大數據資料科學家養成班
第二次專案討論

Costco好市多 商品經驗老實說

簡報者：許X寧

第五組：林X霖
許X寧
呂X蓁

中華民國 111年 05 月

前情提要

專案討論僅供課後留存分享使用

藉由大數據分析來協助信用卡行銷

更精確的評估受眾

更即時的調整優惠

更客觀的預測未來趨勢

BI商業智慧-女性受眾

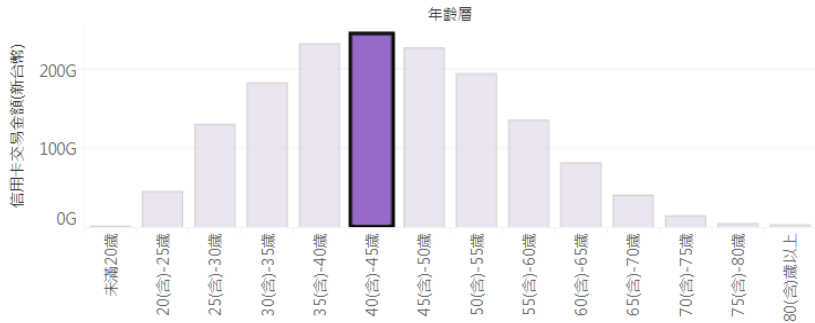
最具顯著效益之行銷推廣組合

針對40-45歲女性提高百貨的優惠

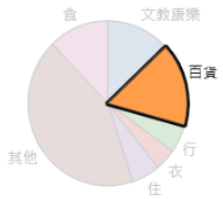
全台縣市消費熱度

針對最具消費力之年齡層推廣總收益占比第一之產業

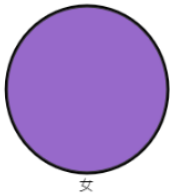
年齡X性別消費比較



產業別消費比例



各性別交易金額



年度各年齡層消費金額

BI商業智慧-男性受眾

最具顯著效益之行銷推廣組合

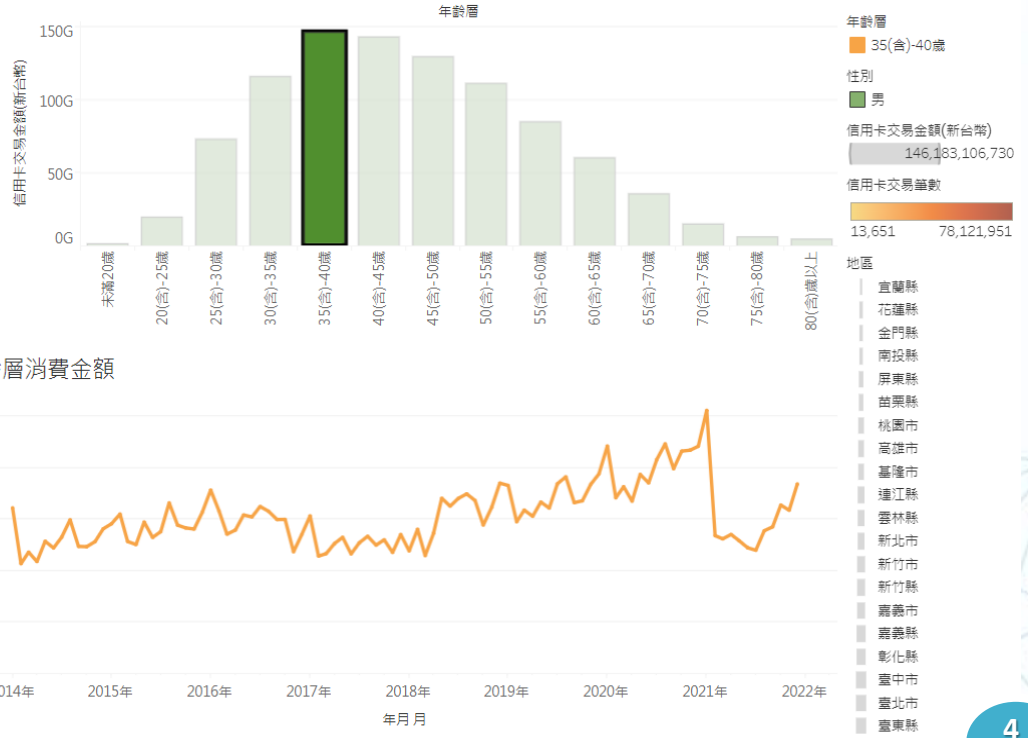
針對35-40歲男性提高食的優惠

使用

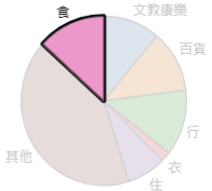
全台縣市消費熱度

針對最具消費力之年齡層
推廣總收益占比第一
之產業

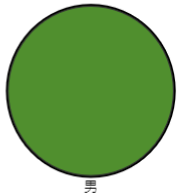
年齡X性別消費比較



產業別消費比例



各性別交易金額



年度各年齡層消費金額

簡報大綱

專案討論僅供課後留存分享使用

研究動機

壹、程式開發流程圖

貳、開發過程描述

(1)資料來源

(2)爬蟲

(3)資料前處理

(4)模糊比對

(5)使用者操作介面

參、成果展示

肆、現行問題

伍、未來展望

研究動機

專案討論僅供課後留存分享使用

從自身需求出發

通路選擇：COSTCO、全聯、購物網站

別人有買，我也要

錢要花在刀口上，精準消費

節省比價時間

全球第一家Costco

1983年於美國華盛頓州西雅圖市

臺灣第一家Costco

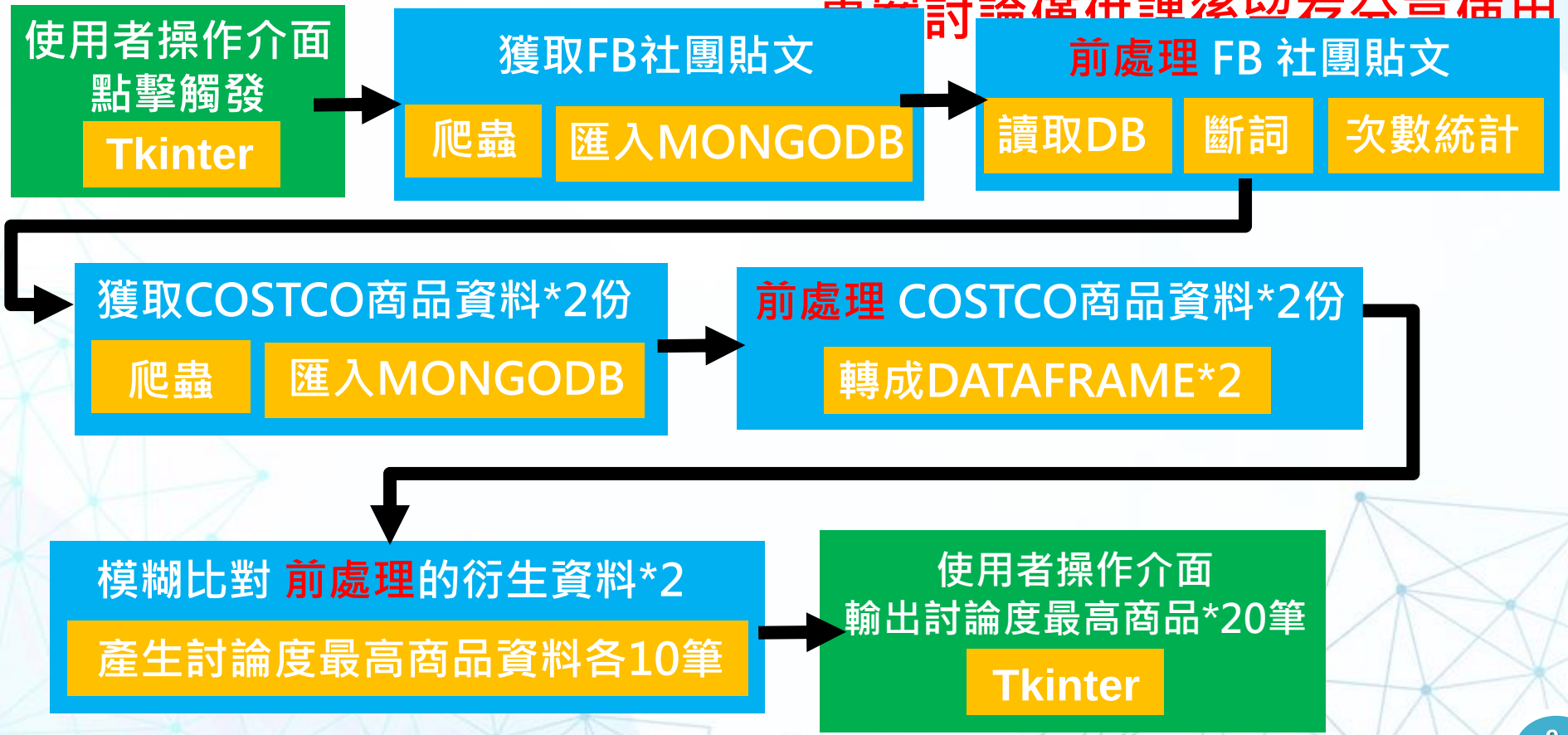
專案討論僅供課後留存分享使用

1997年於臺灣高雄市前鎮區

熱門度的定義

壹、程式開發流程圖

前端 後端 採用語法



貳、開發過程描述

(1)資料來源

資料來源(FB公開社團)

專案討論僅供課後留存分享使用

1 選擇原因

參考價值：活躍度最高

2 目標資料

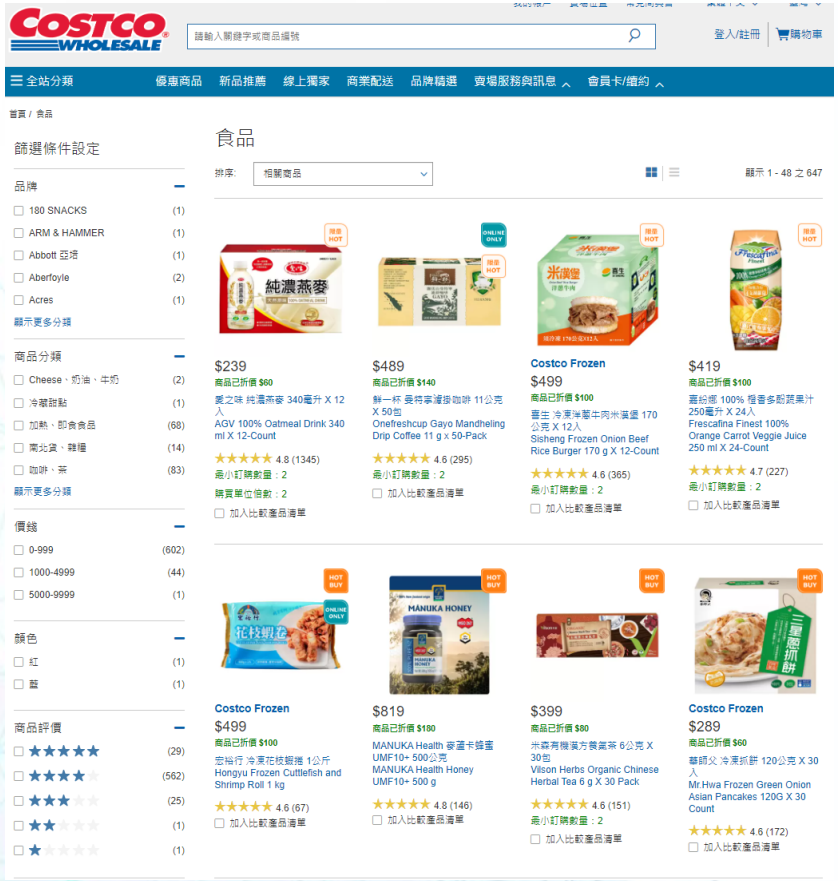
剛才逛好市多線上，發現飛利浦這台萬用鍋有特價了!
之前吃過朋友用這台燉的滷肉，肉很嫩湯汁也很濃，當時看他開鍋直接用來炒菜也覺得很方便，感覺可以燉可以炒真的頗萬用，被燒到很想買一台XD
一直蹲等終於被我等到優惠，也上來分享分享

買了一罐松露醬，家裡除了我沒有一個人受得了那個味道，明明就很香😢😢😢
看大家都說很容易壞掉，搞得我三餐卵起來加，松露蛋餅，松露臭豆腐，松露蛋炒飯...
現在已經升級到松露炒米粉了😟😟😟
跪求各種好吃的方法！！！

技術考量：公開社團在爬蟲的可行性較高

資料來源(COSTCO官網)

專案討論僅供課後留存分享使用



1. 選擇品項：飲食
2. 販賣範圍：全通路
   賣場限定

貳、開發過程描述

(2)爬蟲

(3)資料前處理

爬蟲(FB公開社團)

專案討論僅供課後留存分享使用

函式庫

```
def scrap():
    
    #FB 爬蟲匯入 mongoDB
    import pandas as pd
    import os
    import time
    import requests
    from bs4 import BeautifulSoup
    from selenium import webdriver
    from webdriver_manager.chrome import ChromeDriverManager
    import random
    import pandas as pd
    import pymongo
    from pymongo import MongoClient

    global df_extract
    global df
    global df2
```

連接MONGODB，並覆蓋舊資料

```
    client = pymongo.MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
    db = client.costco
    col = db.FB_pythons

    x = col.delete_many({})
    print(x.deleted_count, "個已刪除")
    #wb=openpyxl.Workbook()
    #ws=wb.active
```

避開彈出通知

```
# 防止跳出通知
chrome_options = webdriver.ChromeOptions()
prefs = {
    "profile.default_content_setting_values.notifications": 2
}
chrome_options.add_experimental_option("prefs", prefs)

# 使用 ChromeDriverManager 自動下載 chromedriver
driver = webdriver.Chrome(
    ChromeDriverManager().install(), chrome_options=chrome_options)
```

爬取FB貼文

```
    # 進入 Costco 好市多 商品經驗老實說

driver.get("https://www.facebook.com/groups/1260448967306807?sorting_setting=CHRONOLOGICAL")

    time.sleep(5)

    # 往下滑 10 次，讓 Facebook 載入文章內容
    for x in range(20):
        driver.execute_script("window.scrollTo(0,document.body.scrollHeight)")
        print("scroll",x)
        time.sleep(random.randint(3,5))

    root = BeautifulSoup(driver.page_source, "html.parser")

    # 定位文章標題
    titles = root.find_all(
        "div", class_="ecm0bbzt hv4rvrfc ihqw7lf3 dati1w0a")
    for title in titles:
        # 定位每一行標題
        posts = title.find_all("div", class_="kvgmc6g5 cxmmr5t8 oygrvhab hcukyx3x c1et5uql ii04i59q")
        # 如果有文章標題才印出
        if len(posts) != 0:
            for post in posts:
                #print(post.text)
                #ws.append([post.text])
                #新增單筆資料
                st={"FB 內文":post.text
                    }
            result=col.insert_one(st)
```

前處理01(FB公開社團貼文)

專案討論僅供課後留存分享使用

讀取DB中FB貼文

```
#FB 爬蟲斷詞 Dataframe

from pymongo import MongoClient
import pandas as pd
#連接 MongoDB
client =
pymongo.MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
#指定資料庫
db = client.costco
#指定資料表
col=db.FB_pythons
#直接從 MongoDB 查詢
post=col.distinct("FB 內文")
# data=pd.DataFrame({"FB 內文":post})
# print(data)
```

將貼文進行斷詞

```
#將 FB 內文進行斷詞
import jieba
word=[]
for i in post:
    a=jieba.cut(i)
    word+=a
print(word)
```

斷詞結果

```
['#', '吉室', '三星', '蔥', '牛', '軋米餅', '#', 'costco', '新品', '推薦', '#', '檬', '是', '女兒', '賊', '嗎', '?', '?', '這次', '回家', '看到', '我媽', '買', '兒', '賊', '立刻', '搜刮', '一半', '咖啡', '還蠻', '香', '的', '\u3000', '不會', '好喝', '100', '倍', ' ', '覺得', '好', '不錯', '!', '!', '!', '!', '…', '…', '排', '我', '想', '應該', '是', '我', '的', '廚藝出', '了點', '問題', '！', '#', ', '莎莎亞', '椰奶', '2', '.', '沙威隆', '3', '.', '葡萄柚', '汁', '4', '.', '草店', '快要', '被', '搬', '完', '了', '…', '…', '', 'Apple', ' ', 'iPad', ' ', '的', '時候', '還沒有', '看到', '，', '沒', '想到', '悄悄的', '在', '線', '上', '大概', '便宜', '千元', '左右', '，', '搭配', '折扣', '碼', '可以', '再折', '300'
```

前處理02(FB公開社團貼文)

統計每個詞語的出現次數

```
#統計每個字串出現次數
#把每個字串當成 key 去計算出現的次數(當成值)
dic_w={}
#判斷 KEY(ele)是否在 dic_w{}裡面
for ele in word:
    #如果 KEY(ele)不在 dic_w{}裡面,如果 KEY 沒出現過，就新增一筆 KEY 並讓值從 1 開始
    if ele not in dic_w:
        dic_w[ele]=1
    #如果 KEY 重複出現，值(AKA 次數)就+1
    else:
        dic_w[ele]+=1
print(dic_w)

#調整顯示方式
#因為 dic 是無序的，所以要用.items 來排序呼叫
for ele in dic_w.items():
#       print(ele[0],ele[1])
    #兩個字以上的，才會出現
    if len(ele[0])>=2: # and ele[1]>=2
        num=list()
        word=list()
        #因為經過.items 的關係，變成(0,1)=(前面,後面)
#           print(ele[0],ele[1])
```

統計結果

```
{'#': 6, '吉室': 1, '三星': 1, '蔥': 1, '牛': 1, '軋米餅': 1, 'cos
'誰': 1, '跟': 7, '我': 21, '一樣': 5, '是': 23, '女兒': 2, '賊':
3, '買': 24, '了': 38, '一箱': 2, '冰萃': 1, '咖啡': 2, '身為': 2,
1, '的': 82, '\u3000': 1, '不會': 7, '很苦': 1, '很': 10, '澀': 1,
' ': 39, '覺得': 3, '好': 10, '不錯': 2, '!': 8, '…': 31, '顯示':
7, '應該': 1, '廚藝出': 1, '了點': 1, '問題': 2, '！': 14, '陪': 1
'莎莎亞': 1, '椰奶': 2, '2': 3, '沙威隆': 1, '3': 2, '葡萄柚': 1,
5, '5': 3, '/': 8, '新莊店': 1, '快要': 1, '被': 3, '搬': 1, '完':
```

前處理03(FB公開社團貼文)

排序次數(從少到多)

```
#要排序次數
import operator
#itemgetter 是指出現該函數內第幾個的意思(從 0 開始數)
```

```
#.itemgetter()從 0 開始會變成依筆畫排列
sort_w=sorted(dic_w.items(),key=operator.itemgetter(1),reverse=True)
for ele in sort_w:
    num=list()
    word=list()
    for i in range(len(sort_w)):
        a=sort_w.pop()
        num.append(a[1])
        word.append(a[0])
df_extract=pd.DataFrame({'num':num,'word':word})
print(df_extract)
    return df_extract
```

回傳排序結果

排序結果

	num	word
0	1	哈啾
1	1	滿毛
2	1	中怖
3	1	空氣
4	1	毛
...	...	...
1080	31	…
1081	38	了
1082	39	
1083	82	的
1084	110	，

[1085 rows x 2 columns]

爬蟲(COSTCO官網)

專案討論僅供課後留存分享使用

函式庫

```
#Costco 爬蟲食品清單 1

#       import requests #請求
#       from bs4 import BeautifulSoup #BS 要大寫
#       import time
#       import random
#       import pymongo
#       import openpyxl #匯入 EXCEL 格式

    #至少要有一筆資料，資料庫才能存在(沒辦法建立空資料庫)
    #建立資料庫
```

連接MONGODB，並覆蓋舊資料

```
    client = pymongo.MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
    db = client.costco
    #建立 collection
    col=db.commodity
    x = col.delete_many({})

    print(x.deleted_count, "筆舊資料已刪除")
```

爬取COSTCO商品資料

```
    # print(soup.find_all("ul",class_="product-listing product-grid")[0].find_all("li",class_="product-list-item product-list-item--grid vline ng-star-inserted")[0].prettify())

    for commodity in soup.find_all("ul",class_="product-listing product-grid")[0].find_all("li",class_="product-list-item product-list-item--grid vline ng-star-inserted"):
        #商品名稱
        a=commodity.find_all("a",class_="lister-name js-lister-name")[0].text
        #商品價格
        b=commodity.span.text
        #網址
        c=commodity.a["href"]

        #新增單筆資料
        st={"商品名稱":a,
            "商品價格":b,
            "網址":c
            }
        result=col.insert_one(st)
```

讀取DB

```
#       from pymongo import MongoClient
#       import pandas as pd
    #連接 MongoDB
    client = MongoClient("mongodb+srv://test:test@cluster0.j7nzi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority")
    #指定資料庫
    db = client.costco
    #指定資料表
    col=db.commodity
    #直接從 MongoDB 查詢
    name=col.distinct("商品名稱")
    price=col.distinct("商品價格")
    web=col.distinct("網址")

    df = pd.DataFrame(list(db.commodity.find({},{ "_id": 0, "商品名稱": 1, "商品價格": 1, "網址": 1 })))
    print(df)
#       return df
```

前處理：產生DATAFRAME

貳、開發過程描述

(4)模糊比對

(5)使用者操作介面

模糊比對01

專案討論僅供課後留存分享使用

函式庫

```
#Costco 清單 1 模糊比對結果
def shoplist1(): #定義按鈕使用功能
    import pandas as pd
    from fuzzywuzzy import fuzz
    from fuzzywuzzy import process

    global dfx
    global final
    global complete
```

FB貼文斷詞V.S.COSTCO商品列表

```
    #FB 貼文的斷詞
#       chart=pd.DataFrame(df_extract) #為什麼要再 dataframe 一次
    chart=df_extract
    #Costco 商品列表
#       chart2=pd.DataFrame(df) #為什麼要再 dataframe 一次
    chart2=df

    def fuzzy_merge(chart, chart2, key1, key2, threshold=20, limit=1):
        """
        :param df_1: the left table to join
        :param df_2: the right table to join
        :param key1: key column of the left table
        :param key2: key column of the right table
        :param threshold: how close the matches should be to return a match, based on Levenshtein distance
        :param limit: the amount of matches that will get returned, these are sorted high to low
        :return: dataframe with boths keys and matches
        """
        s = chart2[key2].tolist()
        m = chart[key1].apply(lambda x: process.extract(x, s, limit=limit))
        chart['matches'] = m
        m2 = chart['matches'].apply(lambda x: ', '.join([i[0] for i in x if i[1] >= threshold]))
        chart['matches'] = m2
        chart['merge_key'] = chart['matches']
        chart2['merge_key'] = chart2[key2]
        dfx = pd.merge(chart, chart2, how='left',on='merge_key')
        return dfx

    dfx = fuzzy_merge(chart,chart2,'word','商品名稱',20)
    final=dfx.drop(['matches','merge_key'],axis=1,inplace=True)
    print(final)
```

比對精準度設定為20%

模糊比對02

資料清洗

預設為撈取第一筆出現對應字詞的商品資料

存分享使用

出現次數TOP10

```
#資料清洗(讀取筆數要改成讀 10 筆(0~9))
complete=dfx.dropna().sort_values(["num"],ascending=0,ignore_index=True).drop(["num","word"],axis=1).drop_duplicates()[0:10]

final=dfx.dropna().sort_values(["num"],ascending=0,ignore_index=True).drop(["num","word","網址"],axis=1).drop_duplicates()[0:10]
    final.index=[1,2,3,4,5,6,7,8,9,10]
    print(dfx)
    l1.config(text=final,justify="left")
    return complete
```

使用者操作介面

函式庫

```
from tkinter import *
import tkinter as tk
import pandas as pd

#背景
window = Tk()

window.geometry("1447x728")
window.configure(bg = "#FFFFFF")
canvas = Canvas(
    window,
    bg = "#FFFFFF",
    height = 728,
    width = 1447,
    bd = 0,
    highlightthickness = 0,
    relief = "ridge")
canvas.place(x = 0, y = 0)

background_img = PhotoImage(file = f"background.png")
background = canvas.create_image(
    723.5, 364.0,
    image=background_img)

#點擊按鈕會有反應
def btn_clicked():
```

視窗建置

點擊觸發爬蟲

按鈕建置

```
#背景清單 1 按鈕
img1 = PhotoImage(file = f"img1.png")
b1 = Button(
    image = img1,
    borderwidth = 0,
    highlightthickness = 0,
    command = shoplist1,
    relief = "flat")

b1.place(
    x = 1123, y = 364,
    width = 276,
    height = 53)

l1=Label()
l1.place(
    x = 471, y = 77,
    width = 961,
    height = 269)
```

專案討論僅供課後留存分享使用

```
#背景清單 2 按鈕
img2 = PhotoImage(file = f"img2.png")
b2 = Button(
    image = img2,
    borderwidth = 0,
    highlightthickness = 0,
    command = shoplist2,
    relief = "flat")

b2.place(
    x = 1131, y = 13,
    width = 295,
    height = 63)

l2=Label()
l2.place(
    x = 471, y = 435,
    width = 961,
    height = 287)


window.resizable(False, False)
window.mainloop()
```

參、成果展示

肆、現行問題

討論僅供課

發現問題	導致現況	預期解決方式
爬蟲相關 ◆ FB社團爬取的貼文內容不完整 ◆ FB社團爬蟲的連接不穩定 ◆ 商品資料會出現非食品	◆ 影響模糊比較結果 ◆ 推薦商品中可能會包含非食品	◆ 完善FB社團與COSTCO官網的爬蟲設定
◆ 斷詞不夠精確(例：柯克蘭) ◆ 民眾發文時，常使用簡稱或口語化說法	◆ 斷詞數量統計會失真(算錯或沒算到) ◆ 模糊比對結果容易失真(例：可能民眾討論的是蜂蜜工廠的蜂蜜，卻比對出蜂蜜餅乾)	◆ 字典訓練(新增專有名詞，提高斷詞精準度) ◆ 比對貼文的附圖或留言，提高熱門討論的精準度
◆ GUI的版面配置不夠精確 ◆ (字形、字體、位置) ◆ 未提供商品連結	◆ 產生的商品清單過小，看不清楚 ◆ 民眾須另行查詢購買方式，失去節省時間的初衷	◆ 優化GUI排版 ◆ 新增商品購買連結

伍、未來展望

專案討論僅供課後留存分享使用

- 新增優惠折扣的標籤
- 熱門類別去排名，例：生活用品、家電等等
- 討論群去做清洗，例： Ptt、Dcard、Instagram
- 其他通路做比較，例：家樂福、全國電子

專案討論僅供課後留存分享使用

簡報結束
謝謝聆聽