Homework6 report

組名:工科機械
組員:方宣翔 E94066173 工科 110
楊明學 E94066165 工科 110
陳奕佑 E14066436 機械 110

競賽敘述與目標:

此次競賽的任務是要透過分析客戶的多項經濟行為，預測出哪些客戶有可能會不再使用此銀行進行交易。由於資料的 Output 存在不平衡，為了有更好的評估方法，此次作業的評估方法分別使用不同的評估指標(Accuracy, Precision, and F-Score)，並且給予各個指標不同的權重，得出最後的 Final Score。

資料前處理:

將一些無參考價值(不影響結果)的特徵去除，刪除的特徵: RowNumber、CustomerId、Surname，因為 RowNumber 和 CustomerId 每個人都有自己獨立的數字，故並不適合當作特徵，而 Surname 只是匿名化的名字，也不是適合當作參考的特徵。

特徵處理與分析:

在刪除完沒有參考價值的特徵後，最先需要處理的是類別型的資料。這裡使用 Pandas 中 get_dummies()函式，將類別型的資料(Geography、Gender)轉成 One-Hot 的形式，方便後續的處理。

將類別型資料處理完後，由於經觀察後發現各個特徵的數值範圍相差很大，故使用 StandardScaler()函式將資料做標準化的處理，讓後續將各個特徵的數值放入模型中訓練時可以更精確。

預測訓練模型：

處理完特徵後，最後就是找出最適合的訓練模型。一開始試了許多種模型(嘗試過 KNN、Logistic Regression、Decision Tree 、Nonlinear SVC、Random Forest)。在最一開始的嘗試中，上述使用過模型所使用的超參數都是預設值。

Out[4]:

	model name	accuracy	precision	F_Score
0	KNN	0.8500	0.6271	0.5522
1	Logistic_Regression	0.8325	0.6250	0.3738
2	Nonlinear_svm	0.8700	0.7674	0.5593
3	Decision_Tree	0.8150	0.5062	0.5256
4	Random_Forest	0.8600	0.6792	0.5625

圖一、不同模型在使用模型預設的超參數時的各項評估指標表現

在第一輪在嘗試中發現 KNN、Nonlinear SVM、Random Forest 這三種模型有較高的 Final Score，故選定這三種模型做更深入的分析。

一開始在測試上述選定的三種模型(KNN、Nonlinear SVC、Random Forest)時，都是使用 GridSearchCV()函式來自動尋找各個模型的最佳超參數，但是找出來的結果上傳自網站時卻總是不如預期，後來了解到 GridSearchCV()只透過 Accuracy 做評估指標，很明顯的並不適用於此資料，也不適用於網站的 Final Score。

經過考量後，決定分為兩種方式做評估。一種是將資料自己切割成 train data、test data 評估(80% train、20% test)，另一種方法是先透過 StratifiedKFold()函式分割資料，將資料分割成三份做 Cross-Validation 來評估。上一段有提到，單純只看 Accuracy 的結果上傳到網站後得不到高分，所以在此處改良後，決定使用網站上的三項評估指標(Accuracy、Precision、F-Score)都拿來使用。透過計算出各個模型的各種參數以及不同評估方法(Train-Test 切割方法和 Cross-Validation 的方法)，算出上述三個評估指標，並且使用網站所提供的權重，算出各個模型各個參數的 Final Score，藉此來尋找最適合的模型及參數。

第一個嘗試的是 KNN 的模型，透過上述的評估方法找出 KNN 模型在 n_neighbors=11 時為最精確的參數，並且將資料上傳到網站後得到: accuracy=0.8500, precision=0.6923, F_Score=0.4737 的分數。

隨後嘗試 Nonlinear SVM 的模型，並且在: C=100, gamma=0.01, kernel='rbf' 的超參數條件下，上傳自網站後得到 accuracy=0.8750, precision=0.8049, F_Score=0.5690 的分數。

最後在 Random Forest 的模型中得到最好的分數，發現 Random Forest 並且在 max_depth=9，而 n_estimators 為預設值時有最好的 Final Score。值得注意的是，一開始 max_depth=9，n_estimators 設定為預設值時得到了最好的表現 (accuracy=0.8725, Precision=0.7727, F_Score=0.5714)，隨後將 n_estimators 也納入待測的超參數，超參數的列表分別為 max_depth=[2,3,4,8,9,10] ,n_estimators=[100,150,300,350,400,450,500]，並使用兩個 for 迴圈去一一交叉比對，並且透過圖表看出可能潛在的最好的超參數，但是嘗試多次仍找不到比第一次好的超參數。

隨後又將第一次 max_depth=9 的模型再拿去訓練一次，並且上傳去網站做評估，發現得到更高的分數，也就是最後最高的分數(accuracy=0.8800, precision=0.8140, F_Score=0.5932)，隨後上網看了官方文件才發現，當沒有指定 n_estimators 時，n_estimators 的值會是隨機的，落在 20~100 之間，所以先前設定 n_estimators=[100,150,300,350,400,450,500]明顯超出 20~100，故跑不出更好的指標。且 Random Forest 中各個決策樹的資料也是隨機分配的，所以就算 max_depth 和 n_estimators 都給定了，每次跑出來的結果仍然不一樣。

最後在 Random Forest 中 max_depth=9 時跑出最好的分數，但由於 n_estimators 是隨機指定的(20~100)，所以並不知道最好的 n_estimators 落在哪裡。

預測結果分析:

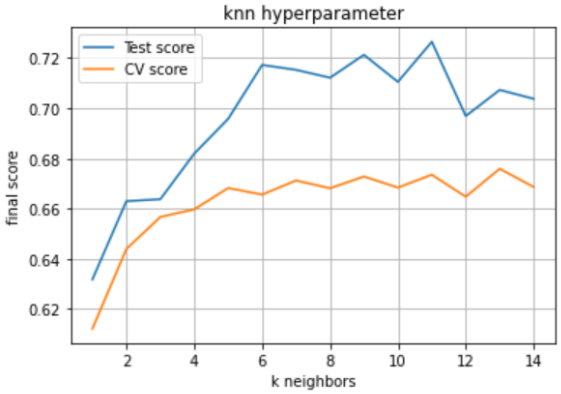
前一段有提到，為了找出每種模型最好的超參數，會用表格搭配圖表判斷。第一個使用的模型，KNN 所畫出的表格及圖表如下圖所示:

	acc	precision	f1	final_score
1	0.796875	0.508711	0.473258	0.631850
2	0.820625	0.686957	0.355056	0.662993
3	0.817500	0.593137	0.453184	0.663778
4	0.825000	0.708333	0.377778	0.681944
5	0.828750	0.657303	0.460630	0.695798
6	0.833125	0.760331	0.407982	0.717263
7	0.834375	0.704403	0.458078	0.715305
8	0.831250	0.767857	0.389140	0.712174
9	0.835625	0.727891	0.448637	0.721299
10	0.830625	0.770642	0.382688	0.710520
11	0.836250	0.753731	0.435345	0.726505
12	0.827500	0.754717	0.366972	0.696926
13	0.831250	0.734375	0.410480	0.707317
14	0.828750	0.769231	0.368664	0.703783

圖二、KNN 各個 K 值所得到的分數 (Index 為 K 值)，此表格為自己切的 Train-Test

	acc	precision	f1	final_score
1	0.789688	0.481429	0.456866	0.612225
2	0.818594	0.661302	0.334599	0.643939
3	0.817032	0.579979	0.448980	0.656693
4	0.822188	0.671528	0.361170	0.659736
5	0.822501	0.617201	0.436482	0.668223
6	0.823438	0.687054	0.359391	0.665609
7	0.824376	0.643157	0.416662	0.671240
8	0.823595	0.706457	0.346205	0.668143
9	0.825001	0.666260	0.397017	0.672811
10	0.823595	0.710078	0.343257	0.668412
11	0.825157	0.682064	0.382868	0.673520
12	0.822345	0.721476	0.323670	0.664762
13	0.825782	0.696463	0.373977	0.675911
14	0.823125	0.733022	0.321265	0.668653

圖三、KNN 各個 K 值所得到的分數 (Index 為 K 值)，此表格為 CV score



圖四、KNN 的圖表，由圖可看出 K=11 時表現最好，實際上傳至網站表現也為最好的

將 K=11 上傳到網站後得到: accuracy=0.8500, precision=0.6923, F_Score=0.4737 的分數。

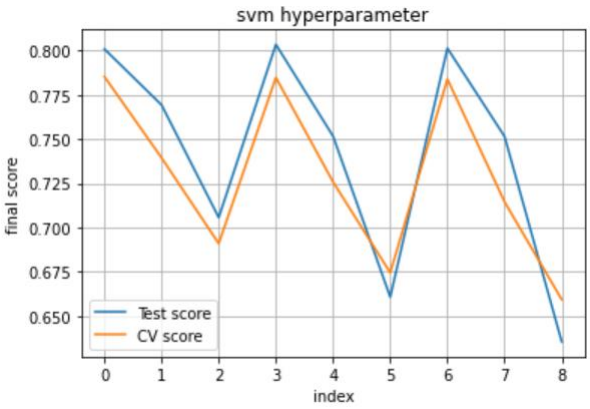
使用 Nonlinear SVM 得到的結果如下圖所示:

	acc	precision	f1	final_score	para
0	0.855625	0.851064	0.509554	0.800935	[50, 0.01, rbf]
1	0.851875	0.729064	0.555347	0.769327	[50, 0.05, rbf]
2	0.830000	0.623932	0.517730	0.705665	[50, 0.1, rbf]
3	0.856875	0.848276	0.517895	0.803531	[100, 0.01, rbf]
4	0.846250	0.700000	0.544444	0.751653	[100, 0.05, rbf]
5	0.811250	0.553435	0.489865	0.660695	[100, 0.1, rbf]
6	0.856875	0.838926	0.521921	0.801401	[150, 0.01, rbf]
7	0.846250	0.696262	0.547794	0.751497	[150, 0.05, rbf]
8	0.799375	0.516129	0.472906	0.635427	[150, 0.1, rbf]

圖五、Nonlinear SVM 的表格，此表格為自己切的 Train-Test，其中 para 為各個超參數的值(C, gamma, kernel)

	acc	precision	f1	final_score	para
0	0.854688	0.811290	0.511056	0.785345	[50, 0.01, rbf]
1	0.844063	0.678069	0.537419	0.739414	[50, 0.05, rbf]
2	0.824688	0.585273	0.523595	0.690953	[50, 0.1, rbf]
3	0.855001	0.803551	0.517230	0.784813	[100, 0.01, rbf]
4	0.839376	0.650153	0.534763	0.725779	[100, 0.05, rbf]
5	0.815939	0.553240	0.520942	0.674455	[100, 0.1, rbf]
6	0.855157	0.796100	0.522390	0.783943	[150, 0.01, rbf]
7	0.834844	0.627339	0.532809	0.714512	[150, 0.05, rbf]
8	0.808127	0.530006	0.512058	0.659264	[150, 0.1, rbf]

圖六、Nonlinear SVM 的表格，此表格為 CV score，其中 para 為各個超參數的值(C, gamma, kernel)



圖七、Nonlinear SVM 的圖表，可以看出在 index=3 及 index=6 時有最好的表現，由圖五、圖六兩表格可看出，index=3 時超參數為 [100,0.01,rbf]，而 index=6 時為 [150,0.01,rbf]

將上述找到的兩組超參數上傳至網站後，分別得到[100,0.01,rbf]: accuracy=0.8750, precision=0.8049, F_Score=0.5690，以及[150,0.01,rbf]: accuracy= 0.8650, precision= 0.7234, F_Score= 0.5574。

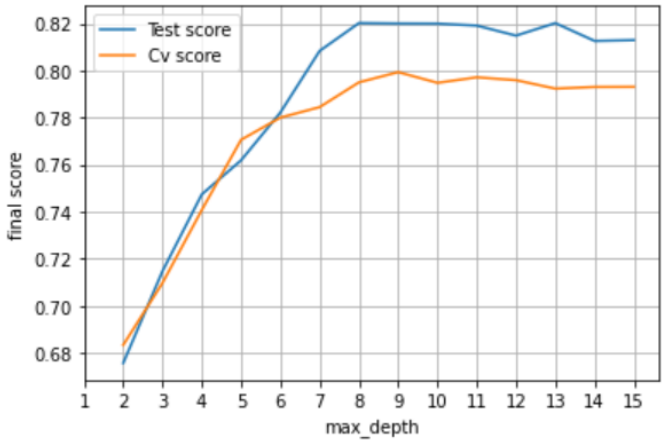
最後我們在 Random Forest 這個模型中找出最好的演算法，一開始只更改 max_depth 這個超參數，n_estimators 設定為預設，並且嘗試了 max_depth=9,10,11…等等的數值，**上傳至網站後最後在 max_depth=9 時跑出最好的分數(accuracy=0.8800, precision=0.8140, F_Score: 0.5932)。**

	acc	precision	f1	final_score
1	0.793750	0.000000	NaN	NaN
2	0.803125	1.000000	0.086957	0.675720
3	0.816250	0.973684	0.201087	0.714783
4	0.831250	0.916667	0.328358	0.747385
5	0.840000	0.870000	0.404651	0.761860
6	0.847500	0.864407	0.455357	0.782156
7	0.858125	0.855172	0.522105	0.808349
8	0.863750	0.845679	0.556911	0.820161
9	0.864375	0.838323	0.563380	0.819994
10	0.865000	0.831395	0.569721	0.819947
11	0.865000	0.827586	0.571429	0.819106
12	0.863750	0.821839	0.567460	0.814845
13	0.866250	0.818681	0.582031	0.820160
14	0.863750	0.811111	0.572549	0.812589
15	0.864375	0.805405	0.578641	0.812931

圖八、Random Forest 的表格，此表格為自己切的 Train-Test，其中 index 為 max_depth。

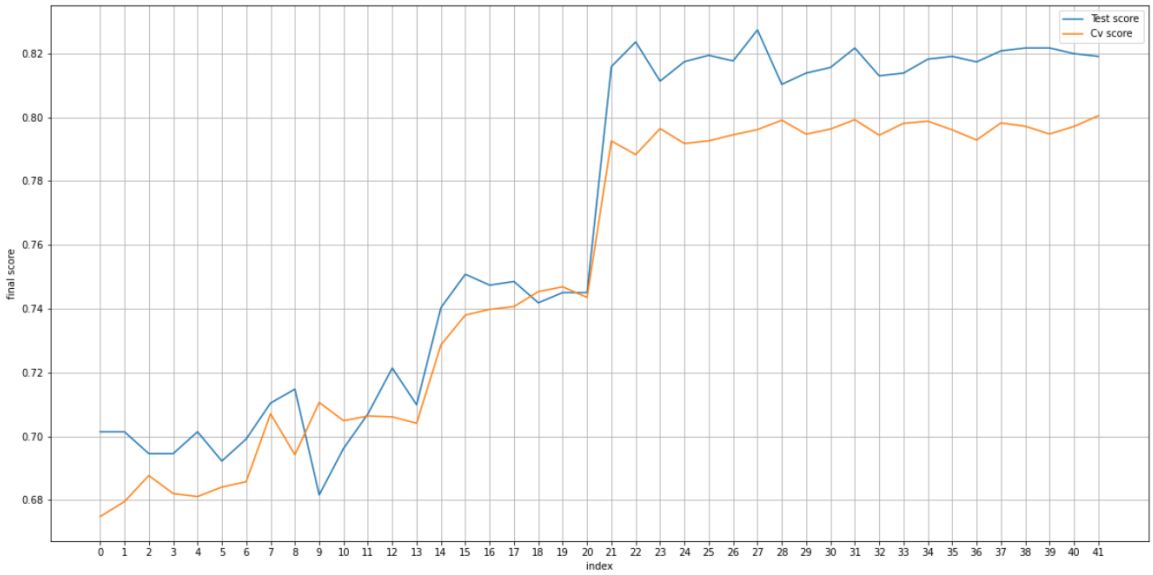
	acc	precision	f1	final_score
1	0.796562	0.000000	NaN	NaN
2	0.807657	1.000000	0.102729	0.683389
3	0.817969	0.964191	0.196726	0.709758
4	0.832187	0.906922	0.320809	0.740748
5	0.845156	0.867320	0.425264	0.770580
6	0.850781	0.837742	0.474181	0.780004
7	0.854376	0.811370	0.508991	0.784457
8	0.858438	0.809227	0.534393	0.794980
9	0.860158	0.809133	0.544084	0.799334
10	0.859533	0.794400	0.547920	0.794788
11	0.860470	0.794672	0.552771	0.797118
12	0.860470	0.788177	0.556268	0.795919
13	0.859688	0.778886	0.557157	0.792324
14	0.860157	0.776510	0.560949	0.793031
15	0.860314	0.774704	0.562788	0.793091

圖九、Random Forest 的表格，此表格為 CV score，其中 index 為 max_depth。



圖十、Random Forest 的表格，其中可看到 max_depth>8 後的 Final Score 都差不多

隨後將 n_estimators 也納入更改的超參數，並將範圍設在 max_depth=[2,3,4,8,9,10], n_estimators=[100,150,300,350,400,450,500] 繪製出來的圖表為:



圖十一、Random Forest 的圖表，可以看出在 index=21~41 時有較好的表

21	0.857501	0.810397	0.527907	0.792572	[8, 100]
22	0.856407	0.802504	0.525938	0.788299	[8, 150]
23	0.858595	0.814984	0.532203	0.796453	[8, 300]
24	0.857345	0.807721	0.528745	0.791790	[8, 350]
25	0.857501	0.810001	0.528414	0.792616	[8, 400]
26	0.858126	0.811648	0.531011	0.794502	[8, 450]
27	0.858439	0.815393	0.531117	0.796136	[8, 500]
28	0.859845	0.811076	0.541753	0.799085	[9, 100]
29	0.859064	0.800395	0.542101	0.794718	[9, 150]
30	0.859064	0.808075	0.538402	0.796310	[9, 300]
31	0.859845	0.812052	0.541189	0.799250	[9, 350]
32	0.858595	0.804379	0.537629	0.794382	[9, 400]
33	0.859845	0.806954	0.543439	0.798111	[9, 450]
34	0.860001	0.809034	0.542883	0.798767	[9, 500]
35	0.860001	0.794907	0.550248	0.796062	[10, 100]
36	0.858751	0.795620	0.542488	0.792868	[10, 150]
37	0.860470	0.800663	0.549576	0.798237	[10, 300]
38	0.860001	0.800577	0.547386	0.797186	[10, 350]
39	0.859064	0.800437	0.542191	0.794770	[10, 400]
40	0.860157	0.798567	0.549125	0.797124	[10, 450]
41	0.860782	0.807249	0.548351	0.800475	[10, 500]

圖十二、Random Forest 的表格，此表格為 CV score，其中可看到 index=21~41 時都為 max_depth>8 的值，只是 n_estimators 不一樣而已

將上述 index=21~41 的結果一一上傳至網站後，發現跑出來的分數仍沒有比把 n_estimators 設定為預設值高(max_depth=8,n_estimators=500 時跑出 accuracy=0.8750, precision=0.7907, FScore=0.5763) (max_depth=10,n_estimators=350 時跑出 accuracy=0.8700, precision=0.7556, FScore=0.5667 等等)。有此可知，若使用 Random Forest 的話，n_estimators 的範圍最好使用預設值，也就是 n_estimators=20~100 之間的隨機之最佳。

由於 n_estimators 使用預設值時為隨機選取 20~100 之間的數，所以並不知道確切的 n_estimators 應該用多少最好。而雖然 max_depth=9 時跑出最好的分數，但是由於 Random Forest 的各個決策樹的資料也是隨機分配的，故 max_depth=9 時仍然會每次跑出不同的分數。但是透過上面的圖表以及不斷的上傳至網站測試後，推測 Random Forest 在預測此資料時應使用 max_depth=8~12 而 n_estimators 應使用預設值，跑出來的效果為最佳。

感想與心得:

方宣翔: 此次競賽學到了很多關於 Scikit-Learn 的套件如何使用，以及各個超參數的意義等。此次也自學到了上課沒教到的 SVM 模型，並且跑出了不錯的成績。我認為此次作業花最多時間的地方在如何找到最好的超參數，有時花了很長時間找，結果卻往往不如預期，但是有時看到跑出來的分數超出先前的最高分數時還是很有成就感，此次作業讓我受益良多。此次作業也有嘗試使用深度學習來訓練，但是由於對於深度學習本身不太了解，所以訓練出來的結果比起 Scikit-learn 都還要差，最後還是放棄了，若未來有機會我或許仍會自主研究深度學習的相關知識。

陳奕佑: 這次資料科學導論的競賽讓我學到很多。作為 python 初學者，這堂課前面的幾次 HW 練習對於這次的競賽相當有幫助。競賽中，除了透過實際操作來更加了解機器學習許多細節，也讓我認識許多不同的 model 及他們運作的方式。分析資料的過程也訓練我思考的邏輯和分類事務的規則，藉由簡單的數學定義，就能讓電腦自己得到 feedback，不斷進步的過程也相當有成就感。也謝謝我凱瑞的組員們，在遇到問題時也都很熱心地給予我幫助，細心地回答我的問題，非常謝謝他們。

楊明學: 這個競賽作業讓我學到很多東西，從最開始的資料分析與處理，再來選擇適合的函式庫並建立各種不同模型來訓練，最後還需要各種嘗試來找到最佳的超參數。最大的收穫就是大概學會利用 keras 建立類神經模型，對於未來通訊領域最佳化的幫助非常大。花最多的時間應該就是在找到最好的模型，因為 python 與過去習慣的 C 不同，不太知道程式每一步背後的運作原理，再加上

對於數據分析的不熟悉導致看不懂模型超參數的選項。 這也是第一次參加這種競賽型的作業，利用競賽的方式讓我們更加認真去學習提升自己程式的品質。
也謝謝教授與助教這學期的教導。