國立雲林科技大學資訊管理系

資料探勘-作業四

Department of Information Management

National Yunlin University of Science & Technology

Assignment

使用聚類分析Banana資料集與sizes3資料集之績效

M11223018黃裕鳴

M11223020游棨翔

M11223043葉桔良

指導老師：許中川 博士

Advisor: Chung-Chian Hsu, Ph.D.

中華民國113年1月

January 2024

**摘要**

本研究旨在深入比較Apriori演算法與FP-Growth演算法在小型實際交易資料集中的效能。雖然Apriori演算法和FP-Growth演算法在大型數據集上皆有其優勢，然而在小型資料集中的實際表現卻可能不盡相同。研究動機來自於對這兩種常見資料探勘演算法在實際情境中的效能差異的好奇心。

本研究選擇一個真實的交易資料集，其中包含大量發票號碼、商品資訊等交易相關資訊，總計157,397筆資料。透過Apriori演算法和FP-Growth演算法，本研究將挖掘這些資料中的關聯規則。為了達到比較的目的，本研究使用相同的超參數條件，並探討兩者在運行時間、效能等方面的異同。

初步觀察結果顯示，Apriori演算法在小型資料集上的效能相對較佳，運行時間較短。然而，FP-Growth演算法雖然在大型數據集上優於Apriori，但在這個特定情境下花費較多時間建構FP-Tree，導致執行效率稍遜。這些比較結果展示了在實際應用中，選擇適當的演算法需考量資料集的大小和特性，而非一概而論。最終，本研究希望提供資料科學領域實際可行的參考，協助選擇適用於不同情境的資料探勘方法。

**關鍵字**：關聯規則、Apriori、FP\_Growth

1. **緒論**

在當今資訊時代，數據成為企業和組織日常運營的核心資源。然而，隨著數據量的急速增長，如何從大量的數據中提煉出有價值的信息成為一個關鍵挑戰。資料探勘技術應運而生，旨在通過發現數據中的隱藏模式、趨勢和知識，從而為企業制定科學合理的策略提供支持。

1. **動機**

研究動機源於對於資料探勘中Apriori演算法和FP-Growth演算法效能比較的興趣。儘管這兩者在不同情境下皆有優勢，本研究關注在相對小型的實際交易資料集中的表現。Apriori演算法是經典的頻繁項目集挖掘方法，而FP-Growth演算法則以樹結構儲存資料，在大型數據集上表現更為優越。

然而，一般性的觀念可能無法全面理解兩者在實際應用中的優勢。因此，本研究計劃深入探討這兩種演算法在特定資料集上的表現，尤其是在現實且多變的交易資料集中，以提供更具體且實用的見解。這樣的比較不僅有助於深化對兩者特性的理解，也能為資料探勘決策提供更具體的指導，尤其在處理小型但複雜的資料集時。透過這項研究，期望能為資料科學領域中選擇適當演算法提供更多實用的參考依據。

1. **目的**

在當今大數據時代，資料探勘成為了從龐大資料中萃取有價值知識的關鍵技術之一。關聯規則挖掘作為資料探勘的一個重要分支，在商業、市場分析等領域發揮著重要作用。其中，Apriori演算法和FP-Growth演算法是兩種經典且廣泛應用的關聯規則挖掘方法。

然而，這兩種演算法在不同資料集大小和特性下可能表現出不同的效能。因此，本研究旨在深入研究並比較Apriori演算法和FP-Growth演算法在實際應用中的優勢和劣勢。透過探討其理論基礎、演算法原理，以及在交易資料集上的應用，期望能夠提供對於這兩種演算法適用性的更深入了解。

1. **資料集**
2. **真實資料集**
3. 交易資料集介紹

一個人工資料集，有許多發票號碼，相同的發票號碼表示同一次的交易，總計157,397筆資料。每筆資料均包含相應交易所購買的商品資訊。下方顯示本研究所使用之資料集的部分資料以及欄位介紹。

**表1**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| INVOICE\_NO | CUST\_ID | ITEM\_ID | ITEM\_NO | PRODUCT\_TYPE | TRX\_DATE | QUANTITY |
| CX47348203 | 3218 | 3217532 | M25P40-VMN6TPB | MEMORY\_EMBEDED | 2016/7/26 | 2500 |
| CX47346522 | 2470 | 3326781 | AU80610006237AASLBX9 | CPU / MPU | 2016/7/11 | 50 |
| CX47348534 | 16135 | 740487 | MMBD2837LT1G | DISCRETE | 2016/7/27 | 3000 |
| A20160700174 | 999999999 | 3434776 | IHLP1616ABER2R2M11 | PEMCO | 2016/7/29 | 0 |
| CX47346184 | 2356 | 70072 | MMBT3906LT1G | DISCRETE | 2016/7/6 | 12000 |

交易資料集部分資料

**表2**

交易資料集欄位介紹

|  |  |  |
| --- | --- | --- |
| 欄位 | 屬性 | 內容 |
| 0 | INVOICE\_NO | Nominal |
| 1 | CUST\_ID | Continuous |
| 2 | ITEM\_ID | Continuous |
| 3 | ITEM\_NO | Nominal |
| 4 | PRODUCT\_TYPE | CPU / MPU, DISCRETE, PEMCO, LOGIC IC, LINEAR IC, OPTICAL AND SENSOR, CHIPSET / ASP, MEMORY\_EMBEDED, MEMORY\_SYSTEM, OTHERS |
| 5 | TRX\_DATE | Date |
| 6 | QUANTITY | Continuous |

1. **方法**
2. **實作說明**

在判斷關聯規則模型的實驗中，本研究首先將「交易資料集」做數據的前置處理，其包括將同個發票號碼存入同個陣列，並使用Apriori演算法與FP-Growth演算法來探勘出關聯規則的輸出，將此規則輸出至csv存檔。事後再將此關聯規則讀入，將可輸入其中一樣商品，程式會判斷出可能會一起購買的商品。

1. **操作說明**

本研究執行環境皆為Python3.11.5，並且使用 Visual Studio Code與 Jupyter Notebook作為開發工具，利用 pandas與numpy來讀取資料以及做資料的前處理，探勘模型利用mlxtend套件來建構，使用到Apriori演算法與FP-Growth演算法作為輸出關聯規則的演算法。

1. **實驗**
2. **前置處理**

**圖1**

交易資料集前置處理流程圖

1. 檢測缺失值：檢測是否有欄位有缺失值，若有缺失值則移除此列資料。
2. 檢測0值：檢測是否有欄位有0，若有0則移除此列資料。
3. 轉換數據型態：將具有相同發票的內容合併為一個陣列，形成整體的二維陣列。
4. **實驗設計**

本次實驗使用Apriori演算法以及FP-Growth演算法，來推敲關聯規則，並且比較花費的時間。本研究主要有使用兩種超參數，一種是支持度另一個是置信度，設定方式於下方有說明。在最後的結果中是使用Apriori演算法並且使用巢狀迴圈的方式，將支持度數值設定為0.001至0.01每次增加0.001，置信度數值設定為0.008至0.012每次增加0.001，使用迭代的方式，將所有數值測試一遍，並列出min\_support、min\_confidence、rule\_count與execution\_time，將在實驗結果做進一步的展示。

**4.2.1. 推敲關聯規則流程步驟**

下圖(圖3)展示交易資料集在Apriori演算法以及FP-Growth演算法應用上的流程步驟，需比較不同演算法的運行時間。

**圖2**

流程圖

載入資料

訓練模型

比較

運行時間

Apriori

FP-Growth

**4.2.2. 不同演算法超參數設定**

1. Apriori演算法超參數設定方式

使用Apriori演算法挖掘頻繁項目集，將min\_support設定了最小支持度閾值為0.004，表示頻繁項目集至少應在0.4% 的交易中出現。再來使用association\_rules函數生成關聯規則。指定了計算關聯規則的信心度閥值為0.01，表示只保留信心度大於0.01的規則。

1. FP-Growth演算法超參數設定方式

使用FP-Growth演算法挖掘頻繁項目集，將min\_support設定了最小支持度閾值為0.004，表示頻繁項目集至少應在0.4% 的交易中出現。再來使用association\_rules函數生成關聯規則。指定了計算關聯規則的信心度閥值為0.01，表示只保留信心度大於0.01的規則。

**4.2.3. 超參數設定說明**

透過上節(4.2.2)可以得知在Apriori演算法與FP-Growth演算法在於設定超參數的數值幾乎雷同，故在此節多做說明。

1. min\_support與規則數量的關係

隨著支持度的增加，關聯規則數減少，這是因為支持度的遞增會將低於此支持度的規則過濾，因此匹配到的規則會減少。當min\_confidence不改變，min\_support提高時，規則數量將減少。

1. min\_confidence與規則數量的關係

隨著置信度的增加，關聯規則的數量也隨之減少，較高的置信度將過濾更多的規則數。當門檻提高時，符合條件的規則數量減少當min\_support不改變時，min\_confidence提高時，規則數量將減少。

1. **實驗結果**

**4.3.1. 關聯規則分析**

透過下表可以發現，較高的support和confidence可以得到更強烈關連規則，但規則數會減少，較低的支持度可以得到更多的關連規則，但這些規則的強度會較弱。

**表3**

關聯規則分析表

|  |  |  |  |
| --- | --- | --- | --- |
| **min\_support** | **min\_confidence** | **rule\_count** | **execution\_time** |
| 0.001 | 0.008 | 216 | 0.059 |
| 0.001 | 0.009 | 207 | 0.042 |
| 0.001 | 0.010 | 201 | 0.045 |
| 0.001 | 0.011 | 197 | 0.049 |
| 0.002 | 0.008 | 144 | 0.036 |
| 0.002 | 0.009 | 137 | 0.033 |
| 0.002 | 0.010 | 135 | 0.030 |
| 0.002 | 0.011 | 131 | 0.030 |
| 0.003 | 0.008 | 72 | 0.027 |
| 0.003 | 0.009 | 72 | 0.023 |
| 0.003 | 0.010 | 72 | 0.022 |
| 0.003 | 0.011 | 72 | 0.021 |
| 0.004 | 0.008 | 36 | 0.021 |
| 0.004 | 0.009 | 36 | 0.023 |
| 0.004 | 0.010 | 36 | 0.023 |
| 0.004 | 0.011 | 36 | 0.021 |
| 0.005 | 0.008 | 30 | 0.016 |
| 0.005 | 0.009 | 30 | 0.019 |
| 0.005 | 0.010 | 30 | 0.022 |
| 0.005 | 0.011 | 30 | 0.016 |
| 0.006 | 0.008 | 16 | 0.013 |
| 0.006 | 0.009 | 16 | 0.013 |
| 0.006 | 0.010 | 16 | 0.012 |
| (續) | | | |
| (續上表) | | | |
| 0.006 | 0.011 | 16 | 0.014 |
| 0.007 | 0.008 | 14 | 0.014 |
| 0.007 | 0.009 | 14 | 0.013 |
| 0.007 | 0.010 | 14 | 0.013 |
| 0.007 | 0.011 | 14 | 0.012 |
| 0.008 | 0.008 | 12 | 0.012 |
| 0.008 | 0.009 | 12 | 0.014 |
| 0.008 | 0.010 | 12 | 0.012 |
| 0.008 | 0.011 | 12 | 0.014 |
| 0.009 | 0.008 | 12 | 0.013 |
| 0.009 | 0.009 | 12 | 0.012 |
| 0.009 | 0.010 | 12 | 0.011 |
| 0.009 | 0.011 | 12 | 0.011 |

**4.3.2.** **Apriori演算法實驗結果**

1. Apriori演算法關聯規則表

結束了上述實驗設計的部分有得出一個關聯規則表，以下顯示部分關聯規則表之內容。這些規則表示了當antecedents存在時有confidence的機率consequents也會存在，提升度為lift。表示antecedents和confidence之間有輕微正相關性。其他指標提供了規則的額外洞察。

**表4**

Apriori演算法關聯規則表部分內容

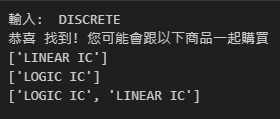
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| **antecedents** | (CHIPSET / ASP) | (CPU / MPU) | (LINEAR IC) | (CPU / MPU) | (LOGIC IC) |
| **consequents** | (CPU / MPU) | (CHIPSET / ASP) | (CPU / MPU) | (LINEAR IC) | (CPU / MPU) |
| **antecedent support** | 0.044112 | 0.110746 | 0.260398 | 0.110746 | 0.121541 |
| **consequent support** | 0.110746 | 0.044112 | 0.110746 | 0.260398 | 0.110746 |
| **support** | 0.005343 | 0.005343 | 0.007672 | 0.007672 | 0.005206 |
| **confidence** | 0.121118 | 0.048243 | 0.029461 | 0.069273 | 0.042831 |
| **lift** | 1.093658 | 1.093658 | 0.266026 | 0.266026 | 0.386754 |
| **leverage** | 0.000458 | 0.000458 | -0.02117 | -0.02117 | -0.00825 |
| **conviction** | 1.011802 | 1.004341 | 0.916248 | 0.79465 | 0.929047 |
| **zhangs\_metric** | 0.089589 | 0.096302 | -0.7886 | -0.75625 | -0.64349 |

1. 輸出結果

本研究將關聯規則存入csv檔之中，再讀取這個csv檔來做一使用者輸入的方式，最終會得出想要的結果，可參考rules.csv檔案內的關聯規則。

**圖3**

Apriori使用者輸出結果



**4.3.2. FP\_growth演算法實驗結果**

1. FP\_growth演算法關聯規則表

結束了上述實驗設計的部分有得出一個關聯規則表，以下顯示部分關聯規則表之內容。這些規則表示了當antecedents存在時有confidence的機率consequents也會存在，提升度為lift。表示antecedents和confidence之間有輕微正相關性。其他指標提供了規則的額外洞察。

**表5**

FP\_growth演算法關聯規則表部分內容

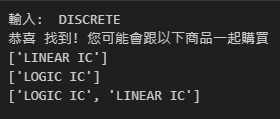
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 |
| **antecedents** | (CHIPSET / ASP) | (CPU / MPU) | (LINEAR IC) | (CPU / MPU) | (LOGIC IC) |
| **consequents** | (CPU / MPU) | (CHIPSET / ASP) | (CPU / MPU) | (LINEAR IC) | (CPU / MPU) |
| **antecedent support** | 0.044112 | 0.110746 | 0.260398 | 0.110746 | 0.121541 |
| **consequent support** | 0.110746 | 0.044112 | 0.110746 | 0.260398 | 0.110746 |
| **support** | 0.005343 | 0.005343 | 0.007672 | 0.007672 | 0.005206 |
| **confidence** | 0.121118 | 0.048243 | 0.029461 | 0.069273 | 0.042831 |
| **lift** | 1.093658 | 1.093658 | 0.266026 | 0.266026 | 0.386754 |
| **leverage** | 0.000458 | 0.000458 | -0.02117 | -0.02117 | -0.00825 |
| **conviction** | 1.011802 | 1.004341 | 0.916248 | 0.79465 | 0.929047 |
| **zhangs\_metric** | 0.089589 | 0.096302 | -0.7886 | -0.75625 | -0.64349 |

1. 輸出結果

本研究將關聯規則存入csv檔之中，再讀取這個csv檔來做一使用者輸入的方式，最終會得出想要的結果，可參考rules.csv檔案內的關聯規則。

**圖4**

Apriori使用者輸出結果



**4.3.3. Apriori演算法與FP\_growth演算法比較差異**

透過這個資料集可以發現對於Apriori演算法的處理效率會是較佳的。由於這個資料集是屬於較小的資料集，不太適合使用FP\_Growth演算法。

FP\_Growth演算法會比Aapriori演算法在大規模數據上效能還好，因為FP\_Growth只需要有兩次掃描，一次建構FP-tree，依次用於挖掘頻繁集建構FP-tree時也減少了候選集，相較之下Apriori需要多次掃描。

時間上的差異會發現Apriori演算法是屬於速度較快的，不過此資料集對於FP\_Growth演算法來說數據集較小，在數據集小的情況下建出fp-tree，在少量頻繁集的情況下不需要建構這麼複雜的結構，在簡單數據集的情況下建立如此複查的樹將會花費相當多的時間，也是因為這樣，在此數據集中，FP\_Growth會比Apriori花費更多執行時間。下表來顯示他們的差異。

**表6**

Apriori與FP\_Growth運行時間差異表

|  |  |  |
| --- | --- | --- |
|  | Apriori | FP\_Growth |
| time(s) | 0.0296 | 0.2372 |

1. **結論**

在這份研究中，運用一個包含157,397筆資料的人工交易資料集，每筆交易包括發票號碼、客戶ID、商品ID、商品編號、產品類型、交易日期和數量等相關資訊。為了進行關聯規則的探勘，本研究先進行了資料的前置處理，將相同發票號碼的交易整合成同一陣列。

接著，本研究使用了Apriori演算法和FP-Growth演算法，兩者皆是常見的關聯規則探勘方法。實驗中，調整了超參數，包括支持度和置信度，以比較兩種演算法的效能。

本研究發現使用Apriori演算法的過程中，成功地挖掘出了資料集中的關聯規則。值得注意的是，由於資料集相對較小，Apriori演算法在效能上呈現優越表現，達到了更短的運行時間。這表明Apriori演算法對於小型資料集的處理速度相對較快，是一個有效的選擇。FP-Growth演算法在大規模數據集上擁有優越的效能，然而在這個相對較小的資料集中，FP-Growth演算法的效能稍遜，主要原因在於建構FP-Tree所花費的時間相對較多。這顯示FP-Growth演算法在處理小型資料集時可能會因為建構FP-Tree的成本而不如Apriori演算法效率高。

綜合實驗結果，得知對於這個規模的資料集，Apriori演算法在效能上具有優勢。然而，這同時提醒所有人在不同資料集大小和特性下，選擇適當的演算法和超參數是至關重要的。因此，演算法的選擇應該依賴於資料集的規模和特性，以達到最佳的探勘效果。

1. **參考文獻**

andy6804tw (2021年04月02日)。[機器學習] 離群值處理。https://andy6804tw.github.io/2021/04/02/python-outliers-clean/

Renesh Bedre (2023, Feburary 6). DBSCAN clustering algorithm in Python (with example dataset). https://www.reneshbedre.com/blog/dbscan-python.html