Churn prediction

統計112　H24081040 梁鈞翔:
https://github.com/ForwardCourage/IDS-HW4

統計112　H24086040 張易安:https://github.com/ian0416/HW4?fbclid=IwAR1PQgP2lSxAvcC-rvXDqmsRhy1pA5ryRgb9r8GTT65_8W-zJFsQbHT4T1U

競賽敘述與目標

此次競賽的目的是要用bank customer churn prediction資料集，以train資料訓練出最佳模型，並以此模型預測test資料，已達成最佳準確率與final score。

資料前處理

資料敘述:

- Rownumber:原資料中第幾行(不重要)
- CustomerId:銀行客戶的ID(不重要)
- Surname:客戶的姓氏(不重要)
- CreditScore:信貸評分，連續型
- Geography:居住國家，類別型
- Gender:性別，類別型
- Age:年齡，連續型
- Tenure:已使用這家銀行的服務時間
- Balance:餘額，連續型
- NumOfProducts:購買產品數，連續型
- HasCrCard:是否持有信用卡，類別型
- IsActiveMember:是否為活躍用戶，類別型
- EstimatedSalary:該客戶薪資的估計量，連續型
- Exited:是否退戶，1為是

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	551	15806307	Trevisano	720	Spain	Male	38	5	114051.97	2	0	1	107577.29	0
1	6897	15709621	Martin	682	France	Female	54	4	62397.41	1	1	0	113088.60	1
2	4588	15619340	Palmer	672	France	Female	31	5	119903.67	1	1	1	132925.17	0
3	291	15620746	Napolitani	592	Spain	Female	40	4	104257.86	1	1	0	110857.33	0
4	1673	15646372	Yao	753	Spain	Male	42	5	120387.73	1	0	1	126378.57	0

特徵處理與分析

已經確認train、test皆無NA，故先將認定無相關的RowNumber、CustomerId、Surname去除後，依照分類法決定變數處理方式:

- Tree-based model: 將類別型以LabelEncoder()做處理，連續型變數保持原樣。
- 其他: 類別型以One-hot encoding處理，連續型變數則進行標準化。
- 雖然NumOfProducts為連續型變數，但是只有1~4四種數值，我們嘗試將其以One-hot encoding處理後，發現預測結果比直接進行標準化的效果更好。
- 我們雖直覺認定tenure、balance、estimated salary與是否離開為負相關，但並未深入解析。

預測訓練模型

嘗試過的演算法:Neural Network, SVM, Decision Tree, Random Forest, XGBoost, CatBoost, Gradient Boosting, AutoML.

AutoML為python的module兼演算法，可根據給定條件依序找出效果最佳演算法，並依權重整合模型。

各模型在初步上傳至競賽網站測試的最佳結果如右表所示。

最終使用模型:XGBoost

演算法	Accuracy	Precision	f-Score
Neural Network	0.8700	0.7119	0.6176
SVM rbf	0.8775	0.8333	0.5882
SVM poly	0.8800	0.8222	0.6066
Decision Tree	0.7900	0.4588	0.4815
Random Forest	0.8700	0.6984	0.6286
XGBoost	0.8875	0.7759	0.6667
Gradient Boosting	0.8775	0.7500	0.6316
AutoML	0.8825	0.7778	0.6412

SVM

因為資料為線性不可分，所以我們並未使用線性SVM，使用的是kernel為rbf與poly的非線性SVM。

為了嘗試減少overfitting，我們試著將超參數C降低至0.8，發現相較起預設值1會有較好的表現。

另外gamma從預設的scale改成auto時在rbf的表現有顯著的提升，不過在poly的f-Score卻是大幅下降。

整體來說SVM的效果不算差，不過相較XGBoost還是遜色一些，因此最終我們選擇將重心放在XGBoost上面。

選用參數	Accuracy	Precision	f-Score
kernel=rbf gamma=scale, C=1	0.8725	0.7826	0.5854
kernel=rbf gamma=scale, C=0.8	0.8725	0.7955	0.5785
kernel=rbf gamma=auto, C=0.8	0.8775	0.8333	0.5882
kernel=poly gamma=scale, C=1	0.8775	0.7917	0.6080
kernel=poly gamma=scale, C=0.8	0.8800	0.8222	0.6066
kernel=poly gamma=auto, C=0.8	0.8525	0.9091	0.4040

Neural Network

此方法以pytorch.nn 進行實作，因為對torch的經驗尚淺，故以簡單的Linear 和 ReLU 建構模型，並在最後加入Sigmoid進行轉換(因為此次競賽任務為二元分類)

Optimizer 採用 Adams

Loss function 採BCELoss

然而此方式雖然可簡單實作，卻在三次調整架構之後便出現overfitting，在訓練集表現越好，預測表現越差。

因為nn的架構本身由使用者確定，對架構使用的心得不足，故無法繼續研究下去，只好放棄

CatBoost

僅以catboost 套件簡單進行預測並檢查上傳結果，確認其效益遠低於XGBoost後便放棄CatBoost。

AutoML

AutoML可在輸入X, y之後，嘗試許多分類方法，並依照分類表現進行排名與權重分配，而後 automl本身會成為整合模型。選出的方法數與條件中給定的總運行時間 (time_left_for_this_task)和每個方法允許的最大運行時間 (per_run_time_limit)相關。此次競賽中以 time_left_for_this_task = 180 & per_run_time_limit = 30 執行。

註: 此方法除了將不必要的欄位去除，不須另外做前處理。執行過程會自行決定處理方式。

結果: 排名結果顯示 gradient boosting 和 random forest 分別為第一與第二，並以此去預測，其 F-score 僅次於XGBoost，並另外以gradient boosting 再預測。

Gradient Boosting

基於AutoML的排名與權重，另外以gradient boosting 嘗試，但效果不佳，故放棄gradient boosting。

XGBoost(最終選定模型) Hyperparameter Tuning

在做完第一次預測後，我們發現XGBoost在參數皆為預設值之下就能達到我們試過的模型中最好的表現。之後我們在其他參數保持不動的情況下試著將max_depth稍微降低，發現在4時可以將表現提高不少，eta也在嘗試降低之後又將表現提高了一些，因此我們決定用cross-validation去尋找此兩個參數的最佳組合作為最終的模型。

選用參數	Accuracy	Precision	f-Score
max_depth=6, eta=0.3 (default)	0.8650	0.6885	0.6087
max_depth=4, eta=0.3	0.8875	0.7581	0.6763
max_depth=4, eta=0.08	0.8875	0.7759	0.6667

在調整超參數的過程中，以GridSearchSV()尋找最佳的max_depth、eta(其他超參數因尚未了解其意義而決定暫時保留)，cross-validation 的選組方式以ShuffleSplit(n = 5, test_size = 0.2)

- max_depth: 3~8
- eta: 0.00~0.99

總運行次數為(8-3+1) * 100 * 5 = 3000次

最後得到最佳參數 max_depth 為 5 , eta 為 0.0 (最終上傳的模型)

最終模型	Accuracy	Precision	f-Score
max_depth=5, eta=0.0	0.8925	0.8148	0.6718

感想與心得

這次競賽一開始，我天真的認為只要直接把資料做好前處理之後，直接丟進 nn model裡便能輕易跑出好結果，畢竟 neural network的概念十分強大。然而，現實卻是在不斷調整結構後，預測結果一次比一次慘澹。考慮到自己並不擅長 nn model，加上組員已經以其他方法不斷嘗試，並已經不斷在改善 xgboost效果，因此決定轉戰 xgboost。我原本並不清楚 over-fitting和 cross-validation的效益，但在這次 nn model 的挫折，以及實際操作 cross-validation後，模型預測效果有了卓越的提升，排名也從第17名一次躍升至第 5名。我認為 cross-validation最大的意義在於將 validation set分割出來，藉由犧牲部分資訊以考驗模型的精確度以及提高對雜訊的容忍度。既是實際預測的模擬，也是盡可能消去 training set 和 testing set之間的歧異。實際上，這次我們只針對兩個超參數進行調整。另外，我們並不完全了解 XGBoost的運作模式，但在 python的官方 documentation中，eta 是類似於 learning rate的參數。因此，以 cross-validation調參後得出最佳 eta=0時，我們都感到非常驚訝。雖然這次 XGBoost是最好的模型，但實際上必然有其他更好的方法。

-統計112 梁鈞翔

在開始這個競賽時，因為算是這堂課第一次在沒有太多指示的情況下要對一筆真實的資料進行預測，一開始跟組員決定先分開做各自的嘗試時發現其實要考慮的東西真的還不少，像是資料該如何做前處理比較適當，或是到底該使用哪一種演算法，在沒有太多經驗的情況下決定一一做實際的嘗試，不斷上傳結果後選擇出表現最好的繼續著手做更多的調整。 **在創建模型時，雖然都有套件可以使用，但是我認為比較困難的是如何對參數進行調整來提高預測表現，以** XGBoost為例，因為能 夠調整的參數其實非常多，也有很多互相影響、搭配的問題，所以其實花了不少時間上網自學，了解各個參數的意義，最後在做了一些嘗試之後決定以cross-validation來選出最好的 eta與max depth，雖然最後找出的組合將我們的排名升上第五，不過其實我們還是沒有完全了解造成這個結果的原因，更不用 說還有很多其他更細微的參數來不及做更多嘗試，不過在做這份作業時與組員討論以及上網自學的過程中真的在無形中累積了不少知識。

-統計112 張易安