期末作業

第 15 組： 金企三甲 溫庭妤 409411337
金企三甲 王樂詠 409411600
金企三甲 林嘉柔 409411167

(a)議題陳述 與 議題重要性

因為 P2P 平台為去中介化、提供媒合服務給借貸雙方的配對機制，風險由借貸雙方自行承擔，而透過許多特徵變數來預測出借方是否會違約並致力於降低錯誤放貸率是很重要的事（減少錢借出去但收不回而造成損失的機率）。

也就是說，我們需要預測出在放入不同 X 變數下以及更動超參數的數字及比例後，哪種搭配能使錯誤放貸率降至最低，並擁有良好的績效。也就是以平台貸款契約特徵變數與貸款者的特徵變數來分析哪些變數會顯著的影響放貸率，並嘗試讓錯誤放貸率降低。

(b)資料處理流程 與 分析步驟

我們分別使用了決策樹、隨機森林、XGBoost 來處理資料以及建立模型，並且在最後以 XGBoost 的結果獲得三種模型之中最高的排名。

首先我們將原始資料以羅吉斯回歸跑出相關係數，此步驟目的為挑出相關係數>0.1，即和 loan_status 有較高相關性的變數，分別為 dti 0.12, grade 0.22, funded_amnt 0.14, installment 0.13, open_acc 0.10, all_util 0.12, bc_util 0.13, num_actv_rev_tl 0.14, percent_bc_gt_75 0.13 九項。而後嘗試從此九項變數中分別挑選放入模型中，來測試何種變數能最顯著影響錯誤放貸率。

id	-0.022123	funded_amnt	0.135617	open_rv_12m	0.029929	mths_since_recent_inq	-0.048043
annual_inc	-0.004028	installment	0.129428	max_bal_bc	-0.004310	num_accts_ever_120_pd	-0.002437
dti	0.121064	delinq_2yrs	0.055791	all_util	0.116824	num_actv_rev_tl	0.139472
inq_last_6mths	0.052314	mths_since_last_delinq	-0.075349	inq_fi	0.015337	num_op_rev_tl	0.090677
loan_status	1.000000	open_acc	0.102328	bc_open_to_buy	-0.076652	num_rev_accts	0.042810
home_ownership	-0.001151	pub_rec	0.035150	bc_util	0.126457	num_tl_120dpd_2m	0.026228
purpose	0.040938	revol_bal	0.028899	delinq_amnt	0.066165	num_tl_op_past_12m	0.035449
verification_status	0.008728	acc_now_delinq	0.023536	mo_sin_old_rev_tl_op	-0.003537	percent_bc_gt_75	0.130202
grade	0.219680	tot_coll_amt	0.041515	mo_sin_rcnt_rev_tl_op	-0.033117	pub_rec_bankruptcies	-0.027829
		open_acc_6m	-0.025323	mort_acc	-0.037884	tot_hi_cred_lim	-0.009542
		open_act_il	0.062135				

XGBoost 的流程步驟為: 我們首先將原始資料(train)放入後讀檔，用平均值去補變數中的缺失值，並於建立模型中將 X 變數裡放入 funded_amnt, percent_bc_gt_75, num_actv_rev_tl, inq_last_6mths 四個，y 為

loan_status。並將測試集的比例從 0.25 改為 0.2，n_estimators 從 100 改為 60，learning_ rate 從 0.1 改為 0.2，其他維持不變。最後在 XGBoost 的模型中得到最低 0.1574 的錯誤放貸率。

```
[ ]  1 #  預測變數:  x
     2 #  目標變數:  y
     3 x=  df2[['funded_amnt','percent_bc_gt_75','num_actv_rev_tl','inq_last_6mths']]
     4
     5 #  最小值最大值標準化
     6 from  sklearn.preprocessing  import  MinMaxScaler
     7 scaler  =  MinMaxScaler(feature_range=(0,  1)).fit(x)
     8 x2=  scaler.transform(x)
     9
    10 y=  df2['loan_status']
```

```
1 #  訓練集:  測試集  =  0.8:0.2
2 from  sklearn.model_selection  import  train_test_split
3 x_train,x_test,y_train,y_test=train_test_split(x2,y,test_size=0.2,random_state=10)
```

```
1 #  建立模型
2 #  設置決策樹的結構
3 from  xgboost.sklearn  import  XGBClassifier
4 clf=XGBClassifier(n_estimators=60,max_depth=20,learning_rate=0.2)
```

(c)資料變數定義 與 基本統計量分析

1. 我們選出的 X 變數為
   funded_amnt, percent_bc_gt_75, num_actv_rev_tl, inq_last_6mths 4 項; Y 變數為 loan_status。定義如下:

- funded_amnt: 資金提供總額，總額越高，收不回的風險也就越高，最終越不利於平台貸款狀態。
- percent_bc_gt_75: 動用信用額度超過 75%的卡數，此類卡數越多代表越有信用風險，越不利於平台貸款狀態。也就是說那些錢會更沒有機會收回來，因為放出去的太多。(信用卡額度使用率是「信用餘額」除以「信用卡的總額度上限」的比例。建議將使用額度控制在 10%~30%之間)
- num_actv_rev_tl:目前持續使用循環利息帳戶數量，帳戶數量越多，越不利於平台貸款狀態。因為這些帳戶並未在當期繳完應繳金額，反而持續延後使用循環利率，到最後金額會越來越大，帳戶使用者將更負擔不起。而導致貸款單位風險越大，可能會收不回此筆費用。
- inq_last_6mths:過去 6 個月的查詢次數，若查詢次數太高，會被銀行列為高風險族群，也就越不利於平台貸款狀態。(聯徵紀錄查詢的目的是讓銀行等金融機構了解這個人的信用狀況好或不好，可以用來參考評估，要不要借錢給這個人。若次數太多金融機構會認為為什麼申請這麼多次還要再來申請新的信用貸款)
- (Y)loan_status:平台貸款狀態(1 代表違約, 0 代表正常)

最終得出**錯誤放貸率為 0.1574，放款總量=108** 的結果。在三種模型下，此四項 X 變數在 XGBoost 模型裡會顯著的影響平台貸款狀態，最終使平台貸款狀

態趨近正常（錯放率低）。

2. 基本統計量分析（Y 與挑選的 X）：

	loan_status	funded_amnt	percent_bc_gt_75	num_actv_rev_tl	inq_last_6mths
count	700.00	700.00	692.00	700.00	700.00
mean	0.28	13985.43	41.61	5.28	0.63
std	0.45	8626.40	36.75	3.10	0.87
min	0.00	1000.00	0.00	0.00	0.00
25%	0.00	7200.00	0.00	3.00	0.00
50%	0.00	12000.00	33.30	5.00	0.00
75%	1.00	20000.00	67.88	7.00	1.00
max	1.00	40000.00	100.00	24.00	4.00

在此五項變數的基本統計量發現：funded_amnt 最大值高達 40000，平均為 13985.43（雖然標準差看起來略大，但認為與其他變數相比平均值也較大且以經濟直覺來看覺得此變數滿有影響力，因此選擇放入看看）。
percent_bc_gt_75 的平均值為 41.61%，而有 33.3%的人累積使用額度 50%。
num_actv_rev_tl 平均值 5.28，累積 50%為 5.00。最後 inq_last_6mths 平均為 0.63，最大值為 4。

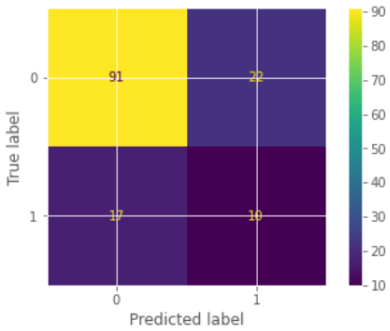
(d)演算法簡單概念敘述 與 參數設定

1. XGBoost 演算法簡單概念：為先產生一棵樹，根據這棵樹的誤差產生第二棵樹，再依照前兩棵樹的誤差產生第三棵樹...依此類推，最後再將所有結果相加，就是使用 XGBOOST 所得到的預測結果(接近真實值)。
也就是說， 第一次先預測真實的 Y，再計算出真實 Y 與我們第一次結果預測的 Y 之間的殘差(殘差 1)，將殘差 1 拿去跑第二次預測的結果，兩者之間的差距(前者減後者)即為殘差 2，而第三次預測即是預測殘差 2...以此類推，最後再將每次預測出的結果相加，通常就會很接近真實的 Y)。

2. 參數設定：如(b)所提及，我們想要使用較原先設置的 0.75 還要多一點的資料去訓練，因此讓訓練集:測試集的比為 0.8:0.2；而發現在這個資料內樹越多其實效果沒有變得更好，因此不斷調整最後讓 n_estimators 從 100 減少為 60，最後是 learning_ rate 為 0.2 時績效最好，
max_depth 為 20 時效果也最佳因此無調整，其他參數維持不變。

(e)混淆矩陣與預測績效指標(自己選擇)

在預測測試集中，我們預測出來的精確度 Accuracy 為 0.7214，得出的混淆矩陣如右圖。如先前提及，錯放率為 0.1574，放款總量為 108。



而我們在調整變數或參數跑模型時，將預測績效指標設定為 Accuracy>0.7，錯放率<0.16，放款總量希望能夠大於 100。

(f)若使用不只一個演算法，或調整參數，可比較不同演算法或有沒有調整參數的績效，敘述最後結論。

依照我們所嘗試的演算法與績效排名來看，最佳的是 XGBoost（錯放率預測為 0.1574，實際績效為 0.2685），再來是隨機森林（錯放率預測為 0.1843，實際績效為 0.2795），最後則是決策樹（錯放率預測為 0.1568，實際績效為 0.2813），雖然決策樹的預測績效是三者中最好的，但實際績效卻是最低的。可能是資料太多與變數選擇不同（績效好的同樣變數在另外兩個模型中績效皆變差），還有模型背後概念的不同而產生影響。

而我們調整參數後的績效大多都比預設的還要好（如下題表格），決策樹則是未調整時較佳（test size=0.25,max_depth=5）。

(g)看看找出的預測規則或變數(若能看變數重要性)結果是否合理。

從我們使用的模型中挑出排名最高的前四次來加以比較：

測試模型	放入變數	test size	參數
第三次 XGBoost	funded_amnt percent_bc_gt_75 num_actv_rev_tl inq_last_6mths	0.2	n_estimators=60 max_depth=20 learning_rate=0.2
第十二次 隨機森林	funded_amnt delinq_2yrs dti installment grade_4 grade_5 annual_inc (加了類別轉虛擬的變數 grade)	0.23	n_estimators=45 max_depth=12

第三次 決策樹	funded_amnt percent_bc_gt_75 installment dti annual_inc	0.25	max_depth=5
第 18 次 XGBoost	funded_amnt all_util num_actv_rev_tl inq_last_6mths grade_4 grade_5 dti	0.2	n_estimators=55 max_depth=12 learning_rate=0.11

選擇的變數主要都有 funded_amnt 資金總額，最具影響力，而其他放入次數較多的變數有 dti 申貸時每月支付債務佔薪水比率，installment、annual_inc 等變數。我們認為 funded_amnt 對於績效的高低有一定程度的影響力，因資金提供總額越高，收不回的風險也就越高，因此覺得此結果是合理的。

而我們在測試的過程中發現訓練集與測試集的比例對於績效表現也有很大的影響力，依照測試集比例與績效來看，從第三名至第一名的測試集比例分別為 0.25、0.23、0.2，而結果與我們認為拿相對多一點資料去訓練會使績效較好的預期結果一致。因此我們選擇的訓練集比例大多在 0.2 左右。

變數重要性的部分，我們在第十二次隨機森林中有依照變數重要性增加了最高的 delinq_2yrs(過去兩年內)發生過逾期 30 天以上的帳戶數量，第 18 次 XGBOOST 也有依照變數重要性做調整，結果確實有使績效增加，但若再多放幾個卻會使績效降低。

(h)總結:

1. 最終嘗試出績效最高是使用 XGBoost 模型。
2. 預測變數使用：funded_amnt、percent_bc_gt_75、num_actv_rev_tl、inq_last_6mths。
3. 訓練集與測試集的比例為 0.2 與 0.8，n_estimators=60，max_depth=20，learning_rate=0.2。
4. 在訓練資料中得出的績效為:
   錯誤貸放率 = 0.1574074074074074
   放款總量 = 108
5. 在測試資料中得出的績效為:
   wrong_loan_ratio = 0.2685185185
   loan_ratio = 0.72
   →整體績效算佳

（下一頁為其他 extra work 或想法說明）

(i)其他 extra work 或想法說明

王樂詠：

三種模型我們都嘗試了很多次，在訓練資料中也測試出比我們排行第一測試資料更低的錯誤放貸率，有更佳的表現，但在最終測試資料的排名反而出現在十分後面，其原因之一可能是資料選擇不夠具有代表性，如放款量太低，測試集與訓練集比例懸殊，導致績效不佳，無法套用到其他資料上，最終績效是否良好，除了要考慮到放入的變數，其他參數的比例抉擇也會影響最後的績效與模型的適用性。

溫庭妤：

在挑選 X 變數時，雖然是從九項與 loan_status 有高相關係數的變數中選取，但無法保證可以完全的使錯誤放貸率降低。必須要考慮到這九個變數之間的相關係數，若彼此之間相關係數過高則會影響到結果。此外，我們也有試著加入非此九項的變數來觀察模型是否會受到影響。例如:我們所選的四個變數其中之一的 inq_last_6mths，它與 loan_status 的相關係數只有 0.052，但在放入模型後卻能使錯誤放貸率降低。所以我們多方的嘗試放入不同 X 變數，最終才決定了績效最佳的一組。

林嘉柔：

- 因為覺得變數太多不知道從哪裡開始，於是挑選與 loan_status 相關係數大於 0.1 的前五個變數，再刪除共線性較高的其中一個變數(剔除績效差的那個)，雖然一開始績效不錯，但後來發現依照經濟直覺加入與調整變數組合後會使績效更好。
- 試著使用中位數去補資料中的缺失值，但發現績效與使用平均數去補的差別不大。
- 在第 12 次隨機森林與第 18 次 XGBoost 中加入了相關係數與變數重要性皆較高的類別型轉虛擬變數 grade_4 與 grade_5，發現績效有變好，但若是放在第 3 次 XGBoost 的變數組合裡還是會變差。
- 類別型變數像是 purpose 借款人申貸時勾選目的，選擇了 purpose_2 繳信用卡，覺得一個人貸款去繳信用卡違約機率應該滿高的，但加入此變數的績效卻沒有變好，與經濟直覺不太符合的感覺。
- 調整參數的部分，發現 n_estimators 大約在 45~60 會較好，在我們測試的情況下如果增加甚至大於 100 反而會變差；而 max_depth 原本想說不要放太多層，因為擔心會有過擬合的狀況所以一直都降低設在 12 左右，但後來發現最好的績效其實是 20，可能是因為資料與其特徵、模型訓練等關係，20 並不算太多反而是最佳的(不過也發現 20~30 的績效都是相同的)。
- 試過調整決策樹的深度，像是 max_depth 設置在 7 或 8，雖然預測的錯放率會降低到 0.14 甚至是 0.13 左右，但實際績效非常差，可能就是因為過擬合的關係。
- 也曾試過剔除統計表中 p 值不夠顯著(大於 0.1)的變數去跑羅吉斯迴歸但績效不佳，後來換去跑 XGBoost，而參數皆與第三次 XGBoost 相同，雖然預測績效有變得較好但實際績效卻非常差，或許是剩下的變數還是

太多了，與模型的懲罰機制有關。

- 最後則是依照第三次 XGBoost(績效最好)的變數與參數調整，試試看能不能找到更好的組合，不過就算預測出的錯放率有降低，混淆矩陣預測與真實相同的兩格增加、不同的減少，放貸率也有增加，但是實際績效卻都沒有上述表格中前四名的績效好。因此無論增加還是減少抑或是修改，我們嘗試過績效最好的變數組合還是 funded_amnt、percent_bc_gt_75、num_actv_rev_tl、inq_last_6mths。