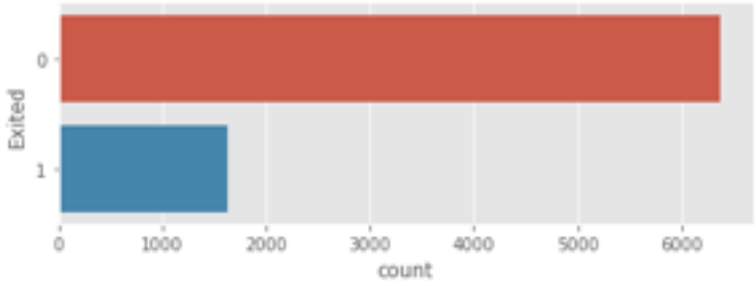
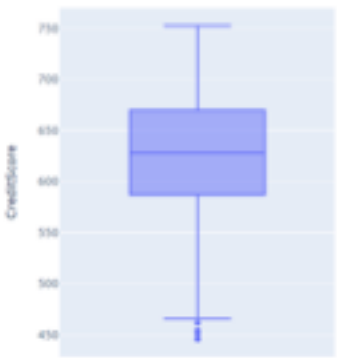
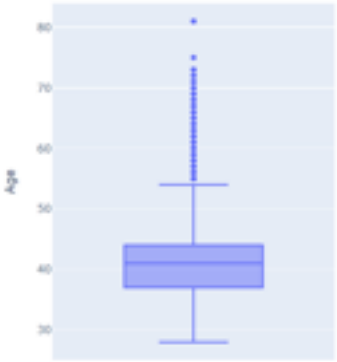
[Github 連結]

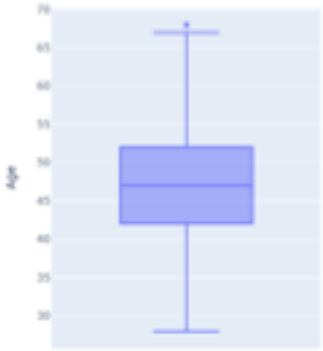
1. 姓名：林昕霏 系級：工資 112 學號：H34084040
   姓名：徐圓媛 系級：工資 112 學號：H34086115
2. 競賽敘述與目標：
   預測銀恆客戶流失。用已知的銀行客戶行為、金融資料來預測顧客是否最終（未來）會不再使用銀行的服務。
3. 資料前處理：
   將類別變數 HasCrCard, IsActiveMember 0 改成 -1 以增加係數相關性；名目變數 Geography, Gender 以 onehot encoding 的方式轉換；去掉 RowNumber, CustomerId, Surname 變數；連續變數則以 minMax 的方式去標準化。上述的轉換都是為了讓模型能夠更好的理解數據。
4. 特徵處理與分析：
   在開始處理特徵前，我們針對資料進行一些統計分析，給予初步建議
   I. 統計 Exited 比例發現為 0.796 : 0.204

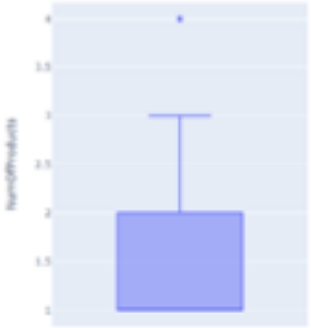


   II. 觀察各特徵的離群值：
      A. CreditScore: Exited = 1 outlier < 466 → fig(A)
      B. Age: Exited = 0 outlier > 54 →fig(B-1) ; Exited = 1 ouutlier > 67 → fig(B-2)
      C. NumOfProducts: Exited = 1 outlier > 3 → fig(C)


fig(A)


fig(B-1)


fig(B-2)


fig(C)

   III. 名目變數的相關性：

A. geography 整題而言位於法國的使用者佔將近半數，由比例可知，使用者最多的國家其退出比率最低；其餘兩國皆佔各四分之一使用者比率，其中德國退出比例有 3 成，可能是德國的服務較其他地區較為不好，是未來需要改進的方向。→ fig(III-A)

B. Gender Female 的退出率比 Male 的高。→ fig(III-B)

C. IsActiveMember 不活躍用戶相較於活躍用戶，更容易退出服務不活躍用戶的比例高達將近半數，銀行需要考慮未來如何將這個族群轉為活躍用戶。→ fig(III-C)

	Geography	RowNumber	Exited
0	France	4006	0.161008
1	Germany	2018	0.321606
2	Spain	1976	0.171053

fig(III-A)

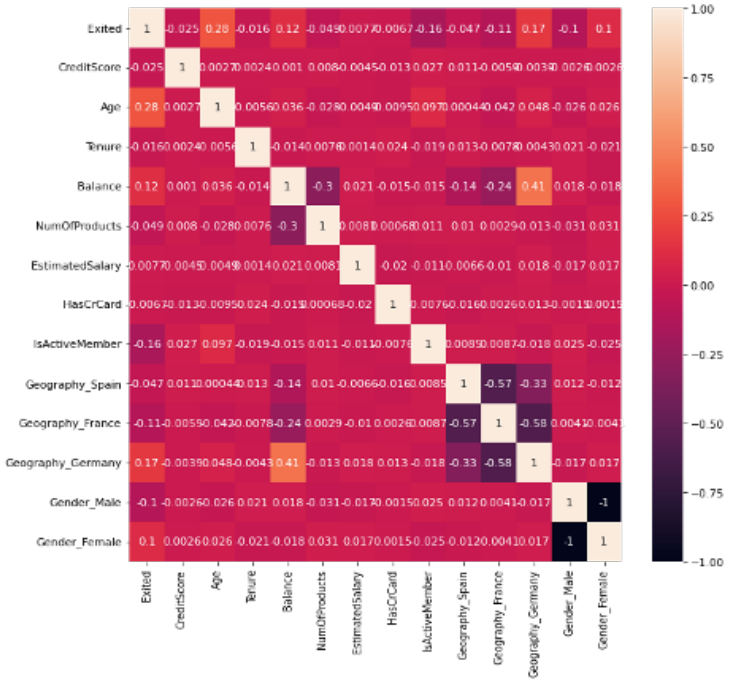
	Gender	RowNumber	Exited
1	Male	4371	0.165637
0	Female	3629	0.250207

fig(III-B)

	IsActiveMember	RowNumber	Exited
1	1	4135	0.142201
0	0	3865	0.270116

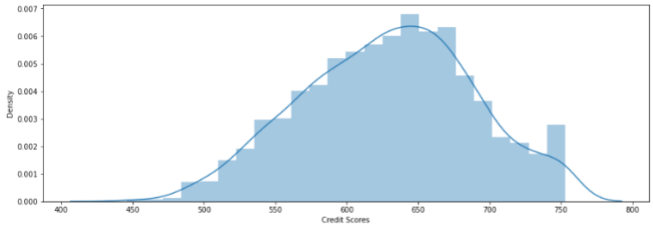
fig(III-C)

IV. 特徵相關性比較：

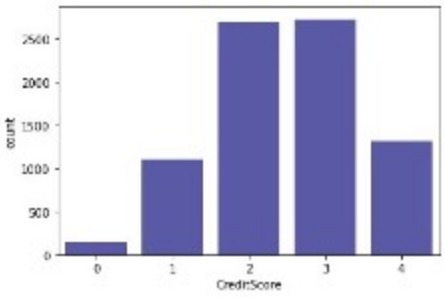


V. 連續變數與轉換成離散型態後的分佈：

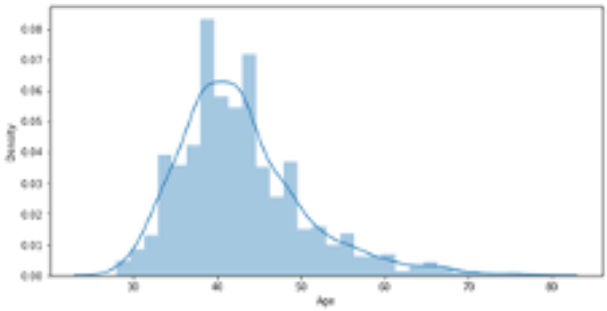
CreditScore:



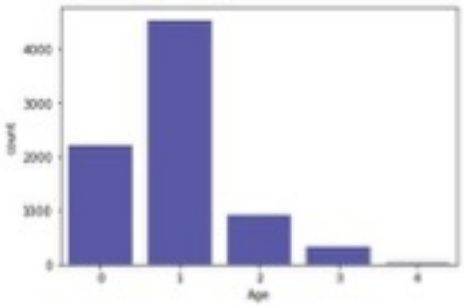
→



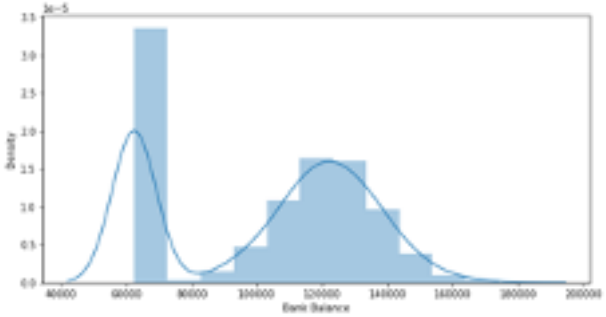
Age:



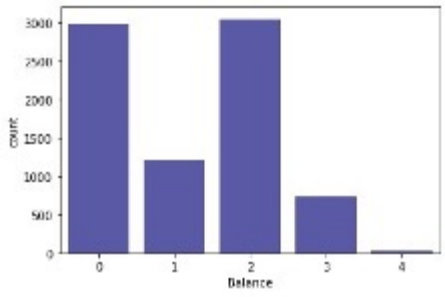
→



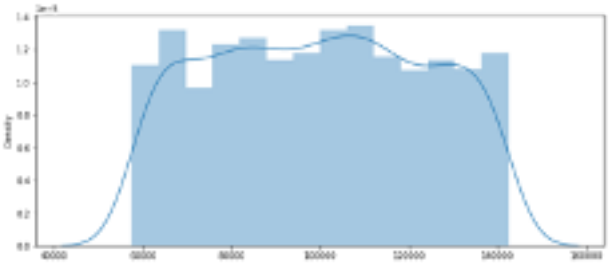
Bank Balance:



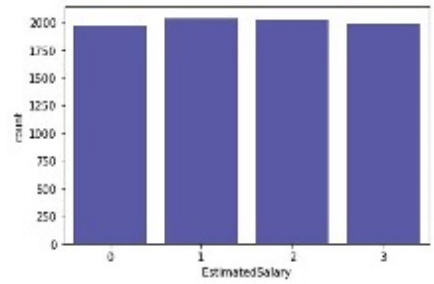
→



Estimated Salary:



→



我們總共嘗試了六種不同的資料處理方式得到的數據，分別為下列的 a~f：

a、 Original data (13) [hw4_library.py]

b、 New feature (13+3) [newfeature_3.ipynb]

在進行資料分析的過程中我們發現有些參數的相關系數較低，因此希望可以藉由各特徵的比例創造新的特徵再從中挑選相關性高的特徵。所有特徵中 Tenure、HasCrCard、EstimatedSalary 的相關性分別為，-0.016、0.0067、0.0077 為當中較低的，因此選擇先以他們去做特徵擴增。由於 Balance 和 Age 為其餘的特徵中相關性交高者，所

以我們去對照上述三特徵與他們的關聯得出應增加 TenureByAge、BalanceSalaryRatio、CreditScoreGivenAge。

c、 More feature (13+8) [newfeature_8.ipynb]
由於增加了三個特徵後模型訓練的結果有些許的提升，因此我們嘗試了增加其他五個特徵: BalanceAge、EstimatedSalaryAge、CreditScoreSalaryRatio、TenureSalaryRatio、NumofProductsSalaryRatio。

d、 Continuous → discrete (13) [preprocess_contodis.ipynb]
將所有連續變數 CreditScore、Age、Balance、EstimatedSalary 轉換成離散型的數字處理，這個方法可能會丟失一些重要的資訊，但同時可以更好的套入線性的模型處理。

e、 Original outlier
經過資料分析後發現，部分資料有一些離群值，因此我們也嘗試去除他們。

f、 Original corr
去掉相關係數低於 0.01 的特徵

5. 預測訓練模型：[model.ipynb]
我們嘗試了共七種模型，為了節省嘗試的時間使用了 GridSearch 網格搜尋，來尋找最佳參數。我們只呈現 More feature 的結果。

a、 Logistic Regression：
- 平均得分：0.821375
- 最佳參數：LogisticRegression(C=50, max_iter=250, tol=1e-05)

b、 SVM RBF：
- 平均得分：0.8483744951986028
- 最佳參數：SVC(C=150, gamma=0.01, probability=True)

c、 SVM poly：
- 平均得分：0.8549995110326138
- 最佳參數：SVC(C=100, degree=2, gamma=0.1, kernel='poly', probability=True)

d、 XGBoost（只截取重要參數）：
- 平均得分：0.85725
- 最佳參數：XGBClassifier(gamma=0.01, learning_rate=0.05, n_estimators=100, max_depth=7, min_child_weight=10)

e、 Random forest：
- 平均得分：0.8526250000000001
- 最佳參數：- 最佳參數：RandomForestClassifier(max_depth=8, max_features=9,

min_samples_split=6, n_estimators=50)

f、 Light GBM：

- 平均得分：0.849753328574923
- 最佳參數：LGBMClassifier(learning_rate=0.05, max_depth=10, n_estimators=50, num_leaves=20)

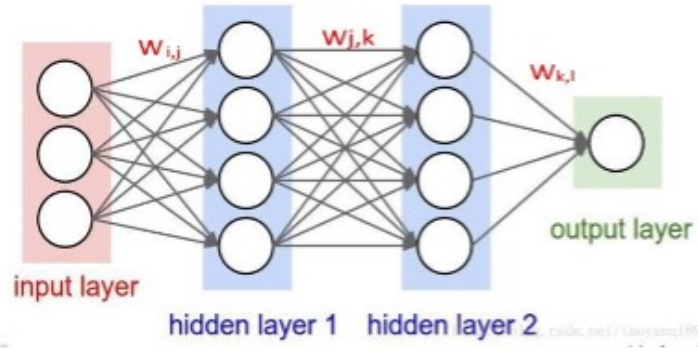
全部的測試當中，由於神經網路的結果相對前述的六種方法較好，因此我們後面主要都在做相關的參數設定、測試。

g、 NN [NN_model.ipynb]

Neural Network 是參考生物神經系統的結構，神經元(Neuron)之間互相連結，由外部神經元接收信號，再層層傳導至其他神經元，最後作出反應的過程。

Input Layer 為接受信號的神經元、Hidden Layer 為隱藏層、Output Layer 是做出反應的輸出層，而各神經元傳導的力量大小為權重。

若僅單純做個層的節點加權總和，就是簡單回歸；若要解決非線性的問題則會在整個公式之前加上一個非線性函數 Activation Function。當我們希望學習的效果更顯著，可以使用多層的 Hidden Layer 模擬生物神經系統，也就是常見的 Deep Learning。(大致的架構概念如下)



過程中主要決定的就是 hidden layer 有幾層、分別又有多少神經元存在。因為並沒有特定的標準答案應該要如何設定，因此我們只能一一測試。

6. 預測結果分析：

經過個 model 對處理過後的六種資料實驗後我們得到以下的結論

- 機器學習模型結果

a、 每組模型適用的資料不同，沒有最佳的資料（螢光筆標示為最佳結果）

logistic					
data	accuracy	precision	recall	fscore	final
original	0.815	0.542857143	0.13571	0.339285714	0.54307
original_c	0.8275	0.681818182	-0.0758	0.303030303	0.57401
nf_3	0.8175	0.833333333	-0.5924	0.120481928	0.54344
nf_3_corr	0.8275	0.681818182	-0.0758	0.303030303	0.57401
nf_8	0.82	0.857142857	-0.5714	0.142857143	0.56029
nf_8_corr	0.8225	1	-0.7108	0.144578313	0.60458
con_dis	0.8125	0.541666667	-0.0268	0.257425743	0.50922
con_dis_c	0.8125	0.541666667	-0.0268	0.257425743	0.50922

svm rbf					
data	accuracy	precision	recall	fscore	final
original	0.8625	0.729166667	0.39083	0.56	0.7015
original_c	0.87	0.745098039	0.4424	0.59375	0.72203
nf_3	0.86	0.698113208	0.44035	0.56923	0.69513
nf_3_corr	0.8675	0.722222222	0.46862	0.59542	0.71508
nf_8	0.86	0.705882353	0.41912	0.5625	0.69476
nf_8_corr	0.8725	0.76	0.43685	0.59843	0.72912
con_dis	0.86	0.818181818	0.16364	0.49091	0.69982
con_dis_c	0.8625	0.823529412	0.18548	0.5045	0.70761

lightgbm					
data	accuracy	precision	recall	fscore	final
original	0.8725	0.740740741	0.48063	0.610687023	0.72825
original_c	0.8675	0.714285714	0.48872	0.601503759	0.71514
nf_3	0.8675	0.714285714	0.48872	0.601503759	0.71514
nf_3_corr	0.87	0.727272727	0.48485	0.606060606	0.72161
nf_8	0.8675	0.722222222	0.46528	0.59375	0.71442
nf_8_corr	0.8725	0.740740741	0.48063	0.610687023	0.72825
con_dis	0.8625	0.775	0.28483	0.52991453	0.70322
con_dis_c	0.86	0.744186047	0.32248	0.533333333	0.69459

xgb					
data	accuracy	precision	recall	fscore	final
original	0.8775	0.725806452	0.56916	0.647482014	0.73998
original_c	0.87	0.698412698	0.55873	0.628571429	0.72195
nf_3	0.88	0.745762712	0.54835	0.647058824	0.74655
nf_3_corr	0.87	0.711864407	0.52343	0.617647059	0.72162
nf_8	0.8675	0.714285714	0.48872	0.601503759	0.71514
nf_8_corr	0.8775	0.75	0.51316	0.631578947	0.74088
con_dis	0.8625	0.703703704	0.4566	0.580152672	0.70192
con_dis_c	0.855	0.70212766	0.36239	0.532258065	0.68004

rf					
data	accuracy	precision	recall	fscore	final
original	0.8675	0.714285714	0.48872	0.6015	0.71514
original_c	0.8725	0.740740741	0.48063	0.61069	0.72825
nf_3	0.8675	0.760869565	0.37734	0.56911	0.71615
nf_3_corr	0.875	0.745454545	0.49697	0.62121	0.73462
nf_8	0.87	0.745098039	0.4424	0.59375	0.72203
nf_8_corr	0.875	0.775510204	0.43084	0.60317	0.73642
con_dis	0.87	0.765957447	0.39533	0.58065	0.72305
con_dis_c	0.86	0.705882353	0.35863	0.53226	0.68267

b、 我們分別針對原始資料、去除相關係屬小於 0.01 的特徵之資料討論（HasCrCard, EstimatedSalary 為去除的兩個特徵）。從各項預測結果來看，XGB 與 RandomForest 為除了深度學習外表現最好的模型。（螢光筆標示為最佳結果）

original					
model	accuracy	precision	recall	fscore	final
logisticRe	0.815	0.54286	0.135714	0.33929	0.543071
SVM rbf	0.8625	0.72917	0.390833	0.56	0.7015
XGB	0.8775	0.72581	0.569158	0.64748	0.739985
RandomF	0.8675	0.71429	0.488722	0.6015	0.715137
LightGBM	0.8725	0.74074	0.480633	0.61069	0.728247
HasCrCard, EstimatedSalary					
model	accuracy	precision	recall	fscore	final
logisticRe	0.8275	0.68182	-0.07576	0.30303	0.574008
SVM rbf	0.87	0.7451	0.442402	0.59375	0.722029
XGB	0.87	0.69841	0.55873	0.62857	0.721952
RandomF	0.8725	0.74074	0.480633	0.61069	0.728247
LightGBM	0.8675	0.71429	0.488722	0.6015	0.715137

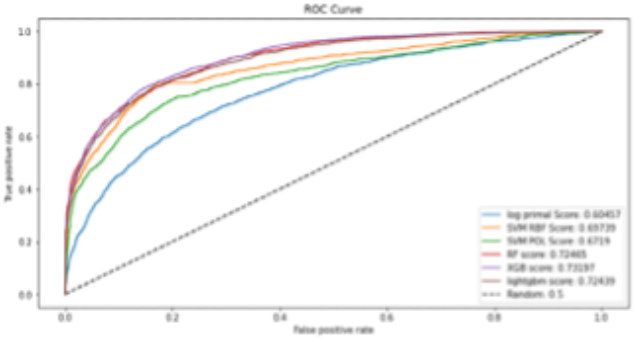
nf_3					
model	accuracy	precision	recall	fscore	final
logisticRe	0.8175	0.83333	-0.59237	0.12048	0.543443
SVM rbf	0.86	0.69811	0.440348	0.56923	0.695126
XGB	0.88	0.74576	0.548355	0.64706	0.746552
RandomF	0.8675	0.76087	0.377342	0.56911	0.716153
LightGBM	0.8675	0.71429	0.488722	0.6015	0.715137
HasCrCard, EstimatedSalary					
model	accuracy	precision	recall	fscore	final
logisticRe	0.8275	0.68182	-0.07576	0.30303	0.574008
SVM rbf	0.8675	0.72222	0.468617	0.59542	0.715085
XGB	0.87	0.71186	0.52343	0.61765	0.721618
RandomF	0.875	0.74545	0.49697	0.62121	0.734621
LightGBM	0.87	0.72727	0.484848	0.60606	0.721606

nf_8					
model	accuracy	precision	recall	fscore	final
logisticRe	0.82	0.85714	-0.57143	0.14286	0.560286
SVM rbf	0.86	0.70588	0.419118	0.5625	0.694765
XGB	0.8675	0.71429	0.488722	0.6015	0.715137
RandomF	0.87	0.7451	0.442402	0.59375	0.722029
LightGBM	0.8675	0.72222	0.465278	0.59375	0.714417
HasCrCard, EstimatedSalary					
model	accuracy	precision	recall	fscore	final
logisticRe	0.8225	1	-0.71084	0.14458	0.604581
SVM rbf	0.8725	0.76	0.43685	0.59843	0.72912
XGB	0.8775	0.75	0.513158	0.63158	0.740882
RandomF	0.875	0.77551	0.430839	0.60317	0.736423
LightGBM	0.8725	0.74074	0.480633	0.61069	0.728247

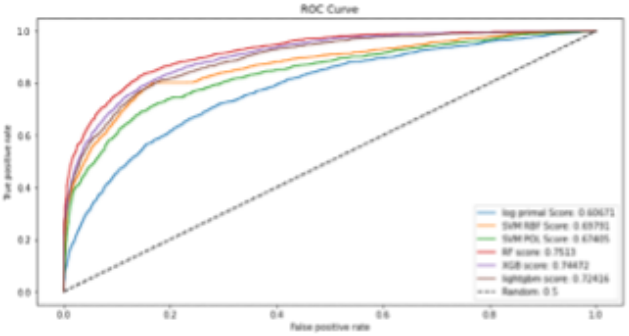
con_dist					
model	accuracy	precision	recall	fscore	final
logisticRe	0.8125	0.541666667	-0.0268	0.257425743	0.50922
SVM rbf	0.86	0.818181818	0.16364	0.490909091	0.69982
XGB	0.8625	0.703703704	0.4566	0.580152672	0.70192
RandomF	0.87	0.765957447	0.39533	0.580645161	0.72305
LightGBM	0.8625	0.775	0.28483	0.52991453	0.70322
HasCrCard, EstimatedSalary					
model	accuracy	precision	recall	fscore	final
logisticRe	0.8125	0.541666667	-0.0268	0.257425743	0.50922
SVM rbf	0.8625	0.823529412	0.18548	0.504504505	0.70761
XGB	0.855	0.70212766	0.36239	0.532258065	0.68004
RandomF	0.86	0.705882353	0.35863	0.532258065	0.68267
LightGBM	0.86	0.744186047	0.32248	0.533333333	0.69459

c、 ROC curve

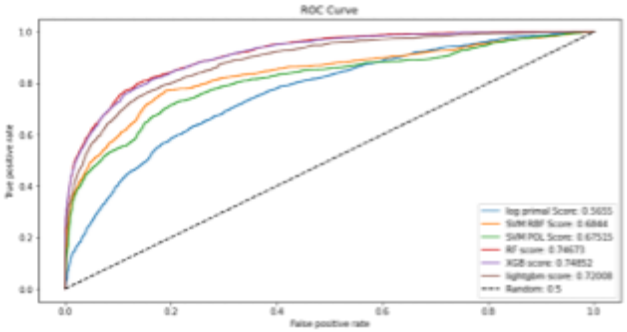
在所有模型結果之後，我們將 ROC curve 視覺化發現 RandomForest, XGB 的 AUC 分數在所有狀況下皆為最高的兩個，歸納此二者確實為表現最佳的兩種模型，以下為分別針對去除兩個小於 0.01 相關係數的特徵與原資料分析後的 ROC curve



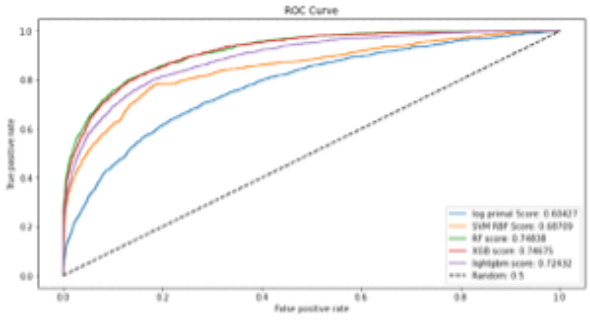
a. Original data



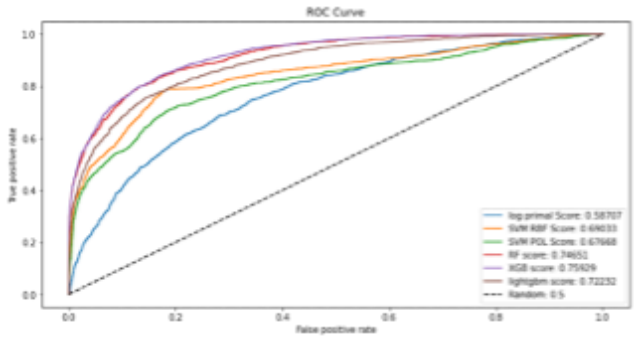
a. Original data without 2 features



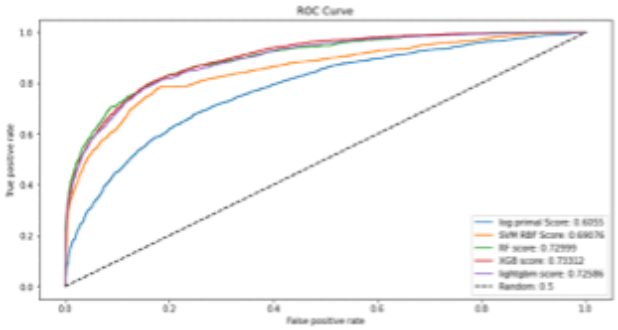
b. New feature



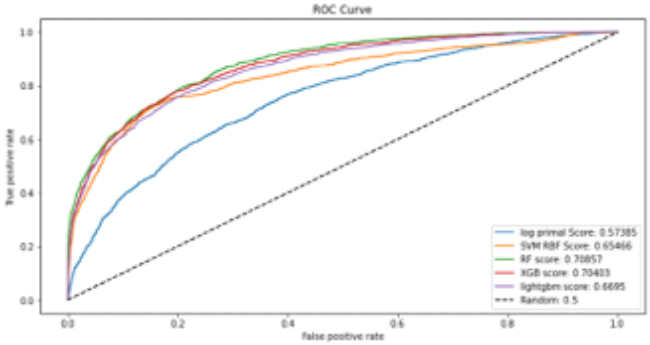
b. New feature without 2 features



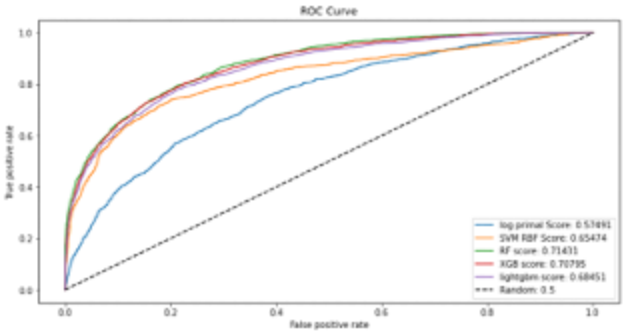
c. More feature



c. More feature without 2 features



d. Continuous → discrete



d. Continuous → discrete without 2 features

- 類神經網路結果：

  上一部分我們提到了類神經網路的基本設定沒有標準答案，因此我們針對神經元數量、hidden layer 層數、threshold，分別調整測試。

  a、 神經元數量

  我們發現隱藏層的神經元數目應該要維持同樣的數量對於整體預測結果會比較好。單純對於 a. original 處理的結果來說兩層、六個神經元預測結果會是最好的

hidden la	neuron	batch	data	threshold	accuracy	precision	fscore	final
5	16_32_64_32_16	32	16	0.5	0.7825	0.447916667	0.497109827	0.56797
3	16_32_16	32	16	0.5	0.845	0.62295082	0.550724638	0.66068
3	16_32_64	32	16	0.5	0.8525	0.68	0.535433071	0.67392
2	5_128	32	16	0.5	0.8675	0.8	0.547008547	0.71905
2	16_32	32	16	0.5	0.8725	0.740740741	0.610687023	0.72825
2	6_6	32	16	0.5	0.8725	0.770833333	0.592	0.7298

  b、 Batch 大小討論

  通常會以 2 的次方設定，然而在整理資料、爬找原因的時候我們發現其實沒有用 2 次方設定也不一定訓練的效率會較差。在我們的實驗中，我們只有針對 32、64、128 去測試，發現 64 對我們的資料會得到最好的訓練結果，因此後續我們都用 64 來做。

hidden la	neuron	batch	data	threshold	accuracy	precision	fscore	final
2	6_6	128	16	0.5	0.8675	0.74	0.582677165	0.71532
2	6_6	32	16	0.5	0.8725	0.770833333	0.592	0.7298
2	6_6	64	16	0.5	0.875	0.846153846	0.568965517	0.74393

c、 Hidden Layer 層數討論

hidden lay	neuron	batch	data	threshold	accuracy	precision	fscore	final
20	6	64	21	0.55	0.86	0.7442	0.5333	0.694589
16	6	64	21	0.55	0.87	0.8205	0.5517	0.727844
8	6	64	21	0.55	0.855	0.7021	0.5323	0.680042
4	6	64	21	0.55	0.865	0.7674	0.55	0.709733

在多次實驗後我們發現在設定為 16 層的時候結果是最好的

d、 Threshold 設定

一般設定的 threshold 數值為 0.5 當預測為真的機率大於時=1，不然就會自動歸為 0。我們是透過增加、減少 threshold 的值去比對他的 score 來決定要用哪一組結果。

hidden lay	neuron	batch	data	threshold	accuracy	precision	fscore	final
4	6	64	21	0.48	0.8675	0.7143	0.6015	0.715137
4	6	64	21	0.5	0.865	0.7091	0.5909	0.708591
4	6	64	21	0.55	0.8775	0.7917	0.608	0.74395
4	6	64	21	0.6	0.8675	0.7857	0.5546	0.717813
4	6	64	21	0.7	0.8675	0.875	0.5138	0.728255
4	6	64	21	0.75	0.865	0.8966	0.4906	0.724692
4	6	64	21	0.8	0.86	1	0.4286	0.729429

然而在整理資料、找尋原因後我們發現這個方法可能會使結果 overfitting，因此若還有機會我們之後會以 CrossValidation 的方式去找 threshold 值。

在所有結果中，我們預測最好的是用 c. More Feature 16 層 NN threshold = 0.8 得到的結果。

- 其他

a、 針對資料 e. original_outlier 的討論，一開始我們認為去除 outlier 能使預測結果更為精準。然而，由結果可以得知，每個模型去除離群值後總體的分數皆下降，因此最後並未採用此手法處理資料。

	original					without outlier				
	accuracy	precision	recall	f1-score	final	accuracy	precision	recall	f1-score	final
Logistic	0.8325	0.692	0.23334	0.349	0.59695	0.815	0.555	0.19445	0.288	0.5262
SVM rbf	0.865	0.755555556	0.44156	0.557377049	0.70912	0.83	0.695652174	0.20779	0.32	0.5857
SVM poly	0.8675	0.75	0.46753	0.576	0.71565	0.825	0.666666667	0.18182	0.28571	0.56179
RF	0.86	0.684	0.50647	0.582	0.696	0.8225	0.625	0.19478	0.297	0.55305
XGB	0.86	0.666666667	0.54545	0.6	0.698	0.825	0.666	0.1921	0.286	0.5617

b、 PCA 降維處理，考量到特徵過多可能會造成模型訓練的困難度增加，我們嘗試用表現最好的 XGBoost 做降維資料訓練。效果較先前所有的結果都更差，此方法並不能增加模型的表現

XGB-original data	accuracy	precision	fscore	final
1	0.735	0.3505	0.3908	0.481976
2	0.75	0.2326	0.1667	0.361434
3	0.77	0.1739	0.08	0.315174
4	0.7075	0.2674	0.2822	0.405366
5	0.7725	0.3409	0.2479	0.433196
6	0.715	0.2716	0.2785	0.407374
7	0.7475	0.3065	0.2734	0.425538
8	0.755	0.3091	0.2576	0.422258
9	0.7575	0.1875	0.1101	0.327537

7. 感想與心得：

ａ、 徐圓媛

這次的競賽嘗試了許多種不同的機器學習模型、類神經網路，相較於以往做過的專案更知道自己應該要如何進行每一步的動作，再次印證資料科學領域的知識真的是從做中學的事實。這次花比較多時間的是在深度學習模型的了解，由於過去沒有相關的經驗，所以會花比較多時間在了解數學運算原理、調整參數以及函數；對於深度學習有一定的認識，之後有機會應該會更深入地去實作其他的題目。花最多時間的應該是在資料的前處理、特徵工程上，相對於模型的使用，前處理並沒有一個一定的方式、特徵工程也沒有固定的 SOP，所以最後才產生了六種不同的處理後資料做後續的訓練、預測。最困難的地方應該還是在於要如何在不過擬何的情況下讓預測結果更好，這一點是我們沒有做好的。至於競賽平台的建議，我建議可以加入刪除檔案的按鈕，像這次我們發現加權成績最好的結果有過擬合的現象，卻沒有辦法拿掉這個檔案，就覺得有點可惜。

ｂ、 林昕霏

本次競賽中學到作多的我認為是模型的部分。之前有學過人工智慧導論，其中都只有大概敘述模型的種類，以及可以做到的功能等等，幾乎沒有實作跟解說的部分。即便如此，也燃起我對這部分的興趣，也有想過之後要再網路上找尋相關課程自修。透過同學的推薦，休息本門課，發現完完全全就是我想要上的課程，因此過程中，雖然會有點吃力，但是很滿足。尤其在這次的作業，有了機會可以更加深入了解背後的意義，並且也學習很多種不同的模型，更加了解這個領域的知識。除了老師的課程外，期間也上網查了許多資料，像是從 kaggle 上可以學到模型的使用，或是一些新的技巧，如果是自己想的話一定想不到。在這次的競賽中，對於特徵的修改或是新增也是我沒有碰觸過的，之前都會認為 data 就是長那樣，應該也不用去做修正。此次競賽也了解到，應該要針對感興趣的部分，去進行不同的修正，達到做好的效果，也為了能讓模型更 general，要去避免 overfitting 的狀況。但因為能力不足的關係，在本次競賽中沒有辦法達到更好的分數，希望未來能夠再透過進修相關知識，補足自己缺失的地方。透過這次競賽，在讓寫作業的同時，看到排名上升獲得的成就感，也是很特殊的體驗。最後，自己在本次競賽中學習到很多相關知識，更學到團隊合作的默契，謝謝隊友的支持。