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9975 entries, 0 to 9974
Data columns (total 11 columns):
id          9975 non-null object
context     9975 non-null object
time        9975 non-null object
shares      9975 non-null int64
likes       9975 non-null int64
love        9975 non-null int64
wow         9975 non-null int64
haha        9975 non-null int64
angry       9975 non-null int64
sad         9975 non-null int64
comments    9975 non-null int64
dtypes: int64(8), object(3)
memory usage: 857.3+ KB
```

圖 9-37 查看資料 info()

整理資料



計算按讚數量加總，共有下面六種指標，新增欄位 likes_count，為六種按讚指標的加總。

```
07  df['likes_count'] =
08  df['likes']+df['love']+df['wow']+df['haha']+df['angry']+df['sad']
```

再次查看資料確實成功新增了 likes_count 的欄位。

```
09  df.head(1)
```

	id	context	time	shares	likes	love	wow	haha	angry	sad	comments	likes_count
0	1246163309068001560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😱😱\n\n影片來源...	2017-06-05T03:09:40+0000	560	1144	11	757	21	292	86	589	2311

圖 9-38 查看資料 head()

整理時間資料

查看原先的時間格式，這是 FB 存放時間的資料格式，要先進行些許轉換後續分析才好使用。

```
10  df['time'][0]
```

執行結果：

```
'2017-06-05T03::09:40+0000'
```

去除多餘的時間符號以及去除秒數後四位。

```
df['time'] = df['time'].str.replace('T','').str.replace('-','')
df['time'] = df['time'].str.replace(':','').str.split('+').str[0]
df['time'][0]
```

執行結果：

```
'20170605030940'
```

轉換成 pandas 中的時間格式 datetime。

```
df['time'] = pd.to_datetime(df['time'],format='%Y%m%d%H%M%S')
df['time'][0]
```

執行結果：

```
Timestamp('2017-06-05 03:09:40')
```

FB 預設的時區為中原標準時間，也就是倫敦的時區，因此需要額外再加 8 小時才是台灣的時區。

```
import datetime
df['time'] = df['time']+datetime.timedelta(hours = 8)
df['time'][0]
```

執行結果：

```
Timestamp('2017-06-05 11:09:40')
```

新增欄位 hour 存放小時，且轉為 object 資料型態。

```
df['hour'] = df['time'].dt.hour
df['hour'] = df['hour'].astype('object')
df['hour'][0]
```

執行結果：

```
11
```

新增欄位 weekday 存放星期 (預設為數字 0~6)，數字從 0~6 代表星期一至日，而我們將其取代成英文字串較好觀看。

```
df['weekday'] = df['time'].dt.weekday
df['weekday'] = df['weekday'].replace([0, 1, 2, 3, 4, 5, 6],
['Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunday'])
df['weekday'][0]
```

執行結果：

```
'Monday'
```

再次查看資料，成功新增了 hour 與 weekday 兩個欄位。

```
df.head(1)
```

	id	context	time	shares	likes	love	wow	haha	angry	sad	comments	likes_count	hour	weekday
0	1246163309068001560_5011973182 99	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😂😂\n\n影片來源...	2017-06-05 11:09:40	560	1144	11	757	21	292	86	589	2311	11	Monday

圖 9-39 查看資料 head()

取出小編名稱

新聞的粉專都會有個奇特的現象，在每篇文章中都會有個 tag 來標示是哪一位小編所發的文章。這是一個很有趣的現象，我們也特別把它單獨取出來，後續分析看看有沒有什麼有趣的發現。

圖 9-40 取出小編名稱

每篇貼文的最後都會出現 #XX 編，因此加以切開取出，存入變數 curator 的 list 當中。迴圈中使用 if 判斷式是因為不是所有的 # 字號都是小編的名稱，因此要先確定其為小編名稱才加入 list 當中。

```
curator = []
for i in df['context'].str.split('#').str[1].str.split('：').str[0]:
    try:
        if i[-1] == '編':
            curator.append(i)
        else:
            curator.append(None)
    except:
        curator.append(None)
```

在 dataframe 當中新增欄位 curator 加入小編名稱。

```
df['curator'] = pd.DataFrame(curator)
```

查看資料後確實成功取出小編名。

```
df.head(1)
```

	id	context	time	shares	likes	love	wow	haha	angry	sad	comments	likes_count	curator
0	1246163309068001560_5011973182 99	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😂😂\n\n影片來源...	2017-06-05 11:09:40	560	1144	11	757	21	292	86	589	2311	要打統編

圖 9-41 查看資料 head()

描述性統計

匯入套件

matplotlib 和 seaborn 皆為 Python 的繪圖套件 。

```
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.font_manager import FontProperties
```

設定中文字型

繪圖套件本身並不支援中文，因此我們需要事先設定中文的路徑。

OS X 環境 (Mac 用戶)

```
font = FontProperties(fname=r'/System/Library/Fonts/STHeiti Light.ttc')
```

Windows 環境

```
font = FontProperties(fname=r'C:/windows/Fonts/msjh.ttc')
```

開始進行基本統計計算

連續型資料統計指標：表格之中包含數量、平均、標準差、四分位距以及最大最小值。

```
df.describe()
```

	shares	likes	love	wow	haha	angry	sad	comments	likes_count
count	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000	9975.000000
mean	277.510877	3125.289925	115.573634	118.844411	262.752381	147.826065	111.721704	585.611830	3882.008120
std	1171.178025	6770.009492	500.730071	323.255901	1105.812423	942.289195	967.558397	2545.403304	8278.568139
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	6.000000	384.000000	7.000000	8.000000	3.000000	0.000000	0.000000	10.000000	465.000000
50%	40.000000	1017.000000	15.000000	28.000000	11.000000	1.000000	1.000000	52.000000	1246.000000
75%	160.000000	2795.500000	43.000000	95.000000	77.000000	10.000000	8.000000	254.000000	3515.000000
max	39140.000000	144475.000000	11394.000000	8474.000000	30265.000000	30751.000000	48997.000000	64206.000000	156850.000000

圖 9-42 連續型資料統計指標

小編出現次數排名：可以發現 B 編是最常發文的一位小編。

```
df['curator'].value_counts().head(11)
```

```
B編          410
內編          397
條紋編         383
悠悠編         299
哩厝編         276
惡魔在身編       276
M編          275
哈姆編         266
周二編         266
閃編          265
西瓜挖大編       248
```

圖 9-43 小編出現次數排名

計算小編數量：發現東森新聞在這段期間內共有 86 位小編！

```
44  len(df['curator'].value_counts().index)
```

```
執行結果：
86
```

針對小編進行進一步分析

以圖表方式呈現發文數量前 10 名的小編。

```
45  sns.countplot(data=df, x='curator',
46      order=df['curator'].value_counts().iloc[:10].index)
47  plt.xticks(fontproperties=font, size=10)
48  plt.title('小編發文數量', fontproperties=font, size=12)
49  plt.show()
```

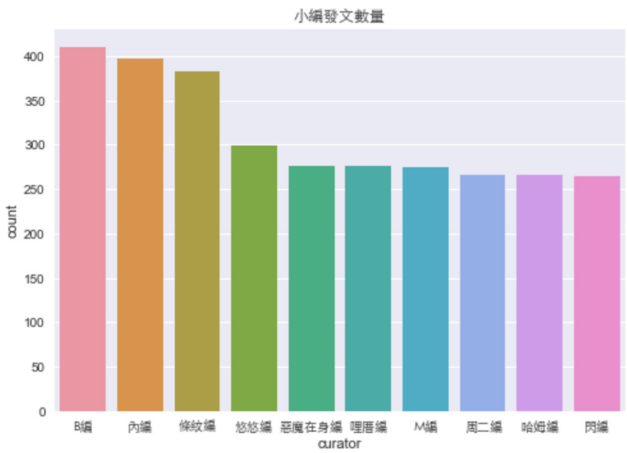


圖 9-44 小編發文數量

以圖表方式呈現發文按讚數量平均前 10 名的小編 (須先計算每一位小編各自的平均按讚數量)。

```
likes_avg = []
for i in df['curator'].value_counts().index:
    likes_avg.append([i,(df[df['curator']==i])['likes_count'].
mean()])

df2 = pd.DataFrame(likes_avg, columns=['curator','mean_likes'])

df3 = df2.sort_values('mean_likes',ascending=False).head(10)
sns.barplot(x='curator', y='mean_likes', data=df3)
plt.xticks(fontproperties=font, size=10)
plt.title('按讚數量前 10 名小編 ', fontproperties=font, size=12)
plt.show()
```

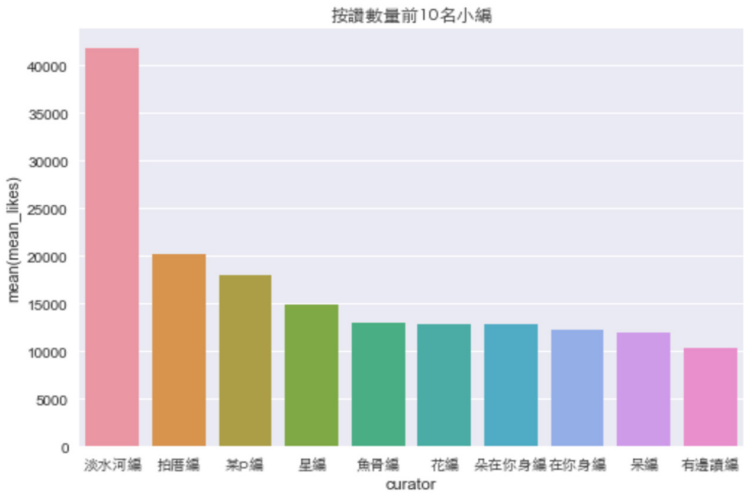


圖 9-45 按讚數量前 10 名小編

時間因子所造成的影響

以圖表方式呈現各個發文時間的統計

```
sns.countplot(data=df, x='hour')
plt.xticks(fontproperties=font, size=10)
plt.title('發文時間統計', fontproperties=font, size=12)
plt.show()
```

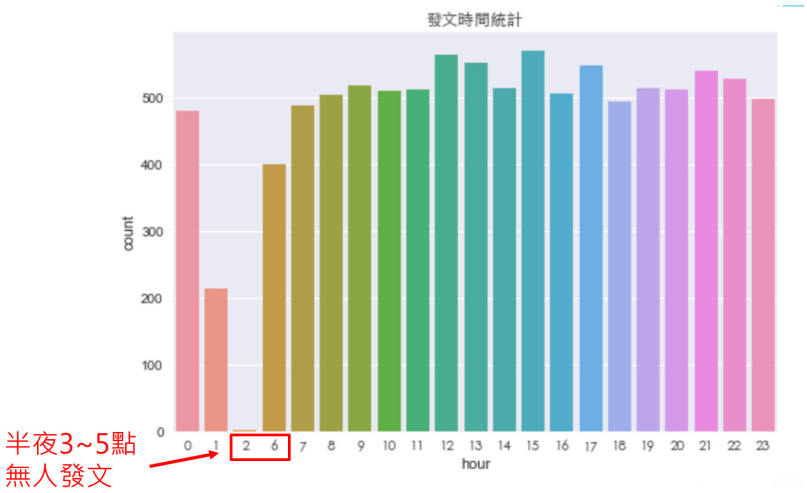


圖 9-46 發文時間的統計

以圖表方式呈現各個時段發文的按讚成效 (須先計算每個時段的按讚平均)。

```
likes_avg_hour = []
for i in df['hour'].value_counts().index:
    likes_avg_hour.append([i,(df[df['hour']==i])['likes_count'].mean()])
df4 = pd.DataFrame(likes_avg_hour, columns=['hour','mean_likes'])
sns.barplot(x='hour', y='mean_likes', data=df4)
plt.xticks(fontproperties=font, size=10)
plt.title(' 各時段發文按讚成效 ', fontproperties=font, size=12)
plt.show()
```

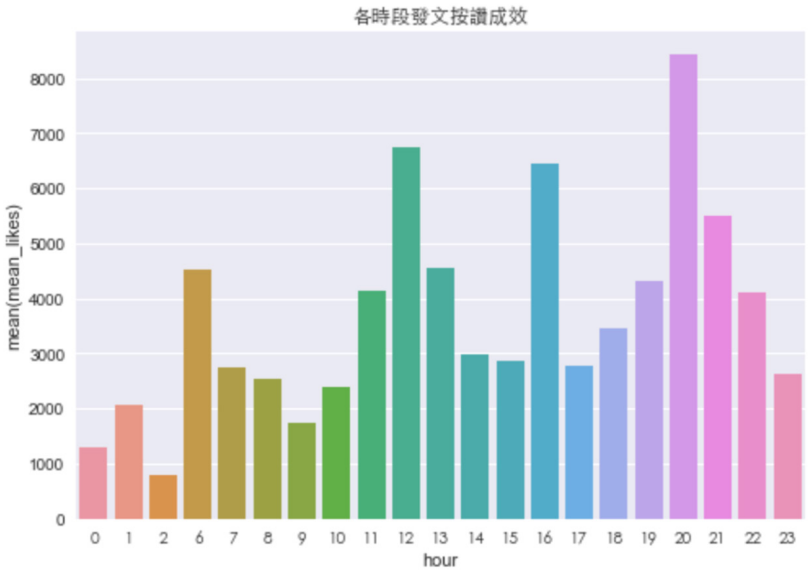


圖 9-47 各個時段發文的按讚成效

以圖表方式呈現各個發文星期的統計。

```
sns.countplot(data=df, x='weekday')
plt.xticks(fontproperties=font, size=10)
plt.title('發文星期統計', fontproperties=font, size=12)
plt.show()
```

以圖表方式呈現各個星期發文的按讚成效(須先計算每個星期的按讚平均)。

```
likes_avg_week = []
for i in df['weekday'].value_counts().index:
    likes_avg_week.append([i,(df[df['weekday']==i])['likes_count'].mean()])
df4 = pd.DataFrame(likes_avg_week, columns=['weekday','mean_likes'])
sns.barplot(x='weekday', y='mean_likes', data=df4)
plt.xticks(fontproperties=font,size=10)
plt.title('各星期發文按讚成效',fontproperties=font,size=12)
plt.show()
```

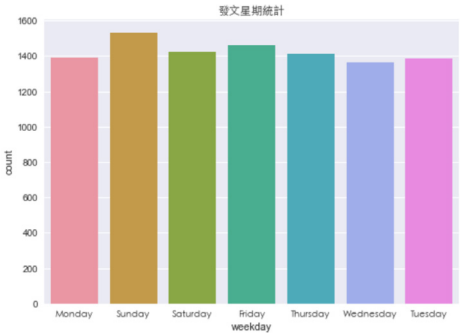
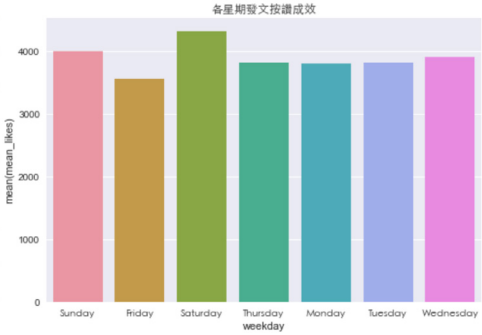


圖 9-48 發文星期的統計圖



9-49 各個星期發文的按讚成效

使用熱點圖 (Heatmap) 看兩兩之間的關係

各個星期 vs. 各個時間點的發文數量。

```
df6 = pd.crosstab(df['hour'], df['weekday'])
sns.heatmap(df6, annot=True,cmap='Oranges')
plt.show()
```

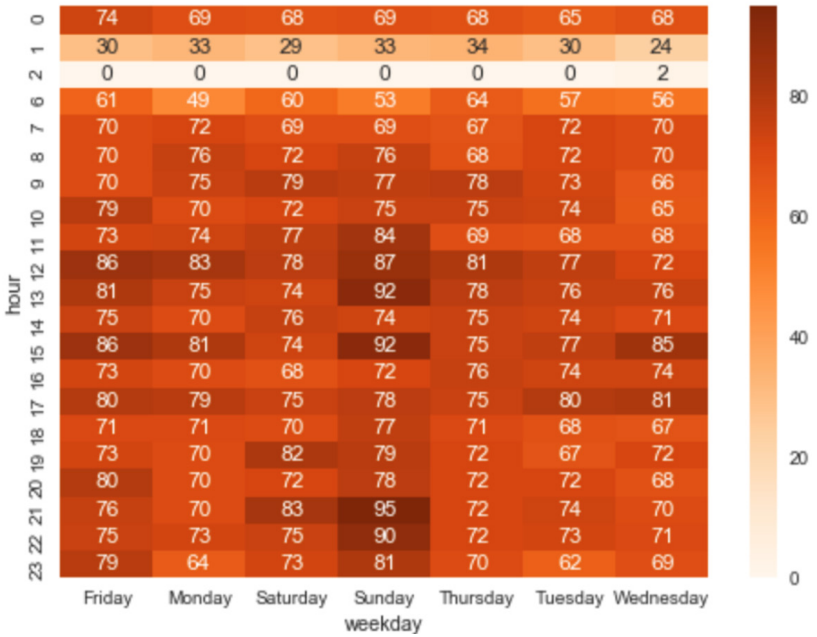


圖 9-50 熱點圖查看關係

各個星期 vs. 各個時間點的按讚數量。

```
df7 = pd.pivot_table(df, index=['hour'],
    columns=['weekday'], values=['likes_count'])
sns.heatmap(df7, annot=True, cmap='Oranges')
plt.show()
```

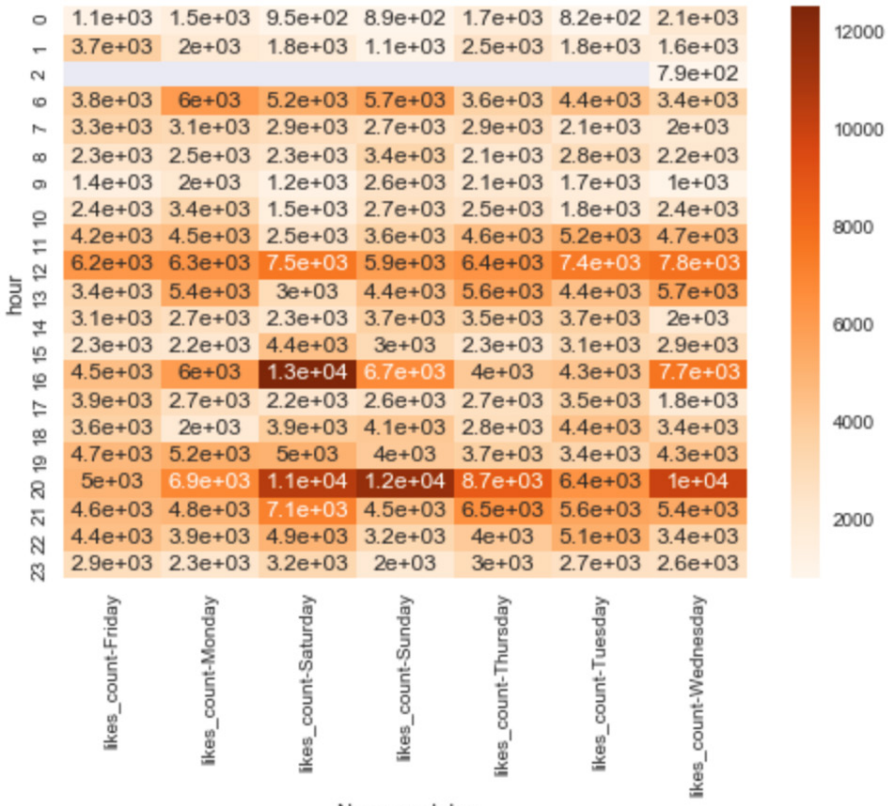


圖 9-51 熱點圖查看關係

匯出檔案：

最後匯出經過整理好的資料

```
92 df.to_excel('fanpage_clean.xlsx', index=False)
```

小結

我們在本節中示範了 Python 清理資料的方法以及使用圖表呈現一些統計性分析，當然各位讀者也可以針對各自有興趣的欄位畫圖查看有無有趣的發現，我們在這邊只列出幾張統計性圖表而已。以上述的分析為例，可以發現常常發文的小編顯然按讚的迴響並不高，反而只要淡水阿編一發文就可以收到很大的按讚數量，當然這也與發文次數的多寡有關係，不過藉由這種分析就可以很容易的查看不同變量間的關係。此外從時間與按讚數量的關係可以發現在中午十二點、下午四點以及晚上八點能收到最大的按讚數量，推測原因可能是中午午休時間、快下班的時間以及吃飽飯後是最多人會打開 FB 查看訊息的時間點，因此藉由這些簡單的統計圖表，其實就可以帶給東森新聞粉專的小編很多不同的資訊，利用這些資訊來調整發文的方式等等。甚至更深入一點可以針對留言與分享數量進行分析，或是發文的議題為何等等，這些部分就留給讀者繼續思考囉！

9-4 資料工程：文字探勘 Text mining

本節將會為各位讀者介紹什麼是文字探勘以及如何使用 Python 進行一些簡易的實作。文字探勘簡而言之就是針對文字的資料進行處理與分析，而我們使用的資料同樣是上一章節的東森新聞粉專資料；接著將從文字的斷字斷詞開始帶領實作，同時帶入兩款處理文字常見的演算法 (TFIDF, Word2Vec)，以下將會一一詳細介紹。

什麼是文字探勘

文字探勘其實是自然語言處理 : (Natural Language Process, NLP) 的其中一項範疇。自然語言處理的研究歷史相當久遠，就維基百科顯示最早可以追溯到 1950 年代，其所需具備的領域知識更是廣闊，包含計算機科學、統計學、人工智慧、語言學等等。不過現階段自然語言處理的應用其實已經融合在我們的日常生活當中囉！像是我們常常會使用的 Google 翻譯、手機的輸入法選字等等 (圖 9-52)

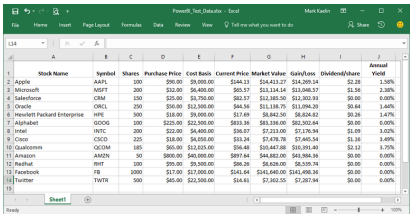
圖 9-52 Google 翻譯、手機的輸入法

1. 文本朗讀（Text to speech）/語音合成（Speech synthesis）
2. 語音識別（Speech recognition）
3. 自動分詞（word segmentation）
4. 詞性標註（Part-of-speech tagging）
5. 句法分析（Parsing）
6. 自然語言生成（Natural language generation）
7. 文本分類（Text categorization）
8. 信息檢索（Information retrieval）
9. 信息抽取（Information extraction）
10. 文字校對（Text-proofing）
11. 問答系統（Question answering）
12. 機器翻譯（Machine translation）
13. 自動摘要（Automatic summarization）
14. 文字蘊涵（Textual entailment）

圖 9-53 自然語言處理應用範疇

自然語言處理的應用範疇就像圖 (圖 9-53) 非常廣闊 (參閱維基百科 https://zh.wikipedia.org/wiki/ 自然語言處理)，而我們今天所說的文字探勘 (text mining) 也被稱為文本挖掘、文字採礦、智慧型分析等，其為自然語言處理的其中一部分領域。學術上的表達也就是從非結構化的文字資訊中，萃取出重要的資訊或知識；若白話一點說明則是試著從大量的文章中找出隱含的寶藏、有用的資訊。

- 結構化資料
  --> Excel, csv, json 檔...
- 非結構化資料
  --> 文字檔，圖片檔，影音檔...



Stock Name	Symbol	Shares	Purchase Price	Cost Basis	Current Price	Market Value	Gain/Loss	Dividend/share	Annual Yield
Apple	AAPL	100	$90.00	$9,000.00	$144.13	$14,413.27	$14,209.14	$2.28	1.58%
Microsoft	MSFT	200	$32.00	$6,400.00	$65.57	$13,114.14	$13,048.57	$1.56	2.38%
Salesforce	CRM	150	$25.00	$3,750.00	$82.57	$12,385.50	$12,302.93	$0.00	0.00%
Oracle	ORCL	250	$50.00	$12,500.00	$44.56	$11,138.75	$11,094.20	$0.64	1.44%
Hewlett Packard Enterprise	HPE	500	$18.00	$9,000.00	$17.69	$8,842.50	$8,824.82	$0.26	1.47%
Alphabet	GOOG	100	$225.00	$22,500.00	$833.36	$83,336.00	$82,502.64	$0.00	0.00%
Intel	INTC	200	$22.00	$4,400.00	$36.07	$7,213.00	$7,176.94	$1.09	3.02%
Cisco	CSCO	225	$18.00	$4,050.00	$33.24	$7,478.78	$7,445.54	$1.16	3.49%
Qualcomm	QCOM	185	$65.00	$12,025.00	$56.48	$10,447.80	$10,391.40	$2.12	3.75%
Amazon	AMZN	30	$800.00	$40,000.00	$897.64	$44,882.00	$44,984.36	$0.00	0.00%
Redhat	RHT	100	$95.00	$9,500.00	$86.28	$8,628.00	$8,339.74	$0.00	0.00%
Facebook	FB	1000	$17.00	$17,000.00	$141.64	$141,640.00	$141,498.36	$0.00	0.00%
Twitter	TWTR	500	$45.00	$22,500.00	$16.61	$7,302.55	$7,287.94	$0.00	0.00%

圖 9-54 結構化和非結構化資料

文字探勘進行的方式

現在我們已經稍微理解什麼是文字探勘了，接下來我們來談談究竟電腦 (機器) 是怎麼閱讀人類的文字呢？首先第一步要先理解的是文字的斷字斷詞，處理文字前一定要先能分辨一個句子或一篇文章中的所有單字

以英文為例，單字與單字中間有空格分開，只要透過程式將空白處分別切開，即可取得單獨的單字。那中文的處理方式呢？現在我們有以下例句，他應該是由四個不同的單詞所組成的，但該怎麼將他們分開？

例 : 我來自台灣東吳大學

我 / 來自 / 台灣 / 東吳大學

解決方式就是我們必須擁有一份中文字詞的字典，而這份字典裡需要有中文單字、權重與詞性，若權重越高代表優先切開的字詞。為什麼會需要有權重呢？我們來看以下例子：

例 : XX 是開發中國家

1. XX / 是 / 開發中國家
2. XX / 是 / 開發中 / 國家
3. XX / 是 / 開發 / 中國 / 家

開發中國家這五個字詞可以是單純的一個字，也可以是由開發中、國家兩個字所組成，甚至可以是開發、中國與家三個字所組成，這就是中文在斷字斷詞中最困難的地方，因此我們需要有權重的機制幫助我們決定優先切開的字詞為何。這份字典的範例如圖 (圖 9-55) 所示：

```
製造場 3 n
製造家 26 n
製造局 3 n
製造廠 194 n
製造廠商 3 n
製造懸念 3 n
製造業 847 n
製造業者 3 n
製造機 2 n
製造矛盾 3 l
製造糾紛 3 z
製造者 71 n
製造術 3 n
製造費用 3 n
```

圖 9-55 字典範例

當成功斷完中文的字詞，事情並沒有這麼簡單就結束了，還必須要去除停止詞 (stopwords) 才行。什麼是停止詞呢？也就是在中文當中沒有意義的字詞，例如我、你、它、當、的等等，以「這堂課是文字探勘」為例，正確的切出字詞應該是 (這堂、課、文字探勘) 才對。因此我們同樣需要一份停止詞的字典來告訴我們什麼樣的詞是沒有意義的 (圖 9-56)

```
可
可以
可是
可能
可見
各
各個
各人
各位
各地
各種
各級
各自
合理
```

圖 9-56 停止詞字典

這些字典都已經有人整理過並公開在網路上開放下載，當然大家整理的字典都會有些許不同，甚至有些專有名詞並不會被收錄在這種通用型的字典裡面，因此當今天如果我們要處理的文字是醫學期刊的文章，那字典就需要另外再找，甚至是自己建立了！

文字探勘套件介紹

NLTK 套件

```
01 import nltk
```

nltk 是 Python 非常大的自然語言處理套件庫，包含的套件非常多，若全部下載將會非常花時間，因此我們個別下載需要用的就好。

```
02 nltk.download()
```

跳出新的下載視窗並點選 All Packages

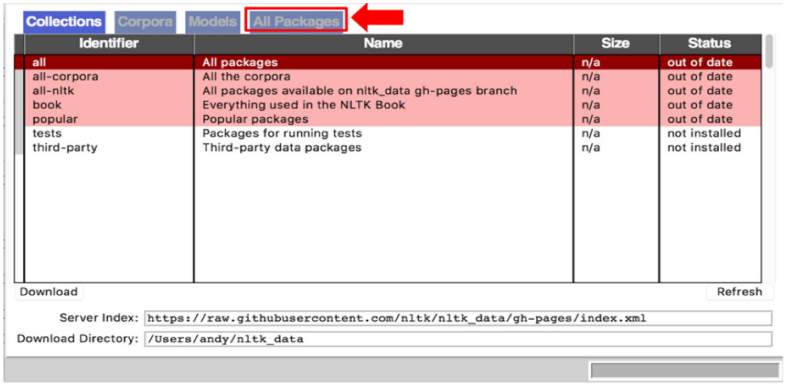


圖 9-57 點選 All Packages

往下滑找到 stopwords，點選左下角 Downloads 即下載完成。

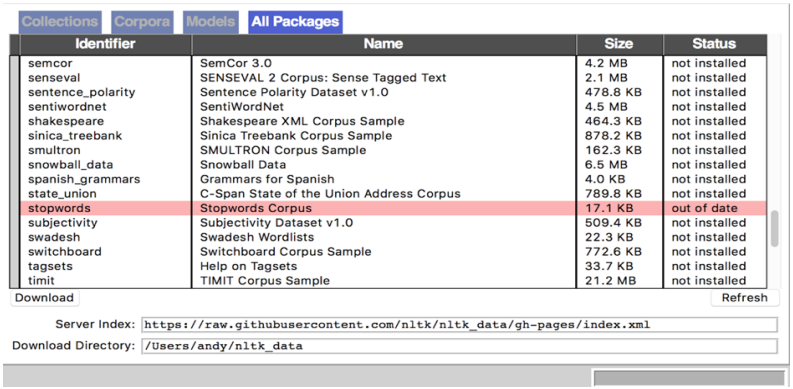


圖 9-58 找到 stopwords

Jieba 套件介紹

Jieba 是 Python 中處理中文斷字斷詞的套件，其他還有像是 Stanford CoreNLP 以及中研院的 CKIP 等等，在這邊選用 Jieba 是因為其安裝方便快速且使用起來非常簡單，那我們就直接來看程式吧！

匯入 Jieba 套件以及兩份下載好的字典

```
import jieba
from nltk.corpus import stopwords
jieba.set_dictionary('dict.txt.big.txt')
stop=stopwords.words('stop_words2.txt')
```

在 Jieba 當中有三種斷字斷詞的模式

全模式：會把所有認為可能是單字的詞都切出來。

```
word = jieba.cut('我來到台灣台北東吳大學', cut_all=True)
print('/'.join(word))
```

執行結果：

```
我 / 來到 / 台 / 灣 / 台北 / 北東 / 東吳 / 東吳大學 / 大學
```

精確模式：只會切出最有可能在這個句子中的單字。

```
word = jieba.cut('我來到台灣台北東吳大學', cut_all=False)
print('/'.join(word))
```

執行結果：

```
我 / 來到 / 台灣 / 台北 / 東吳大學
```

PS：若無設定 cut_all 參數，默認為精確模式。

搜尋模式：切出有可能會被搜尋的字詞。

```
word = jieba.cut_for_search('我來到台灣台北東吳大學')
print('/'.join(word))
```

執行結果：

```
我 / 來到 / 台灣 / 台北 / 東吳 / 大學 / 東吳大學
```

各位可以試著到網路上找一篇長篇文章，使用精確模式看切出來的字詞效果如何。就此篇文章而言可以發現有些棒球的專業術語並未成功切出。

```
13 word = jieba.cut_for_search("""目前暫居龍頭的中信兄弟今
14 天在主場迎戰富邦悍將，1 局下黃衫軍精神領袖彭政閔就擊出 3 分砲，
15 助隊在開賽就攻下 3 比 0 領先。8 月 6 日是洽洽的 40 歲生日，在年紀
16 越來越大的情況下，近年不斷傳出他可能從球員身分退役的消息，雖已
17 接近不惑之年，彭政閔本季目前為止打擊率仍有 3 成 76，還看得到年
18 輕時期「4 割男」的影子。彭政閔本季至今天賽前 229 打數擊出 72 支
19 安打包含 4 發全壘打、貢獻 36 打點、打擊率 3 成 76，上回洽洽開轟是
20 6 月 14 日面對富邦悍將投手張耿豪所擊出。今天同樣是面對富邦，苦
21 主則換成前隊友五鐸 (Bryan Woodall)。( 中時電子報 )""")
22 print('/'.join(word))
```

執行結果：

```
目前 / 暫居 / 龍頭 / 的 / 中信 / 兄弟 / 今天 / 在 / 主場 / 迎戰 / 富邦 / 悍將 / ，
/1/ 局下 / 黃 / 衫 / 軍 / 精神領袖 / 彭政閔 / 就 / 擊出 /3/ 分 / 砲 / ，/ 助隊 / 在 / 開賽
/ 就 / 攻下 /3/ 比 /0/ 領先 / 。/8/ 月 /6/ 日 / 是 / 洽洽 / 的 /40/ 歲 / 生日 / ，/ 在 /
年紀 / 越來越 / 大 / 的 / 情況 / 下 / ，/ 近年 / 不斷 / 傳出 / 他 / 可能 / 從 / 球員 / 身
分 / 退役 / 的 / 消息 / ，/ 雖 / 已 / 接近 / 不惑之年 / ，/ 彭政閔 / 本季 / 目前 / 為止 /
打擊率 / 仍 / 有 /3/ 成 /76/ ，/ 還看 / 得到 / 年輕 / 時期 /「/4/ 割 / 男 /」/ 的 / 影子 / 。
/ 彭政閔 / 本季 / 至 / 今天 / 賽前 /229/ 打數 / 擊出 /72/ 支安打 / 包含 /4/ 發 / 全壘打 / 、
/ 貢獻 /36/ 打點 / 、/ 打擊率 /3/ 成 /76/ ，/ 上 / 回洽 / 洽開 / 轟 / 是 /6/ 月 /14/ 日 /
面對 / 富邦 / 悍將 / 投手 / 張耿豪 / 所 / 擊出 / 。/ 今天 / 同樣 / 是 / 面對 / 富邦 / ，/
苦主 / 則 / 換成 / 前 / 隊友 / 五鐸 /(/Bryan/ /Woodall/)/。/(/ 中時 / 電子報 /)
```

加入自定義字典

Jieba 除了有內建辭庫外，也可以自行加入沒有在字典的字詞，也就是有不同的語料庫。因此我們打開記事本，輸入剛剛文章中沒有切好的詞 (圖 9-60)。

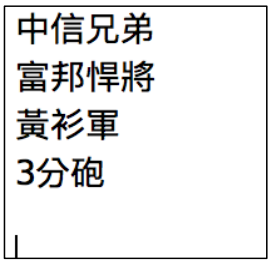


圖 9-60 沒有切好的字

接著重新載入自定義的字典

```
23  jieba.load_userdict('addword.txt')
```

重跑一次剛剛的程式，就可以發現正確切開了新加入的字詞。

```
24  word = jieba.cut_for_search("""目前暫居龍頭的中信兄弟今
25  天在主場迎戰富邦悍將，1 局下黃衫軍精神領袖彭政閔就擊出 3 分砲，
26  助隊在開賽就攻下 3 比 0 領先。8 月 6 日是洽洽的 40 歲生日，在年紀
27  越來越大的情況下，近年不斷傳出他可能從球員身分退役的消息，雖已
28  接近不惑之年，彭政閔本季目前為止打擊率仍有 3 成 76，還看得到年輕
29  時期「4 割男」的影子。彭政閔本季至今天賽前 229 打數擊出 72 支安
30  打包含 4 發全壘打、貢獻 36 打點、打擊率 3 成 76，上回洽洽開轟是 6
31  月 14 日面對富邦悍將投手張耿豪所擊出。今天同樣是面對富邦，苦主
32  則換成前隊友五鐸 (Bryan Woodall)。( 中時電子報 )""")
33  print('/'.join(word))
```

執行結果：

目前 / 暫居 / 龍頭 / 的 / 中信兄弟 / 今天 / 在 / 主場 / 迎戰 / 富邦悍將 / ，/1/ 局下 / 黃衫軍 / 精神領袖 / 彭政閔 / 就 / 擊出 /3 分砲 / ，/ 助隊 / 在 / 開賽 / 就 / 攻下 /3/ 比 /0/ 領先 / 。/8/ 月 /6/ 日 / 是 / 洽洽 / 的 /40/ 歲 / 生日 / ，/ 在 / 年紀 / 越來越 / 大 / 的 / 情況 / 下 / ，/ 近年 / 不斷 / 傳出 / 他 / 可能 / 從 / 球員 / 身分 / 退役 / 的 / 消息 / ，/ 雖 / 已 / 接近 / 不惑之年 / ，/ 彭政閔 / 本季 / 目前 / 為止 / 打擊率 / 仍 / 有 /3/ 成 /76/ ，/ 還看 / 得到 / 年輕 / 時期 / 「/4/ 割 / 男 / 」/ 的 / 影子 / 。/ 彭政閔 / 本季 / 至 / 今天 / 賽前 /229/ 打數 / 擊出 /72/ 支安打 / 包含 /4/ 發 / 全壘打 / 、/ 貢獻 /36/ 打點 / 、/ 打擊率 /3/ 成 /76/ ，/ 上 / 回洽 / 洽開 / 轟 / 是 /6/ 月 /14/ 日 / 面對 / 富邦悍將 / 投手 / 張耿豪 / 所 / 擊出 / 。/ 今天 / 同樣 / 是 / 面對 / 富邦 / ，/ 苦主 / 則 / 換成 / 前 / 隊友 / 五鐸 /(/Bryan/ /Woodall/)/ 。/(/ 中時 / 電子報 /)

去除停止詞

除了將中文字詞切開外，第二步要做的就是去除停止詞。

- 去除前：

```
34  word = jieba.cut('目前暫居龍頭的中信兄弟今天在主場迎戰富邦悍將')
35  print('/'.join(word))
```

執行結果：

目前 / 暫居 / 龍頭 / 的 / 中信兄弟 / 今天 / 在 / 主場 / 迎戰 / 富邦悍將

- 去除後：

```
word = jieba.cut('目前暫居龍頭的中信兄弟今天在主場迎戰富邦悍將')
stoptext = ''
for words in word:
    if words not in stop:
        stoptext += '/'+words
print(stoptext)
```

執行結果：

```
/暫居/龍頭/中信兄弟/主場/迎戰/富邦悍將
```

分析粉絲專頁資料

我們已經介紹完如何處理中文的斷字斷詞以及去除停止詞了，那接下來就來實際跑跑看真實的資料吧！首先匯入 pandas 套件與粉專資料，其中 fanpage_clean.xlsx 是前一節創造的成果。

```
import pandas as pd
df = pd.read_excel('fanpage_clean.xlsx')
df.head()
```

	id	message
0	124616330906800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞 #要打統編：小編真的是快嚇死了...😂😂 影片來源：騰訊視頻 #草食性 #羊
1	124616330906800_1560454417322977	被黑了！李毓芬演唱「大落拍」網友卻意外發現「亮點」 #條紋編：這一段應該是昨天的亮點表演之一吧～ #李毓芬 #落拍 #唱歌
2	124616330906800_1559870414048044	誰說牠呆？心機月月調虎離山 網友讚影帝 #樂無編：最萌心機鬼～(*˘v˘)~❤ 影片來源：秒拍 #萌 #心機 #調虎離山計

圖 9-64 粉專資料

因為後面使用到的 TFIDF 與 Word2Vec 演算法所需要的字詞格式並不同，因此在這邊將匯入的資料做斷字斷詞，分成兩部分 (text, text2)。第一層 for 迴圈處理的是每一篇文章中的 Jieba 斷詞，第二層 for 迴圈處理的是判斷每一個字是否為停止詞，最後將結果存入 jieba_text 的列表當中。

```
jieba_text = []
for index in range(len(df)):
    words = jieba.cut(str(df['context'][index]))
    text, text2 = [], ''
    for word in words:
        if word not in stop:
            text.append(word)
            text2 += ' '+word
    jieba_text.append([text,text2,len(text)])
```

100% 9975/9975 [00:20<00:00, 498.10it/s]

圖 9-65 將匯入的資料做斷字斷詞

結果呈現，放回原本的 Dataframe 裡面。

```
df['jieba_text'] = pd.DataFrame(jieba_text)[0]
df['jieba_text2'] = pd.DataFrame(jieba_text)[1]
df['jieba_count'] = pd.DataFrame(jieba_text)[2]
df.head(3)
```

	id	message	jieba_text	jieba_count	jieba_text2
0	124616330906800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😩😩\n\n影片來源...	[阿娘, 威, 披, 羊皮, 狼, 竟大口, 嚼, 小雞, \n, #, 統編, 小編, 真...	30	阿娘 威 披 羊皮 狼 竟大口 嚼 小雞 \n # 統編 小編 真 的 快 嚇死 ... 😩...
1	124616330906800_1560454417322977	被黑了！李毓芬演唱「大落拍」網友卻意外發現「亮點」\n#條紋編：這一段應該是昨天的亮點表演...	[黑, 李毓芬, 演唱, 「, 大落, 拍, 」, , 網友, 卻, 意外, 發現, 「,...	37	黑 李毓芬 演唱 「 大落 拍 」 網友 卻 意外 發現 「 亮點 」 \n # 條紋...
2	124616330906800_1559870414048044	誰說牠呆？心機月月調虎離山 網友讚影帝\n#樂無編：最萌心機鬼～(*´∀`)~♥\n\n影片...	[誰, 說, 牠, 呆, 心機, 月, 月, 調虎離山, , 網友, 讚, 影帝, \n,...	43	誰 說 牠 呆 心機 月 月 調虎離山 網友 讚 影帝 \n # 樂無編 最萌 心機 ...

圖 9-66 結果呈現

TF-IDF 演算法

TF-IDF

接下來我們將實作兩種處理文字的演算法，首先是 TF-IDF。TF-IDF 想要達成的目標是找出一段文字或一篇文章中有哪些字詞是較重要的。它其實是 TF 和 IDF 的結合，公式如下看起來很複雜，不過我們將其分開來介紹。

Term Frequency (TF)

TF 計算的是一個字詞 t 在一篇文章 d 中所出現的總次數 $tf_{t,d}$，並取 log 降維，但文字出現的越多不代表越重要，有可能是不重要的字詞　如 : 我、你、它等等 (若沒事先去除停止詞的話)。

$$\mathrm{TF} = \begin{cases} 1+\log_{10} tf_{t,d} & \text{if } tf_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

Inverse Document Frequency (IDF)

IDF 計算的是 [總文件數 / 文字出現在文件內的次數]，並取 log。也就是把雖然出現頻率很高但卻是不重要的字詞降低權重。

$$\mathrm{IDF} = \log_{10} \frac{N}{df_t}$$

Term	df_t	IDF (*N*=1000000)
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

圖 9-67 Inverse Document Frequency

所以 TF-IDF 就是將 TF 與 IDF 兩個相乘 (其中一種算法，其實還有很多衍伸型)。

$$\mathrm{TF-IDF} = \mathrm{TF} \times \mathrm{IDF} = (1+\log(tf_{t,d})) \times \log(N / df_t)$$

如果還是看不太懂公式，下圖表達應該較好理解 (圖 9-68)。

Word	出現總次數	出現文件數
ferrari	10422	17←較高的稀有性 (高資訊量)
insurance	10440	3997

圖 9-68 TF-IDF

如果已經稍微理解TF-IDF是什麼了的話，那我們就進入Python的實作吧！

首先我們需要 scikit-learn 這個套件，安裝後匯入其中的 TF-IDF 模組：

```
from sklearn import feature_extraction
from sklearn.feature_extraction.text import TfidfVectorizer
```

接著將文字轉換成矩陣模式，使用到的文字格式是剛剛處理好以空白切開每個字詞的字串格式。

```
vectorizer = TfidfVectorizer()
tfidf = vectorizer.fit_transform(df['jieba_text2'])
```

計算 TF-IDF，我們顯示每篇文章中的前五個關鍵字。

```
words = []
words2 = vectorizer.get_feature_names()
for i in tqdm_notebook(range(len(df['jieba_text2']))):
    print('======= Post'+str(i+1)+' =======')
    temp_array = tfidf[i,:].toarray()
    for l in temp_array:
        print([(words2[x],l[x]) for x in (l*-1).argsort()][:5])
```

執行結果：

```
======= Post1 =======
[('草食性', 0.3886164663330653), ('竟大口', 0.3886164663330653),
('小雞', 0.37205588933099115), ('羊皮', 0.37205588933099115),
('阿娘', 0.31216568557942365)]
======= Post2 =======
[('李毓芬', 0.5393296772291432), ('亮點', 0.47428487993368196),
('大落', 0.28166788829793676), ('昨天', 0.2611485393335714), ('
演唱', 0.2611485393335714)]
======= Post3 =======
[('心機', 0.6875944679814824), ('調虎離山', 0.5748856520127635),
('影帝', 0.25649296680540506), ('最萌', 0.22919815599382748),
('樂無編', 0.2181098701502329)]
```

將執行的結果存到變數 words 裡面。

```
words = []
words2 = vectorizer.get_feature_names()
for i in tqdm_notebook(range(len(df['jieba_text2']))):

    temp_array = tfidf[i,:].toarray()
    for l in temp_array:
        words.append([(words2[x],l[x]) for x in (l*-1).argsort()][:5])
```

轉成 DataFrame 形式。

```
# 每篇文章的前五關鍵詞
df2 = pd.DataFrame(words,columns=['Keyword1','Keyword2',
                    'Keyword3','Keyword4','Keyword5'])
df2.head()
```

與原本的資料合併。

```
df3 = pd.concat([df, df2], axis=1)
df3.head(1)
```

	id	message	jieba_message	Keyword1	Keyword2	Keyword3	
0	124616330906800_1560501197318299	阿娘威！披羊皮的狼？竟大口嚼小雞\n#要打統編：小編真的是快嚇死了...😊😊\n\n影片來源...	/阿娘/威/披/羊皮/狼/竟大口/嚼/小雞/\n/統編/小編/真的/嚇死/\n/\n/\n/...	(草食性, 0.41657326701385844)	(竟大口, 0.41657326701385844)	(羊皮, 0.3988213335189045)	(小雞, 0.398821

圖 9-69 顯示結果

評估 TF-IDF 的結果好壞有一種方法：「若我們只看這五個或是額外延伸至十個關鍵字是否有辦法猜出原本的文章大致內容呢？」就從目前的結果而言，可以發現還是有斷字斷詞並未處理乾淨的狀況發生。處理中文字詞最難的地方往往是在斷字斷詞身上，也有很多研究針對如何讓電腦自行加入自定義的字詞等，或是也可以嘗試用用看別種的中文斷詞方法，也許最後的結果也會不同，這邊就留給有興趣的讀者繼續研究囉！

Word2Vec 演算法

接下來我們來介紹另一種處理文字的演算法 Word2Vec。還記得我們前面有提到究竟電腦 (機器) 是怎麼閱讀人類的文字嗎？關鍵就在於將文字向量化 (Word Embedding) ！電腦終究只看得懂數字符號而已，因此需要有方法將文字資料轉換成數學向量模式，其中一種方式就是透過 Word2Vec 演算法 (Word to Vector)。

Word2Vec 概念

我們可以想像把所有的文字字詞轉換成向量投射到高維度的空間中，在這個高維度的空間中越接近的文字其相似度應該會越高，且類似的文字應該會有相同的距離，如下左圖 (圖 9-70) (MAN 與 WOMAN 的距離等同於 KING 與 QUEEN 的距離)。此外文字的單複數也同樣適用於這個規則，如下右圖 (圖 9-70)(KING 與 QUEEN 的複數應該擁有相同的空間距離。)

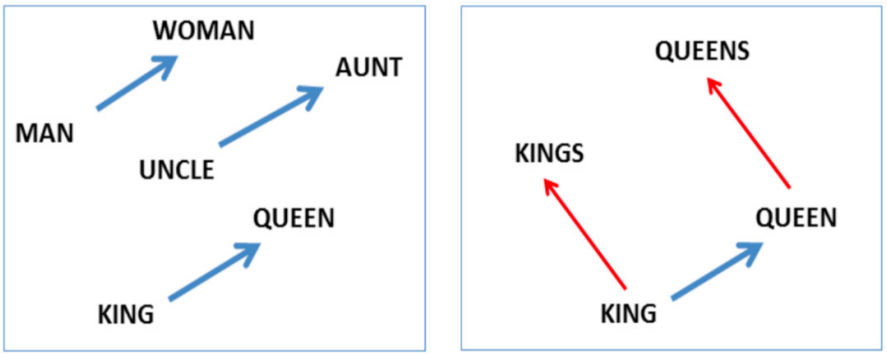


圖 9-70

Word2Vec 在做的就是找尋字與字之間的關聯性，以及讓同概念的詞其距離能夠越接近越好。每一個字詞都是在空間中的一個向量，其維度相對也是屬於高維的向量。又如同下圖的範例 (圖 9-71)，國家之於其首都大致都是同樣的距離，而國家字詞都在左邊，城市之詞都在右邊，Word2Vec 就是在實現這種想法。

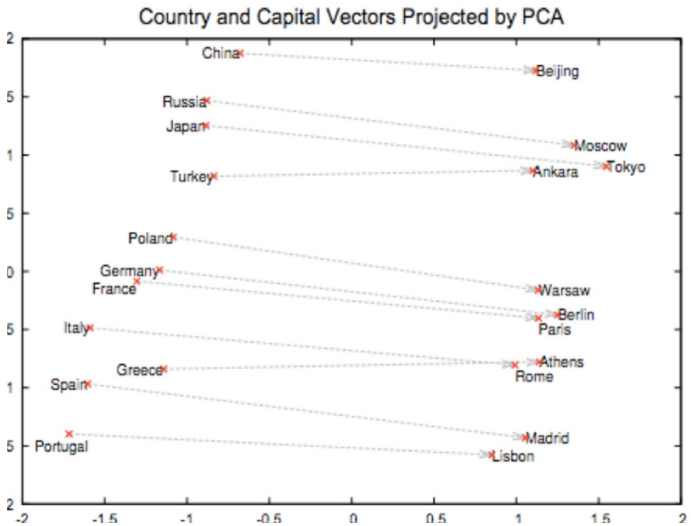


圖 9-71

Word2Vec 計算距離的方法是向量間的餘弦 (cos) 值，也就是夾角角度。

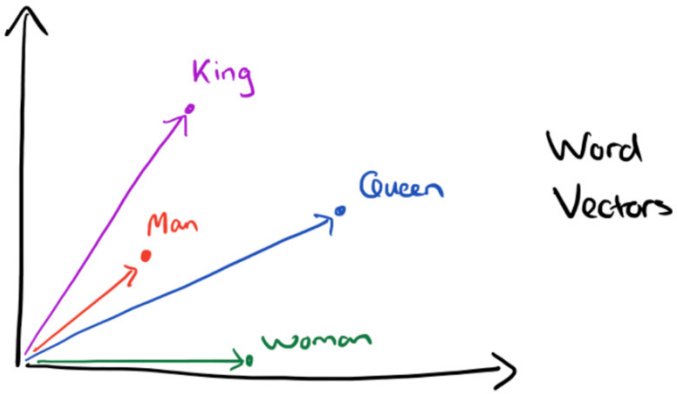


圖 9-72

若對 Word2Vec 有興趣，想要了解其詳細演算過程，可以再從網路上找尋其它教材，已有很多網友整理與介紹，在這邊就不一一詳細講解囉，我們就直接帶領各位進行 Python 的實作吧。

首先，我們會需要 genism 這個套件，安裝完成後匯入 Word2Vec 模組。

```
from gensim.models.word2vec import Word2Vec
```

接著建立 Word2Vec 模型，使用到的文字格式是先前處理好以串列 list 儲存每個字詞的格式，執行完這行，模型就建立完成囉！

```
model = Word2Vec(df['jieba_text'])
```

另外建立一個函式來顯示不同字詞前 10 個相關的字詞有哪些，for 迴圈依序處理每一個要搜尋的字詞，wv.most_similar 則是會判斷最接近此字詞的前 n 個字並存入 similar_words 變數當中，最後以 Dataframe 方式呈現。倘若字詞並未出現在模型當中，則會顯示 not found in Word2Vec model! 字詞

```
def most_similar(w2v_model, words, topn=10):
    similar_df = pd.DataFrame()
    for word in words:
        try:
            similar_words = pd.DataFrame(
            w2v_model.wv.most_similar(word, topn=topn),
            columns=[word, 'cos'])

            similar_df = pd.concat([similar_df, similar_words], axis=1)
        except:
            print(word, 'not found in Word2Vec model!')
    return similar_df
```

查看其結果，以這 9 個字詞來看各自最接近的前 10 字詞有哪些：

```
most_similar(model, ['新聞','體育','娛樂','政治',
'XDD','小編','夜市','明星','女孩'])
```

	新聞	cos	娛樂	cos	政治	cos	XDD	cos	小編	cos	夜市	cos	明星	cos	女孩	cos
0	直播	0.967324	S	0.998950	地方	0.994920	哩厝編	0.994200	好	0.984673	環島	0.999220	技巧	0.999714	噴發	0.998853
1	中心	0.965926	櫻花	0.998900	社會	0.994582	狗狗	0.991577	萌死	0.978975	飯店	0.999144	髮	0.999616	爸爸	0.998728
2	綜合	0.956663	起司	0.998894	T	0.990361	柯基	0.991063	曉	0.977118	高空	0.999091	最美	0.999566	Q	0.998684
3	育樂中心	0.943752	役	0.998769	NCC	0.990037	XDDD	0.990727	想	0.972329	餐廳	0.999069	安娜	0.999490	抱	0.998549
4	愛玩	0.932010	害怕	0.998734	壯壯	0.989895	誰准	0.990455	人	0.971244	團	0.999040	茶	0.999484	洋蔥	0.998500
5	吳念樺	0.928818	飛行	0.998684	禽流感	0.987717	好笑	0.989693	寶寶	0.970022	鍋貼	0.998937	V	0.999439	心	0.998497
6	朱姵慈	0.928699	手工	0.998654	肇逃	0.987193	愛	0.988566	吃	0.963738	驚悚	0.998870	倍	0.999332	床	0.998367
7	台	0.917264	抓到	0.998643	侵權	0.986938	養樂多	0.988533	真的	0.962447	巴黎	0.998865	父	0.999247	有趣	0.998241
8	提醒您	0.909181	性別	0.998492	毒品	0.986920	牠	0.988344	Jay	0.962223	跑車	0.998857	編髮	0.999242	瞬間	0.998178
9	李欣	0.907590	約會	0.998433	巨蛋	0.986402	Baby	0.988283	長	0.959666	板車	0.998834	馬尾	0.999203	挑戰	0.998119

圖 9-73 最接近的前 10 字詞

我們在建立模型時可以修改幾個參數

1. size 代表詞向量的維度大小

2. iter 代表訓練的次數多寡

修改後會發現跟原先跑出的結果也不盡相同

```
model_d250 = Word2Vec(df['jieba_text'], size=250, iter=10)
most_similar(model_d250, ['新聞','體育','娛樂','政治',
'XDD','小編','夜市','明星','女孩'])
```

```
體育 not found in Word2Vec model!
```

	新聞	cos	娛樂	cos	政治	cos	XDD	cos	小編	cos	夜市	cos	明星	cos	女孩	cos
0	Nick	0.908591	Keigo	0.995358	監獄	0.978106	愛	0.963409	好	0.927528	逢甲	0.979087	原味	0.994482	掛在	0.968459
1	節目	0.908183	Youtube	0.991555	香港	0.972059	牠	0.961356	想	0.902362	溫泉	0.971208	跟風	0.994479	汪汪	0.964284
2	直播	0.898731	邦	0.990808	美國	0.972043	可愛	0.958527	感覺	0.901491	步道	0.968835	上身	0.994263	卡哇伊	0.956831
3	愛玩	0.891666	紙牌	0.990400	青瓦台	0.971166	xD	0.957824	真的	0.870075	牛肉	0.967882	出奇	0.994263	這招	0.956034
4	房趣	0.887777	ㄑ	0.989025	谷	0.967853	XDDDD	0.948542	人	0.869911	殺手	0.967501	出招	0.993517	主人	0.955768
5	調查	0.821492	媒合	0.988894	汙	0.967761	嘗試	0.943750	媽媽	0.849948	沖繩	0.966578	境界	0.993509	表情	0.955542
6	中心	0.813243	ㄟ	0.988487	中部	0.966782	der	0.940438	厲害	0.846849	品牌	0.966445	完美	0.993505	挑戰	0.953859
7	挑	0.810430	Mihara	0.988282	政府	0.966702	嘴編	0.937272	耶	0.843780	駁火	0.966388	放閃	0.993127	小	0.953715
8	車	0.797249	咖哩	0.988211	跨界	0.964899	脆嘀叔	0.933207	好吃	0.841185	商圈	0.965920	材料	0.993045	隻	0.952203
9	馨	0.788445	錕	0.988182	車站	0.964738	萌死	0.930417	統編	0.840771	麻辣鍋	0.965484	電眼	0.992904	肚臍	0.951553

圖 9-74 修改後結果

Word2Vec 屬於神經網路的一環，這類型的演算法都很仰賴大量的資料進行訓練，越多的資料對於其結果會越準確，不過同時也需要有好的設備才容易進行訓練，此外也同樣需要準確的斷字斷詞結果才行。

儲存模型

當資料量很大時避免每次都要重複訓練模型，可以先儲存起來

```
52 model.save('word2vec.model')
```

當下次還想使用時可以直接匯入模型，再進行其他的分析

```
01 from gensim.models.word2vec import Word2Vec
02 model = Word2Vec.load('word2vec.model')
```


小結

我們在本節中介紹了文字探勘的一些基本概念，以及處理中文斷字斷詞的流程，另外也介紹了兩個與文字相關的演算法。從兩者最後的結果當中我們可以得知在進行這類型的演算時有幾個重要的先決條件，其一為需要擁有準確的斷字斷詞前處理，其二為大量的資料對於訓練的結果影響也很巨大，這一點與下一節的機器學習相同。不過在這邊我們希望各位讀者能夠吸收到的是如何使用 Python 程式來實現這些分析方法，當要套用回自己的資料時是可以輕易跑出結果的，同時也希望各位讀者能夠有能力解讀跑出來的結果所代表的意義。有時候我們並非是高深的統計學家，甚至可能也沒學過微積分和線性代數等等課程，因此若要理解演算法背後的運算方式是有些難度的。但若我們至少能夠理解演算法的概念，以及修改的參數對於結果會造成哪些影響，最重要的是能夠解釋運算結果所呈現的意義，就大部分的應用層級而言已經是相當足夠的了。若讀者並非是數學背景出生的，建議可以朝這個方向前進呦！

9-5 進階分析：資料建模 Data modeling

本節將為各位讀者介紹什麼是機器學習、機器學習有哪些演算法、如何透過 Python 建立機器學習模型進行預測、以及可以透過哪些指標來評估模型的好壞。我想近幾年各位對於大數據這一詞一定不陌生，不論是從哪個領域當中我們都可以發現大數據的蹤跡，像是商業行為、醫療、交通、犯罪、工廠等等。大數據的背後與本章節所提及的機器學習息息相關，綜觀而言，機器學習其實是 2、30 年前就已經出現的技術，近年來電腦運算的速度不斷進步、成本也逐漸下降，不論是企業、學校、乃至於個人現階段都有能力使用具有高速運算能力的電子產品，同時使用機器學習技術的門檻也逐漸降低，我們並不需要是資訊背景、統計背景的專家才可以使用，只要有能力撰寫 Python、R 等程式語言都可以輕易實現，不過當然若能理解機器學習背後的運算邏輯會更有幫助。現在我們就直接進入我們的最後一節：資料建模吧！

機器學習的出現

從圖 9-75 中我們可以發現早在 1950 年代就開始出現人工智慧 AI 的想法了，那時候因為戰爭的影響因此各國都很仰賴電腦能幫助他們進行作戰，而到了 1980 年代逐漸出現了所謂的專家系統，也就是在那陣子開始出現機器學習的概念，希望電腦可以習得人類的相關知識來簡化人類的負擔，而到了 2012 年因為一場電腦視覺競賽的結果造就了現階段最夯的深度學習出現。

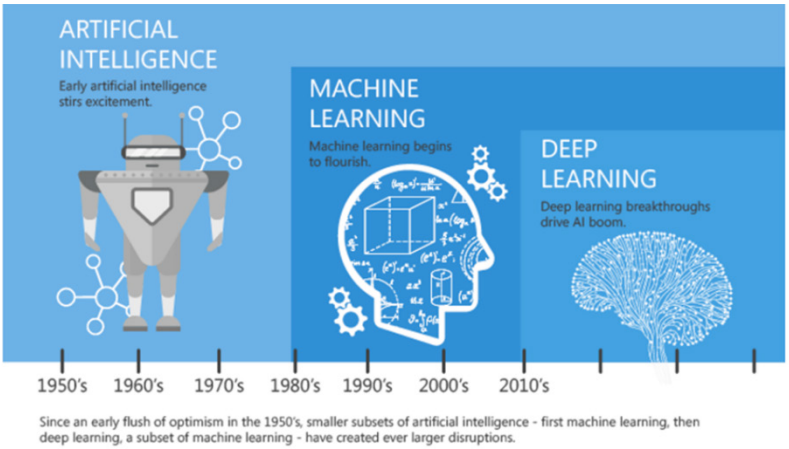


圖 9-75 機器學習發展 (圖片來源 : Kapil Tandon (2016) AI & Machine Learning: The evolution, differences and connections)

機器學習的概念

其實我們不用把機器學習想的太複雜，簡單透過一句話表達就是從過往的資料和經驗中讓電腦學習並找到其中運行的規則。這些過往的資料我們稱之為「訓練資料 (training data)」，訓練資料中會包含兩部分「特徵 (feature)」和「目標 (label)」。如圖 (圖 9-76) 我們想要預測瓶中是哪一種酒，我們可以使用酒的顏色與酒精濃度當作特徵，建立出模型後再進行預測，而酒的種類就是所謂的目標。

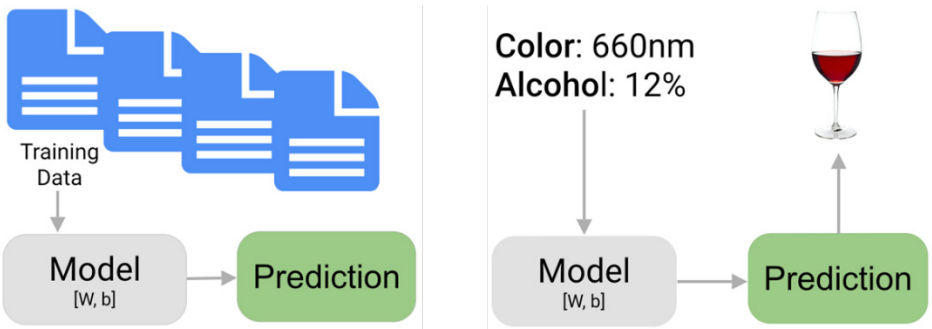


圖 9-76 機器學習的概念

機器學習的分類

機器學習可以分為監督式學習與非監督式學習。監督式學習包含所謂的分類模型與迴歸模型，這類的模型在訓練時能夠事先得知正確的答案為何；例如我們想要建立預測酒的品種模型，我們會事先知道每一種酒的顏色與酒精濃度，也就是在有已知答案的情況下讓電腦學習其中的規則為何。非監督式學習屬於在沒有正確的答案下所建立的模型，稱之為分群模型，例如我們現在有上百筆電影的資料，我們想將他們分類但卻又不知道該怎麼進行分類，因此透過分群演算法電腦會自動幫我們分成預先設定好的群體數量 (如 : 分成四類)，分完後再由人去看這些群體所代表的個別意義為何，可能其中一群都是動作片，其他群可能是恐怖片、動畫片等等。在後續的教學我們會先以監督式學習當中的迴歸演算法當作示範。

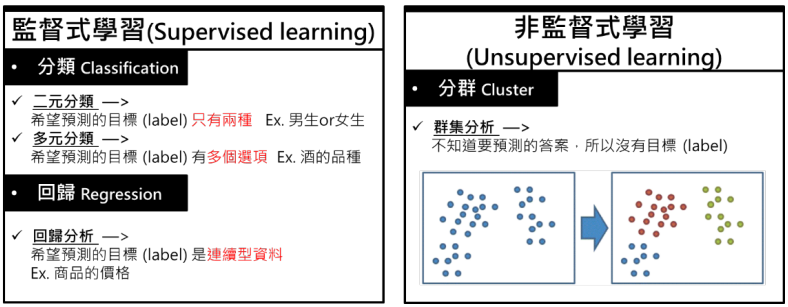


圖 9-77 監督式與非監督式學習

機器學習架構圖

每一種機器學習方法都分別對應有不同的演算模型，像是分類模型有 SVM、NB、KNN 模型等，迴歸模型有 LR、SVR、GPR 等，分群模型則有 K-Means、Hierarchical 等。每一種模型都有其各自的優缺點以及適合使用的資料，至於每一種資料適合哪一種演算法除了需事先了解背後的演算邏輯外，還有各自的建模經驗所帶來了經驗法則，因此各位讀者若對於這塊有興趣的話，不妨多參考網路上所分享的分析經驗，以及試著多使用看看不同的資料實際建立模型會更有所收穫呦。

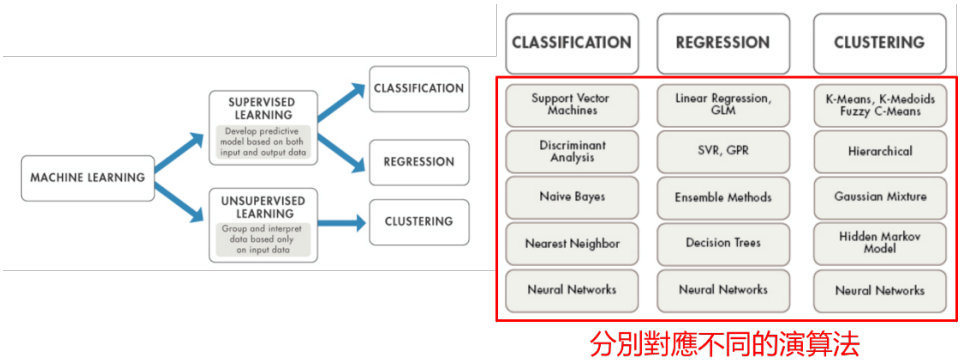


圖 9-78 不同的演算法

機器學習的流程

前面提及了我們在建立模型時需要擁有訓練資料 (training data)，而此資料裡面又會分為特徵及目標。在建立模型時通常只會使用資料的 70% 進行訓練，剩下的 30% 則會當作測試資料 (testing data)，也就是用來驗證模型的好壞。

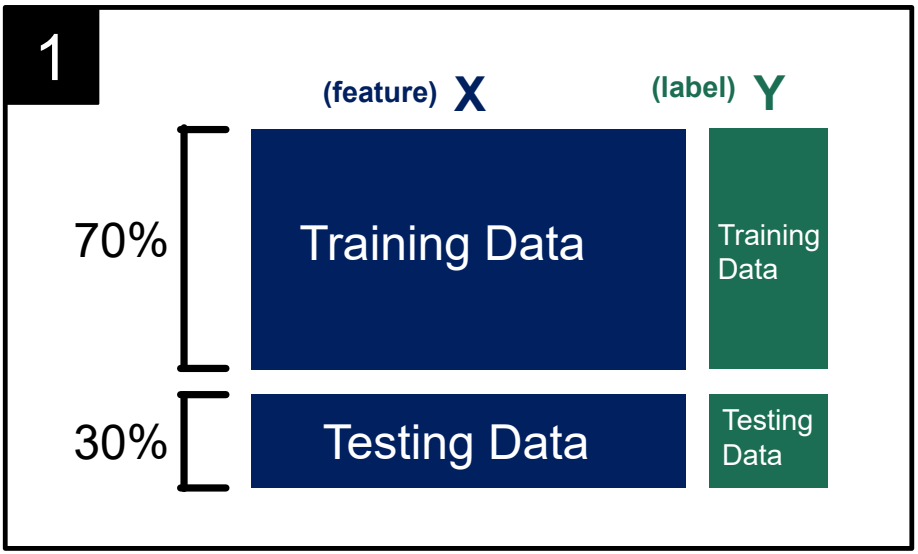


圖 9-79 70% 訓練，30% 測試

將 70% 的訓練資料用於建立機器學習模型：

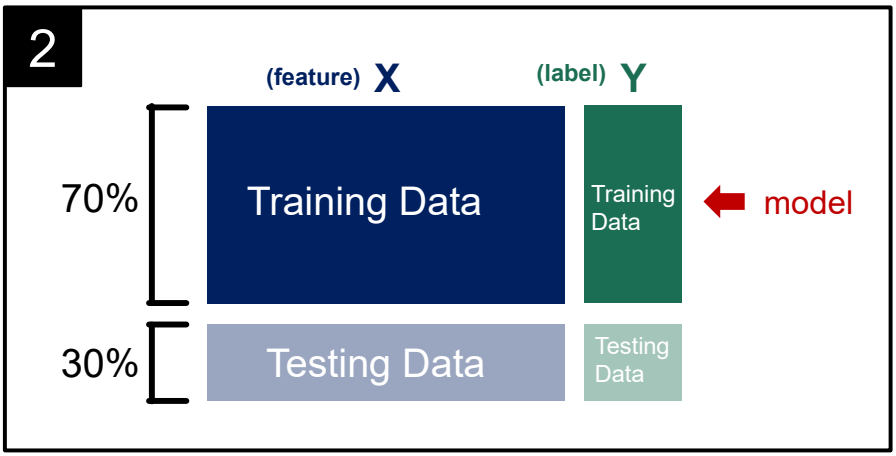


圖 9-80 70% 的訓練資料

當我們訓練好模型，接著就會拿 30% 的測試資料預測看看會得到什麼樣的結果：

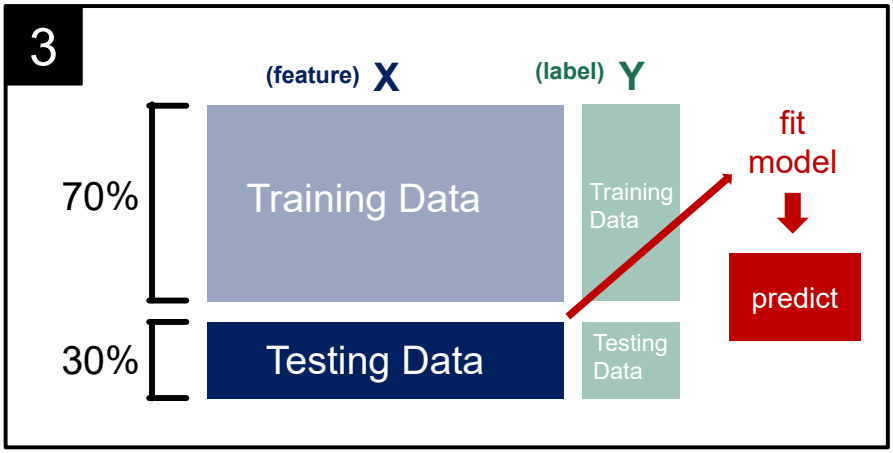


圖 9-81 30% 的測試資料

最後預測完成就會將具有真實答案的測試資料與模型所預測出的結果進行比對，來看看模型的效果是否具有準確預測能力。

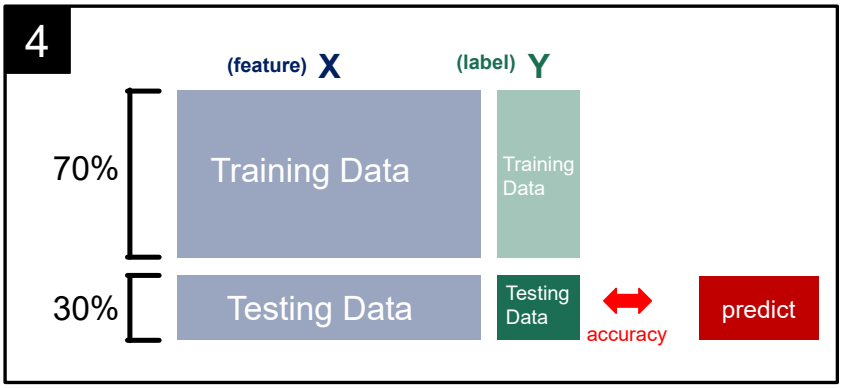


圖 9-82 預測的結果進行比對

以上就是建立機器學習的大致流程，現在已經學會了那我們就開始使用 Python 來進行實踐吧！不過在建立模型前還是要先經過資料前處理的步驟才行。

資料前處理

匯入套件與資料

首先匯入 pandas 套件以及 9-3 節經過整理的粉專資料。

```
import pandas as pd
df = pd.read_excel('fanpage_clean.xlsx')
```

查看資料：

```
df.head()
```

	id	message
0	124616330906800_1560501197318299	阿娘威!披羊皮的狼?竟大口嚼小雞\n#要打統編:小編真的是快嚇死了... \n\n影片來源..
1	124616330906800_1560454417322977	被黑了!李毓芬演唱「大落拍」網友卻意外發現「亮點」\n#條紋編:這一段應該是昨天的亮點表演..
2	124616330906800_1559870414048044	誰說牠呆?心機月月調虎離山網友讚影帝\n#124616330906800_1559870414048044樂無編:最萌心機鬼~(*^Y)~y\n\n影片..

圖 9-83 查看資料 head()

將資料中的特徵 X 與目標 Y 分開

我們將會使用分享數、留言數、小編、發文時間以及發文星期來試著預測看看文章的按讚數量多寡，因此前面的資料就會是特徵 X，而按讚數量則是目標 Y。

```
04 X = df[['shares','comments','curator','hour','weekday']]
05 y = df['likes_count']
```

檢查資料

接著檢查特徵資料當中有無遺失值，發現小編資料並非每筆文章當中都有。

```
06 X.info()
```

```
<class 'pandas.core. frame. DataFrame'>
Range Index: 9975 entries, 0 to 9974
Data columns (total 5 columns):
shares 9975 non-null int64
comments 9975 non-null int64
curator 7232 non-null object
hour 9975 non-null int64
weekday 9975 non-null object
dtypes: int64(3), object(2)
memory usage: 389.7+ KB
```

圖 9-84 檢查資料

處理遺失值

因此將具有空值的小編填入字串未知：

```
07 X['curator'] = X['curator'].fillna('未知')
```

再次檢查，發現已成功填入空值：

```
08 X.info()
```

```
<class 'pandas.core. frame. DataFrame'>
Range Index: 9975 entries, 0 to 9974
Data columns (total 5 columns):
shares 9975 non-null int64
comments 9975 non-null int64
curator 9975 non-null object
hour 9975 non-null int64
weekday 9975 non-null object
dtypes: int64(3), object(2)
memory usage: 389.7+ KB
```

圖 9-85 處理遺失值

資料標準化

接著將連續型的資料進行標準化，用意為將資料的範圍壓縮在一定的範圍內，可以減少資料上運算的複雜度。我們的特徵中屬於連續型資料的有分享數量以及留言數量兩個欄位。

```
from sklearn import preprocessing
standard = preprocessing.StandardScaler()
X[['shares','comments']] =
    standard.fit_transform(X[['shares','comments']])
```

其餘的欄位是類別型的資料，我們將其轉換為 Dummy 變數，Dummy 變數也就是我們將不同的小編名稱轉換為電腦看得懂的 0,1 數字，否則電腦將無法讀懂 B 編、淡水阿編這種特殊字眼。

```
X_1 = pd.get_dummies(X)
```

轉換後的結果

```
X_1.head()
```

	shares	comments	hour	curator_BG編	Curator_B編	curator_M編	curator_七條編	curator_人間四月編	curator_什麼編	curator_傻編	...	curator_高光編
0	0.241213	0.001331	11	0	0	0	0	0	0	0	...	0
1	-0.212199	-0.206505	11	0	0	0	0	0	0	0	...	0
2	0.677548	0.564335	10	0	0	0	0	0	0	0	...	0
3	-0.224154	-0.215934	10	0	0	0	0	0	0	0	...	0
4	-0.212199	-0.208076	10	0	0	0	0	0	0	0	...	0

圖 9-86 轉換後的結果

Training & Testing data

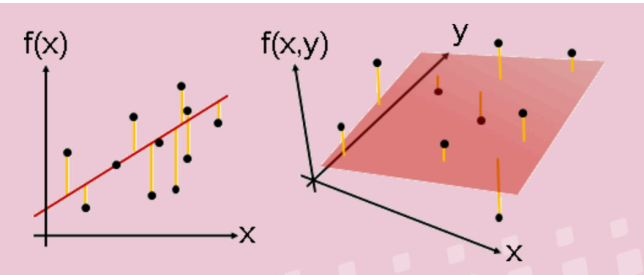
將資料分成 training data 以及 testing data，不一定要 70% 比 30% 的比例切，也有些人分成 80% 與 20% 等。透過 test_size 參數可以協助設定切分的比例。

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X_1, y, test_size = 0.3, random_state = 2)
```

預測模型：Linear Regression

線性迴歸介紹

我們來介紹第一個要使用到的機器學習模型 — 線性迴歸。相信各位在國高中時應該都有學過迴歸函數，也就是找一個函數，盡量能夠使其符合手邊的一堆數據。當資料是屬於二維度的，此迴歸函式就是一直線 (y = ax+b)，當資料是屬於三維度的，則迴歸函數會變成是平面的空間，資料維度越大，迴歸的函式就會越複雜。



- 二維：直線
- 三維：平面
- 多維：超平面

圖 9-87 線性迴歸

簡單線性迴歸公式如下，也就是我們熟悉的 y = ax + b，希望每一個資料點距離此條線的誤差加總能夠最小化。

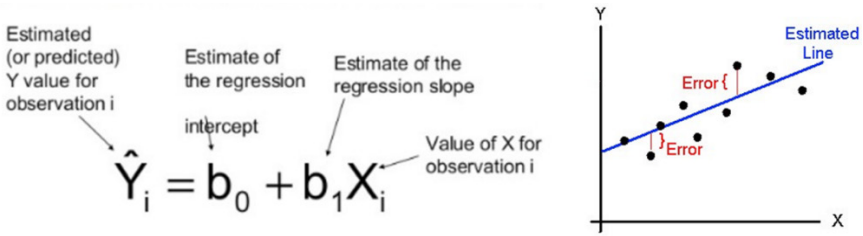


圖 9-88 線性迴歸公式

複迴歸：當我們的資料特徵值不只一項時，就會使用到複迴歸，公式中的每一個 X 都代表一個資料特徵，例如酒精顏色、酒精濃度等，每一個 X 都會搭配一個權重 β(Beta) 來代表此特徵的重要性高低。而我們所要建立的也就是複

迴歸公式，接下來就開始進入 Python 程式吧。

$$Y_i = \beta_0 + \beta_1 X_{i1} + \beta_2 X_{i2} + \varepsilon_i$$

圖 9-89 複迴歸

匯入迴歸模型套件

從 sklearn 套件中匯入線性迴歸模型，並且將此模型存入 lm 變數當中。

```
18 from sklearn.linear_model import LinearRegression
19 lm = LinearRegression()
```

接著進行模型訓練，使用到的是前面所切分的 70% 訓練資料

```
20 lm.fit(X_train.values, y_train.values)
```

執行結果：

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1
, normalize=False)
```

有沒有發現建立模型居然只要一行程式就建立完畢了！真的相當簡單。

查看模型

迴歸模型的截距項，也就是公式中的 β_0。

```
21 print(lm.intercept_)
```

執行結果：

```
3620.626025474225
```

每一個特徵變相的迴歸係數，也就是公式中的 β_1~ β_n。

```
22 print(lm.coef_)
```

執行結果：

```
[ 3.00462596e+03   4.72062851e+03   2.01721953e+01  -2.28276444e+03
  2.07920813e+03   1.48858548e+02  -1.46823617e+03  -2.81825555e+03
  3.51436775e+03  -1.33455026e+03   4.27693492e+02   3.67418636e+02
 -1.60739471e+03  -3.07750639e+02  -1.51851326e+03   1.48574584e+03]
```

進行預測

建立好模型後，我們接著來預測看看 30% 的測試資料。

```
y_predict = lm.predict(X_test)
```

將預測按讚數與真實按讚數列出前五筆作比較，可以發現沒有很準確。

```
pd.DataFrame(list(zip(y_test.values, y_predict)),
             columns=['Measured','Predicted']).head()
```

	Measured	Predicted
0	6938	221.857708
1	860	2083.246286
2	128	273.179627
3	4851	5543.968663
4	491	12672.784547

圖 9-90 前五筆資料

將按讚數量的分佈透過圖形呈現，可以發現大多都聚集在 20,000 個讚以下。

```
import matplotlib.pyplot as plt
plt.scatter(y_test.values,y_predict,s=2)
plt.plot([y_test.values.min(), y_test.values.max()],
         [y_test.values.min(), y_test.values.max()], 'k--', lw=2)
plt.ylabel('Predicted')
plt.xlabel('Measured')
plt.show()
```

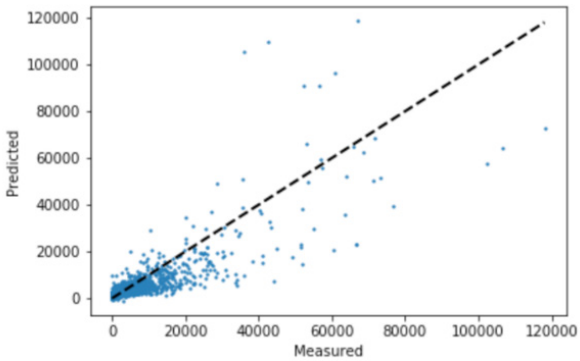


圖 9-91 按讚數量

檢驗迴歸模型

我們的資料總共有將近一萬筆，不太可能一筆一筆的確認其預測的是否準確，因此可以透過一些統計指標來協助我們評估模型的準確性。

均方誤差 MSE，也就是所有的按讚數量與真實按讚數量之間的誤差總和：

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test.values, y_predict)
print('MSE : ',mse)
```

執行結果：

```
MSE : 23446608.538704727
```

均方根誤差 RMSE，將 MSE 開根號：

```
from math import sqrt
rms = sqrt(mean_squared_error(y_test.values, y_predict))
print('RMSE : ',rms)
```

執行結果：

```
RMSE : 4842.169817210537
```

判定係數 R-square 與調整後的判定係數 Adjusted R-square：判定係數用於評估所使用的特徵對於目標 Y 的解釋能力有多少比例，當特徵不只一項時查看 Adjusted R-square 較為準確。因此從以下結果中可以得知分享數、留言數、小編、發文時間以及發文星期在線性迴歸模型當中只能解釋目標 Y 的 69% 而已。

```
R_2 = lm.score(X_train, y_train)
print('R-squared  : ',R_2)
```

執行結果：

```
R-squared: 0.6991904295741282
```

```
adj_R_2 = R_2 - (1 - R_2) *
          (X_train.shape[1] / (X_train.shape[0] - X_train.shape[1] - 1))
print('Adjusted R-squared : ',adj_R_2)
```

執行結果：

```
Adjusted R-squared:  0.6949518287125203
```

現在我們已經成功建立出第一個機器學習模型了，不過我們並不容易透過這些指標來得知究竟模型好不好，因為沒有其他比較的基準，因此現在我們就來使用另一項迴歸模型，隨機森林樹 (Random Forest Regression) 來比較看看模型的好壞吧！

Random Forest Regression

隨機森林樹迴歸介紹

此模型是基於決策樹 (decision tree) 的樹狀模型，當我們資料中具有較多的類別特徵，且這些類別特徵有可能對於目標 Y 會有顯著影響時非常適合使用。從第三節的 EDA 分析可以發現 PO 文的小編以及發文的時間確實會影響到按讚的數量，因此可以設想採用此模型效果有機會更好。

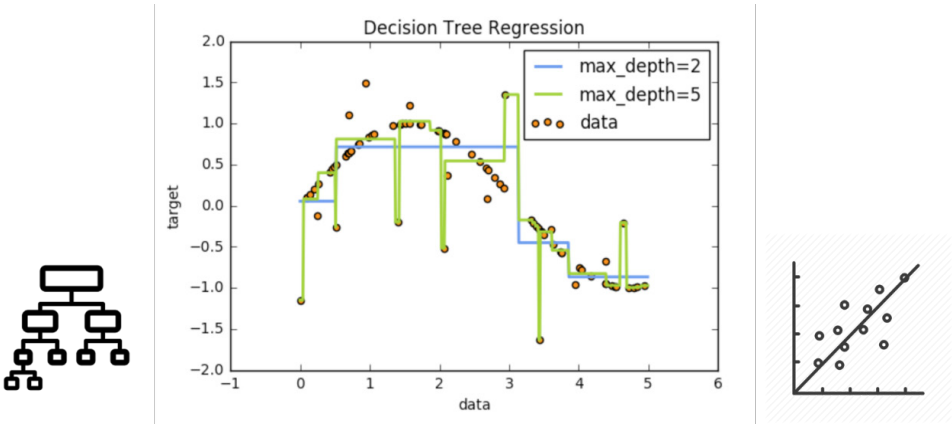


圖 9-92 隨機森林樹迴歸

匯入樹迴歸模型套件

從 sklearn 套件中匯入隨機森林樹迴歸，並且將此模型存入 rfr 變數當中。

```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
```

接著進行模型訓練，使用到的是前面所切分的 70% 訓練資料。

```
46  rfr.fit(X_train.values, y_train.values)
```

執行結果：

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto',
                      max_leaf_nodes=None, max_samples=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=100,
                      n_jobs=None, oob_score=False, random_state=None,
                      verbose=0, warm_start=False)
```


進行預測

建立好模型後，我們接著來預測看看 30% 的測試資料：

```
47  y_predict_rfr = rfr.predict(X_test)
```

將預測按讚數與真實按讚數列出前五筆作比較，可以發現好像兩者之間有較為接近的感覺囉！

```
48  pd.DataFrame(list(zip(y_test.values, y_predict_rfr)),
49                 columns=['Measured','Predicted']).head()
```

	Measured	Predicted
0	6938	3221.5
1	860	755.8
2	128	292.6
3	4851	5626.1
4	491	730.9

圖 9-93 前五筆資料

檢驗迴歸模型

接著我們同樣透過一些統計的指標來評估此模型的準確程度：

均方誤差 MSE

```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(y_test.values, y_predict_rfr)
print('MSE : ',mse)
```

```
執行結果：
MSE :17156586.352579575
```

均方根誤差 RMSE

```
from math import sqrt
rms = sqrt(mean_squared_error(y_test.values, y_predict_rfr))
print('RMSE : ',rms)
```

```
執行結果：
MSE :4142.050983821852
```

判定係數 R-square

```
R_2 = rfr.score(X_train, y_train)
print('R-squared  : ',R_2)
```

```
執行結果：
R-squared : 0.9591639892499428
```

調整後的判定係數 Adjusted R-square

```
adj_R_2 = R_2 - (1 - R_2) *
            (X_train.shape[1] / (X_train.shape[0]- X_train.shape[1] - 1))
print('Adjusted R-squared : ',adj_R_2)
```

```
執行結果：
Adjusted R-squared: 0.9585885835203154
```

我們可以從結果中發現不論是 MSE、RMSE 或是判定係數都比線性迴歸模型來的優秀，甚至判定係數高達 94%，這也就是前面所提及的，當我們對於所要分析的資料具有一定的了解，以及懂得不同演算法背後的運算邏輯時，在選用模型上確實可以收到不錯的效果，當然這並非代表隨機森林樹就是最適合此份資料的演算法。相信加入更多的特徵 X，以及進行更多的資料前處理後會有更好的預測結果，我們將這部分稱之為資料特徵工程 (feature engineering)。因

此，想要擁有好的預測結果，除了選對模型之外，最重要的就是挑選特徵變項。特徵變項並不是越多越好，而是要能找到最能解釋目標 Y 的幾項特徵才行，這部分就要透過第三節的 EDA 分析來看特徵與目標之間的關係是否有顯著影響，此外就是依靠我們本身的領域知識 (domain knowledge) 了，當我們對資料越加熟悉，越容易得知什麼樣的特徵對於目標會有影響。

小結

我們在本章介紹了機器學習的一些基本概念、以及如何透過 Python 程式成功實踐。相信各位讀者讀完此章後會發現建立機器學習的程式相當簡單，但中間所需具備的領域知識卻是相對高深且複雜的，沒有一定的實務分析經驗並不容易建立出一個好的機器學習模型。此外我們也只介紹了機器學習當中的迴歸模型，尚未介紹分類模型與分群模型，雖然使用 Python 程式同樣相當的簡單與簡短即可成功，不過這兩種模型又各有指標用於評估模型的好壞，因此要成為一名機器學習工程師，其實是需要許多時間磨練經驗的！不過各位讀者也不用太擔心這條路途會非常遙遠，網路上有許多的教材甚至影片，坊間也有不少的書籍。也不用擔心沒有資料可以練習分析。在這邊相當推薦 Kaggle 這個網站，他是一個資料分析競賽的網站，裡面有各式各樣的數據集資料，像是鐵達尼號、電影、旅館、交通等等，即使比賽已經結束了仍然能下載其資料來進行分析，此外每個比賽都會有許多高手分享他們的分析方法以及建模過程，因此相當推薦有興趣的讀者參考及學習。

總結

我們的教學就告一段落了，所有內容都是相當的實用的，從最初的資料爬取到資料分析、建立模型等等，將中間的所有過程整理起來會是相當完整的一份分析報告書呢！我們希望藉由此種方式可以讓各位讀者了解什麼叫做資料分析，在資料分析的過程當中會是以什麼樣的方式在進行。即使讀者對於資料科學領域並未有太大的興趣，但至少也習得了一套完整的資料分析流程。現階段各行各業都在想盡辦法跟上這一波大數據的浪潮，相信各位在未來不論是與數

據分析師交談亦或是與公司內部的 IT 人員討論時，可以更有自信地表達自己的看法，希望本章可以讓各位讀者具有知識上或是技術上的收穫。

參考資料

1. API 介紹：https://cola.workxplay.net/what-is-an-api/
2. 圖形 API 測試工具的官方文件說明：
   https://developers.facebook.com/docs/graph-api/using-graph-api/#fieldexpansion
3. 斷字斷詞字典下載位置：https://github.com/fxsjy/jieba/raw/master/extra_dict/dict.txt.big
4. 停止詞字典下載位置：https://github.com/chdd/weibo/blob/master/stopwords/ 中文停用词库 .txt
5. 圖形 API 參考資料：https://developers.facebook.com/docs/graph-api/reference
6. word2vec 簡介：
   https://medium.com/pyladies-taiwan/ 自然語言處理入門 -word2vec 小實作 -f8832d9677c8
7. word2vec 參數
   (1) 官方文件：https://radimrehurek.com/gensim/models/word2vec.html
   (2) 中文參考：https://www.kaggle.com/jerrykuo7727/word2vec