

金融科技導論 期末報告 Group 20

信用卡盜刷偵測

組員： b07901014 陳希格、b08902062 魏皓融、r08921108 鄒家禾、
r08921059 陳鴻智、b08902118 譚國慶

一、問題定義

在消費型態改變與手機行動支付結合的推波助瀾之下，信用卡交易金額以每年約10%的幅度成長，銀行業者除提高信用卡使用率與金額的同時，如何降低盜刷與違約風險並保有獲利顯得格為重要。

本專題利用各種不同資料清洗的方式，以及不同的模型來進行盜刷行為的預測。

資料：150萬筆training data (with label), 46萬筆testing data(no label)

欄位：X(23項，包含一些交易資訊及註記)，Y(1項，即是否盜刷)

評分標準：F1 Score

二、方法描述

我們主要先透過資料分析，得知特徵分布和相關性。再經過前處理後和模型訓練的循環，找出最優的前處理方法和模型的搭配，並用其預測結果(test set)。



三、資料分析

由於test data的fraud_id沒給，因此只有分析train data。

1. Imbalanced Data

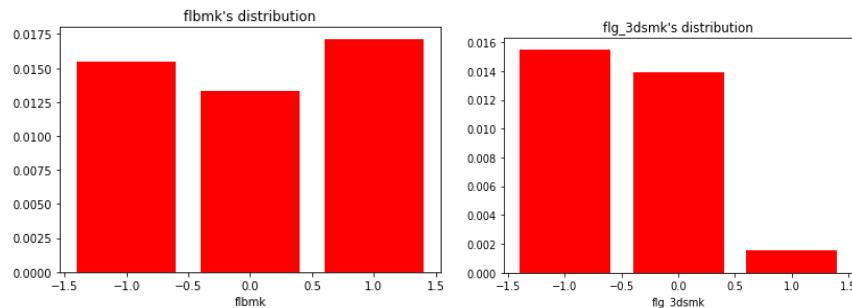
這次使用的資料集共約150萬筆的資料，23個欄位

- 其中1欄為盜刷註記(fraud_ind)
- 盜刷註記資料比例為98.66 : 1.34
- 資料極度不平衡

	盜刷	正常
筆數	20,355	1,501,432
佔比	98.66%	0.34%

2. 缺值分析

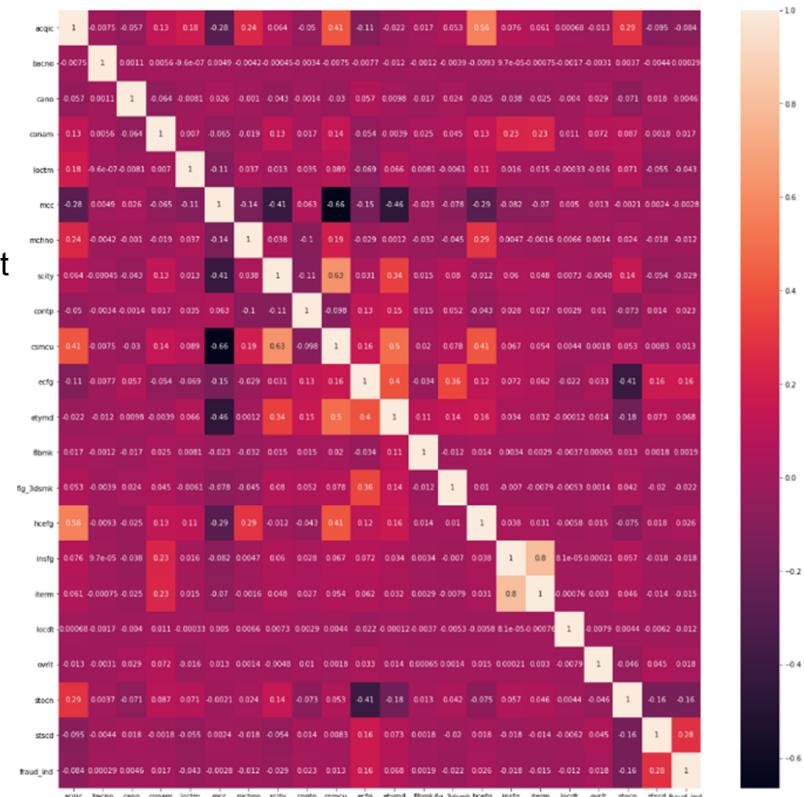
資料集缺值的欄位有"fbmk"、"flg_3dsmk"，缺值所佔比例為整體的0.83%。這是一個值為N,Y的欄位，由盜刷分布難以判斷fbmk較可能的值，flg_3dsmk則較可能為N。這是合理的，因為這該column的資料分布，N的數量是Y的數十倍。



上圖為盜刷分布，(-1,0,1)分別為(NA,N,Y)

3. 特徵相關性分析

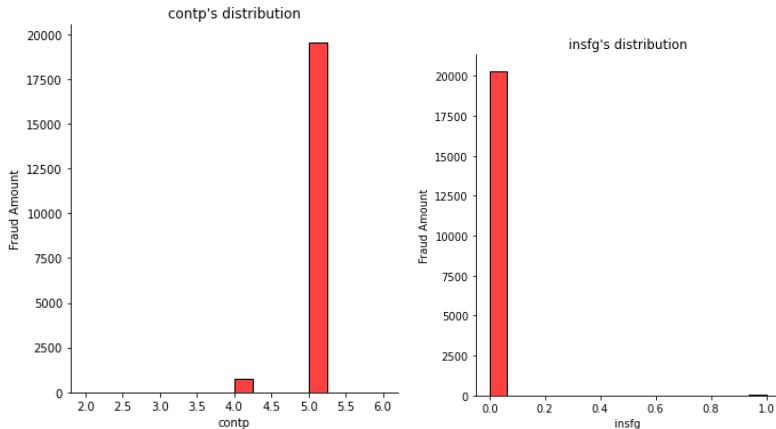
- 我們採用 Pearson correlation coefficient 進行相關性分析
- 繪製correlation heatmap如右圖



4. 特徵分布

● 極端、明顯的分布

- 盜刷集中在交易型態5(圖1)
- 由分期交易標註得知，盜刷中幾乎沒有分期交易(圖2)



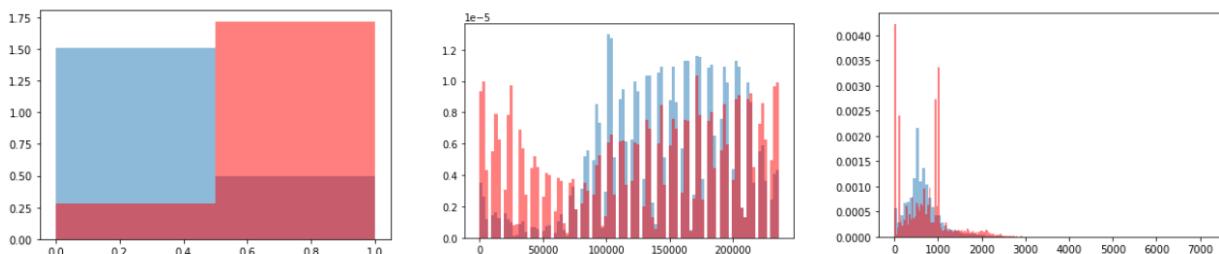
(圖1)

(圖2)

● 明顯有別的分布

以下展示一些正常與盜刷交易之分布有顯著差異的column。以下紅色為盜刷的分布，藍色為非盜刷的分布，偏紫的顏色為盜刷及非盜刷重疊。

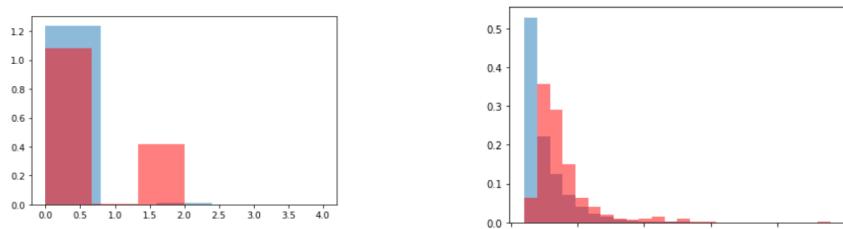
- 較多盜刷為網路交易，與正常交易相反。(圖1)
- 相較正常交易，盜刷在凌晨的交易較多。(圖2)
- 盜刷金額分布與正常交易不同，但非正相關或副相關。(圖3)
- 盜刷在狀態碼2的比例較正常交易高。(圖4)
- 單一卡號所到訪的國家數，結果顯示盜刷的卡號普遍去過超過一個國家，代表盜刷可能大多來自國外，因此會有本人消費及盜刷消費，共至少兩個國家。(圖5)



(圖1)

(圖2)

(圖3)



(圖4)

(圖5)

四、資料前處理

1. 缺值

資料集中缺值的欄位有 "flbmk"、"flg_3dsmk"，而我們嘗試了三種補值方式

- "NA", 即補入一個值代表其缺值
- mode, 用眾數補值
- mice, 即用Multiple Imputation by Chained Equations的方式補值

而實驗結果顯示"NA"是最好的補值方式(使用lightgbm model)

	"NA"	mode	mice
AUC	0.9717	0.9712	0.9712
precision	0.5270	0.5398	0.5398
recall	0.4971	0.4480	0.4480
f1_score	0.5116	0.4896	0.4896

2. 增加特徵欄位

由於原本的時間來為只有locdt(授權日期)、loctm(授權時間HHMMSS)，又從資料分析中我們可以發現盜刷跟時間高度相關，所以我們嘗試增加了特徵欄位：

- abs_time : 絕對時間，= day*86400+hour*3600+minute*60+second
- hours : 即HH
- minutes : 即MM
- seconds: 即SS

而實驗結果顯示增加特徵欄位沒有助於f1_score的提升(使用 lightgbm model)

	增加特徵欄位	不增加特徵欄位
AUC	0.9709	0.9717
precision	0.5939	0.5270
recall	0.4423	0.4971
f1_score	0.5070	0.5116

3. 刪除特徵欄位

我們嘗試刪除部分低相關性欄位，只保留高相關性的8個欄位。以下是我們用 Pearson correlation coefficient 方法篩選出跟盜刷高相關性的8個欄位：

['acqic', 'loctm', 'scity', 'ecfg', 'etymd', 'hcefg', 'stocn', 'stscd']

而實驗結果顯示刪除特徵欄位沒有助於f1_score的提升(使用 lightgbm model)

	刪除特徵欄位	不刪除特徵欄位
AUC	0.9584	0.9717
precision	0.6931	0.5270
recall	0.3629	0.4971
f1_score	0.4764	0.5116

4. standardize 和 one hot encode

我們嘗試針對資料型態為連續型的資料進行了standardize，並針對離散型的資料進行了one hot encode，詳細的結果比較請見模型選擇與實驗設計。

5. down sampling

由於資料為imbalanced的数据，所以我們嘗試將訓練資料進行down sampling，詳細的結果比較請見模型選擇與實驗設計。

6. 資料切割

由於沒有test data，我們將訓練資料分割做模型訓練

train : valid : test = 9 : 3 : 4:

分割方式：1.依時間切割 2.隨機分割測試 -> 結果無明顯差異

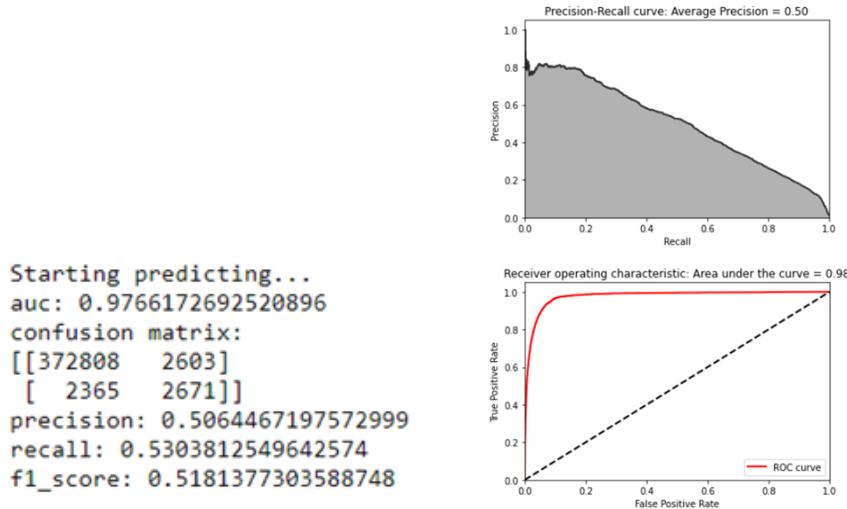
五、模型選擇與實驗設計

先用sample code裡的資料處理方法，將training set的columns分為數值欄位和類別欄位，並且對數值欄位做標準化，而類別欄位則做one hot label encoding，有缺值的['flbmk', 'flg_3dsmk']這兩欄都用'NA'來補值。這時做完欄位會從原來的21欄(不含y target)增長為367欄。

因為沒有test data，所以再將training set切成train與test比值是3:1。但訓練資料集的target('fraud_ind')相當不平衡，因此再將train做down sampling後切為train和validation比值是3:1，初步的前處理就先做到這邊。

1. LightGBM

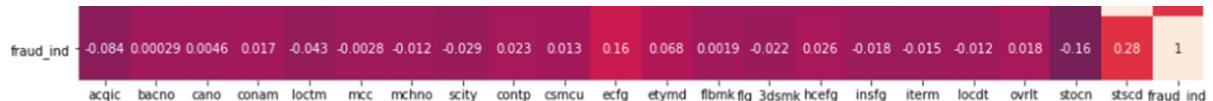
我們一開始的模型選的是LightGBM, 用於預測test資料集結果如下：



可以看到f1 score只有0.5左右, 效果差強人意。

2. LightGBM + Kmeans Clustering + Feature Selection

接下來我們打算先算出所有features對於target欄位的相關係數, 採用Pearson correlation coefficient, 然後繪製correlation heatmap如下：

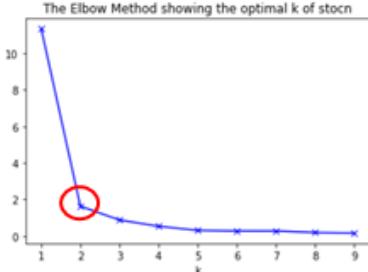
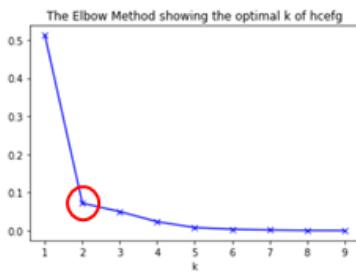
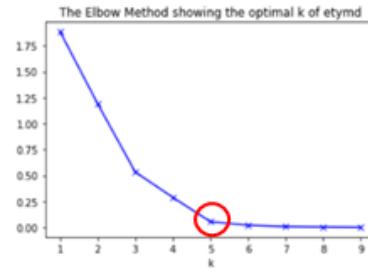
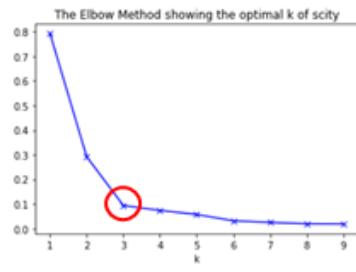
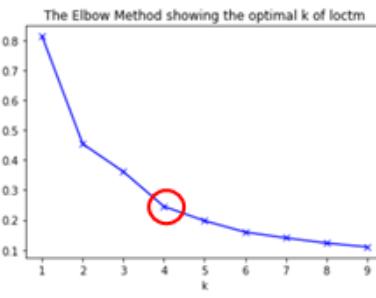
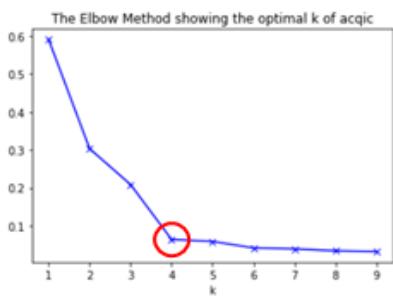


選出相關係數絕對值最高的8個features, 分別是['acqic', 'loctm', 'scity', 'ecfg', 'etymd', 'hcefg', 'stocn', 'stscd'], 其中loctm我們在特徵分析時有發現凌晨時段的盜刷比例較高, ecfg則是因為網路交易的盜刷比例高, 而stscd的狀態碼2在盜刷中佔比比正常交易高。由此可知由相關係數選出的feature是符合邏輯的。

接著我們將選出的features分別做K-means Clustering(ecfg的值只有0,1, stscd只有0-5故不做)。而選出k值的方法我們採用較簡單的Elbow Method, 其概念是基於 SSE(sum of the squared errors, 誤差平方和)作為指標, 去計算每一個群中的每一個點, 到群中心的距離。算法如下：

$$SSE = \sum_{i=1}^k \sum_{p \in C_i} |p - mi|^2$$

再對於每個feature根據K和SSE作圖, 從曲線中由急速下降轉為平緩的點作為選中的K值，如下圖：



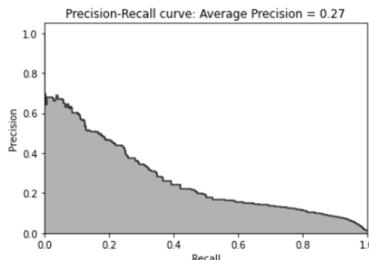
由上面六張圖得到K值分別為acqic:4, loctm:4, scity:3, etymd:5, hcefg:2, stocn:2。

對這些features作完K-means clustering後，只取8個相關係數最高的feature放入LightGBM模型訓練，結果如下(AUROC都大同小異故省略)：

```

Starting predicting...
auc: 0.9424456337289331
confusion matrix:
[[373586 1825]
 [ 3771 1265]]
precision: 0.40938511326860844
recall: 0.2511914217633042
f1_score: 0.311346295840512

```



可以看到結果竟然變差了，從這邊我們可以發現只取少數feature並無益於模型訓練，也可能是我們特徵萃取上做的不夠完善導致此結果。

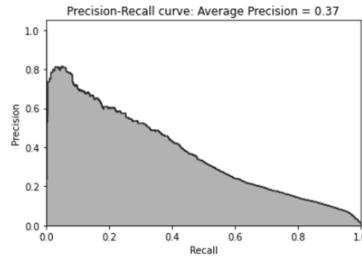
接著我們認為交易金額(conam)雖然在相關係數上是偏低的，但直覺上可能會對盜刷與否有關，所以將conam這欄加入剛剛高相關性的特徵進行訓練，結果如下：

:

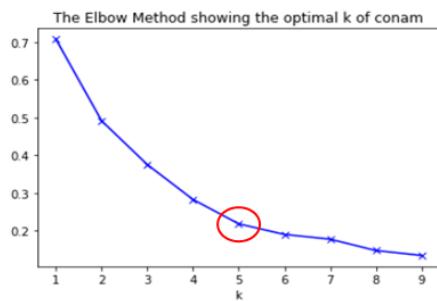
```

Starting predicting...
auc: 0.9566443121150974
confusion matrix:
[[373381  2030]
 [ 3206  1830]]
precision: 0.4740932642487047
recall: 0.36338363780778393
f1_score: 0.4114208633093525

```



發現雖然效果一樣滿差的，但卻比剛剛好了一些，`f1 score`來到0.4。因此我們知道了交易金額應該是對模型訓練有幫助的。所以也對conam做了k-means elbow method得到K值為5，如下圖：



反覆實驗下我們發現兩點：

1. 類別欄位是否做one hot encoding效果都差不多，而且會讓data維度太高。
2. down sampling對結果沒有幫助，做了又會讓training data數量大幅減少。

所以接下來我們前處理都會略過這兩個步驟。

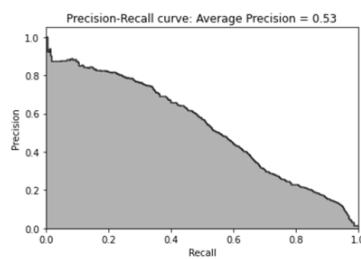
3. LightGBM + Kmeans Clustering

由於只選出相關係數高的特徵對提升效能沒什麼幫助，我們就想說把高相關性的特徵做k-means分群，而其他欄位也保留(除了cano和bacno)。模型一樣用LightGBM，結果如下圖：

```

Starting predicting...
auc: 0.9682996987856249
confusion matrix:
[[373816  1595]
 [ 2629  2407]]
precision: 0.6014492753623188
recall: 0.47795869737887214
f1_score: 0.5326399645939367

```

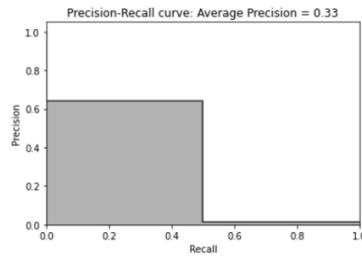


可以看到`f1 score`突然上升到0.53，已經超過最一開始前處理data的效果了。

4. KNN

這時我們也開始嘗試其他模型，像是KNN如下圖：

```
Starting predicting...
auc: 0.7483130904183536
confusion matrix:
[[5908  20]
 [ 36  36]]
accuracy: 0.9906666666666667
precision: 0.6428571428571429
recall: 0.5
f1_score: 0.5625000000000001
```

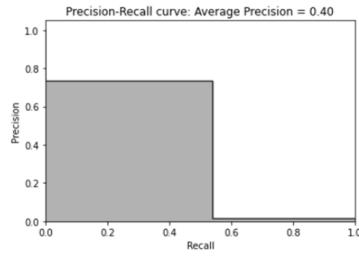


由於KNN模型的預測需要較久時間，所以有從test set裡隨機取樣一定量來做預測。發現KNN效果跟LightGBM差不多。

5. Random Forest

再來試試看Random Forest, 如下圖：

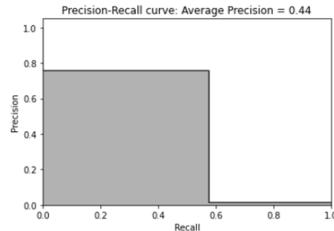
```
Starting predicting...
auc: 0.7684445168719918
confusion matrix:
[[374425   986]
 [ 2319  2717]]
precision: 0.7337294085876317
recall: 0.539515488482923
f1_score: 0.6218102757752603
```



f1 score來到了0.62, 效果比起一開始有了不錯的進步，由於Random Forest效果特別好，接下來就都使用它來做訓練。

但前面做的相關係數分析，交易金額conam並不在高分群，我們卻對他做了k-means。為了驗證交易金額分群是否對結果有影響，接下來嘗試不對conam做分群，直接用標準化後的值去做訓練，結果如下：

```
Starting predicting...
auc: 0.7865823391690323
confusion matrix:
[[374476   935]
 [ 2137  2899]]
accuracy: 0.9919252878850405
precision: 0.7561293688054251
recall: 0.5756552819698173
f1_score: 0.6536640360766629
```

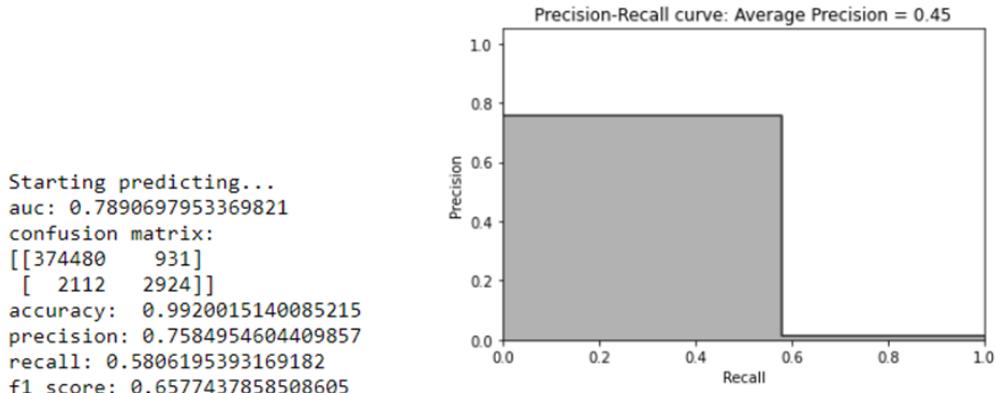


結果f1 score又小小提高到了0.65，所以我們得知只對相關係數高的做分群，效果會比較好。

但其實做了k-means得到的特徵重新計算一次Pearson correlation coefficient後發現，有些特徵分群之後未必相關性會變高，如下圖(後面加上底線k代表是做過分群後的特徵)：

fraud_ind	1.000000	contp	0.023248
stscd	0.279225	flg_3dsmk	0.020809
ecfg	0.161701	insfg	0.018403
stocn	0.161405	ovrlt	0.018177
stocn_k	0.148973	conam	0.016634
acqic	0.083684	iterm	0.014662
etymd_k	0.075384	csmcu	0.013472
etymd	0.067743	mchno	0.012187
loctm_k	0.063976	locdt	0.011773
acqic_k	0.055927	conam_k	0.009801
loctm	0.042688	scity_k	0.007335
scity	0.028715	cano	0.004608
hcefg_k	0.028114	mcc	0.002827
hcefg	0.026210	flbmk	0.002543
		bacno	0.000285

所以我們又從上列取了最高的8個特徵為['acqic', 'loctm_k', 'scity', 'etymd_k', 'hcefg_k', 'stocn', 'ecfg', 'stscd']，其中acqic, scity, stocn都是做了k-means後相關性反而變小，所以取原欄位。而其餘欄位依然保留，結果如下：



結果發現分數只有相當微小的提升，得知此處理可能影響不大。

六、實驗結果及討論

1. 實驗結果：

我們留下幾個相對較好的結果與初步處理的結果做比較，如下圖：

Features	Model	Precision	Recall	F1_score
Original	LightGBM	0.5064	0.5303	0.5181
Clustered conam	LightGBM	0.6014	0.4779	0.5326
Clustered conam	KNN	0.6428	0.5	0.5625
Clustered conam	RandomForest	0.7337	0.5395	0.6218
Standerdized conam	RandomForest	0.7561	0.5762	0.6536
High corr after K-means	RandomForest	0.7584	0.5806	0.6577

基本上我們結果較好的資料處理都是不做down sampling與one hot encoding, 然後對高相關性的特徵做K-means, 其他欄位除了cano和bacno之外都留著, 唯一有不同的就是是否有加入conam的分群。而最後一列則是做完K-means之後再算一次correlation取高的, 其他欄位保留。

可以看出實驗結果最好的是最後兩欄, 對高相關性特徵做分群, 並且保留其他特徵, 而模型選用則是Random Forest效果最好。

2. 錯誤案例分析:

實驗過程中我們發現分別對與target高相關性的特徵做分群, 並且只取這些分完群的特徵, 訓練結果甚至比baseline還差, 這可能是因為過於簡單地萃取特徵對於模型訓練並沒有好處。

如果能進一步做特徵工程, 例如刪除兩兩相關性高的重複特徵, 或是根據其中幾個特徵的特性合併出新的特徵, 或許才是特徵萃取的真諦, 在資料維度下降的情況下得到更好的結果。

再來是模型選取上, LightGBM和Random Forest都是基於決策樹建構出的模型, 直覺上LightGBM如果適當調整參數應該會在性能上勝過Random Forest。

但我們的實驗結果顯示在同樣training set的情況下, Random Forest的分數反而勝過LightGBM, 這可能是因為Random Forest相對沒什麼參數需要調整, 可以比較輕易發揮出它的性能。但由於此次期末報告的時程有點緊迫, 我們並沒有很專注於研究LightGBM要如何調整參數以達到最佳效果。

七、結論及展望

這次是受時間和資源限制, 無法選擇大型的model進行訓練和比較, 只能使用較小且可以快速訓練的model。但依舊取得不錯的成績 F1 score = 0.6577。這也側面證明了簡單好用的模型未必會表現的複雜的神經網路差。而且簡單的模型(Random Forest)帶來許多好處:可快速訓練模型、不須 fine tuning、可解釋性強。

八、附錄

- 參考文獻

<http://blog.tcfst.org.tw/asvda/file/109JQ01/%E4%BF%A1%E7%94%A8%E5%8D%A1%E7%87%9F%E9%81%8B%E6%99%BA%E6%85%A7%E5%88%86%E6%9E%90%E7%B3%BB%E7%B5%B1.pdf>

<http://blog.tcfst.org.tw/asvda/file/JQ04/08.%E9%87%91%E8%9E%8D%E4%BF%A1%E7%94%A8%E5%8D%A1%E7%B5%841204.pdf>

https://skywalker0803r.medium.com/台灣t-brain資料科學比賽-1042f86d995a?fbclid=IwAR0aXUsLQ3k9hwvp8VE5yxBlgRyeJ_tDcHq0P79PMf_tC3jQM7XU71fTHx8

- 原始碼

https://github.com/erictsou/NTU_Fintech_Final

- 小組開會記錄

<https://hackmd.io/RBHIOp6uSF-rz2l3EW5opQ>

- 組員執掌

姓名	學號	Code	報告撰寫	ppt製作	影片剪輯
陳希格	b07901014	前處理	資料前處理、方法、結論、排版	前處理、流程圖、結論、排版	X
魏皓融	b08902062	資料分析	資料分析	資料分析	全
陳鴻智	r08921059	模型選擇	模型選擇與實驗設計	問題定義	X
鄒家禾	r08921108	特徵工程 模型訓練	模型選擇與實驗設計 實驗結果與討論	模型選擇與實驗設計 實驗結果與討論	X
譚國慶	b08902118	X	X	X	X