使用 Python 撰寫程式實現資料挖掘

Data mining with Python

作者：陳宣儒、孫振博

系級：資訊四甲、資訊四丁

學號：D0641962、D0640281

開課老師：林明言

課程名稱：資料倉儲與資料挖掘

開課系所：資電學院 資訊工程學系

開課學年： 109 學年度 第 1 學期

中文摘要

資料挖掘為對極龐大資料的探詢，探詢當中蘊含的規則、模式的方法。而會探究這樣的方法便是因為現今我們能夠蒐集到的資訊是極為龐大的，而想透過人腦直接利用這龐大的資料，讓人一眼便看出當中的規律在數據龐大之下是幾乎不可實現的。而相對的，電腦則是處理龐大數據的行家，因此才會想研究資料挖掘在電腦上的實現方法。

而在現今便有著許多方法被討論出來，其中我們想實現的一是 Apriori 先驗演算法，其方法能讓我們快速的找出資料當中的 frequient patterns 和其 association rules，二則為 DNN 模型，透過訓練模型，試著對資料進行預測分類。

在 Python 上循著這兩種方法實作出可執行的程式後，我們最後可以透過這兩者從資料內尋找到一定的規則。對於找到的規則，有助於我們更瞭解資料間的關係，從中推斷出有利的資訊從而利用之。

關鍵字：Apriori、Data mining、DNN、Python

Abstract

Data mining is a method of discovering patterns in large data sets.We explore such a method because the information we can collect today is extremely large, and if you want to directly use this huge data through the human brain. It is almost impossible under the huge amount of data.On the other hand, computers are experts at handling huge amounts of data, so we want to study the implementation of data mining on computers.

Nowadays, many methods have been discussed. Two of the methods we want to implement are Apriori algorithm and DNN model. Apriori algorithm, which allows us to quickly find the frequent patterns and its association rules in the data sets. DNN model. Train the model and try to predict and classify the data.

After we follow these two methods to implement executable programs on Python, we can finally find certain rules from the data through these two methods. The rules found will help us better understand the relationship between the data, infer beneficial information from it and use it.

Keyword：Apriori、Data mining、DNN、Python

目　次

第一章 目標

1.1 研究背景

在極大量的資料中，我們可以探詢到某種規則、模式存在於這些資料內，而這種探詢資料模式的行為被稱為資料挖掘。透過利用這樣的方法，我們可以在很多事情上獲得啟發，像是在銷售、貿易上的運用，或是人類行為、心理的評估，亦或是股價的跌幅估計，尤其在現今資訊量爆炸性增長的情況下，這種評估精準度與資料量成一定正比關係的方法，也是隨之被愈發的重視，有著許多的論文探討相關的資料、方法。因此，我們想試著從學到的方法中，嘗試實現對資料的挖掘。

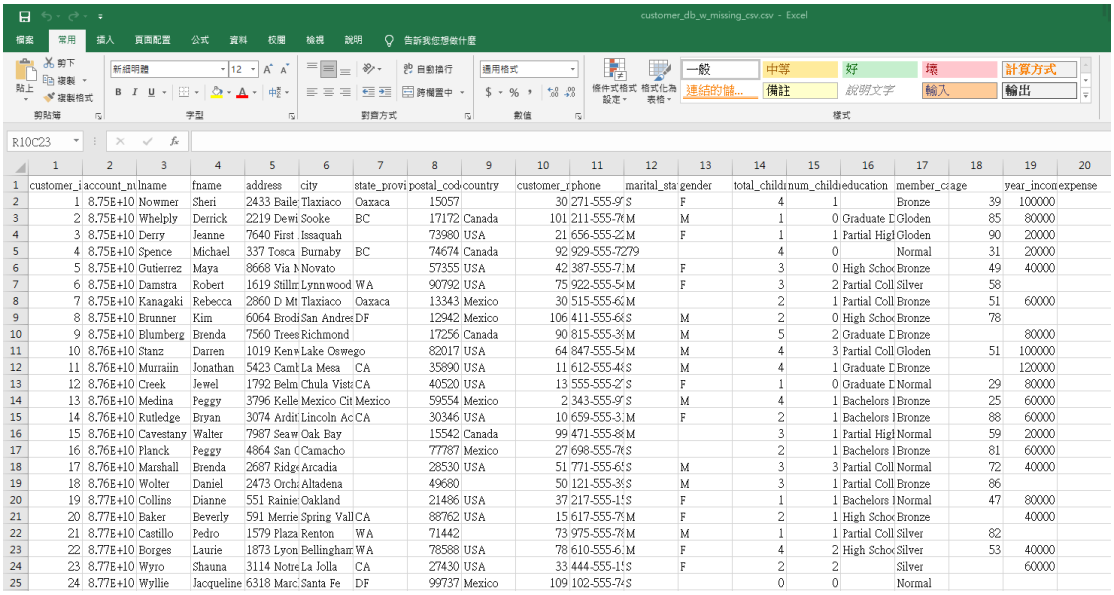
1.2 研究目標

透過對專題的討論，我們想透過 Python 來實現資料挖掘的程式，而主要實現的是兩個，一為使用 Apriori 先驗演算法來對資料進行 frequitent patterns 和其 association rules 上的搜尋，二為將資料投入去訓練 DNN 模型，嘗試使用訓練出的模型進行分類預測。

第二章 資料敘述

2.1 資料來源

此次資料來源 ilearn 2.0 上[資料倉儲與資料挖掘]課程內，林明言老師提供的資料集 customer_db_w_missing_csv.csv，其內部資料為辦卡顧客的資料當中有著 customer_id，account_num，lname　fname，address，city，state_province，postal_code，country，customer_region_id，phone，marital_status，gender，total_children，num_children_at_home，education，member_card，age，year_income，expense 這些欄位，其中還有缺漏資料，缺漏資料將會在後續章節中進行處理。



	customer_i	account_nu	lname	fname	address	city	state_provi	postal_cod	country	customer_r	phone	marital_sta	gender	total_childi	num_childi	education	member_c	age	year_incon	expense
2	1	8.75E+10	Nowmer	Sheri	2433 Baile	Tlaxiaco	Oaxaca	15057		30	271-555-9	S	F	4	1		Bronze	39	100000	
3	2	8.75E+10	Whelply	Derrick	2219 Dewi	Sooke	BC	17172	Canada	101	211-555-7	M	M	1	0	Graduate D	Gloden	85	80000	
4	3	8.75E+10	Derry	Jeanne	7640 First	Issaquah		73980	USA	21	656-555-2	M	F	1	1	Partial Higl	Gloden	90	20000	
5	4	8.75E+10	Spence	Michael	337 Tosca	Burnaby	BC	74674	Canada	92	929-555-7279			4	0		Normal	31	20000	
6	5	8.75E+10	Gutierrez	Maya	8668 Via N	Novato		57355	USA	42	387-555-7	M	F	3	0	High Schoo	Bronze	49	40000	
7	6	8.75E+10	Damstra	Robert	1619 Stillm	Lynnwood	WA	90792	USA	75	922-555-5	M	F	3	2	Partial Coll	Silver	58		
8	7	8.75E+10	Kanagaki	Rebecca	2860 D Mi	Tlaxiaco	Oaxaca	13343	Mexico	30	515-555-6	M		2	1	Partial Coll	Bronze	51	60000	
9	8	8.75E+10	Brunner	Kim	6064 Brodi	San Andres	DF	12942	Mexico	106	411-555-6	S	M	2	0	High Schoo	Bronze	78		
10	9	8.75E+10	Blumberg	Brenda	7560 Trees	Richmond		17256	Canada	90	815-555-3	M	M	5	2	Graduate D	Bronze		80000	
11	10	8.76E+10	Stanz	Darren	1019 Kenw	Lake Oswego		82017	USA	64	847-555-5	M	M	4	3	Partial Coll	Gloden	51	100000	
12	11	8.76E+10	Murraiin	Jonathan	5423 Camt	La Mesa	CA	35890	USA	11	612-555-4	S	M	4	1	Graduate D	Bronze		120000	
13	12	8.76E+10	Creek	Jewel	1792 Belm	Chula Vista	CA	40520	USA	13	555-555-2	S	F	1	0	Graduate D	Normal	29	80000	
14	13	8.76E+10	Medina	Peggy	3796 Kelle	Mexico Cit	Mexico	59554	Mexico	2	343-555-9	S	M	4	1	Bachelors I	Bronze	25	60000	
15	14	8.76E+10	Rutledge	Bryan	3074 Ardit	Lincoln Ac	CA	30346	USA	10	659-555-3	M	F	2	1	Bachelors I	Bronze	88	60000	
16	15	8.76E+10	Cavestany	Walter	7987 Seaw	Oak Bay		15542	Canada	99	471-555-8	M		3	1	Partial Higl	Normal	59	20000	
17	16	8.76E+10	Planck	Peggy	4864 San C	Camacho		77787	Mexico	27	698-555-7	S		2	1	Bachelors I	Bronze	81	60000	
18	17	8.76E+10	Marshall	Brenda	2687 Ridge	Arcadia		28530	USA	51	771-555-6	S	M	3	3	Partial Coll	Normal	72	40000	
19	18	8.76E+10	Wolter	Daniel	2473 Orch	Altadena		49680		50	121-555-3	S	M	3	1	Partial Coll	Bronze	86		
20	19	8.77E+10	Collins	Dianne	551 Rainie	Oakland		21486	USA	37	217-555-1	S	F	1	1	Bachelors I	Normal	47	80000	
21	20	8.77E+10	Baker	Beverly	591 Merrie	Spring Vall	CA	88762	USA	15	617-555-7	M	F	2	1	High Schoo	Bronze		40000	
22	21	8.77E+10	Castillo	Pedro	1579 Plaza	Renton	WA	71442		73	975-555-7	M	M	1	1	Partial Coll	Silver	82		
23	22	8.77E+10	Borges	Laurie	1873 Lyon	Bellingham	WA	78588	USA	78	610-555-6	M	F	4	2	High Schoo	Silver	53	40000	
24	23	8.77E+10	Wyro	Shauna	3114 Notre	La Jolla	CA	27430	USA	33	444-555-1	S	F	2	2		Silver		60000	
25	24	8.77E+10	Wyllie	Jacqueline	6318 Marc	Santa Fe	DF	99737	Mexico	109	102-555-7	S		0	0		Normal			

2.2 感興趣資料

要想找出符合個人目的的資料模式，就必須對要使用資料的納入做出考量，基於這樣的考量，在後面進行 Apriori 和 DNN 模型訓練時主要選擇使用了以下的屬性 total_children，num_children_at_home，education，member_card，year_income，主觀上認為辦卡上的等級應該是與一個人的年收、教育和從家庭撫養孩子數量考量家庭撫養能力等有關，而像是 customer_id 等其它無關屬性則會在後續實際操作時不列入考量。

第三章 工具敘述

3.1 Google Colaboratory

首先在環境上選擇使用了由 Google 提供的 Python 雲端開發環境 Colaboratory，在其內早已內建好大部分使用者經常使用到的函式庫，可以把程式儲存在 Google Drive 上隨使取用，經過共享與組員方便探討程式碼，且在其上環境又是共通的，不會出現在他人電腦可以，這邊電腦卻不行的情況。

圖 3. 1 Colaboratory Logo

3.2 Pandas

Pandas 是 python 的一個開源數據分析函式庫，提供高效能、簡易使用的資料格式(Data Frame)讓使用者可以快速操作及分析資料。

圖 3. 2 pandas Logo

3.3 MLXTEND

MLXTEND 是一個基於 Python 的開源項目，在當中提供一些數據分析的工具，其中提供了此一些函式，讓使用者可以簡單實現 Frequent Patterns 的一些演算法。

圖 3. 3 MLXTEND Logo

3.4 Keras

Keras 是一個在深度學習上目前流行的開放程式庫，當中有個很多深度學習相關的 API 存在，為使用 Python 編寫，網上存在很多相關的教程、資源可以學習，本次 DNN 模型便是使用其實現。

圖 3. 4 Keras Logo

第四章 探勘過程

4.1 Apriori 實現方法

本次實現 Apriori 是在 Colaboratory 上使用 Python 去實作的。為對資料進行 Apriori 演篡法的實現，首先便是要載入資料，這邊主要是透過 Colaboratory 有的 files.upload 方法去載入，可以看到其很簡單便可實現資料的載入。

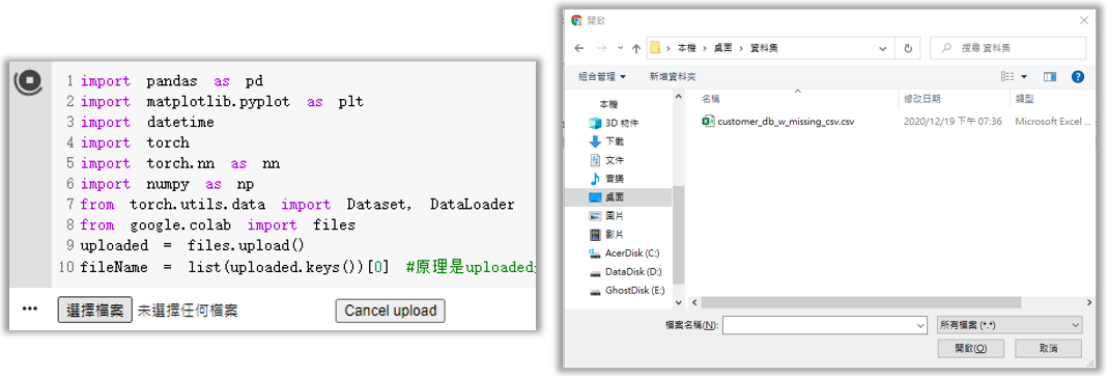


圖 4. 1 files.upload()介面

載入的 csv 資料最後是交給 Pandas 去進行讀入，因 Pandas 在 csv 上提供很多實用方法可在接下來的操作中被使用到。

因為只對當中部分的資料有興趣，所以要先將感興趣的資料標記出來，這邊寫了一介面可以透過勾選的方式簡單的勾選起來感興趣資料，方便隨時修改，而後便可透過 Pandas 將其擷取出來。

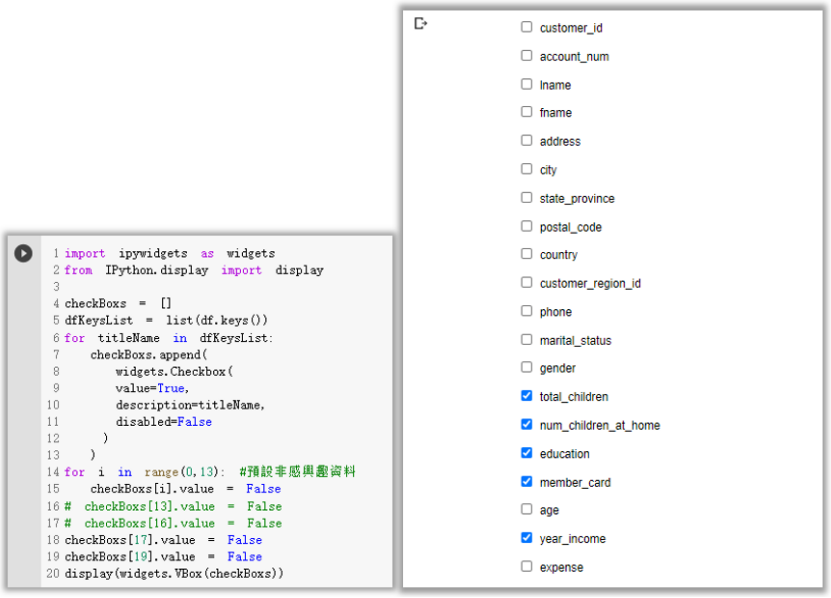


圖 4. 2 感興趣資料勾選

透過 Pandas 將剛才的感興趣資料印出後可看到如圖 4.3 的內容，而觀察它可以發現當中有些缺失資料，所以後續開始進行資料清洗的動作。

```
InterestedData = []
for i in range(0,len(dfKeysList)):
    if (checkBoxs[i].value == True):
        InterestedData.append(checkBoxs[i].description)
df[InterestedData].head(10)
```

	total_children	num_children_at_home	education	member_card	year_income
0	4	1	NaN	Bronze	100000.0
1	1	0	Graduate Degree	Gloden	80000.0
2	1	1	Partial High School	Gloden	20000.0
3	4	0	NaN	Normal	20000.0
4	3	0	High School Degree	Bronze	40000.0
5	3	2	Partial College	Silver	NaN
6	2	1	Partial College	Bronze	60000.0
7	2	0	High School Degree	Bronze	NaN
8	5	2	Graduate Degree	Bronze	80000.0
9	4	3	Partial College	Gloden	100000.0

圖 4.3 感興趣資料

透過 Pandas 中的 dropna()方法，再把資料印出便可看到剛剛含有缺失的資料已經被去除了，剩下 7000 多筆資料。

```
cutedDf = df[InterestedData].dropna()
cutedDf
```

	total_children	num_children_at_home	education	member_card	year_income
1	1	0	Graduate Degree	Gloden	80000.0
2	1	1	Partial High School	Gloden	20000.0
4	3	0	High School Degree	Bronze	40000.0
6	2	1	Partial College	Bronze	60000.0
8	5	2	Graduate Degree	Bronze	80000.0
...	...	...	...	...	...
10273	4	2	High School Degree	Normal	20000.0
10275	0	0	Partial College	Silver	40000.0
10276	4	3	Partial College	Gloden	40000.0
10277	0	0	Partial High School	Normal	20000.0
10280	5	5	Graduate Degree	Gloden	160000.0

7147 rows × 5 columns

圖 4.4 清洗後資料

後續為使用 Mlxtend 來實現 Apriori，必須先將資料轉化為 OneHotEncode 形式，才能符合輸入的條件，於是透過 Pandas，先使用 astype 方法將所有數值先轉換為 object 型態，在使用 get_dummies 方法便可以順利將資料轉換成 OneHotEncode 形式。

```
cutedDf = cutedDf.astype('object')
OneHotEncoderCutedDf = pd.get_dummies(cutedDf)
OneHotEncoderCutedDf
```

	total_children_0	total_children_1	total_children_2	total_children_3	total_children_4	total_children_5	num_children_at_home_0
1	0	1	0	0	0	0	1
2	0	1	0	0	0	0	0
4	0	0	0	1	0	0	1
6	0	0	1	0	0	0	0
8	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...
10273	0	0	0	0	1	0	0
10275	1	0	0	0	0	0	1
10276	0	0	0	0	1	0	0
10277	1	0	0	0	0	0	1
10280	0	0	0	0	0	1	0

7147 rows × 29 columns

圖 4. 5 OneHotEncode 形式資料

再來便可使用 Mlxtend 的 apriori 來使用 apriori 了，如圖 4.6 輸入找尋最小 Itemset 須為 2 以上和 MinSupport 為 0.1 便可得到 frequent pattern 如圖 4.7 的結果。

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

FrequentItemsetNumLimit = 2 #設定找尋的Itemset最小大小
FrequentItemsetMinSupport = 0.1 #設定MinSupport


FrequentItemset = apriori(OneHotEncoderCutedDf, min_support = FrequentItemsetMinSupport, use_colnames=True)
FrequentItemset['length'] = FrequentItemset['itemsets'].apply(lambda x: len(x))
FrequentItemset['absolute support'] = FrequentItemset['support'].apply(lambda x: x * len(OneHotEncoderCutedDf))
LastCol = FrequentItemset.pop(FrequentItemset.columns[-1])
FrequentItemset.insert(1, LastCol.name, LastCol)
FrequentItemset[FrequentItemset['length'] >= FrequentItemsetNumLimit]
```

圖 4. 6 Apriori 程式碼-1

	support	absolute support	itemsets	length
22	0.118931	850.0	(member_card_Bronze, total_children_1)	2
23	0.102561	733.0	(total_children_2, num_children_at_home_1)	2
24	0.117252	838.0	(member_card_Bronze, total_children_2)	2
25	0.100602	719.0	(member_card_Bronze, total_children_3)	2
26	0.187631	1341.0	(member_card_Bronze, num_children_at_home_0)	2
27	0.107598	769.0	(year_income_40000.0, num_children_at_home_0)	2
28	0.206660	1477.0	(member_card_Bronze, num_children_at_home_1)	2
29	0.122709	877.0	(year_income_40000.0, num_children_at_home_1)	2
30	0.132923	950.0	(member_card_Bronze, education_Bachelors Degree)	2
31	0.131524	940.0	(year_income_60000.0, education_Bachelors Degree)	2
32	0.132923	950.0	(education_High School Degree, member_card_Bro...	2
33	0.144536	1033.0	(education_High School Degree, year_income_400...	2
34	0.140479	1004.0	(member_card_Bronze, education_Partial College)	2
35	0.179236	1281.0	(year_income_40000.0, education_Partial College)	2
36	0.203442	1454.0	(member_card_Bronze, year_income_40000.0)	2
37	0.103120	737.0	(member_card_Bronze, year_income_60000.0)	2
38	0.103540	740.0	(member_card_Bronze, year_income_40000.0, educ...	3

圖 4. 7 Apriori 程式碼結果-1

也可改變程式碼，改透過 Absolute support 去得出結果。

```
FrequentItemsetNumLimit = 2 #設定找尋的Itemset最小大小
FrequentItemsetAbsoluteMinSupport = 800 #設定AbsoluteMinSupport

FrequentItemset = apriori(OneHotEncoderCutedDf, min_support = FrequentItemsetAbsoluteMinSupport / len(OneHotEncoderCutedDf) , use_colnames=True)
```

圖 4. 8 Apriori 程式碼-2

	support	absolute support	itemsets	length
19	0.118931	850.0	(member_card_Bronze, total_children_1)	2
20	0.117252	838.0	(member_card_Bronze, total_children_2)	2
21	0.187631	1341.0	(member_card_Bronze, num_children_at_home_0)	2
22	0.206660	1477.0	(member_card_Bronze, num_children_at_home_1)	2
23	0.122709	877.0	(year_income_40000.0, num_children_at_home_1)	2
24	0.132923	950.0	(member_card_Bronze, education_Bachelors Degree)	2
25	0.131524	940.0	(year_income_60000.0, education_Bachelors Degree)	2
26	0.132923	950.0	(education_High School Degree, member_card_Bro...	2
27	0.144536	1033.0	(education_High School Degree, year_income_400...	2
28	0.140479	1004.0	(member_card_Bronze, education_Partial College)	2
29	0.179236	1281.0	(year_income_40000.0, education_Partial College)	2
30	0.203442	1454.0	(member_card_Bronze, year_income_40000.0)	2

圖 4. 9 Apriori 程式碼結果-2

得到 frequent pattern 後，進一步引入 Mlxtend 的 association_rules 便可以用其去得到資料集的 association rule，效果如圖 4.10。

```
from mlxtend.frequent_patterns import association_rules
MinConfidence = 0.5

InterestedAssociationRulesInfo = ["antecedents", "consequents", "support", "confidence"]
AssociationRules = association_rules(FrequentItemset, metric = "confidence", min_threshold = MinConfidence)
AssociationRules[InterestedAssociationRulesInfo]
```

	antecedents	consequents	support	confidence
0	(total_children_1)	(member_card_Bronze)	0.118931	0.605413
1	(total_children_2)	(member_card_Bronze)	0.117252	0.583159
2	(num_children_at_home_0)	(member_card_Bronze)	0.187631	0.582537
3	(num_children_at_home_1)	(member_card_Bronze)	0.206660	0.598218
4	(education_Bachelors Degree)	(member_card_Bronze)	0.132923	0.543168
5	(year_income_60000.0)	(education_Bachelors Degree)	0.131524	0.721966
6	(education_Bachelors Degree)	(year_income_60000.0)	0.131524	0.537450
7	(education_High School Degree)	(member_card_Bronze)	0.132923	0.538244
8	(education_High School Degree)	(year_income_40000.0)	0.144536	0.585269
9	(education_Partial College)	(member_card_Bronze)	0.140479	0.555925
10	(year_income_40000.0)	(education_Partial College)	0.179236	0.507126
11	(education_Partial College)	(year_income_40000.0)	0.179236	0.709302
12	(year_income_40000.0)	(member_card_Bronze)	0.203442	0.575614

圖 4. 10 Association Rules

此外還可以看到 Lift，可透過 metric = “lift” 和 min_threshold = lift 的底線來找出 lift 高於一定值的 association rule。

```
from mlxtend.frequent_patterns import association_rules
MinLift = 1.2

InterestedAssociationRulesInfo = ["antecedents", "consequents", "support", "lift"]
AssociationRules = association_rules(FrequentItemset, metric = "lift", min_threshold = MinLift)
AssociationRules[InterestedAssociationRulesInfo]
```

	antecedents	consequents	support	lift
0	(education_Bachelors Degree)	(year_income_60000.0)	0.131524	2.950196
1	(year_income_60000.0)	(education_Bachelors Degree)	0.131524	2.950196
2	(education_High School Degree)	(year_income_40000.0)	0.144536	1.655946
3	(year_income_40000.0)	(education_High School Degree)	0.144536	1.655946
4	(year_income_40000.0)	(education_Partial College)	0.179236	2.006882
5	(education_Partial College)	(year_income_40000.0)	0.179236	2.006882

圖 4. 11 Association Rules，使用 MinLift=1.2 限制

另外，也可以使用 Pandas 去查看在 year_income_40000.0 的 Association Rules。

```
InterestedDataName = ["year_income_40000.0"]
InterestedAssociationRulesInfo = ["antecedents", "consequents", "support", "confidence", "lift"]
AssociationRules = association_rules(FrequentItemset, metric = "lift", min_threshold = 0)
AssociationRules[AssociationRules["antecedents"] == InterestedDataName][InterestedAssociationRulesInfo]
```

	antecedents	consequents	support	confidence	lift
8	(year_income_40000.0)	(num_children_at_home_1)	0.122709	0.347189	1.005007
17	(year_income_40000.0)	(education_High School Degree)	0.144536	0.408947	1.655946
20	(year_income_40000.0)	(education_Partial College)	0.179236	0.507126	2.006882
23	(year_income_40000.0)	(member_card_Bronze)	0.203442	0.575614	1.127717

圖 4. 12 year_income_40000 的 Association Rules

4.2 DNN 分類預測實現方法

本次實現 DNN 也是在 Colaboratory 上使用，在一開始我們建置一個簡易的 DNN 模型。

```
model = Sequential()
model.add(Dense(units=512, input_dim=5, kernel_initializer='uniform'
))
model.add(Activation('relu'))
model.add(Dense(units=256, kernel_initializer='uniform'))
model.add(Activation('relu'))
model.add(Dense(units=128, kernel_initializer='uniform'))
model.add(Activation('relu'))
model.add(Dense(units=64, kernel_initializer='uniform'))
model.add(Activation('relu'))
model.add(Dense(units=32, kernel_initializer='uniform'))
model.add(Activation('relu'))
model.add(Dense(units=4, kernel_initializer='uniform'))
model.add(Activation('sigmoid'))
model.compile(loss='categorical_crossentropy',
              optimizer='adam', metrics=['accuracy'])
```

圖 4. 1313 模型建構部分 code

在模型中我們需要加入全連接層使模型去對資料中的特徵進行過濾，而這邊我們的激勵函數使用線性整流函數，該函數作用是使輸入的權重值小於 0 的話則輸出 0，而所有大於 0 的權重則依照輸入值直接輸出。

首先第一層的輸入層設定總共有 5 個資料維度，分別是教育程度，年收入，結婚情況，性別，年齡。

接在輸入層後面的有 4 層隱藏層，從第 1 層到 4 層分別有 256，128，64，32 個神經元，每層隱藏層神經元如果越多的話越能對資料的特徵值進行更細緻的提取，不過有時候太多反而會造成過擬合的現象發生。

而最後一層輸出層則是對應了會員卡等級的 4 個級別，分別為普通，銅，銀，金，而訓練最佳化函數選擇最泛用的 adma 函數，而模型測量的基準則以訓練時的

準確度為基準, 全部設定好之後就可以將模型建構起來。

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 512)               3072
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
activation_2 (Activation)    (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
activation_4 (Activation)    (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 32)                2080
_________________________________________________________________
activation_5 (Activation)    (None, 32)                0
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 132
_________________________________________________________________
activation_6 (Activation)    (None, 4)                 0
=================================================================
Total params: 177,764
Trainable params: 177,764
Non-trainable params: 0
_________________________________________________________________
None
```

圖 4. 1414 模型整體結構圖

接下來就是讓模型訓練使其對資料提取特徵。

```
h = model.fit(train_x, train_y, validation_data=(test_x, test_y),batch_size=128, epochs=100, shuffle=True
, verbose=1)
```

圖 4. 1515 模型訓練 code

而經過多次測試後, 最終模型的準確度來到 63. 19%。

```
Epoch 98/100
3598/3598 [==============================] - 0s 59us/step - loss: 1.0999 - accuracy: 0.5901 - val_loss: 1.0382 - val_accuracy: 0.6319
Epoch 99/100
3598/3598 [==============================] - 0s 51us/step - loss: 1.0986 - accuracy: 0.5901 - val_loss: 1.0384 - val_accuracy: 0.6319
Epoch 100/100
3598/3598 [==============================] - 0s 45us/step - loss: 1.0984 - accuracy: 0.5901 - val_loss: 1.0379 - val_accuracy: 0.6319
Evaluation on test data: loss = 1.037881 accuracy = 63.19%
```

圖 4. 1616 模型訓練 code

第五章 結果與討論

5.1 結果

透過引用 MLXTEND 我們可以很簡單的便實現了 Apriori 先驗演算法來找出資料的 frequitent patterns 和其 association rules，甚至還有 Lift 方法去看資料間的提升度。而從資料訓練出來的 DNN 模型，也可對資料進行一定的預判。

5.2 討論

透過 Apriori 能看到資料間的關係，但由於資料種類上頗大，每個種非在每次出現，所以 support 的限制必須低些，才能看的到資料的關聯。而在 DNN 模型上由於我們不能準確的得知會員卡的等級是依照哪幾樣標準去評定的,所以我們選擇了看起來比較有可能的幾項,不過一開始準確度只有 50 出頭,經過幾番嘗試後也發現,這些資料中,總共有幾個孩子跟現在還在扶養的孩子數量並不會對結果造成影響,而性別與結婚狀況分別可以讓準確度提升 9%及 4%,所以我們對資料調整後並重新訓練該模型,發現準確度上升到了 63%。

參考文獻

[1]刘建平（民 106 年 02 月 20 日）深度神经网络（DNN）模型与前向传播算法, from https://www.cnblogs.com/pinard/p/6418668.html

[2]Keras 入門（六）模型訓練實時視覺化（民 109 年 02 月 24 日), from https://www.mdeditor.tw/pl/pxc0/zh-tw

[3]Python 實戰篇：Apriori Algorithm（民 108 年 12 月 10 日), from https://artsdatascience.wordpress.com/2019/12/10/python-%E5%AF%A6%E6%88%B0%E7%AF%87%EF%BC%9Aapriori-algorithm/

[4]KERAS 官方文檔(無日期), from https://keras.io/

[5]随机森林 VS 神经网络：哪个更好？(民 108 年 06 月 10 日), from https://www.toutiao.com/a6700795235598860811/