分 析 說 明 書

資料處理與特徵選取

Data Cleaning

首先我們將使用「低、中、高」等漢字表示程度的特徵值，依序轉換為「1、2、3」等數值形式，以維持其程度差異的展現；對於其中出現的 NA 值，則是以該變數之**整體平均值**作爲替代。

以漢字表示程度的特徵值		
AGE	APC_1ST_AGE	INSD_1ST_AGE
LIFE_CNT	REBUY_TIMES_CNT	RFM_R

同時也將諸如性別、聯絡地址等以字母代碼表示的分類資料，利用 **Label Encoder** 轉化為數值變數；對於遺漏值，則賦予「-1」的分類注記。

以字母代碼表示的類別資料	
GENDER	CUST_9_SEGMENTS_CD
CHARGE_CITY_CD	CONTACT_CITY_CD

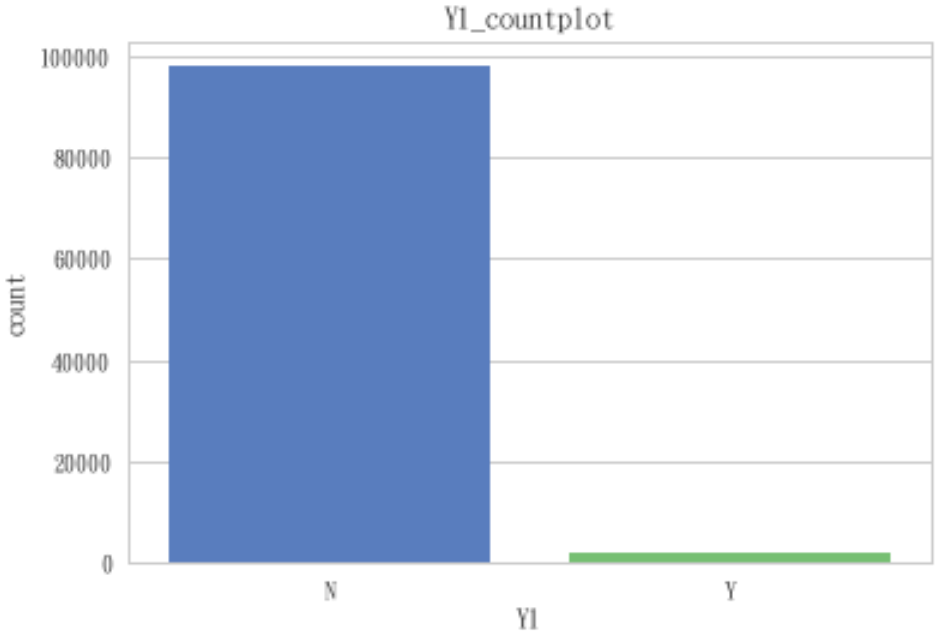
另外也將以「N」、「Y」表示的判斷型資料轉成「0、1」的數值形式；對於這些變數中的 NA 值，亦以獨立數字「-1」作爲替代表示。

以「N、Y」表示的判斷型資料		
LAST_A_CCONTACT_DT	LAST_A_ISSUE_DT	LAST_B_ISSUE_DT
LAST_B_CONTACT_DT	IF_ISSUE_A_IND	IF_ISSUE_B_IND
A_IND	B_IND	C_IND
族繁不及備載	Y_1	共 80 項

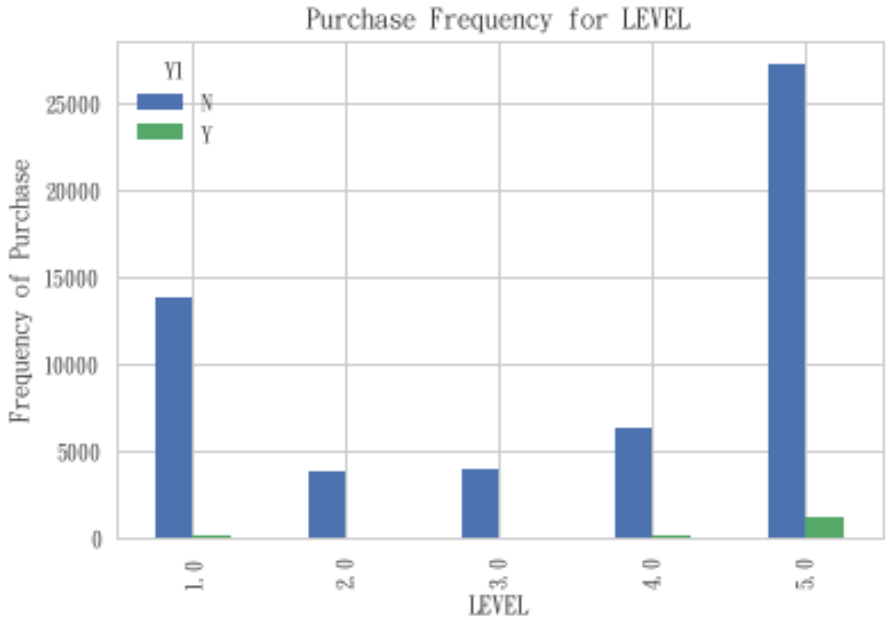
剩餘的資料（42 項），皆是具有**連續性質**的數值資料，故在處理 NA 值時，我們使用了該變數的**平均值**作爲替代。

Data Processing

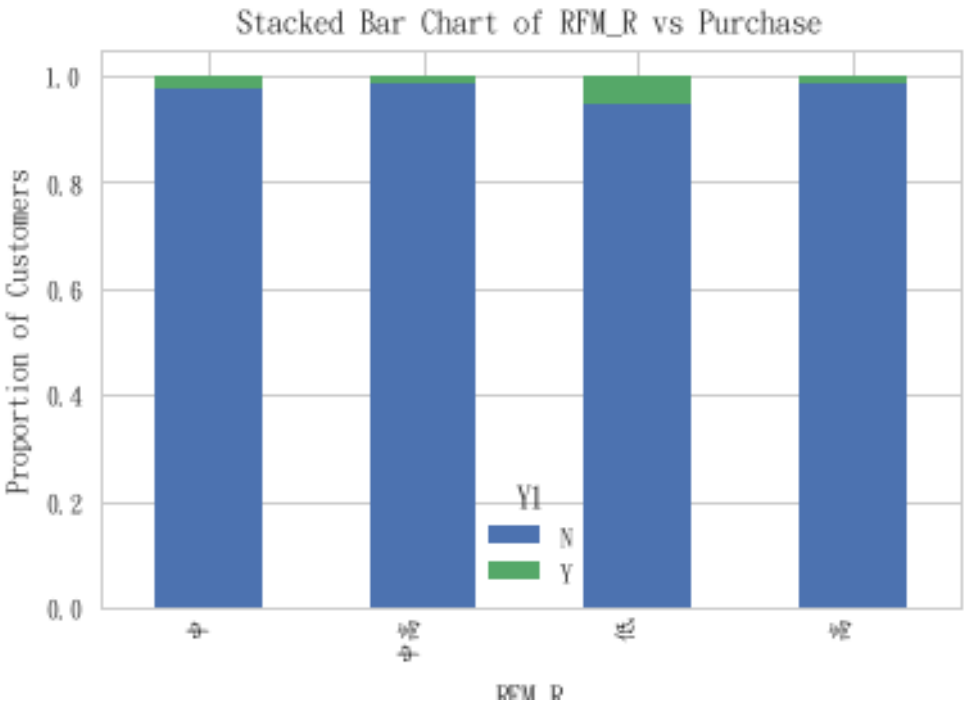
A. 基礎資料分析



→可見**資料不均衡**狀態相當明顯，Y_1欄位數值為 1 的個數十分稀少。

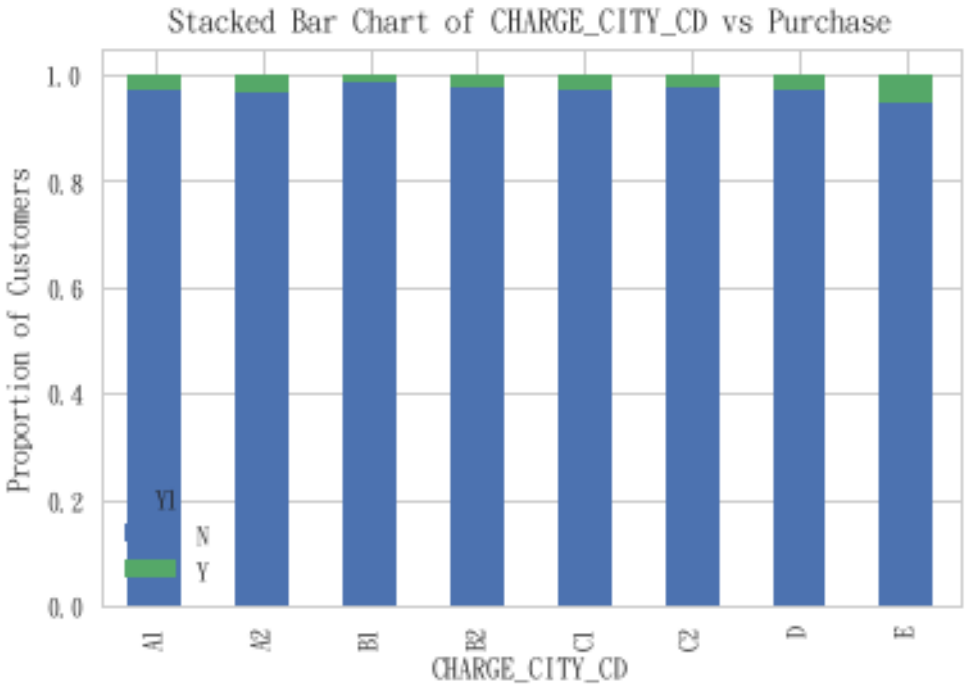


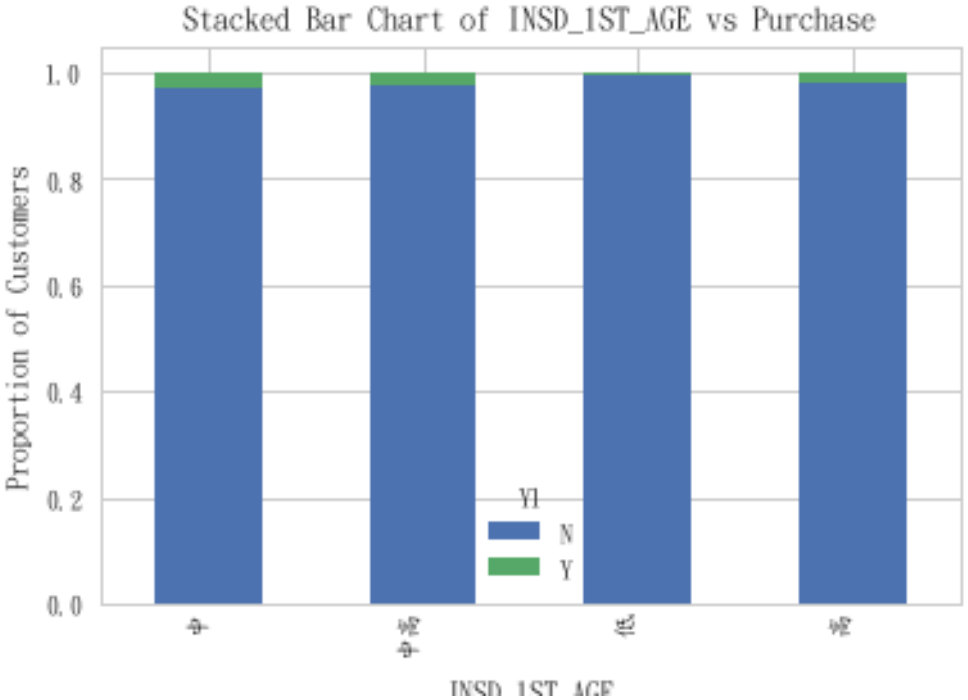
→往來關係等級**最高者**最有機會購買。



→上次要保人身份投保距今間隔時間**<u>級距愈低</u>**，購買機會愈大。

→假如近三年**<u>有</u>**透過 A 通路投保新契約，更有機會購買。





→CHARGE_CITY_CD、INSD_1ST_AGE 皆與 Y_1 呈現高相關性，然而經討論後我們認為此關聯性為**<u>偽相關</u>**。主要原因是這兩個特徵值的欄位**幾乎沒有** Y_1=1 的值，然而 Y_1 項中有 98%為 Y_1=0，故相關性才會異常的高，因此我們將這兩個欄位刪除，並認定它們並非重要的特徵。

B. 產生新變數

首先使用 **One Hot Encoding** 將先前用 Label Encoder 轉換完的所有「**名目欄位**」轉換成啞變數，例如：

LEVEL
1
2
3

→

LEVEL_1	LEVEL_2	LEVEL_3
1	0	0
0	1	0
0	0	1

目的為：

- 藉此將各名目欄位中的遺失值分別獨立成一個欄位再捨棄。
- 確保之後利用 imbalanced-learn 的 **SMOTE** 增加樣本數以解決 imbalanced data 的狀況時，特徵空間中類別變數的數值大小**不會被當作連續變數**考慮而出現不存在的類別。

C. 平衡訓練集

(1) Over-Sampling—SMOTE

「SMOTE」主要運用 KNN 演算法合成少數類別的假樣本。其宗旨為：取任一資料點周遭的 K 個鄰居，從中隨機擇一計算與樣本的歐式距離，再將原數值加上該向量的隨機 0~1 倍而得新樣本點。透過此種方法製造出多筆 Y_1=1 的人工樣本，使 Y_1=1 與 Y_1=0 的**個數平均**。

- Length of oversampled data is 137206
- Number of people who will purchase critical illness insurance in oversampled data is 68603
- Number of people who will not purchase critical illness insurance in oversampled data is 68603
- Proportion of people purchasing in oversampled data is 0.5
- Proportion of people not purchasing in oversampled data is 0.5

(2) Under-Sampling

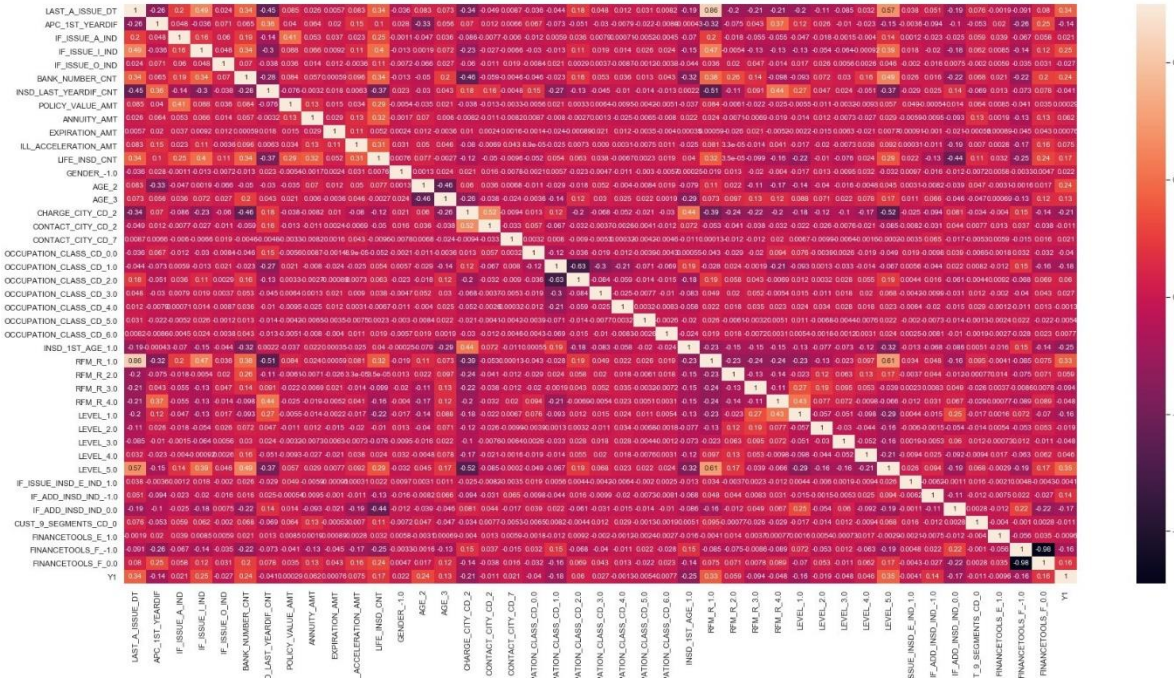
訓練集的樣本數龐大，總共有 100000 筆，然而資料是極度**不均衡**的；因此我們透過**減少 Y_1=0** 的樣本數來達成平衡，使 Y_1=1 與 Y_1=0 皆為 2000 筆。此種方法的主要問題在於有可能刪掉過多的樣本，導致最後樣本數過少，因而**不滿足大數法則**，不過我們仍然採用此種方法來測試。

Feature Selection

由於在資料處理時已用 pandas.get_dummies 使欄位個數從 131 驟增到 268，故需要做特徵選擇來**降維**。

我們使用 **Recursive Feature Selection** 以挑選特徵，藉由迭代的方式（採預設，即一次刪減一欄）篩選掉影響力較低的特徵，並選擇以 scikit-learn 套件的**羅吉斯回歸**（Logistic Regression）作為 RFE 的估計式（去衡量準確率）；而目標挑選出的個數則為 50。

而後再用統計模型套件 StatsModels 的羅吉斯回歸（Logit Model）做出這 50 個參數的統計報告，並刪掉 p-value 大於 0.005、判定為**不顯著**的參數，最後得到 42 項要丟入模型中訓練的特徵。



※上圖為用 Seaborn 套件做出的相關係數 Heatmap 圖表，可以觀察到更多**預測變項**與**目標變數**間的關聯性，如：

```
top_corr_feature=corr_matrix.index[abs(corr_matrix["Y1"])>0.3]
print(top_corr_feature)
```

```
Index(['LAST_A_ISSUE_DT', 'RFM_R_1.0', 'LEVEL_5.0', 'Y1'], dtype='object')
```

→**LAST_A_ISSUE_DT**、**RFM_R_1.0**、**LEVEL_5.0** 與 Y_1 的相關程度最高。

Logistic Regression

第一步僅用 RFE 將特徵數量減少至 30 個；並利用 Statsmodel 把不顯著的特徵過濾掉；再將此**不含任何假樣本的訓練集（100000, 19）**用 Scikit-Learn 套件的 train_test_split 隨機分成 7:3 的形式，7 成當作 Training set、3 成當作 Cross-Validation set。

就交叉驗證出的準確率 **98%**而言，模型似乎相當精準，但如果以 Classification_report 觀察即可發現：

	precision	recall	f1-score	support
0	0.98	1.00	0.99	29397
1	0.33	0.00	0.01	603
accuracy			0.98	30000
macro avg	0.66	0.50	0.50	30000
weighted avg	0.97	0.98	0.97	30000

儘管模型達到近乎 100%的準確率，由圖表可見目標變數中對於 1 的**精確率**與**召回率相當低**，故可推得此模型並未從訓練集中學習到足夠關於 minor class 的資訊。

為了讓模型能夠取得更多**少數類別**的資訊，第二步我們運用 Over-Sampling 方法中的 **SMOTE** 去合成假樣本；並將這些含人工樣本的訓練集再度藉由 RFE 刪減至 30 個；值得注意的是，這次利用 Statsmodel 觀察 p 值時，因**每一欄皆顯示顯著**故選擇全數保留；最後同樣利用交叉驗證檢視模型時，發現準確率縱使降低至 **73%**，但以 Classification_report 觀察即可發現：

	precision	recall	f1-score	support
0	0.71	0.75	0.73	20471
1	0.74	0.70	0.72	20691
accuracy			0.73	41162
macro avg	0.73	0.73	0.73	41162
weighted avg	0.73	0.73	0.73	41162

模型對於「Y_1」欄位中 1 的精確率與召回率**明顯的提升**了，減少了 imbalanced data 對於預測的負面影響。最後嘗試以**啞變數**把類別欄位全數展開成新的欄位，得到了以下更高的精確率、召回率與 f1-score，並在測試集得到 **74%**的準確率。

	precision	recall	f1-score	support
0	0.78	0.76	0.77	20471
1	0.77	0.79	0.78	20691
accuracy			0.78	41162
macro avg	0.78	0.78	0.78	41162
weighted avg	0.78	0.78	0.78	41162

XGBoost

首先將**沒有製造人工樣本的訓練集（100000, 276）**隨機分成 7:3 的形式，7 成當作 Training set，3 成作為 Cross-Validation set。在模型**未調整任何參數**的情形下以 Training set 去配適，並就 Cross-Validation set 的結果分別去看預測出來的 Y_1=1 與 Y_1=0 之 **f1-score**：

	precision	recall	f1-score	support
0	0.98	1.00	0.99	29413
1	0.43	0.01	0.02	587
accuracy			0.98	30000
macro avg	0.70	0.50	0.51	30000
weighted avg	0.97	0.98	0.97	30000

可以從上面的結果得知，模型預測 Y_1=1 的結果非常差，經過討論後我們將此偏誤歸咎於訓練集屬於 **imbalanced data**。Y_1=1 所占比重太少（2%），因此模型學習不到如何去預測 Y_1=1 的情形；換言之，模型就算全部預測結果為 Y_1=0，在訓練集中仍舊會有 98%的準確率。

經過討論後，我們組內得出兩種想法。第一種是 SMOTE，第二種是 Under-Sampling。

(1)SMOTE

接下來，我們將原本的訓練集透過 SMOTE 轉為 Y_1=1、Y_1=0 個數相同的**數據合成訓練集（137206, 276）**。同樣我們將其隨機分成 7:3 的形式，並在模型**未調整任何參數**的情形下以 Training set 去配適，並就 Cross-Validation set 的結果分別去看預測出來的 Y_1=1 與 Y_1=0 之 **f1-score**：

	precision	recall	f1-score	support
0	0.97	1.00	0.99	20545
1	1.00	0.97	0.99	20617
accuracy			0.99	41162
macro avg	0.99	0.99	0.99	41162
weighted avg	0.99	0.99	0.99	41162

可以從上圖得知，模型預測成效極佳，無論是 Y_1=1、Y_1=0 或是 f1-score 皆取得非常高的分數，然而這不禁讓我們懷疑是否有出現 **Overfitting** 或是其他資料處理問題的嫌疑。果然當我們將 testing set 套用到配適好的模型上時，最後結果顯示只預測出了 321 個 Y_1=1。很明顯地，模型仍然無法正確地預測 Y_1=1 的情形。

(2) Under-Sampling

最終，我們使用 Under-Sampling 的方法將 Y_1=1、Y_1=0 的個數平衡，只挑取了 4000 個樣本（Y_1=1、Y_1=0 各 2000 筆）。同樣的我們將其隨機分成 7:3 的形式，並在模型**未調整任何參數**的情形下以 Training set 去配適，並就 Cross-Validation set 的結果分別去看預測出來的 Y_1=1 與 Y_1=0 之 **f1-score**：

	precision	recall	f1-score	support
0	0.72	0.73	0.73	569
1	0.75	0.74	0.75	631
accuracy			0.74	1200
macro avg	0.74	0.74	0.74	1200
weighted avg	0.74	0.74	0.74	1200

可以從上圖發現，儘管相較於前一個方法，f1-score 的分數降低了不少，然而可以發現此時模型預測出地 Y_1=1、Y_1=0 是相較均衡的，而且直觀上較無 Overfitting 的問題。

因此之後我們以 Undersampling 的方法處理訓練集，並**利用網格搜尋開始調整 XGBoost 的參數**：max_depth、min_child_weight、gamma、subsample、colsample_bytree、reg_alpha。

以下解釋調整各個參數的意義：

- 其中「max_depth」代表樹的最大深度，是用來避免過度擬合的參數，參數愈高，模型能夠學到更具體的樣本。
- 「min_child_weight」決定了小葉子節點的樣本權重和。同樣的這個參數用於避免過度擬合，當這個參數愈高，可以避免模型學到局部的特殊樣本。
- 「gamma」決定了節點分裂時所需的最小損失函數下降值；這個值愈大，算法愈保守。
- 「subsample」控制對於每顆樹，隨機採樣的比例；這個參數愈小，算法愈保守，並避免過度擬合。
- 「colsample_bytree」用來控制每棵樹隨機採樣的列數占比（每一列為一個特徵）。
- 「reg_alpha」代表 L1 正則化項，愈大模型愈保守。

調整完參數後，結果如下。

- **max_depth=3,**
- **min_child_weight=5,**
- **gamma=0.1,**
- **subsample=0.65,**
- **colsample_bytree=0.75,**
- **reg_alpha=0.00001**

而後將這些參數放入模型去配適出最佳的結果，並進行測試集的預測：

```
# 使用更多決策樹(5000)並用較小的學習速率(0.01)
xgbc4 = XGBClassifier(
 learning_rate =0.01,
 n_estimators=5000,
 max_depth=3,
 min_child_weight=5,
 gamma=0.1,
 subsample=0.65,
 colsample_bytree=0.75,
 reg_alpha=1e-05,
 objective= 'binary:logistic',
 nthread=4,
 scale_pos_weight=1,
 seed=27)
modelfit(xgbc4, train, predictors)
```

```
Model Report
Accuracy : 0.7772
AUC Score (Train): 0.865101
```

```
pred4 = xgbc4.predict(X_test)
sum(pred4)
#預測結果顯示總共41311個Y=1
```

```
41311
```

可以從上圖看到，我們的模型在訓練集上取得 **77.72%**的準確度，並得到了 **86.51%** 的 **AUC Score** （AUC 值愈大的分類器，正確率愈高）。最後我們再將測試集預測出的結果（pred4）上傳到網站上獲得了 **76.847%**的準確率，高於先前 Logistic Regression 預測出來的 **74%**，故最終採用的模型為 XGBoost。