資料探勘 – Project1 Report

- 實作 Apriori 和 FP-growth 兩種方法來挑出 Best Rules
- Dataset 使用 Kaggle dataset 和 IBM quest data
- 建立 Dataset 的 arff檔
- 將結果與 SPMF (an open-source data mining library) 作比較

目錄

Dataset

Kaggle dataset: mushrooms

```
data/mushrooms.csv
```

IBM quest data:

```
data/pat.ntrans_5.tlen_10.nitems_0.02
```

Setup

語言：python3

需安裝套件: liac-arff

```
pip install liac-arff
```

Run Process

Apriori:

```
python3 apriori.py [data_type] [min_support] [min_confidence]
```

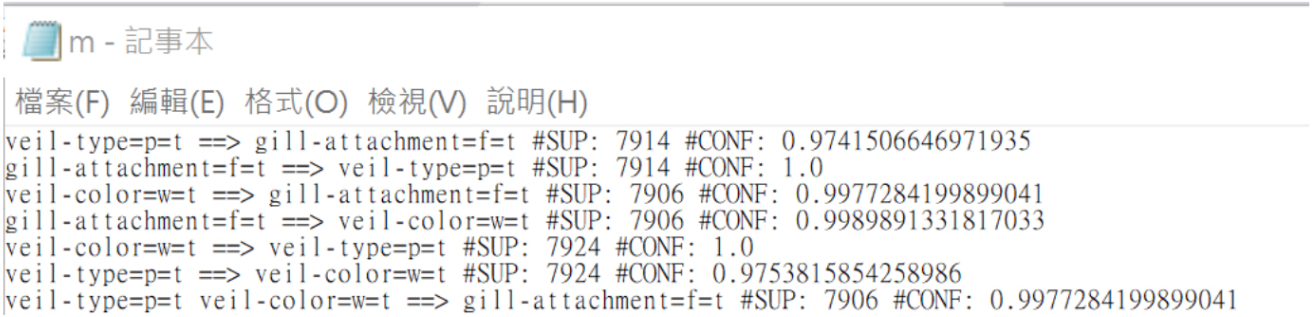
- [data_type] = 0 --> 使用 Kaggle dataset
  [data_type] = 1 --> 使用 IBM quest data
- [min_support] 和 [min_confidence] 的值都介於0~1

執行結果：

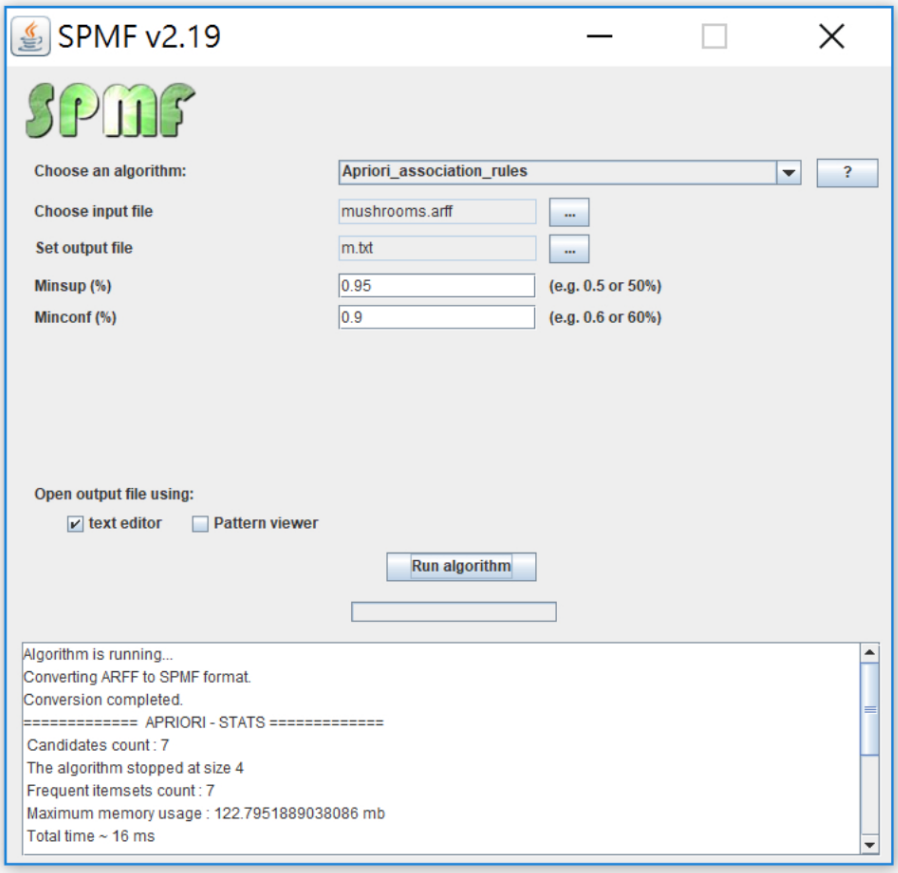
1. 使用自己的演算法套用在kaggle dataset

```
annie  project1  python3 apriori.py 0 0.95 0.9
Min Support: 0.95
Min Confidence: 0.9

Best rules:
1. gill-attachment:f -> veil-color:w   [ conf: 0.999 ]
2. veil-color:w -> gill-attachment:f   [ conf: 0.998 ]
3. gill-attachment:f -> veil-type:p   [ conf: 1.000 ]
4. veil-type:p -> gill-attachment:f   [ conf: 0.974 ]
5. veil-color:w -> veil-type:p   [ conf: 1.000 ]
6. veil-type:p -> veil-color:w   [ conf: 0.975 ]
7. veil-color:w -> gill-attachment:f, veil-type:p   [ conf: 0.998 ]
8. gill-attachment:f -> veil-type:p, veil-color:w   [ conf: 0.999 ]
9. veil-type:p -> gill-attachment:f, veil-color:w   [ conf: 0.973 ]
10. gill-attachment:f, veil-color:w -> veil-type:p   [ conf: 1.000 ]
11. veil-type:p, veil-color:w -> gill-attachment:f   [ conf: 0.998 ]
12. gill-attachment:f, veil-type:p -> veil-color:w   [ conf: 0.999 ]

Process time: 0.3832848072052002 seconds
```

2. 使用spmf，套用在kaggle dataset



```
gill-attachment=f=t veil-color=w=t ==> veil-type=p=t #SUP: 7906 #CONF: 1.0
gill-attachment=f=t veil-type=p=t ==> veil-color=w=t #SUP: 7906 #CONF: 0.9989891331817033
veil-color=w=t ==> gill-attachment=f=t veil-type=p=t #SUP: 7906 #CONF: 0.9977284199899041
veil-type=p=t ==> gill-attachment=f=t veil-color=w=t #SUP: 7906 #CONF: 0.9731659281142294
gill-attachment=f=t ==> veil-type=p=t veil-color=w=t #SUP: 7906 #CONF: 0.9989891331817033
```

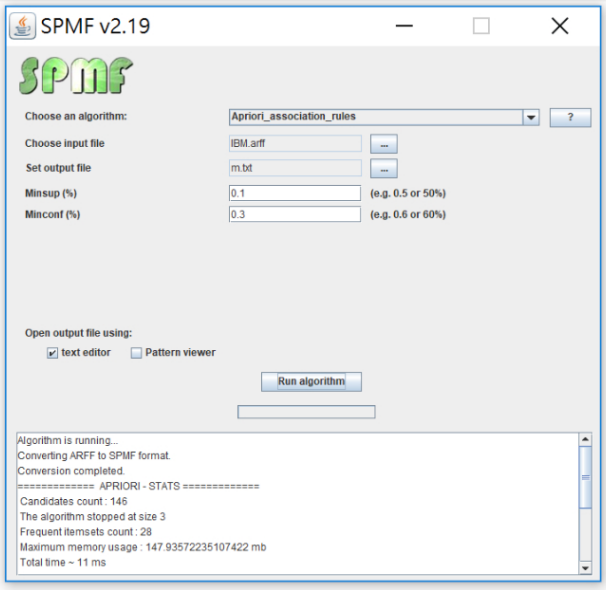


3. 使用spmf，套用在IBM dataset

m - 記事本

檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)

```
9=t ==> 8=t #SUP: 371 #CONF: 0.3260105448154657
14=t ==> 8=t #SUP: 391 #CONF: 0.3709677419354839
11=t ==> 8=t #SUP: 481 #CONF: 0.3465417867435158
8=t ==> 11=t #SUP: 481 #CONF: 0.3646702047005307
17=t ==> 8=t #SUP: 465 #CONF: 0.3705179282868526
8=t ==> 17=t #SUP: 465 #CONF: 0.3525398028809704
3=t ==> 8=t #SUP: 417 #CONF: 0.347210657785179
8=t ==> 3=t #SUP: 417 #CONF: 0.3161485974222896
9=t ==> 11=t #SUP: 394 #CONF: 0.3462214411247803
17=t ==> 9=t #SUP: 392 #CONF: 0.31235059760956174
9=t ==> 17=t #SUP: 392 #CONF: 0.3444639718804921
14=t ==> 11=t #SUP: 385 #CONF: 0.36527514231499053
3=t ==> 14=t #SUP: 371 #CONF: 0.3089092422980849
14=t ==> 3=t #SUP: 371 #CONF: 0.3519924098671727
17=t ==> 11=t #SUP: 443 #CONF: 0.3529880478087649
11=t ==> 17=t #SUP: 443 #CONF: 0.319164265129683
3=t ==> 11=t #SUP: 463 #CONF: 0.3855120732722731
11=t ==> 3=t #SUP: 463 #CONF: 0.3335734870317003
3=t ==> 17=t #SUP: 403 #CONF: 0.33555370524562866
17=t ==> 3=t #SUP: 403 #CONF: 0.3211155378486056
```



FP-growth:

```
python3 fp_growth.py [data_type] [min_support] [min_confidence]
```

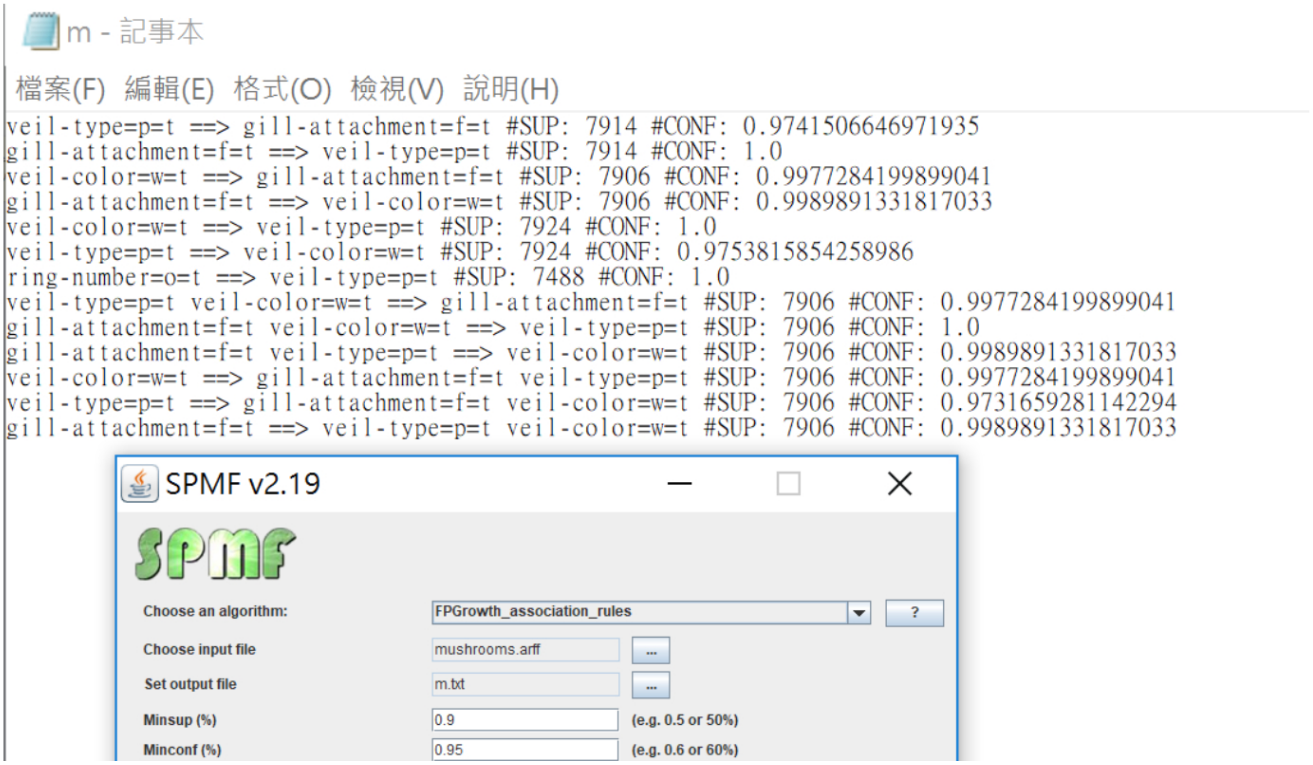
- [data_type] = 0 --> 使用 Kaggle dataset
  [data_type] = 1 --> 使用 IBM quest data
- [min_support] 和 [min_confidence] 的值都介於0~1

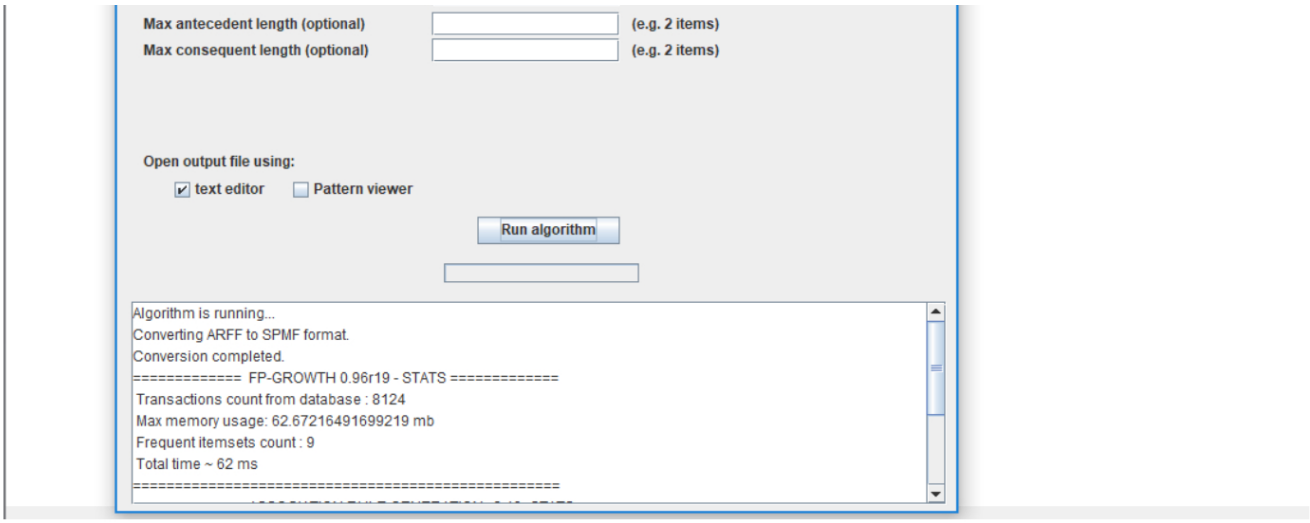
執行結果：

1. 使用自己的演算法套用在kaggle dataset (fp-growth)

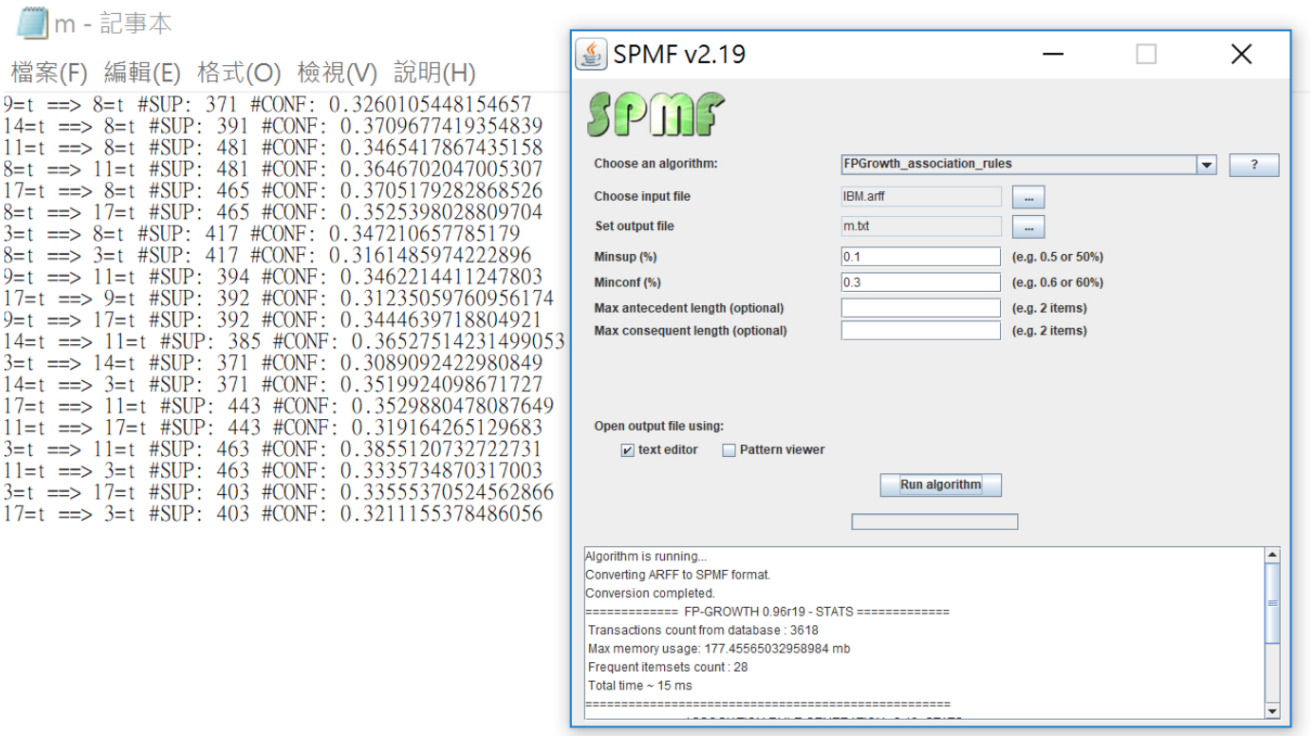
```
annie project1 python3 fp_growth.py 0 0.95 0.9
Min Support: 0.95
Min Confidence: 0.9

Best rules:
1. gill-attachment:f -> veil-color:w   [ conf: 0.999 ]
2. veil-color:w -> gill-attachment:f   [ conf: 0.998 ]
3. veil-type:p -> gill-attachment:f   [ conf: 0.974 ]
4. gill-attachment:f -> veil-type:p   [ conf: 1.000 ]
5. veil-type:p -> veil-color:w   [ conf: 0.975 ]
6. veil-color:w -> veil-type:p   [ conf: 1.000 ]
7. veil-type:p -> gill-attachment:f, veil-color:w   [ conf: 0.973 ]
8. gill-attachment:f -> veil-type:p, veil-color:w   [ conf: 0.999 ]
9. veil-color:w -> veil-type:p, gill-attachment:f   [ conf: 0.998 ]
10. veil-type:p, gill-attachment:f -> veil-color:w   [ conf: 0.999 ]
11. veil-type:p, veil-color:w -> gill-attachment:f   [ conf: 0.998 ]
12. gill-attachment:f, veil-color:w -> veil-type:p   [ conf: 1.000 ]

Process time: 0.22733807563781738 seconds
```

2. 使用spmf，套用在kaggle dataset





3. 使用spmf，套用在IBM dataset (fp-growth)



產生 Kaggle dataset 的 arff 檔:

```
python3 mushroom_to_arff.py
```

比較＆結論：

m=0.95 c=0.9	Apriori (spmf)	Apriori (自己)	FP-Growth (spmf)	FP-Growth (自己)
Kaggle	0.02s	0.38s	0.06s	0.23s
IBM	0.01	0.12	0.01	0.11

IBM	0.01s	0.10s	0.01s	0.11s

m=0.1 c=0.3	**Apriori (spmf)**	**Apriori (自己)**	**FP-Growth (spmf)**	**FP-Growth (自己)**
Kaggle	10min	> 1hr	6min	> 1hr
IBM	0.1s	0.23s	0.01s	> 1hr

[比較1] 在同個演算法下

- 在同個演算法之下，不論是 Apriori 還是 FP-Growth ，都可以看出IBM的資料會比 Kaggle 來的快，我想這應該是因為:
  1. IBM有3600多筆data，並且平均transactions長度只有10，而kaggle則有8000多筆的資料，以及長度為23的transaction。
  2. IBM的資料分布比較稀疏，比較不會有長度較長的frequent itemSet，所以在這塊IBM的資料刪減的會比較快。

[比較2] 在同個 dataSet 下

- 除了很明顯地可以看出由套件跑出的速度會比我自己所寫的來的快以外，還可以看出，在 min_support 以及 confidence 高的時候，fp-growth 並不會比 Apriori 來的快，但是在資料是稀疏以及min_support 小的時候，fp-growth 的效率會比Apriori高上許多。

[比較3] 在不同min_support / confidence下

- 可以看出在 IBM 的資料下所花得時間其實不會差太多，但在 kaggle 的資料就會差上許多，我想這是同樣是因為IBM的資料是稀疏的，刪減後所保留的 dataSet 會下降得很快，而 kaggle 的則相反，由於 min_support太低，刪減後大多數的 itemSet 會繼續保留下來，因此整個時間大幅度的上升。