HW4 Report

1. Members

a. 統計 112 劉恩兆 B14081027

b. 統計 112 宋穎恩 B14081302

2. Introduction

在此次的競賽中，我們需要從多筆銀行客戶的個人資料（如：性別、年齡、資產、擁有信用卡數量等）預測此客戶最終是否會流失，也就是不再有任何交易產生。此次的資料集以 8：2 的比例拆分為訓練集與測試集，分別有 8000 筆集 2000 筆，而我們需要在測試集預測的即為訓練集中的「Exited」欄位；預測方法除了使用課程中老師講授的機器學習方法外，本組也使用了更加進階的機器學習方法以及深度學習，以獲得更加突出的預測表現。以下為訓練集資料簡要圖示：

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
0	551	15806307	S2336	720	S2	Male	38	5	114051.97	2	0	1	107577.29	0
1	6897	15709621	S1500	682	S0	Female	54	4	62397.41	1	1	0	113088.6	1
2	4588	15619340	S1865	672	S0	Female	31	5	119903.67	1	1	1	132925.17	0
3	291	15620746	S1672	592	S2	Female	40	4	104257.86	1	1	0	110857.33	0
4	1673	15646372	S2532	753	S2	Male	42	5	120387.73	1	0	1	126378.57	0
5	648	15649129	S1548	575	S0	Female	42	5	104472.9	1	1	1	71641.38	0
6	6113	15729557	S37	572	S0	Male	37	6	135715.66	1	1	0	115928.95	0
7	8957	15579112	S750	753	S1	Female	34	6	124281.61	1	1	0	89136.06	0
8	1678	15680895	S2079	546	S0	Male	46	3	62397.41	2	1	1	79809.09	0
9	5202	15580935	S252	657	S0	Male	45	4	141238.54	2	0	0	95281.51	0
10	4868	15738150	S1717	617	S2	Male	35	4	62397.41	2	1	1	132607.99	0

To be predicted

3. Data Preprocessing

3-1. 刪除不重要變數

在資料集中有些欄位對於預測效果不具任何意義，如：RowNumber、CustomerId、Surname，因此，直接於訓練集與測試集將此三欄刪除。

3-2. 檢查遺失值

使用「is.na()」檢查是否含有遺失值需要填補，結果於訓練集與測試集皆並未發現含有遺失值。

3-3. Standard Sclaer

將所有連續型變數，如：CreditScore、Balance、EstimatedSalar，標準化，使得數值的平均值為 0，標準差為 1，讓資料不會形成有偏的數據，讓模型快速收斂。

此外，也曾嘗試使用過 normalization，但相較 standard scaler，normalization 於後續模型表現皆較差，因此統一選擇使用 standard scaler 作為連續型變數處理方法。

3-4. One-hot Encoding

經由 nunique()確認後，不論是在訓練集還是測試集中的「Geography」欄位皆只有三種不同的值，分別是法國、西班牙及德國。因此，即可以將沒有程度上差異的 nominal data，如：Geography，使用 one-hot encoding 編碼，避免使用一般 encoding 換成數值的方式造成有大小之分；而總共只有三種不同的數值，也不會造成維度災難。

3-5. Encoding

除了上述資料前處理方法外，還有像是性別變數尚未轉成數值，因此就直接將 Female 編碼成 0、Male 編碼成 1，以方便投入模型。

4. Methodology

4-1. scikit-learn Baseline Models

表一、scikit-learn Baseline Models 表現結果

Model	Accuracy	Precision	Fscore	Final
Logistic Regression	0.8225	0.6000	0.3364	0.5863
SVM	0.3775	0.2190	0.3499	0.3154
MLP (2-layer)	0.8825	0.8125	0.624	0.773
Decision Tree	0.8100	0.5057	0.5366	0.6174
KNN	0.8300	0.5818	0.4848	0.6322
Random Forest	0.8750	0.7547	0.6154	0.7484

表一為經過相同資料處理後，使用 scikit-learn 的機器學習套件，且使用預設參數的結果

4-2. Advanced Models

表二、Advanced Models 表現結果

Model	Accuracy	Precision	Fscore	Final
Gradient Boosting	0.8850	0.7719	0.6567	0.7712
XGBoost	0.8725	0.7031	0.6383	0.7380
LightGBM	0.8725	0.7097	0.6331	0.7384
CatBoost	0.8825	0.7419	0.6619	0.7621
Neural Network	0.8725	0.7955	0.5785	0.7488

表二為經過相同資料處理後，使用較進階的機器學習套件以及深度學習，且使用預設參數的結果，可以發現，相較表一的結果，於 Precision 與 Fscore 皆有顯著提升，帶來更精確的預測。

5. Analysis of Prediction Results

在競賽一開始的策略為使用 Gradient Boosting / tree-based 的模型來取得好成績，而在將 baseline models 與 advanced models 跑完後，就先挑選表現較好的 Gradient Boosting / tree-based 模型來調整參數，以期能衝上前面的名次。

因此，在一開始共挑選了 Gradient Boosting、XGBoost、LightGBM 三種模型來優先調整參數。

5-1. Gradient Boosting Tuning

在 Gradient Boosting 模型中，主要調整的參數為以下四種：learning rate、n_estimators、min_samples_leaf 及 max_depth，而這次使用的調參數方法為依上述順序一次調一種參數，同時其他參數固定不變，在最佳化此參數後即固定不動，依序調整完此四種參數。以下為四種參數的調整紀錄：

【補充說明：之所以不使用 GridSearchCV 是因為認為使用驗證集調參數後還要再上傳才能得到結果，不能依上傳結果彈性調整參數稍嫌不便；但同時，此次使用的調參數方式就不能考慮到參數間的交互影響，導致可能與最佳參數組合擦身而過。】

a. learning rate

表三、Gradient Boosting 調整 learning rate 表現結果

learning rate	Accuracy	Precision	Fscore	Final
0.001	0.8075	0	0	0.2692
0.01	0.8625	0.9583	0.4554	0.7588
0.05	0.8775	0.7800	0.6142	0.7572
0. 1	0.8850	0.7719	0.6567	0.7712
0.25	0.8875	0.7667	0.6715	0.7752

【註 1：其餘參數皆使用預設參數；註 2：表現最佳之參數將使用綠字標示】

由上表可見，在 learning rate=0.25 時模型有最佳表現，因此選定 learning rate=0.25 作為最佳參數。而在挑選參數時傾向優先挑選 Fscore 較大的參數，因為相較 Accuracy / Precision，其較可能在測試集有較穩定的表現。

b. n_estimators

表四、Gradient Boosting 調整 n_estimators 表現結果

n_estimators	Accuracy	Precision	Fscore	Final
10	0.8750	0.8000	0.5902	0.7551
50	0.8750	0.8000	0.5902	0.7551
100	0.8875	0.7667	0.6715	0.7752
500	0.8775	0.7414	0.6370	0.7520
1000	0.8675	0.6875	0.6241	0.7264
5000	0.8625	0.6667	0.6154	0.7149

【註：learning rate=0.25，其餘參數皆使用預設參數】

由上表可見，在 n_estimators=100 時模型有最佳表現，因此選定 n_estimators=100 作為最佳參數。

c. min_samples_leaf

表五、Gradient Boosting 調整 min_sample_leaf 表現結果

min_sample_leaf	Accuracy	Precision	Fscore	Final
30	0.8700	0.7193	0.6119	0.7337
40	0.8725	0.7407	0.6107	0.7413
50	0.8850	0.7541	0.6667	0.7686
60	0.870	0.7193	0.6119	0.7337
70	0.8775	0.7333	0.6423	0.7511

【註：learning rate=0.25、n_estimators=100，其餘參數皆使用預設參數】

由上表可見，在 min_sample_leaf=50 時模型有最佳表現，因此選定 min_sample_leaf=50 作為最佳參數。

d. max_depth

表六、Gradient Boosting 調整 max_depth 表現結果

max_depth	Accuracy	Precision	Fscore	Final
1	0.8800	0.7959	0.6190	0.7650
3	0.8875	0.7667	0.6715	0.7753
5	0.8625	0.6667	0.6154	0.7149
7	0.8775	0.7188	0.6525	0.7496
9	0.8625	0.6897	0.5926	0.7149

【註：learning rate=0.25、n_estimators=100，min_sample_leaf=50】

由上表可見，在 max_depth=3 時模型有最佳表現，因此選定 max_depth=3 作為最佳參數。而最終使用使用的參數分別為：

- learning rate = 0.2
- n_estimators = 100
- min_sample_leaf = 50
- max_depth = 3

模型表現為：

表七、Gradient Boosting 最終表現結果

Model	Accuracy	Precision	Fscore	Final
Gradient Boosting	0.8875	0.7667	0.6715	0.7753

5-2. XGBoost & LightGBM Tuning

XGBoost 與 LightGBM 的調參數方法與 Gradient Boosting 相同，其分別調整及調整後最佳的參數如以下所述：

a. XGBoost
   - learning_rate = 0.01
   - n_estimators = 1000
   - gamma = 0.1
   - max_depth = 9

b. LightGBM
   - learning_rate = 0.005
   - n_estimators = 1000
   - objective = ‘binary’
   - max_bin = 255
   - max_depth = 6

以下整理 Gradient Boosting & XGBoost & LightGBM 之最佳參數表現結果：

表八、Gradient Boosting & XGBoost & LightGBM 最終表現結果

Model	Accuracy	Precision	Fscore	Final
Gradient Boosting	0.8875	0.7667	0.6715	0.7753
XGBoost	0.8750	0.7368	0.6269	0.7462
LightGBM	0.8875	0.7857	0.6617	0.7783

由上表可見，LightGBM 為表現最佳的 Gradient Boosting / tree-based 模型，且相較 XGBoost 有著運算速度較快的優勢，但相較 Gradient Boost 有著 Fscore 較高的劣勢。因此，解決方法可以從 LGBM classifier 改至 LGBM regressor，從原本預測 0 或 1 兩種結果改至預測 0 至 1 的機率值，再將 threshold 從 0.5 往下調整，藉由降低 Precision 來提高 Fscore，也就是稍微放寬認定為 1 的機率，較可能得出平衡的結果。

不過，經由嘗試調整各式 Gradient Boosting / tree-based 模型及參數後，發現始終突破不了上限，排行榜大約都徘徊在第八、第九名附近，因此，選擇改用神經網路來追求更好的預測表現。

5-3. Neural Network Building

這次的神經網路主要是使用 Keras 的框架，主要的設置如下：

- input_dim：12
- layer：2-4 層
- activation：relu、sigmoid
- epoch：200-500

接下來將針對其他設置進行較深層的分析：

a. Layer

經過測試，發現若是 layer 只有兩層，則容易造成 underfitting 的結果，導致模型準確率不足，預測表現不佳；而若是使用 4 層 layer，則又容易造成 overfitting，導致雖然訓練結果良好，但上傳至排行榜後卻得不到好的名次；最終，在此資料集中有較好表現的 layer 數皆為 3 層，推測應與訓練資料筆數為 8000 筆有關。

表九、神經網路不同 layer 表現結果

Layer	Accuracy	Precision	Fscore	Final
2-layer	0.8325	0.5658	0.5621	0.6535
3-layer	0.8675	0.7069	0.6074	0.7273
4-layer	0.8300	0.5570	0.5641	0.6504

b. Neuron

不論在幾層的神經網路，若是設置的神經元個數過高，將使模型複雜度提高，而容易 overfitting；反之，若是設置的神經元個數過低，則將使模型複雜度過低，而導致 underfitting 以下為以三層的神經網路舉例，來看各層設置不同的神經元會有甚麼差異：

表十、3-layer Neural Network 不同神經元表現結果

Neuron	Accuracy	Precision	Fscore	Final
(25,10,1)	0.8800	0.8919	0.5789	0.7836
(15,7,7)	0.8750	0.7647	0.6094	0.7497
(20,10,3)	0.8675	0.8000	0.5470	0.7382
(20,10,1)	0.8750	0.9091	0.5454	0.7765

由上表可見，若是 neuron 設置數量不要相差太多，不同 neuron 設置的表現其實並無相差過大，可能要以相差更多的設置來測試，會有更顯著的表現。

c. Dropout

Dropout 為 Google 提出的一種正規化技術，用來防止過度擬合的問題，其作法是在神經網路的某些層中，隨機丟棄部分的神經元，如此可避免在訓練的過程中有過多的神經元產生複雜的相互適應，讓剩下的神經元在更新資訊後更強健。而在測試中，加入 dropout 的表現得確有所提升，以下同使用 3-layer 神經網路作為示範：

表十一、3-layer Neural Network 有無 Dropout 表現結果

Dropout	Accuracy	Precision	Fscore	Final
Yes	0.8750	0.9091	0.5454	0.7765
No	0.8675	0.9286	0.4952	0.7638

由上表可見，在神經層中夾雜 Dropout 可以得到更好的預測表現。

d. Threshold

一般 threshold 的設定為只要機率值>0.5 即視為 1、機率值<0.5 即視為 0，但在這次的神經網路預測結果中時常會得出 Precision 過高，而犧牲 Accuracy 與 Fscore 的情形，於是我於這次競賽中也有調整 Threshold，以獲得更好的預測表現，以下同使用 3-layer 神經網路作為示範：

表十二、3-layer Neural Network 不同 threshold 表現結果

Threshold	Accuracy	Precision	Fscore	Final
0.5	0.8675	0.9286	0.4952	0.7638
0.48	0.8775	0.9118	0.5586	0.7826
0.46	0.8800	0.9143	0.5714	0.7886
0.44	0.8800	0.8919	0.5789	0.7836

由上表可見，將 Threshold 適度調低可以藉由降低 Precision(也就是放寬認定為 1 的標準)，來提升 Accuracy、Fscore，以得到更好的預測表現。

e. Others

其餘設定如：learning rate、不同資料前處理對神經網路帶來的影響等一些較細微，或是個案類型的模型設定，在這邊礙於篇幅就不過多贅述，詳情可以從 code 來看如何調整。

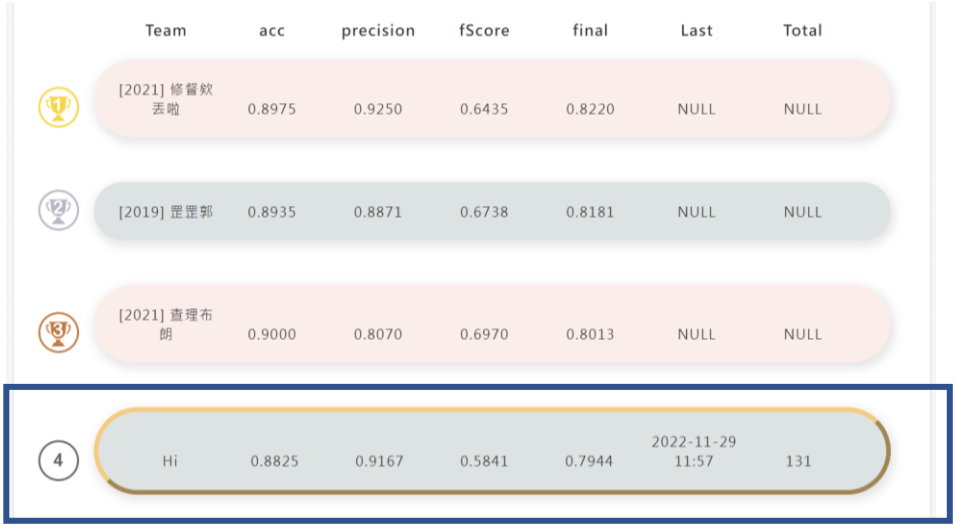
6. Conclusion and Thoughts

6-1. Conclusion

在這次競賽中，原先使用 Gradient Boosting / tree-based 模型就獲得了不錯的排名及表現，但為了追求更高的預測表現，就使用了神經網路，以下為表現最佳之模型設定：

- Model：keras DNN (3-layer)
- Neuron = (25, 10, 1)
- Dropout = (0.5, 0.3)
- Activation = ('relu', 'sigmoid')
- Optimizers = Adam(lr = 0.01)
- Loss = 'binary_crossentopy'
- Metrics = 'accuracy

最終也成功成為這屆 final score 排名第一，排名截圖如下：



6-2. Thoughts

恩兆心得：作為一位大四的統計系學生，同時還有在修習巨量資料分析，這次的競賽難度對我來說不算太高，但還是讓我溫習了基本的機器學習算法。此外，更多的是練習到各種模型的參數調整，以及神經網路的神經元該怎麼設、dropout、層數等等，也很開心能趁機複習 activation 與 loss function 等等基礎卻重要的數學模型。其中，在這次競賽最熱血的是在 leader board 拚排名，也很開心最後成為這屆的第一名，其中的策略調整（像是將 threshold 調低以降低 Precision、提升 Fscore 等）有讓我感到鬥智的感覺，不過也滿可惜沒能超過先前幾屆留下的紀錄，只能說這門課高手如雲 XD