

HTML2021 Final Project

R10521516 廖浚評

R10521517 鍾慕捷

B06501001 陳岳謙

1. 資料前處理

1.1. 數據整理與清洗

此份資料有許多缺失值，若直接對所有缺失填值，恐有太多雜訊混入資料中，因此以下根據不同的資料檔有不同的處理方式，分別進行說明。

- Demographics

性別與結婚兩變數直接轉為0, 1。

年齡、小於三十歲及年長者三變數互相關聯，除非三變數皆遺失，則可互相進行差補(例：小於三十歲者不會是年長者)，插補完成後，小於三十歲與年長者的缺失皆從770筆降至約100筆，對補足資料有很大的幫助。

是否有家屬及家屬人數也可以用類似邏輯補缺失值，家屬人數為0者是否有家屬也為0，但反之，無法確定有家屬者實際家屬人數為何，則統一設為1人。

最後認為已經將顧客進行年輕與年長者之分類，故將年齡變數刪除。

- Location and population

此兩個檔案提供顧客居住城市、經緯度、Zip Code及城市人口，因考量到資料中有超過1000個不同城市會造成後續資料處理的麻煩，因此決定將每個顧客所在城市的人口數作為最終變數。

第一步，因 Zip Code 與經緯度為一一對應，故對 Zip Code、經緯度、經度、緯度四個欄位進行互相差補後，能將缺失的 Zip Code 從770個降為100個。

再將各個顧客的 Zip Code 對應之人口數做為其變數後，就能將其他變數(經緯度、Zip Code)刪除。

- Services

是否推薦朋友與推薦朋友數、使用電話服務與平均每月遠距收費、網路服務與網路類型、網路吃到飽與超額用量收費等兩兩變數皆有一定關係，可透過邏輯判斷互相插補，以減少缺失資料。

另外，費用相關變數也可以透過互相關聯以插補遺失值，經測試資料後，以下公式成立： $\text{總營收} = \text{總收費}(\text{約等於客戶期數} * \text{目前月費}) + \text{總遠距收費}(\text{等於客戶期數} * \text{平均月遠距收費}) + \text{總額外收費} - \text{總折讓費用}$ 。

透過各變數的關聯，都能使我們的資料補足一半以上的缺失值，大幅降低資料的完整性。

- Satisfaction
滿意度並無其他欄位幫助補足缺失值，因此直接作為變數。
- 標準化與補足空值
將變數分為兩類：數字部分，將空值填為平均數，再對所有值進行標準化；若為類別變數，則將空值填上眾數，其中，若該變數之類別超過兩種（如 Offer, Internet Type 等），進行 one-hot encoding，以解決種類變數無線性相關的問題。

另外也嘗試了互相預測的方法來填補空值，使用Scikit-Learn 的 IterativeImputer，進行10次迭代以填補空值。

1.2. 針對 Imbalanced Data 的處理

用於訓練的客戶狀態，有將近74%都屬於無流失、11%因競爭對手而流失，其他的流失類別都只佔資料的5%以下，可見分類之目標非常不平均，因此我們嘗試了兩種方法解決資料不平衡，避免訓練失衡的問題。

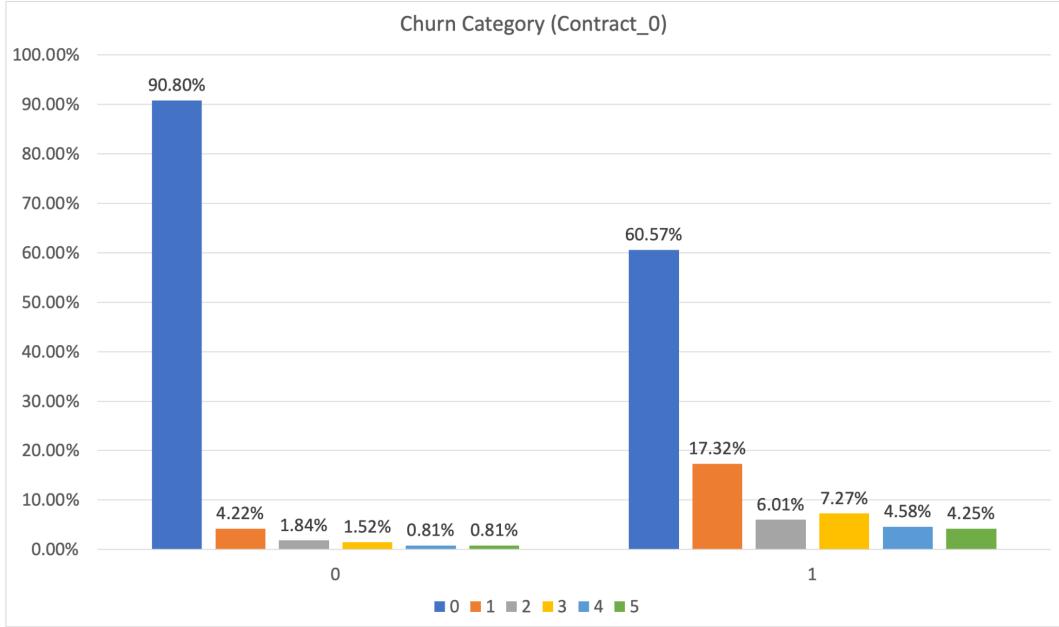
- RandomUnderSampler, RandomOverSampler
Scikit-learn 針對不平衡資料設計的套件，能將過多的類別以隨機抽取的方式降低數量，或是相反的將較少類別的資料，以重複抽取的方式增加資料數量，都可以試圖解決資料類別不平衡對模型預測的影響，兩套件都能夠直接指定各類別所需的資料數，相對簡單明瞭。
- SMOTE
也就是Synthesized Minority Oversampling Technique，這個方法解決了隨機抽取樣本來擴大而容易造成，SMOTE主要概念也就是在少數樣本位置近的地方，人工合成一些樣本。可是缺點也顯而易見，那就是對於所有少數樣本都會做過採樣，然而，大多時候並不是所有少數樣本都失去鑑別度，真正沒有鑑別度的是那些跟多數樣本混合在一起的少數樣本。因此還是難免有overfit的情況發生。

2. 特徵選擇

因為前面做過one_hot_encoding處理或是參數有些是互相相關的，可以知道在眾多參數之中有些是不太重要的而且會影響模型使模型Overfit，因此是需要做一些 Regularized的，以下我們做了兩種方式。

2.1. Excel 條紋分析

我們簡單地透過條紋圖表分析不同變數與流失種類的關聯，挑選比較有影響及解釋力的，這邊舉簽約種類(1為每月續約、0為一年或兩年約)為例。



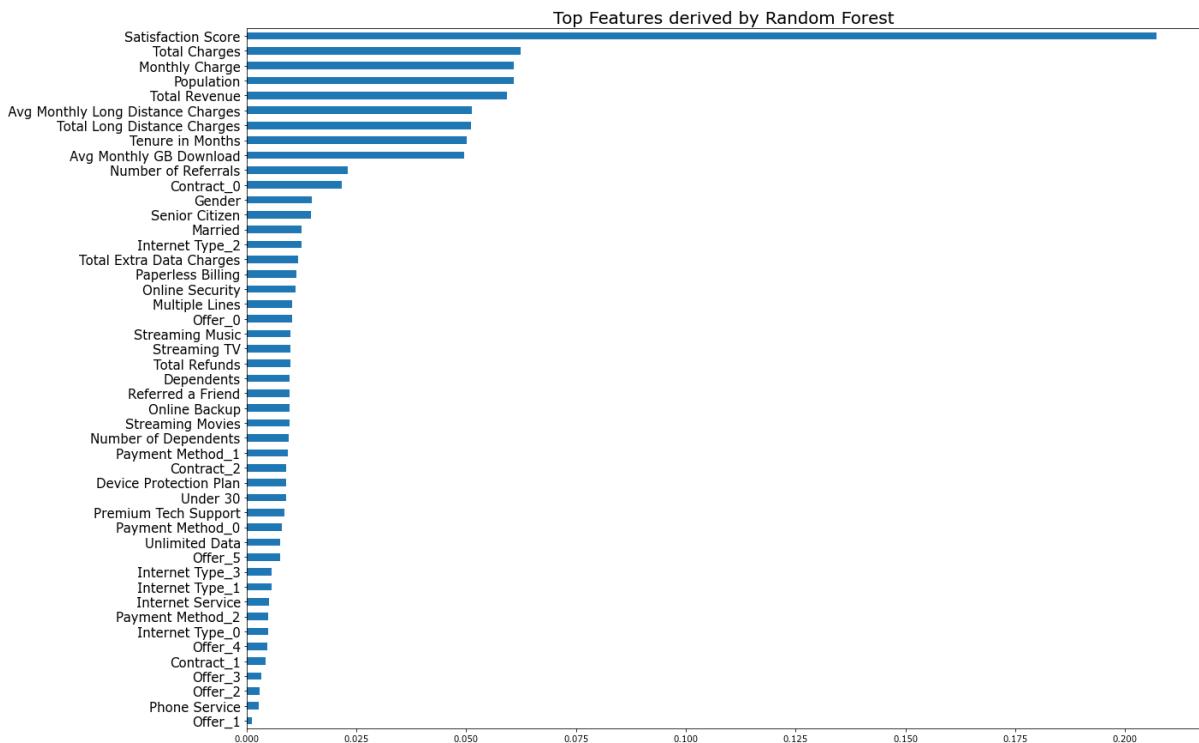
從圖表中可以發現合約若為月份而非一年以上的顧客中，有將近40%的顧客會流失，相較其他兩者的低於10%差距相當大，推測為願意綁約越久的顧客月忠誠，較不易流失，因此這個變數就被視為應選擇之參數，也很合乎邏輯。

2.2. Using lasso regression and random forest and RFE(recursive feature elimination)

Lasso regression是一種透過在線性迴歸中加入L1懲罰函數的回歸模型，目的在於讓模型中不要存在過多的參數，當模型參數越多時懲罰函數的值會越大

$$\text{minimize} \left\{ SSE + \lambda \sum_{j=1}^p |\beta_j| \right\}$$

但是Lasso regression 模型會將不具影響力的變數迴歸係數變成0，等於可以自動化的進行Feature selection，但這不是我們要的，我們認為這樣會犧牲掉太多的模型正確性，也就是刪除過多的feature。因此我們先用一個random forest來決定，在模型上的feature importances，之後得出以下的這張圖。



藉由這張圖，我們經過測試，認為feature留下17個會是一個比較好的選擇。最後利用sklearn內的RFE用lasso regression來回歸刪除feature到只剩下17個feature。

3. 模型訓練與參數調節

3.1. XGBoost Classifier

本節使用XGBoost作為學習的模型。XGBoost全名為eXtreme Gradient Boosting，與隨機森林演算法相似，在生成每一棵樹的時候採取隨機採樣的方式，但同時每一棵樹又能達到修正前面一棵樹犯錯的地方是結合Bagging和Boosting之優點的做法。

XGBoost的參數共分為通用參數、Booster參數及學習目標參數等三類，在本次分析中，我們主要透過調整其中的Booster參數來提升模型的預測力，以下將針對我們有調整的Booster參數進行簡單的介紹。

- n_estimators: 決定樹的數量。
- eta: 就是learning rate, 降低此值可以提升模型的穩定性。
- max_depth: 決定每棵樹的最大深度, 值越大越容易overfit。
- min_child_weight: 決定最小葉子節點之權重和, 提高此值可以避免overfit。

- gamma: 決定節點分裂時所需的最小損失函數下降值，值越大算法越保守。
- subsample: 決定對於每棵樹隨機採樣的比例，降低此值可以避免 overfit。
- colsample_bytree: 類似max_features, 決定每顆隨機採樣的特徵的比例。

我們選擇使用Scikit-learn中的Grid Search作為調整參數的工具，設定5次折疊的交叉驗證，並得到n_estimators=1000、eta=0.01、max_depth=5、min_child_weight=1、gamma=0.1、subsample=0.8、colsample_bytree=0.8的結果。

3.2. Random Forest Classifier

本節使用Random Forest Classifier 作為學習的模型，特徵選擇的方式是Using lasso regression and random forest and RFE。其中有調整的參數有以下

- n_estimators: 決定樹的數量。
- min_samples_leaf : 最小樣本葉片大小。葉是決策樹的末端節點。較小的葉子使模型更容易捕捉訓練數據中的噪聲。
- min_samples_split: 資料數目不得小於多少才能再產生新節點。越大避免overfit
- oob_score : 這是一個隨機森林交叉驗證方法。它和留一驗證方法非常相似，但這快很多。這種方法只是簡單的標記在每顆子樹中用的觀察數據。然後對每一個觀察樣本找出一個最大投票得分，是由那些沒有使用該觀察樣本進行訓練的子樹投票得到。

我們選擇使用Scikit-learn中的Grid Search作為調整參數的工具，設定5次折疊的交叉驗證，並得到n_estimators=1000、min_samples_split = 14、min_samples_leaf = 51 的結果。而我們認為n_estimators 主要影響的是計算時間、應選擇越大越好，所以最後選擇n_estimators=8888

3.3. Ensemble

本節將3.1&3.2的結果利用投票的方式在決定出最後的預測。

4. 預測結果

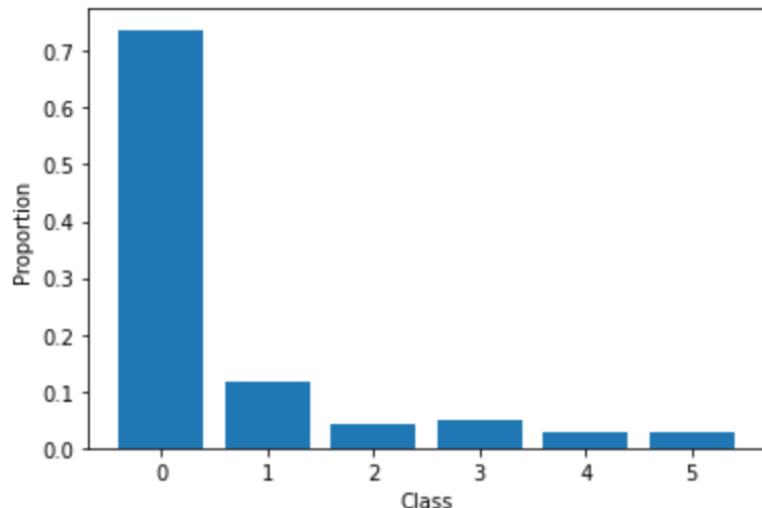
	public score	排名	private score	排名
1.XGBoost	0.32886	73/127	0.32361	55/127
2.RFC	0.34406	52/127	0.33458	43/127
3.XGBoost&RFC	0.30737	100/127	0.32726	54/127

上表是三個模型分別在本次kaggle競賽上的表現。

在分數的表現上，雖然經過資料前處理、模型的調整有得到小幅度的上升，準確率卻都大致在0.30到0.34之間浮動，無法達到足以在業界應用的預測力。關於這方面，我們推測有兩項造成此問題的主要原因：資料的缺失以及分類的不平衡。

首先，在先前資料前處理的章節中提過我們對於各項特徵之缺失值的填補方式，然而實際上在缺失值過多的情況下，不論是填補平均值、眾數或前後值等做法其實都有相當風險，在訓練時容易降低各項特徵對於結果的預測力，讓模型的表現無法通用在測試集之上。

另外，資料類別的不平衡也是我們在模型訓練過程中的一項挑戰。下圖為各分類在原始資料的比例。按照類別的順序分別佔了原始資料的73.8%、11.6%、4.2%、4.8%、2.9%、2.7%。



其中在4226筆原始資料當中，第0類的資料就佔了3118筆，造成模型在預測時容易預測過多的第0類資料。以我們使用的SMOTE方法為例，雖然能透過生成少數類別的數據來緩解這個問題，但其限制在於無法生成現有數據範圍以外的資料點，而是只是在bias和variance之間取得平衡，因此SMOTE在結果上並不會明顯影響模型在預測上的準確率。

最後，在三個模型的比較上，以分數來說RFC表現最好，混合投票不太行，原因應該是兩個模型都是base on 樹理論，應改選用不同的才可能會有比較好的表現。以程式運算能力來說，RFC的樹的數量設置的很高，在運算時間上比起XGBoost花費很久，尤其是縮小數量後能力有明顯下降，這部分我們認為XGBoost在面對更大的數據量時可能可以比較快，在特徵選擇上，若是公司的營運單位，domain knowledge 應該是足夠分辨哪些特徵該怎麼清洗、整理，以泛用性來講，由電腦自動跑出該刪除的特徵也是個不錯的選擇、表現也不錯，可以更簡單的挖掘出大概想要的資訊，因此我們認為RFC會是我們最後的推薦系統。