# 資料探勘專案作業四

# 使用 Python 進行交易資料關聯規則分析


指導教授：

許中川 教授

成員：

M11112075 徐紹鈞

M11121028 陳韋諠


日期：

2022 年 1 月 5 日

# 摘要

本研究利用了交易資料集做關聯規則分析，常應用在商業產品喜愛調查領域上，用途相當的廣泛，在擁有大量數據的資料庫中，找尋資料間彼此的關聯，利用關聯分析(Apriori)分析何種商品放在一起可以增加營業額，並分析不同商品放在一起將發生何種變化。本研究將會利用Apriori及FP-Growth演算法進行分析，並比較兩種方法的處理階段所花費的時間。

# 一、 緒論

## 1.1 動機

在貿易相關產業中為了尋求在直接營銷活動中客戶購買物品的關聯度， 可以利用客戶購買物品來分析資料彼此之間的關聯，變可以為同群的商品設置購買策略，以利於利潤最大化。

## 1.2 研究目的

目前的演算法有相當多種，本研究利用 Apriori 及 FP-Growth 演算法來進行分析，投入商品 ID、商品種類等 6 種屬性資料，希望透過兩種分析方式找到最適合方法，並投入實際情況來做使用 。

# 二、實驗

## 2.1 資料集

### 2.1.1 交易資料集說明

表一 交易資料集 欄位資料說明彙總表

| 欄位名稱 | 欄位說明 |
|---|---|
| ITEM_ID | 商品 ID |
| ITEM_NO | 商品編號 |
| PRODUCT_TYPE | 商品種類 |
| CUST_ID | 客戶編號 |
| TRX_DATE | 交易日期 |
| INVOICE_NO | 發票號碼 |
| QUANTITY | 數量 |

### 2.1.2 實驗數據-交易資料集

| | ITEM_ID | ITEM_NO | PRODUCT_TYPE | CUST_ID | TRX_DATE | INVOICE_NO | QUANTITY |
|---|---|---|---|---|---|---|---|
| 0 | 3217532 | M25P40-VMN6TPB | MEMORY_EMBEDED | 3218 | 2016-07-26 | CX47348203 | 2500 |
| 1 | 3326781 | AU80610006237AASLBX9 | CPU / MPU | 2470 | 2016-07-11 | CX47346522 | 50 |
| 2 | 740487 | MMBD2837LT1G | DISCRETE | 16135 | 2016-07-27 | CX47348534 | 3000 |
| 3 | 3434776 | IHLP1616ABER2R2M11 | PEMCO | 999999999 | 2016-07-29 | A20160700174 | 0 |
| 4 | 70072 | MMBT3906LT1G | DISCRETE | 2356 | 2016-07-06 | CX47346184 | 12000 |

圖(一) 交易資料集

# 三、方法

3.1 程式執行方法

3.1.1 Apriori 演算法

Step1: 將資料進行進行疊代，有了數據庫後接下來進行第一次掃描，可以看出每樣商品出現的次數。 Apriori 的重要假設，當項目集不頻繁那它的子集也不會頻繁。

Step2: 接下來進行第二次掃描，也就是項目集中會有兩樣商品來做分析，一樣列出出現次數後，淘汰掉次數低於預設值的項目集，形成新的數據表。

Step3: 最後進行我們的第三次掃描，項目集中會有三樣商品來做分析，列出出現次數，由於只有一組項目集，最後一樣會形成一個關聯數據表。

3.1.2 FP-Growth 演算法

FP-Growth 演算法是一種高效率發現頻繁集的方法，只須對數據進行兩次掃描

Step1: 首先會先建構 FP 樹，會對原始數據掃描兩遍，第一遍會對所有元素項的出現次數進行記數，會丟棄小於設定的閥值，並將超過閥值的頻繁項集遞減排序。

Step2: 接著 FP 樹中挖掘頻繁項集

Step3:利用推薦系統推薦客戶購買其餘產品，首要要先輸入購買了什麼產品，最後系統將會推薦產品給該顧客。

```
X = input('請輸入要購買的商品:')

empty_list = []
y = list(rules[rules['antecedents']==frozenset({X})]['consequents'].unique())
for i in range(len(y)):
    empty_list.append(list(y[i]))

print('額外推薦:'+str(empty_list))
```

請輸入要購買的商品:

```
X = input('請輸入要購買的商品:')

empty_list = []
y = list(rules[rules['antecedents']==frozenset({X})]['consequents'].unique())
for i in range(len(y)):
    empty_list.append(list(y[i]))

print('額外推薦:'+str(empty_list))
```

```
請輸入要購買的商品:PEMCO
額外推薦:[['LOGIC IC'], ['DISCRETE'], ['LOGIC IC', 'DISCRETE']]
```

# 四、實驗

## 4.1 前置處理

### 4.1.1 交易資料集

在交易資料集中，將相同 INVOICE_NO 的資料及數量為零或負值的值進行刪除，接著在觀察檔案時查看是否有缺值的現象產生，我們將有缺值的資料進行刪除，完成資料清洗。

```
df_pre = df
df_pre = df_pre.drop_duplicates()
```

```
print("Before data-preprocessing: ", df.shape)
print("After data-preprocessing: ", df_pre.shape)
```

```
Before data-preprocessing:  (157396, 7)
After data-preprocessing:  (135661, 7)
```

```
fliter = (df_pre["QUANTITY"] > 0)
df_pre[fliter]
```

| | ITEM_ID | ITEM_NO | PRODUCT_TYPE | CUST_ID | TRX_DATE | INVOICE_NO | QUANTITY |
|---|---|---|---|---|---|---|---|
| **0** | 3217532 | M25P40-VMN6TPB | MEMORY_EMBEDED | 3218 | 2016-07-26 | CX47348203 | 2500 |
| **1** | 3326781 | AU80610006237AASLBX9 | CPU / MPU | 2470 | 2016-07-11 | CX47346522 | 50 |
| **2** | 740487 | MMBD2837LT1G | DISCRETE | 16135 | 2016-07-27 | CX47348534 | 3000 |
| **4** | 70072 | MMBT3906LT1G | DISCRETE | 2356 | 2016-07-06 | CX47346184 | 12000 |
| **6** | 3420352 | TMP103AYFFR | LINEAR IC | 10228 | 2016-07-25 | CX47347899 | 3000 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **157389** | 14691206 | 74HCT365D,653 | LOGIC IC | 2100 | 2016-07-06 | 216071271 | 7030 |
| **157390** | 3249493 | 74HC04D,653 | LOGIC IC | 2197 | 2016-07-21 | 216075314 | 347 |
| **157391** | 133619 | 74LVC2G17GW,125 | LOGIC IC | 2579 | 2016-07-11 | 216072287 | 1644 |
| **157392** | 15231095 | MT41K256M16TW-107:P TRAY | MEMORY_EMBEDED | 43262 | 2016-07-28 | 216077517 | 300000 |
| **157395** | 14852949 | MT41K64M16TW-107:J | MEMORY_EMBEDED | 47302 | 2016-07-05 | 216070810 | 2000 |

111961 rows × 7 columns

圖(二) 交易資料集 Preprocessing

## 4.2 實驗設計

4.2.1 Apriori 演算法

Apriori 是計算頻繁項集的一種演算法，它假設當項集是頻繁的，也就是假設 B 這個物品在數據中是頻繁出現的，那它的子集也會是頻繁的，也就是說{B、C}、{B、C、E}等也是頻繁的，反之就是不頻繁的。

4.2.2 安裝套件

載入 Apriori 演算法至 Jupyter

```
pip install apyori
```

```
Requirement already satisfied: apyori in /Users/fred0522/opt/anaconda3/lib/python3.9/site-packages (1.1.2)
Note: you may need to restart the kernel to use updated packages.
```

```
from apyori import apriori
```

圖(三) Apriori 演算法載入至 Jupyter

4.2.3 設置不同種類關聯集

```
market_data = [['MEMORY_EMBEDED','CPU / MPU','DISCRETE','PEMCO','LOGIC IC','LINEAR IC','OPTICAL AND SENSOR'],
 ['MEMORY_EMBEDED','CPU / MPU'],
 ['DISCRETE','PEMCO','LOGIC IC','LINEAR IC'],
 ['DISCRETE','PEMCO','LOGIC IC'],
 ['MEMORY_EMBEDED','CPU / MPU','DISCRETE','PEMCO','LOGIC IC'],
 ['LINEAR IC','OPTICAL AND SENSOR'],
 ['LOGIC IC'],
 ['DISCRETE','PEMCO'],
 ]
```

圖(四) 將商品隨機分成不同關聯性

4.2.4 參數設定

調整最小支持度(Min Support)、最小信心水準(Min Confidence)、最小提升度(Min Lift)來實現關聯分析，最小提升度要大於 1 才有關聯，所以要設在 1 以上，越大表示關聯性越強，max_length: 用來調整要對幾個商品關聯, max_length = 2，就會有二個商品在 list 中。

```
association_rules = apriori(market_data, min_support=0.2, min_confidence=0.2, min_lift=2, max_length=2)
association_results = list(association_rules)
for product in association_results:
 pair = product[0]
 products = [x for x in pair]
 print(products)
 print("Rule: " + products[0] + " →" + products[1])
 print("Support: " + str(product[1]))
 print("Lift: " + str(product[2][0][3]))
 print("================================")
```

```
['MEMORY_EMBEDED', 'CPU / MPU']
Rule: MEMORY_EMBEDED →CPU / MPU
Support: 0.375
Lift: 2.6666666666666665
================================
['LINEAR IC', 'OPTICAL AND SENSOR']
Rule: LINEAR IC →OPTICAL AND SENSOR
Support: 0.25
Lift: 2.6666666666666665
================================
```

FP-Growth 演算法

4-2-5 載入相關套件至 Jupyter

```
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
from sklearn import datasets
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.frequent_patterns import association_rules
from mlxtend.preprocessing import TransactionEncoder
from apyori import apriori as asp_apr
```

4-2-6 設置不同種類關聯集

```
market_data = [['MEMORY_EMBEDED','CPU / MPU','DISCRETE','PEMCO','LOGIC IC','LINEAR IC','OPTICAL AND SENSOR'],
 ['MEMORY_EMBEDED','CPU / MPU'],
 ['DISCRETE','PEMCO','LOGIC IC','LINEAR IC'],
 ['DISCRETE','PEMCO','LOGIC IC'],
 ['MEMORY_EMBEDED','CPU / MPU','DISCRETE','PEMCO','LOGIC IC'],
 ['LINEAR IC','OPTICAL AND SENSOR'],
 ['LOGIC IC'],
 ['DISCRETE','PEMCO'],
 ]
```

4-2-7 參數設定

調整信心度(confidence)為 0.5，最小支持度(min_support)為 0.01，接著求出各項產品關聯表，如下圖

```
#['support','confidence','lift']

number = 'confidence'
threshold = 0.5
min_support = 0.01
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction |
|---|---|---|---|---|---|---|---|---|---|
| 0 | (PEMCO) | (LOGIC IC) | 0.625 | 0.625 | 0.500 | 0.8 | 1.28 | 0.109375 | 1.875 |
| 1 | (LOGIC IC) | (PEMCO) | 0.625 | 0.625 | 0.500 | 0.8 | 1.28 | 0.109375 | 1.875 |
| 2 | (PEMCO) | (DISCRETE) | 0.625 | 0.625 | 0.625 | 1.0 | 1.60 | 0.234375 | inf |
| 3 | (DISCRETE) | (PEMCO) | 0.625 | 0.625 | 0.625 | 1.0 | 1.60 | 0.234375 | inf |
| 4 | (LOGIC IC) | (DISCRETE) | 0.625 | 0.625 | 0.500 | 0.8 | 1.28 | 0.109375 | 1.875 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1487 | (LINEAR IC, DISCRETE) | (PEMCO, LOGIC IC, OPTICAL AND SENSOR, MEMORY_E... | 0.250 | 0.125 | 0.125 | 0.5 | 4.00 | 0.093750 | 1.750 |
| 1488 | (DISCRETE, CPU / MPU) | (PEMCO, LOGIC IC, OPTICAL AND SENSOR, MEMORY_E... | 0.250 | 0.125 | 0.125 | 0.5 | 4.00 | 0.093750 | 1.750 |
| 1489 | (LINEAR IC, MEMORY_EMBEDED) | (PEMCO, LOGIC IC, OPTICAL AND SENSOR, DISCRETE... | 0.125 | 0.125 | 0.125 | 1.0 | 8.00 | 0.109375 | inf |
| 1490 | (LINEAR IC, CPU / MPU) | (PEMCO, LOGIC IC, OPTICAL AND SENSOR, DISCRETE... | 0.125 | 0.125 | 0.125 | 1.0 | 8.00 | 0.109375 | inf |
| 1491 | (OPTICAL AND SENSOR) | (PEMCO, LOGIC IC, DISCRETE, MEMORY_EMBEDED, LI... | 0.250 | 0.125 | 0.125 | 0.5 | 4.00 | 0.093750 | 1.750 |

4.3 實驗結果

4.3.1 交易資料集評估

| | 花費時間 |
|---|---|
| Apriori | 2 秒 |
| FP-Growth | 1 秒 |

表二 交易資料集評估

## 五、結論

透過上面的實驗可以發現，FP-Growth 在處理大量資料集的運算速度是比 Apriori 還要來的快的。

# 五、參考文獻

[1]Dataframe 刪除全為 0 的行:https://blog.csdn.net/qxqxqzzz/article/details/88786380
[2]資料前處理-缺失值處理方法 https://ithelp.ithome.com.tw/articles/10260675
[3]資料處理與清洗 https://yijutseng.github.io/DataScienceRBook/manipulation.html
[4]Machine Learning -關聯分析-Apriori 演算法 https://chwang12341.medium.com/machine-learning-關聯分析-apriori 演算法-詳細解說啤酒與尿布的背後原理-python 實作-scikit-learn 一步一步教學-76b7778f8f34
[5]資料探勘演算法 - 關聯規則: https://ithelp.ithome.com.tw/articles/10187244
[6]Implementation of Apriori and FPgrowth algorithms in Python: https://www.youtube.com/watch?v=Cryve9ZWbYk
[7]關聯分析（一）FP-Growth:https://www.796t.com/content/1546888205.html