智慧智能化整合
IIE_Final_Project

MEGA 防護罩

109034523 黃浩銓

清華大學工業工程與工程管理學系

摘要

隨著智慧型手機的普及，電子支付也成為了近年的熱門話題，相較於使用現金和實體信用卡，電子支付擁有交易方便、快速等特性，我們只要透過一支手機，就能迅速完成付款的動作；然而使用電子支付卻會有安全性問題，很容易遭受詐騙導致金錢流失或者透過駭客的方式盜取你手機裡綁定信用卡的資訊來盜刷你的信用卡。因此本專案希望從過往資料中透過機器學習的方式找出一套分類模型，可以用來即時監測信用卡是否遭到盜刷。

關鍵字：即時監測，分類模型，RandomForest，Catboost

1. 背景

1.1. 研究背景

現今科技越來越便利，智慧型手機的出現更是讓現代人擁有更加方便的生活。如今隨著互聯網的出現，電子支付也隨之成為現代人耳熟能詳的新名詞。相較於傳統支付是在一個較為封閉的系統，電子支付基於一個開放系統平台(互聯網)上運作，且擁有交易方便、快速等特性，我們只要透過一支手機，就能迅速完成付款的動作。

1.2. 研究動機

然而因為電子支付是在一個開放的系統平台上運作，所以他的安全性也大大降低。有研究指出在 2018-2019 年信用卡盜刷的金額高達 30 億元。而近一步觀察信用卡詐騙的類型，其中以網路交易占比最重，高達 93%，且有時用戶被盜刷了仍渾然未知，直到帳單來了才知道。且往往被盜刷一筆之後就會緊接著被盜刷下一筆款項，造成 2 次傷害。因此如何快速且準確之測出這筆是被盜刷的款項，來降低傷害，是一大課題。

1.3. 研究目的

為了解決信用卡被盜刷卻沒有辦法即時監測的問題，本研究先透過 5 W1H 分析法了解問題，及確認問題解決方法後，再使用機器學習的方式，並根據模型分類結果進行訓練與改善，最終得到分類最好的模型，提供給客戶或是銀行等金融機構。

1.4. 5W1H 分析法

本研究透過 5W1H 分析法，幫助我們了解問題並思考解決問題的方法，對 選定的項目、工序或操作，都要從對象（何事 WHAT）、時間（何時 WHEN）、 人員（何人 WHO）、地點（何地 WHERE）、原因（何因 WHY）、方法（何法 HOW） 等六個方面提出問題進行思考，且對問題進行綜合分析研究，從而得到更具建設 性設的決策。利用此方法，我們主要想針對提問垃圾訊息過多的問題來進行改善。

(1)What?: 要解決甚麼問題? 解決信用卡被盜刷卻沒能被準確的偵測出來

(2)When?: 在甚麼時候要解決? 盜刷信用卡時卻沒有即時偵測出來

(3)Who?: 由誰來改善這些問題? 平台工程師提出此模型，金融機構人員以及用戶同時採用此模型，增加判斷準確率。

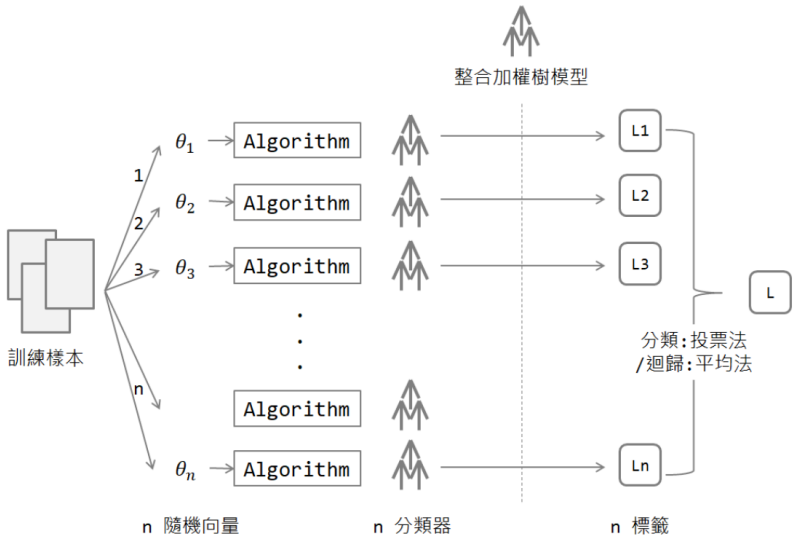
(4)Where?: 在哪種地方可以進行問題改善? 電子商務，金融系統

(5)Why?: 為什麼要進行改善? 讓用戶的信用卡被盜刷時卻無法被即時通知導致無法降低傷害，甚至遭受二次傷害。

(6)How?: 如何解決此問題? 建構分類模型，協助金融機構人員以及用戶快速且準確的得知欣用卡是否有被盜刷或是詐騙。

2. 文獻回顧

2.1. Random Forest



1. 使用 Bagging（又稱 Bootstrap aggregating）演算法將資料生成訓練資料：假設有 N 的樣本，每個樣本有 M 個變數(特徵)，以隨機抽取但會放回的方式取得數個樣本而組成訓練資料，共生成 n 資料集。
2. 對每個訓練資料集，生成不同的隨機向量 θi，隨機選擇 m 的變數(且 m < M)，對其中每個變數都嘗試分割，以選擇達到最小的 Gini 係數的分割方進行分裂，生成 CART 樹。
3. 讓每顆樹生長，不進行剪枝。
4. 對這 n 顆樹的結果進行組合：若為分類資料，則用簡單多數投票法，若為迴歸，則用平均法。

優點：

1. 隨機森林的基礎演算法是基於 CART 演算法，故可以處理類別資料與連續資料
2. 它可以出來很高維度（特徵很多）的數據，並且不用降維，無需做特徵選擇
3. 實現起來比較簡單

缺點：

1. 因需要儲存每顆樹的資訊，所以執行時間較久
2. 隨機森林已經被證明在某些噪音較大的分類或回歸問題上會過擬合。

2.2. CatBoost

CatBoost 是俄羅斯的搜索巨頭 Yandex 在 2017 年開源的機器學習庫，是 Boosting 族算法的一種。CatBoost 是一種基於對稱決策樹（oblivious trees），主要解決的痛點是高效合理地處理類別型特徵，這一點從它的名字中可以看出來，CatBoost 是由 Categorical 和 Boosting 組成。此外，CatBoost

還解決了梯度偏差（Gradient Bias）以及預測偏移（Prediction shift）的問題，從而減少過擬合的發生，進而提高算法的準確性和泛化能力。不同的類別型特徵的任意組合都可視為新的特徵。例如，在音樂推薦應用中，我們有兩個類別型特徵：用戶 ID 和音樂流派。如果有些用戶更喜歡搖滾樂，將用戶 ID 和音樂流派轉換為數字特徵時，根據上述這些信息就會丟失。結合這兩個特徵就可以解決這個問題，並且可以得到一個新的強大的特徵。然而，組合的數量會隨著數據集中類別型特徵的數量成指數增長，因此不可能在算法中考慮所有組合。為當前樹構造新的分割點時，CatBoost 會採用貪婪的策略考慮組合。對於樹的第一次分割，不考慮任何組合。對於下一個分割，CatBoost 將當前樹的所有組合、類別型特徵與數據集中的所有類別型特徵相結合，並將新的組合類別型特徵動態地轉換為數值型特徵。CatBoost 還通過以下方式生成數值型特徵和類別型特徵的組合：樹中選定的所有分割點都被視為具有兩個值的類別型特徵，並像類別型特徵一樣被進行組合考慮。

優點：

1. 可以處理類別型、數值型特徵；
2. 減少了對超參數調優的需求，並降低了過度擬合的機會
3. 性能卓越

缺點：

1. 不同隨機數的設定對於模型預測結果有一定的影響，所以需要不斷嘗試不同的參數調整。

3. 研究架構

3.1. 研究架構

本研究之研究架構，大致可分為四個步驟:(1)資料輸入、(2)資料前處理、(3)模型建構、(4)輸出結果、(5)模型訓練，最後得出分類成效最佳的模型。

3.1.1. 資料輸入

本研究透過 Kaggle 線上開放式資料庫所提供之 BankSim 銀行付款模擬器，基於西班牙一家銀行提供的資料交易樣本。總共有 594,643 筆資料，其中有 587443 筆為正常交易的，7200 筆資料為詐騙。每一筆資料有 10 個項目，見下圖一。

	step	customer	age	gender	zipcodeOri	merchant	zipMerchant	category	amount	fraud
0	0	'C1093826151'	'4'	'M'	'28007'	'M348934600'	'28007'	'es_transportation'	4.55	0

圖一：資料項目

以下分別說明不同項目分別代表的意思：

(1) step：代表這筆資料是從開始收集經過幾天。總共是 0-180，代表 6 個月

(2) customer：用戶 id

(3) age：分類年齡(有 8 種)

(4) gender：分類性別(有 4 種)

(5) ziporderOri：郵遞地址

(6) merchant：貿易商 id

(7) zipMerchant：貿易商郵遞地址

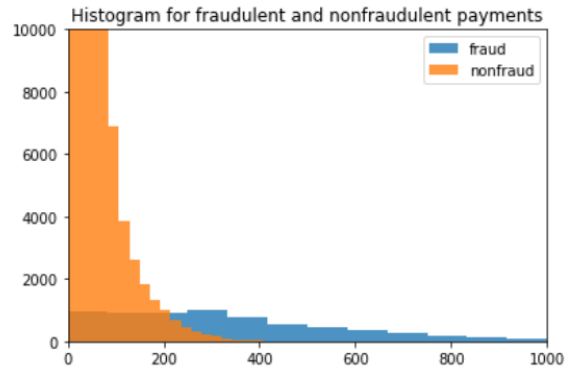
(8) category：分類消費項目(有 15 種)

(9) amount：刷卡金額

(10)fraud：是否被詐騙

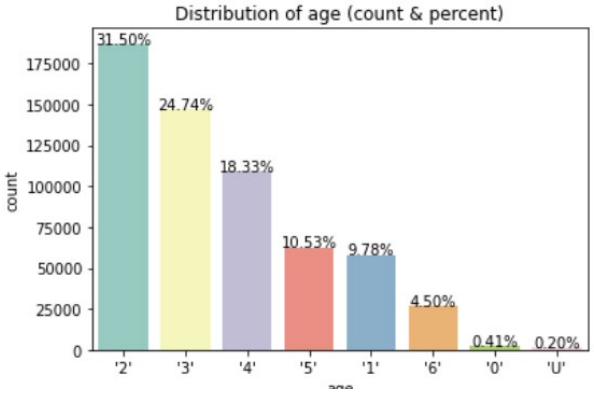
3.1.2. 資料分析

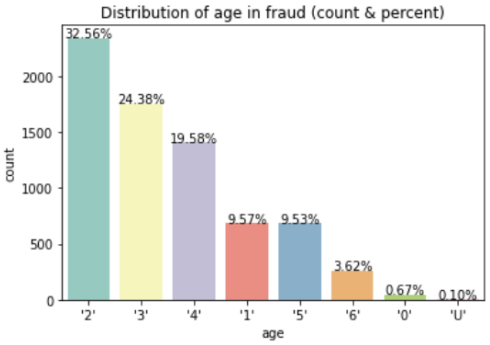
(1) Fraud

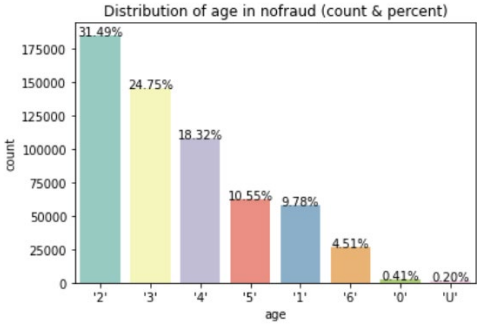


可以看出資料十分不平衡。

(2) age

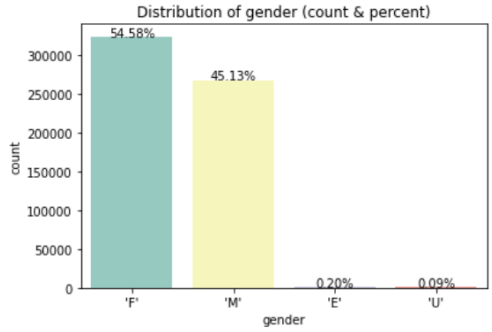


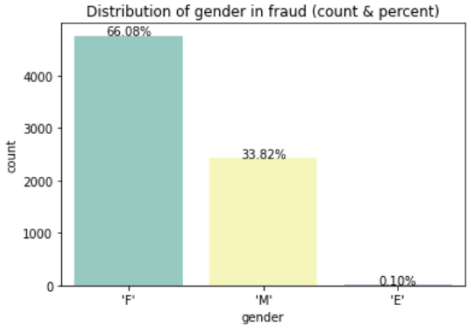


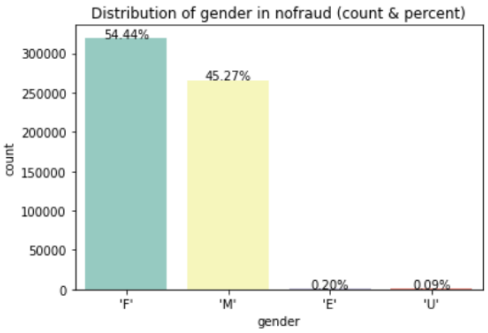


我們可以看出第 0，U 類是最少的而且在底下的兩張圖中受詐騙的也是最少的。

(3) gender

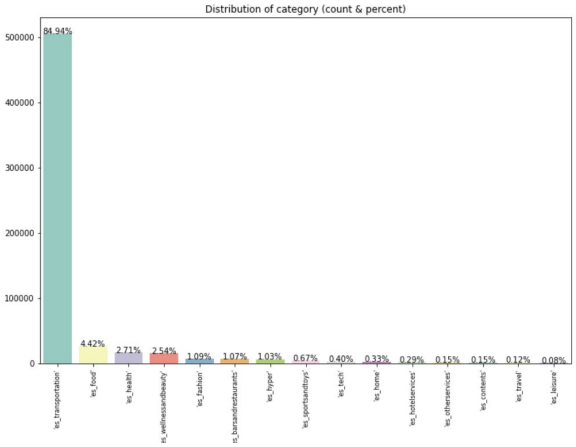


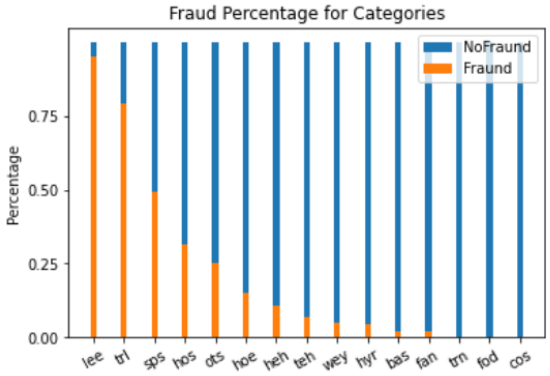




在被詐騙的比例中 U 是沒有的，因此在後座資料處理時將其捨去。

(4) category





可以看出雖然我們 Transportation 是最多的，但是在有殂受騙比例中最高的 leisure

3.1.3. 資料前處理

本研究做之資料前處理包含下列步驟：

(1) 將正常交易做為第 0 類(資料集顯示為 0)，詐騙為第 1 類(資料集顯示為 0)。

(2) 捨棄掉 ziporderOri、zipMerchant 的欄位

(3) 將 gender 欄位中的”M”、”F”、”E”轉成數值 1、2、3，並且捨去”U”。

(4) 將 age 欄位中的”0”、”1”、”2”、”3”、”4”、”5”、”6”、”U”轉成數值 0、1、2、3、4、5、6、7。

(5) 將 category 欄位中的 15 個類分別轉成數值 0-15。

(6) 首先將 594643 筆資料先分出 10%做為測試集，共約 59465 筆資料。

(7) 由於資料集非常不平衡(由 fraud 那一欄位可得知)，會影響到訓練的準確性，因此本研究首先透過過採樣(SMOTE)的方式，共 528686 筆資料。

(8) 將這 528686 筆資料，拆分為 90%做為訓練集，共約 475817 筆。10%做為驗證集，共約 52869 筆

3.1.4. 模型建構

選用 RandomForest 作為我們的模型。

3.1.5. 參數設定

模型建構完成後，本研究對此模型進行訓練，分析不同超參數模型的分類結果。訓練過程改變 tree 的深度 ，n_estimators，做為模型的超參數。詳細參數如下：

```
rf_clf = RandomForestClassifier(n_estimators=200, max_depth=13, random_state=42,
                                verbose=1, class_weight="balanced")
```

3.1.6. 輸出結果

在測試集資料不平衡的情況下，單看準確率並無法完全代表模型的好壞程度，必須同時考慮精確度以及召回率，可以做為評估模型好壞的依據。模型的分類結果如下所示:

Accuracy: 0.988
Precision: 0.491
Recall: 0.927

我們可以看出雖然準確率高達 98.8%，但是在精確度方面卻只有 49.1%，代表模型判斷此筆交易為詐騙的情形，實際上卻只有不到一半的情形是詐騙。就會導致用戶端使用此系統，一直被通知他們的信用卡在被詐騙，但實際上卻沒有。導致用戶失去對此系統的信心。

3.2. 模型改善

由於隨機森林不能夠做出超越訓練集資料範圍的預測，這可能導致在某些特定噪聲的資料進行建模時出現過度擬合，進而對於少樣本數據的預測結果變得更差了，導致準確率很高，但是在精確度方面卻很低的情況。針對覺得模型有不足的地方進改善，改善方法如下

3.2.1. CatBoost

使用不 Catboost 模型，減少了對超參數調優的需求，降低了過度擬合的機會，這也使得模型變得更加具有通用性，同時可以處理類別型特徵(此資料集裡 category 裡全是類別型特徵)，並且可以實現特徵組合，使模型效果更快更好。參數如下：

```
clf = CatBoostClassifier(iterations=100,
                                                    learning_rate=0.01,
                                                    depth=6,
                                                    eval_metric='AUC',
                                                    random_seed = 42,
                                                    bagging_temperature = 0.2,
                                                    metric_period = 20
                                                    )
```

Accuracy: 0.997
Precision: 0.91
Recall: 0.65

3.2.2. 超參數調整

針對 Catboost 進行超參數調整，並且加入過擬合的機制。

```
clf = CatBoostClassifier(iterations=200,
                                                    learning_rate=0.04,
                                                    depth=12,
                                                    eval_metric='Recall',
                                                    random_seed = 42,
                                                    bagging_temperature = 0.2,
                                                    od_type='Iter',
                                                    metric_period = 20,
                                                    od_wait=25
                                                    )
```

加入 od_type，在過擬和發生時可以去檢查迭代次數，並且在最小化損失函數後，可以在迭代 od_wait 次。

超參數調整如下表：

	Lr:0.02 Dep:12	Lr:0.03 Dep:12	Lr:0.04 Dep:12	Lr:0.02 Dep:13	Lr:0.03 Dep:13	Lr:0.04 Dep:13
Accuracy	0.97	0.997	0.997	0.97	0.997	0.997
Precision	0.92	0.92	0.94	0.9	0.89	0.92
Recall	0.81	0.80	0.82	0.82	0.83	0.80
	Lr:0.02 Dep:11	Lr:0.03 Dep:11	Lr:0.04 Dep:11	Lr:0.02 Dep:14	Lr:0.03 Dep:14	Lr:0.04 Dep:14
Accuracy	0.97	0.997	0.997	0.97	0.997	0.997
Precision	0.91	0.92	0.94	0.92	0.92	0.92
Recall	0.8	0.8	0.81	0.81	0.81	0.81

3.2.3. 改變訓練集，測試集，驗證集大小

我們比較 3.2.2 和 3.2.1 的結果可以發現 recall 卻上升了，但是卻與原本 3.1.6 的 recall 還要小，代表實際是詐騙，但是卻沒有發現的機率提高了。因此猜想有可能是因為數量少的類別重複抽取太多次，導致模型發生過擬和，因此將茲療集切分大小做改變改成 Train : Validate : Test = 6 : 2 : 2，然後重複一次超參數調整的步驟，如下表：

	Lr:0.02 Dep:12	Lr:0.03 Dep:12	Lr:0.04 Dep:12	Lr:0.02 Dep:13	Lr:0.03 Dep:13	Lr:0.04 Dep:13
Accuracy	0.997	0.997	0.997	0.997	0.997	0.997
Precision	0.94	0.94	0.94	0.94	0.94	0.94
Recall	0.84	0.86	0.85	0.84	0.84	0.84
	Lr:0.02 Dep:11	Lr:0.03 Dep:11	Lr:0.04 Dep:11	Lr:0.02 Dep:14	Lr:0.03 Dep:14	Lr:0.04 Dep:14
Accuracy	0.97	0.97	0.97	0.997	0.997	0.997
Precision	0.94	0.94	0.94	0.92	0.94	0.94
Recall	0.83	0.84	0.85	0.86	0.85	0.86

4. 研究結果

4.4. 最終模型架構與結果

根據本研究改善後所提出的模型，準確率相比之前測試集準確率達 0.997，比起原先所提出 的測試集 0.988 有微幅的的提升，但是在精確率可以看到我們提出來的新模型達 0.92，比起原本的模型 0.491 還要進步許多。雖然召回率的部分從 0.927 下降至 0.86，但是跟精確率提升的部分相比而言，這點損失我們是承受得起的。

5. 結　　論

5.1. 未來改善

本研究只測試 Catboost 幾個參數而已，但是實際上可調參數高達 102 種，因此未來可以研究不同參數所改變的項目是什麼以及分析不同參數組合所帶來的效果，是值得去探討的。