資料探勘 Project1

工資所 辛政達 R36074121

1. 資料集說明

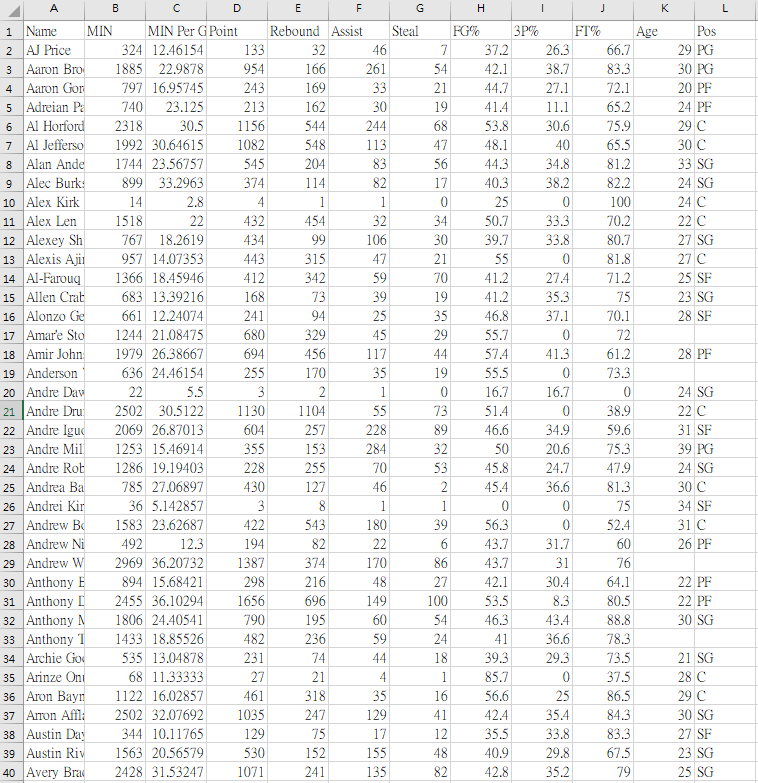
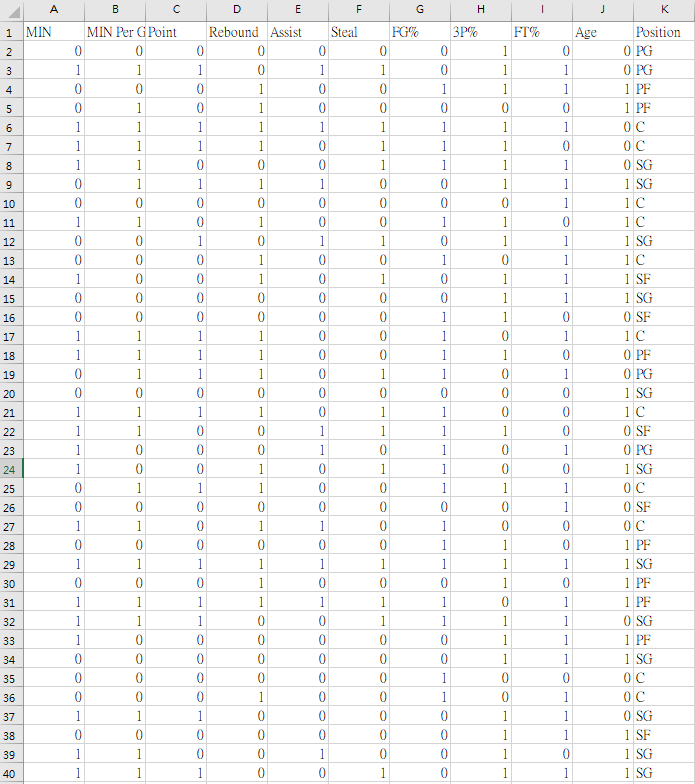
本次作業分別從kaggle.com以及IBM Quest Data Generator取得資料集。

1. Kaggle資料集

從kaggle.com上取得了2014~2015球季的NBA球員數據(nba-players-stats-20142015)，由於此檔案並非關聯式數據，因此對資料做了處理，處理方式為將每一位球員的某項目數據與該項目平均數值做比較，若該球員的數據高於平均，則該項目數值轉換為1反之則為0(binary dataset)。

此資料集總共收錄了491位球員的11項不同數據。

資料如下圖所示，左圖為轉換前球員數據，右圖為轉換後球員數據。

1. IBM Quest Data Generator

使用IBM Quest Data Generator產生資料集，總交易數量為3000筆、每筆交易平均交易10個商品，總共有50個不同商品，輸出資料為兩欄屬性(交易編號與商品編號)的txt檔。

指令如下所示:

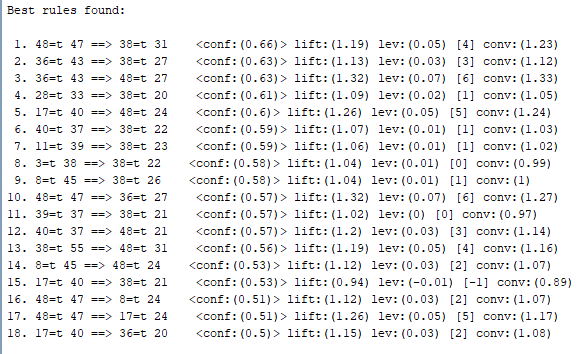


1. 演算法的實作與驗證

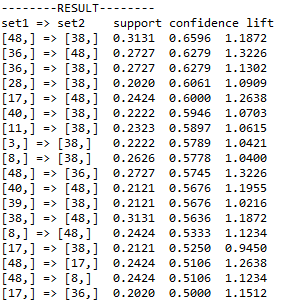
本次作業主要以JAVA進行程式的撰寫，在參考了上課內容以集網路上相關的資源後，分別實作了Apriori以及FPGrowth兩支主程式，在讀檔類型方面則做了讀取CSV和TXT兩種檔案類型的設計，以滿足讀取kaggle和IBM Quest Data Generator資料集的需求。

為了驗證實作的兩個演算法之正確性，我使用IBM Quest Data Generator產生了1筆小量的數據(-ntrans 0.1 -nitems 0.05)，並在設定程式參數support=0.1，confidence=0.6的情況下，將此資料套用至WEKA和自製的Apriori以及FPGrowth演算法中進行比較，結果如下圖所示。

* WEKA-Apriori



* 自製Apriori



在FPGrowth演算法上也有相同的結果，說明了自製的演算法有正確的演算能力。

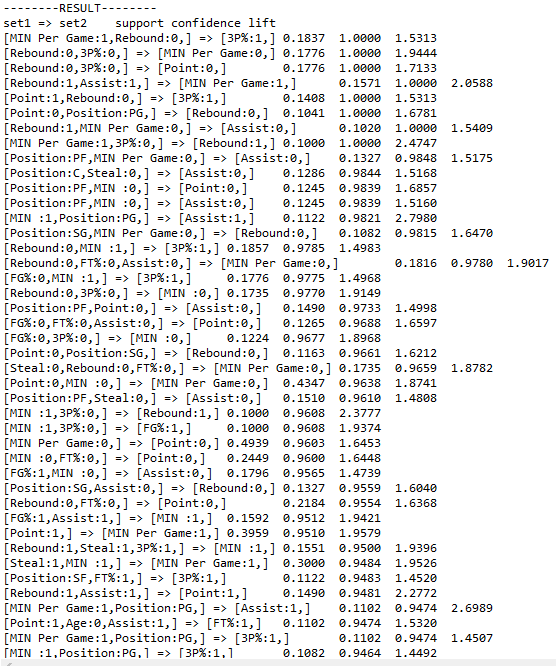
1. 演算法比較

理論上，Apriori演算法需要多次掃描數據資料庫，因此對於大規模數據執行效率很低，而FPGrowth演算法則只需掃描數據資料庫2次，因此對於大規模數據而言會有較優秀的執行效率。

而在實作方面，我分別對IBM Quest Data Generator和kaggle的資料進行測試，並利用簡單的計時工具來進行執行效率的比較。

在IBM Quest Data Generator產生的3000筆交易資料的部分，我使用support=0.1、confident=0.6來進行執行效率的測試，從按下執行到跳出完整的27條關連規則，使用Apriori演算法花費的時間為38.26secs，而使用FPGrowth演算法則只需要 32.08secs，此結果與理論相符。

在kaggle資料集的部分，我使用support=0.1、confident=0.9來進行測試，此次的測試結果也與理論相符，並且更為顯著，在使用Apriori演算法的情況下花費了58.16secs跳出240條關連規則，而FPGrowth演算法則僅花了32.10secs就完成了測試，在此附上執行結果的部分截圖。



此外，值得一提的是，同樣的數據我將之丟入WEKA進行Apriori與FPGrowth演算法的運算，從執行到結果輸出大概只需要2secs的時間，在執行效率上完勝自製的驗算法，這也表示我完成的演算法有很大的優化空間。

1. 小結

這一次的作業我認為最困難的部分莫過於程式撰寫的部分，過去從來沒有獨自完成一個大型程式的我，這一次要搞懂演算法的運作方式並實做出來，在過程中碰上了許多麻煩跟困擾，好在最後透過各方面的協助與網路資源還是完成了。

若往後有機會可以延伸這一次作業的內容，我認為我會主要著重在兩點，第一點我會在測試內容裡面增加記憶體量測的部分，並且處理更大量的數據，如此一來相信能夠更加清楚的認知兩演算法的不同之處。第二點，我會著重在程式優化的部分，目前的程式在讀檔和數據的處理都不甚完善，還有進步的空間。

整體而言我認為這一次的作業讓我對於Apriori和FPGrowth演算法有了更深入的了解，在程式撰寫能力這方面也有所成長，同時也期許自己能夠持續的進步!