Company Bankruptcy Prediction
公司破產預測

(台灣經濟日報1999-2009年破產數據)

組別 : 第五組

組員 : S07353064 束家安

S07240018 溫宏岳

AI實作(一)期末報告

2021/05/31

目錄

1. 資料說明
2. 資料探索
3. 資料前處理
4. 模型架設
5. 模型評估
6. 程式實測
7. 問題說明
8. 本學期心得
9. 組員分工項目

資料說明

1.資料出處

- https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction
- 此數據為統計公司破產預測
- 這數據擷取自<台灣經濟日報>
- 公司破產是根據台灣證卷交易所的業務規定定義的
- 這資料已更新過版本2，數據更好理解

資料說明
2.資料目的

- 此數據的主要目的是利用大量不同的因素來去計算一間公司在哪方面數據下降的時候倒閉機率最高，除了第一項破產與否的1跟0，剩下94項皆為財經類專有名詞，如:營業利潤率，稅前稅後淨利率。

資料說明

3.重要特徵與 Label

- 這些數據其實對於此統計而言都是重要特徵，前面的圖主要是利用Bankrupt製圖，後面推演則各用不同的資料去做演算
- 主要還是照著之前作業與助教教學的部分直接丟下去讓他進行推算
- Label為Bankrupt?(是否有破產)

資料探索

簡單的基本計數圖

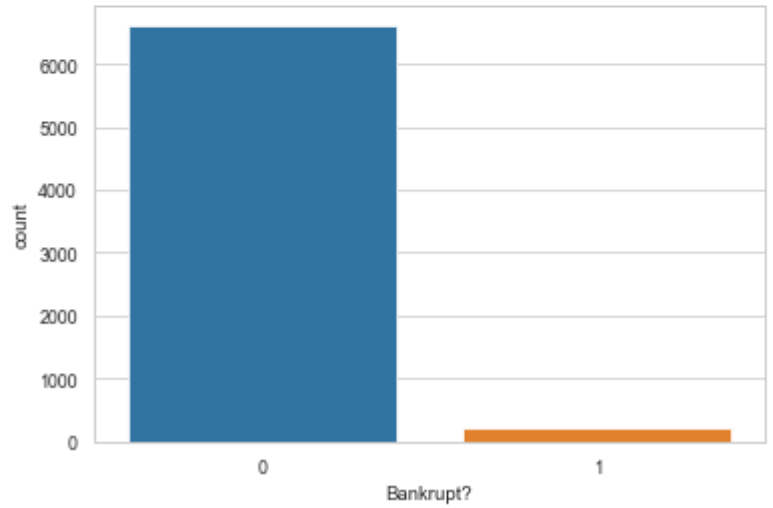
真實是否破產的計數圖(0:沒破產；1:破產)

```
clear_data = data_one_hot.drop(['Bankrupt?'],axis = 1)
label = data_one_hot['Bankrupt?']
df['Bankrupt?'].value_counts()
```

```
0    6599
1     220
Name: Bankrupt?, dtype: int64
```

```
sns.countplot(df['Bankrupt?'])
```

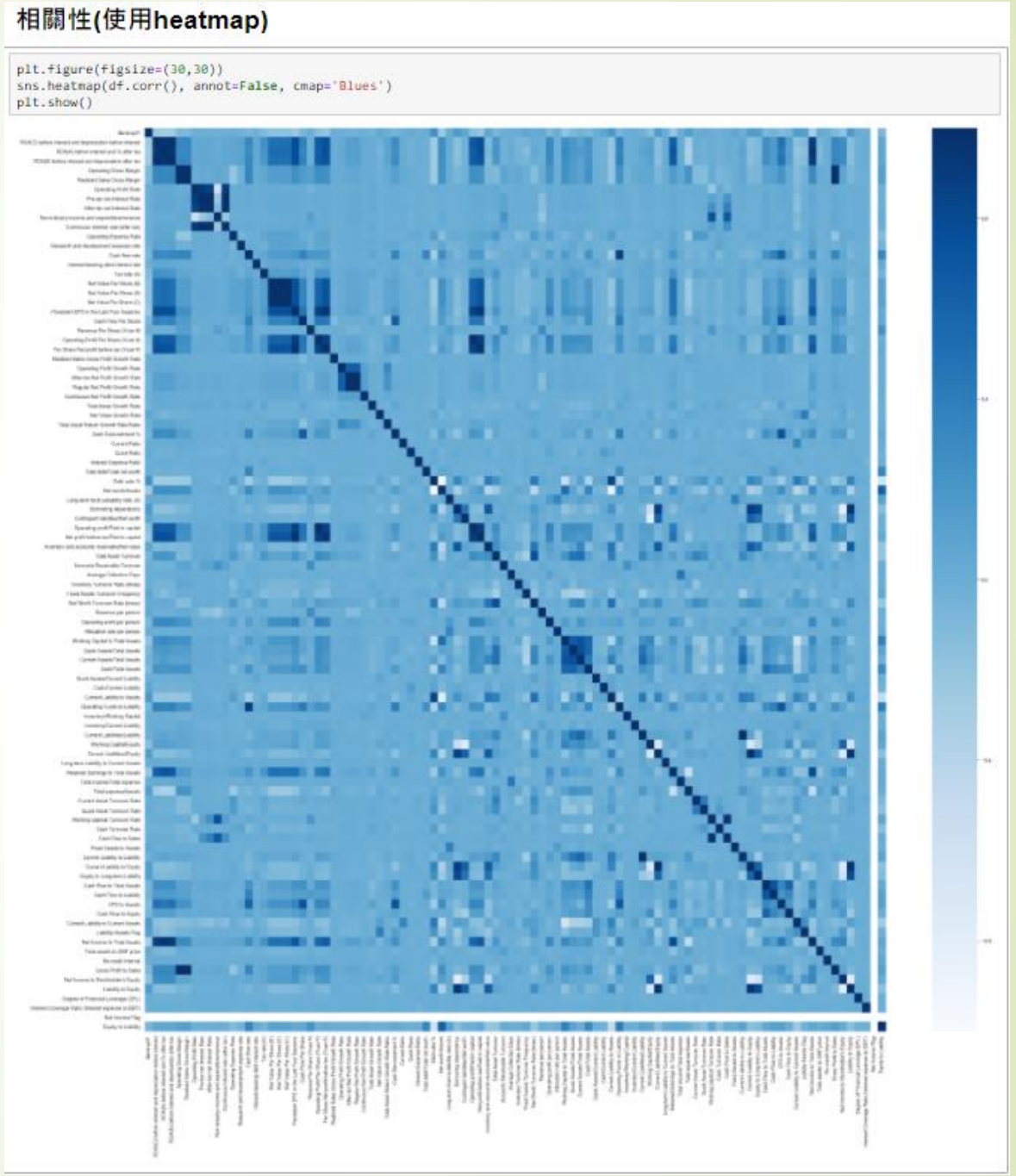
```
<matplotlib.axes._subplots.AxesSubplot at 0x1fb22cab358>
```



資料探索

Heatmap就是所謂的熱力圖

熱力圖在實際中常用於展示一組變量的相關係數矩陣，在展示列聯表的數據分佈上也有較大的用途，通過熱力圖我們可以非常直觀地感受到數值大小的差異狀況。



資料前處理

1.缺失值處理

檢查資料是否有缺失值

```
df.isnull().sum(axis=0)
```

```
Bankrupt?                                                 0
 ROA(C) before interest and depreciation before interest  0
 ROA(A) before interest and % after tax                   0
 ROA(B) before interest and depreciation after tax        0
 Operating Gross Margin                                   0
 Realized Sales Gross Margin                              0
 Operating Profit Rate                                    0
 Pre-tax net Interest Rate                                0
 After-tax net Interest Rate                              0
 Non-industry income and expenditure/revenue              0
 Continuous interest rate (after tax)                     0
 Operating Expense Rate                                   0
 Research and development expense rate                    0
 Cash flow rate                                           0
 Interest-bearing debt interest rate                      0
 Tax rate (A)                                             0
 Net Value Per Share (B)                                  0
 Net Value Per Share (A)                                  0
 Net Value Per Share (C)                                  0
 Persistent EPS in the Last Four Seasons                  0
 Cash Flow Per Share                                      0
 Revenue Per Share (Yuan ¥)                               0
 Operating Profit Per Share (Yuan ¥)                      0
 Per Share Net profit before tax (Yuan ¥)                 0
 Realized Sales Gross Profit Growth Rate                  0
 Operating Profit Growth Rate                             0
 After-tax Net Profit Growth Rate                         0
 Regular Net Profit Growth Rate                           0
 Continuous Net Profit Growth Rate                        0
 Total Asset Growth Rate                                  0
 Current Liabilities/Equity                               0
 Long-term Liability to Current Assets                    0
 Retained Earnings to Total Assets                        0
 Total income/Total expense                               0
 Total expense/Assets                                     0
 Current Asset Turnover Rate                              0
 Quick Asset Turnover Rate                                0
 Working capitcal Turnover Rate                           0
 Cash Turnover Rate                                       0
 Cash Flow to Sales                                       0
 Fixed Assets to Assets                                   0
 Current Liability to Liability                           0
 Current Liability to Equity                              0
 Equity to Long-term Liability                            0
 Cash Flow to Total Assets                                0
 Cash Flow to Liability                                   0
 CFO to Assets                                            0
 Cash Flow to Equity                                      0
 Current Liability to Current Assets                      0
 Liability-Assets Flag                                    0
 Net Income to Total Assets                               0
 Total assets to GNP price                                0
 No-credit Interval                                       0
 Gross Profit to Sales                                    0
 Net Income to Stockholder's Equity                       0
 Liability to Equity                                      0
 Degree of Financial Leverage (DFL)                       0
 Interest Coverage Ratio (Interest expense to EBIT)       0
 Net Income Flag                                          0
 Equity to Liability                                      0
Length: 96, dtype: int64
```

資料前處理
2. label處理

One hot encoding

可以處理數字但不能直接處理字串值，需先將字串對應成數值。

```
data_one_hot = pd.get_dummies(df)
data_one_hot.head()
```

	Bankrupt?	ROA(C) before interest and depreciation before interest	ROA(A) before interest and % after tax	ROA(B) before interest and depreciation after tax	Operating Gross Margin	Realized Sales Gross Margin	Operating Profit Rate	Pre-tax net Interest Rate	After-tax net Interest Rate	Non-industry income and expenditure/revenue	...	Net Income to Total Assets	Total assets to GNP price	No-credit Interval
0	1	0.370594	0.424389	0.405750	0.601457	0.601457	0.998969	0.796887	0.808809	0.302646	...	0.716845	0.009219	0.622879
1	1	0.464291	0.538214	0.516730	0.610235	0.610235	0.998946	0.797380	0.809301	0.303556	...	0.795297	0.008323	0.623652
2	1	0.426071	0.499019	0.472295	0.601450	0.601364	0.998857	0.796403	0.808388	0.302035	...	0.774670	0.040003	0.623841
3	1	0.399844	0.451265	0.457733	0.583541	0.583541	0.998700	0.796967	0.808966	0.303350	...	0.739555	0.003252	0.622929
4	1	0.465022	0.538432	0.522298	0.598783	0.598783	0.998973	0.797366	0.809304	0.303475	...	0.795016	0.003878	0.623521

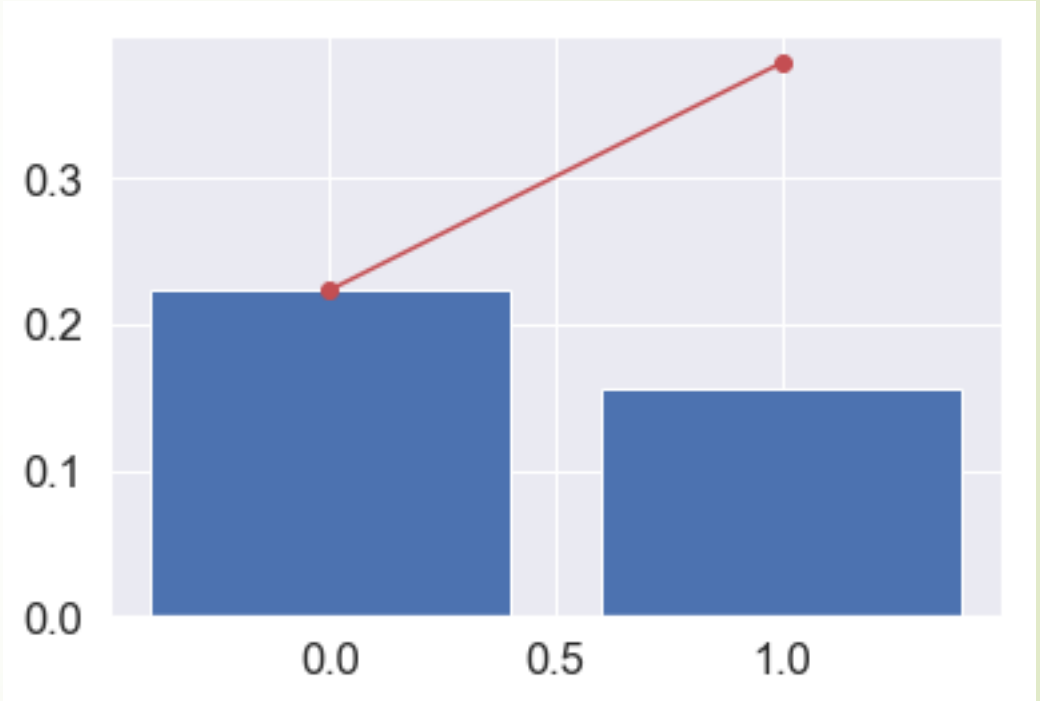
5 rows × 96 columns

Net Income to Total Assets	Total assets to GNP price	No-credit Interval	Gross Profit to Sales	Net Income to Stockholder's Equity	Liability to Equity	Degree of Financial Leverage (DFL)	Interest Coverage Ratio (Interest expense to EBIT)	Net Income Flag	Equity to Liability
0.716845	0.009219	0.622879	0.601453	0.827890	0.290202	0.026601	0.564050	1	0.016469
0.795297	0.008323	0.623652	0.610237	0.839969	0.283846	0.264577	0.570175	1	0.020794
0.774670	0.040003	0.623841	0.601449	0.836774	0.290189	0.026555	0.563706	1	0.016474
0.739555	0.003252	0.622929	0.583538	0.834697	0.281721	0.026697	0.564663	1	0.023982
0.795016	0.003878	0.623521	0.598782	0.839973	0.278514	0.024752	0.575617	1	0.035490

資料前處理
2. PCA

PCA最常見的應用之一就是將高維數據可視化

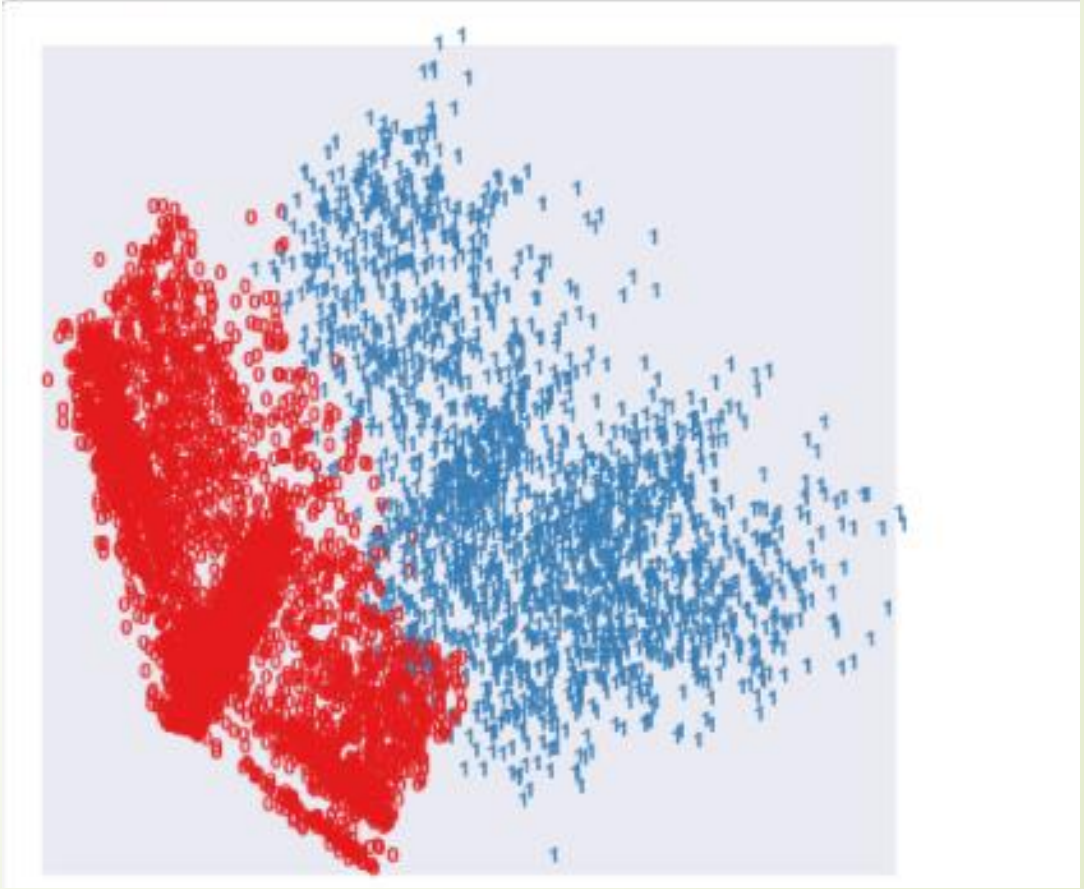
它可以將具有兩個及以上特徵的數據進行可視化
PCA直方圖



資料前處理
2. PCA

PCA降維操作

```
pca = PCA()
X_pca = pca.fit_transform(X)
y = clf.labels_
x_min,x_max = X_pca.min(0),X_pca.max(0)
X_norm = (X_pca-x_min)/(x_max-x_min)
plt.figure(figsize=(8,8))
for i in range(X_norm.shape[0]):
    plt.text(X_norm[i,0],X_norm[i,1],str(y[i]),color=plt.cm.Set1(y[i]),
    fontdict = {'weight':'bold','size':12})
plt.xticks([])
plt.yticks([])
plt.show()
```



模型架設

- 模型分別為1.KNN
  2.SVM
  3.LogisticRegression
  4.Voting(DTree、KNN、SVM)
  5.Gradient_boosting
- 原因：在第八、九次作業中以上的模型表現的比較好，但沒有放在一起比較過，因此才選擇這些模型來使用。

模型評估

1.KNN

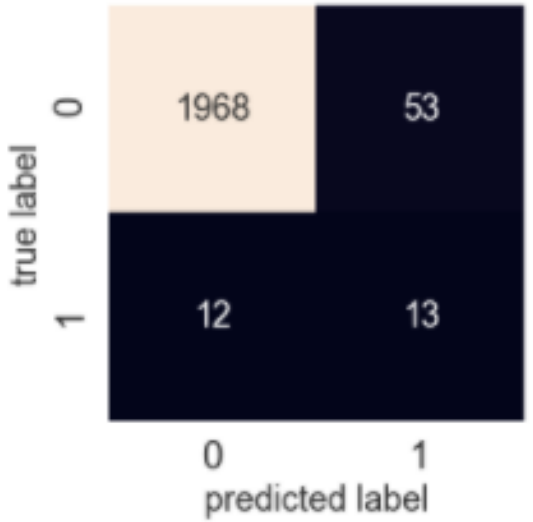
當我們需要預測一個新實例的某個屬性A時，KNN算法會搜索訓練數據集找到K個最相似的實例。這些相似實例的A屬性會被總結歸納，作為新實例A屬性的預測。

```
y_pred = knn.predict(x_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy: ",acc)
```

```
Accuracy:  0.9682306940371457
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report


mat = confusion_matrix(y_test, y_pred)

sns.set(font_scale = 1.5)
sns.heatmap(mat.T, square = True, annot = True, fmt = 'd', cbar = False)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.ylabel('true label')
plt.xlabel('predicted label')
target_names = ['0','1']

print(classification_report(y_test, y_pred, target_names = target_names))
```

```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1980
           1       0.52      0.20      0.29        66

   micro avg       0.97      0.97      0.97      2046
   macro avg       0.75      0.60      0.63      2046
weighted avg       0.96      0.97      0.96      2046
```

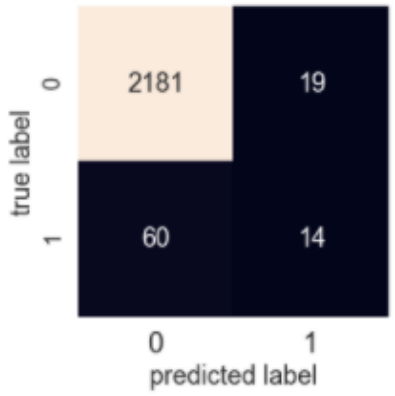


模型評估

KNN 的 K-Fold = 3

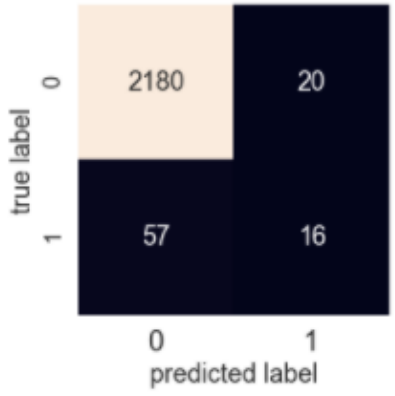
此為第 1 次，K-Fold迴圈:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	2200
1	0.42	0.19	0.26	74
micro avg	0.97	0.97	0.97	2274
macro avg	0.70	0.59	0.62	2274
weighted avg	0.96	0.97	0.96	2274



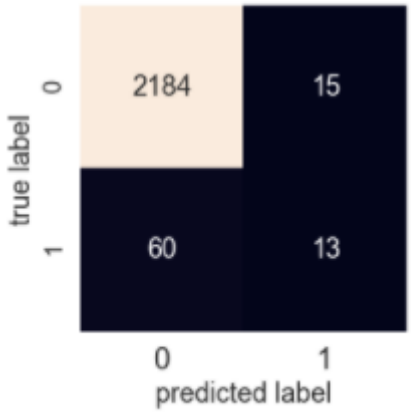
此為第 2 次，K-Fold迴圈:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	2200
1	0.44	0.22	0.29	73
micro avg	0.97	0.97	0.97	2273
macro avg	0.71	0.61	0.64	2273
weighted avg	0.96	0.97	0.96	2273



此為第 3 次，K-Fold迴圈:

	precision	recall	f1-score	support
0	0.97	0.99	0.98	2199
1	0.46	0.18	0.26	73
micro avg	0.97	0.97	0.97	2272
macro avg	0.72	0.59	0.62	2272
weighted avg	0.96	0.97	0.96	2272



用不同的資料評估訓練模組好壞的K-Fold=>交叉驗證

模型評估
2. SVM

SVM的基本原理就是找到一個超平面

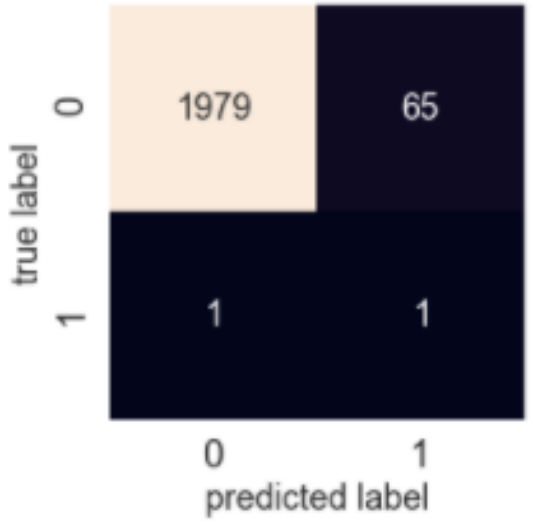
能將數據進行有效的分類，同時保證超平面兩邊的樣本儘可能遠的距離這個超平面。

```
y_pred = model.predict(x_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy: ",acc)
```

```
Accuracy:  0.967741935483871
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

mat = confusion_matrix(y_test, y_pred)

sns.set(font_scale = 1.5)
sns.heatmap(mat.T, square = True, annot = True, fmt = 'd', cbar = False)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.ylabel('true label')
plt.xlabel('predicted label')

target_names = ['0','1']

print(classification_report(y_test, y_pred, target_names = target_names))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      1980
           1       0.50      0.02      0.03        66

   micro avg       0.97      0.97      0.97      2046
   macro avg       0.73      0.51      0.51      2046
weighted avg       0.95      0.97      0.95      2046
```



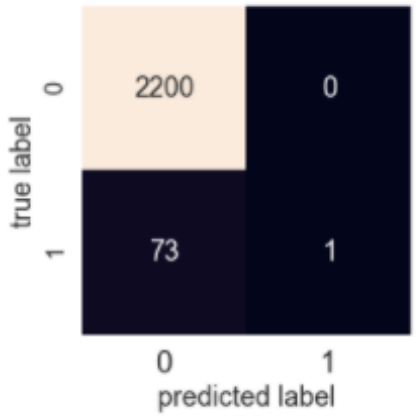
模型評估

SVM 的 K-Fold = 3

此為第 1 次，K-Fold迴圈:

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\
'auto' to 'scale' in version 0.22 to account better for
his warning.
  "avoid this warning.", FutureWarning)
```

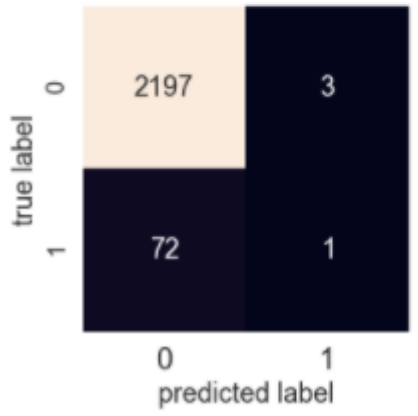
	precision	recall	f1-score	support
0	0.97	1.00	0.98	2200
1	1.00	0.01	0.03	74
micro avg	0.97	0.97	0.97	2274
macro avg	0.98	0.51	0.51	2274
weighted avg	0.97	0.97	0.95	2274



此為第 2 次，K-Fold迴圈:

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\
'auto' to 'scale' in version 0.22 to account better for
his warning.
  "avoid this warning.", FutureWarning)
```

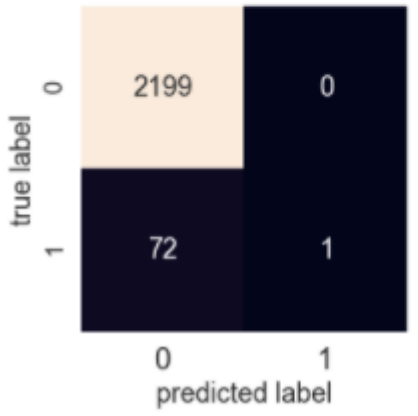
	precision	recall	f1-score	support
0	0.97	1.00	0.98	2200
1	0.25	0.01	0.03	73
micro avg	0.97	0.97	0.97	2273
macro avg	0.61	0.51	0.50	2273
weighted avg	0.95	0.97	0.95	2273



此為第 3 次，K-Fold迴圈:

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\svm\
'auto' to 'scale' in version 0.22 to account better for
his warning.
  "avoid this warning.", FutureWarning)
```

	precision	recall	f1-score	support
0	0.97	1.00	0.98	2199
1	1.00	0.01	0.03	73
micro avg	0.97	0.97	0.97	2272
macro avg	0.98	0.51	0.51	2272
weighted avg	0.97	0.97	0.95	2272



評估訓練模組好壞的K-Fold=>交叉驗證

模型評估
3.LogisticRegression

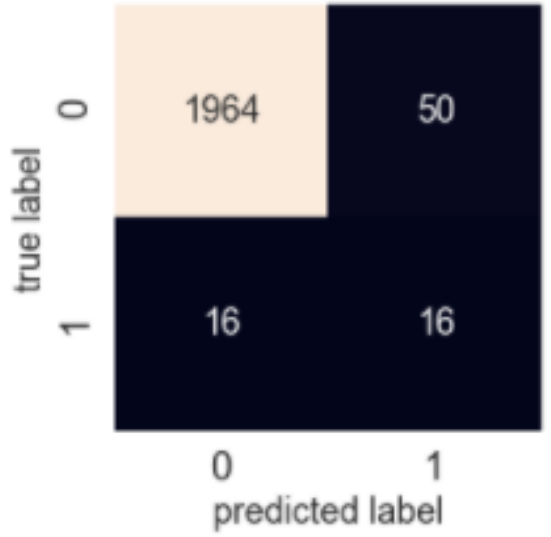
又稱羅吉斯回歸
概念上就是將點帶進去回歸線，回歸線輸出值若是>=0，是一類(target)，值<0是另一類(non-target)這個判斷法其實就是一個unit-step function

```
y_pred = logreg.predict(x_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy: ",acc)
```

```
Accuracy:  0.967741935483871
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

mat = confusion_matrix(y_test, y_pred)

sns.set(font_scale = 1.5)
sns.heatmap(mat.T, square = True, annot = True, fmt = 'd', cbar = False)
plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.ylabel('true label')
plt.xlabel('predicted label')

target_names = ['0','1']
print(classification_report(y_test, y_pred, target_names = target_names))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      1980
           1       0.50      0.24      0.33        66

   micro avg       0.97      0.97      0.97      2046
   macro avg       0.74      0.62      0.66      2046
weighted avg       0.96      0.97      0.96      2046
```



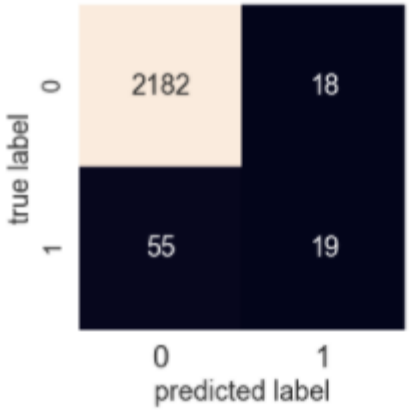
模型評估

LogisticRegression的 K-Fold = 3

此為第 1 次，K-Fold迴圈:

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\lin
to 'lbfgs' in 0.22. Specify a solver to silence this wa
  FutureWarning)
```

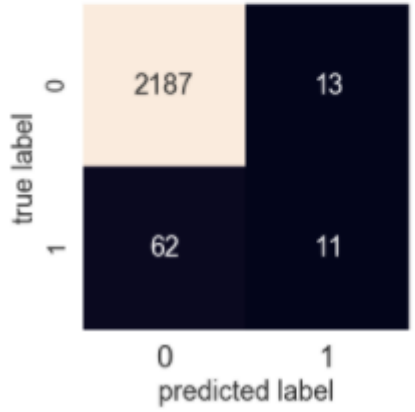
	precision	recall	f1-score	support
0	0.98	0.99	0.98	2200
1	0.51	0.26	0.34	74
micro avg	0.97	0.97	0.97	2274
macro avg	0.74	0.62	0.66	2274
weighted avg	0.96	0.97	0.96	2274



此為第 2 次，K-Fold迴圈:

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\lin
to 'lbfgs' in 0.22. Specify a solver to silence this wa
  FutureWarning)
```

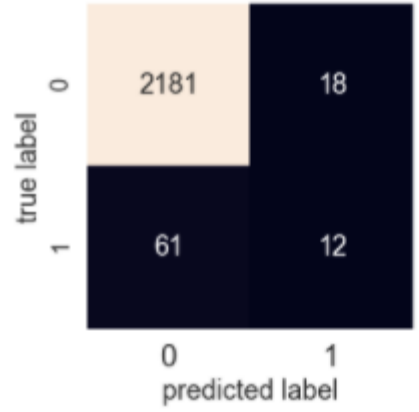
	precision	recall	f1-score	support
0	0.97	0.99	0.98	2200
1	0.46	0.15	0.23	73
micro avg	0.97	0.97	0.97	2273
macro avg	0.72	0.57	0.60	2273
weighted avg	0.96	0.97	0.96	2273



此為第 3 次，K-Fold迴圈:

```
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\line
to 'lbfgs' in 0.22. Specify a solver to silence this wa
  FutureWarning)
```

	precision	recall	f1-score	support
0	0.97	0.99	0.98	2199
1	0.40	0.16	0.23	73
micro avg	0.97	0.97	0.97	2272
macro avg	0.69	0.58	0.61	2272
weighted avg	0.95	0.97	0.96	2272



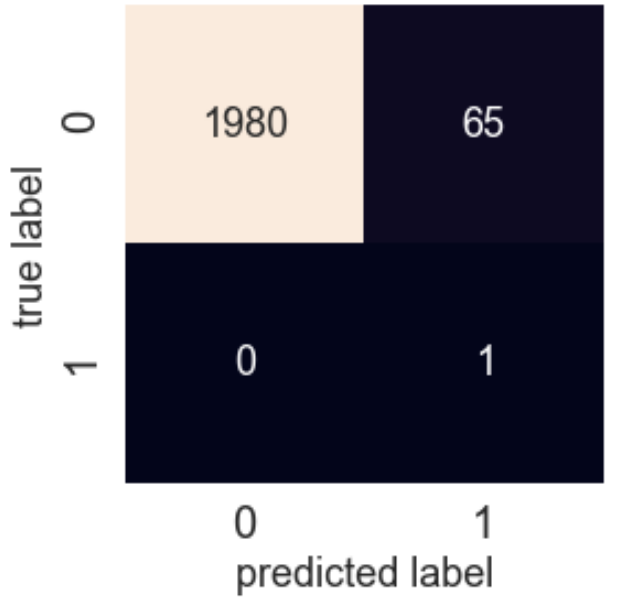
模型評估
4.Voting

中文全名叫摩爾投票演算法，也可以叫做多數投票演算法
每次從陣列中找出一對不同的元素，將它們從陣列中刪除，直到遍歷完整個陣列。

```
acc = accuracy_score(y_test, y_pred)
print("不同演算法Voting(DTree、KNN、SVM)的accuracy: ",acc)
```

```
不同演算法Voting(DTree、KNN、SVM)的accuracy:  0.9682306940371457
```

```
from sklearn.metrics import confusion_matrix

mat = confusion_matrix(y_test,y_pred)

sns.set(font_scale = 1.5)
sns.heatmap(mat.T,square=True,annot = True,fmt = 'd',cbar = False)

plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.ylabel('true label')
plt.xlabel('predicted label');
```



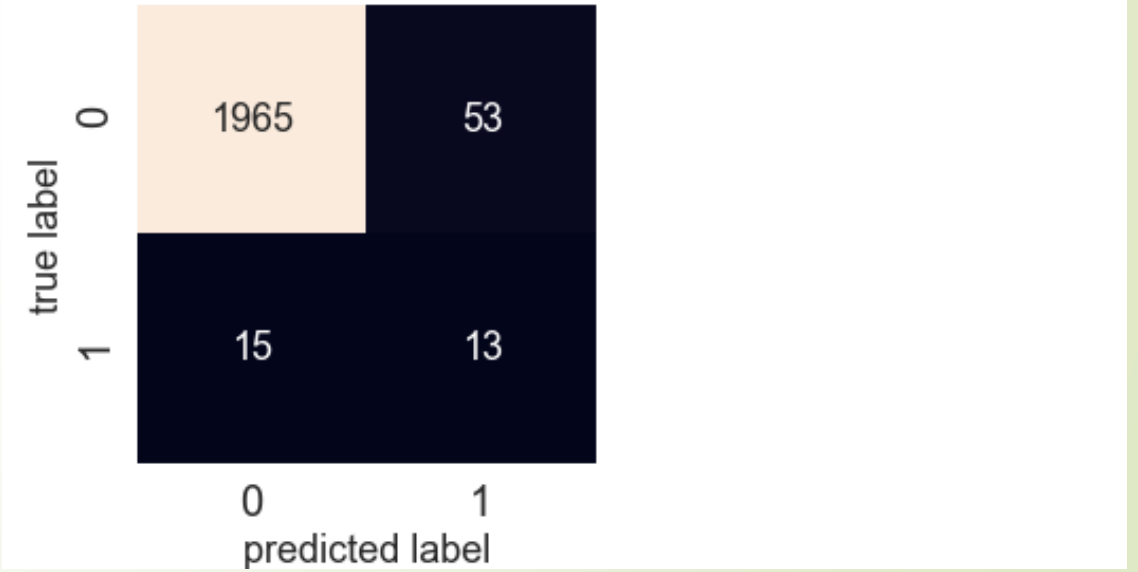
模型評估
5.Gradient_boosting

極限梯度提升 Xgboost ，
Gradient boosting算是一種回歸樹(DTree)的算法
假設今天你有一個Regression問題沒做好，留下了餘數Residual，把這個餘數當Regression問題再次處理，在將結果附到先前的問題中

```
acc = accuracy_score(y_test, y_pred)
print("accuracy: ",acc)
```

```
accuracy:  0.9667644183773216
```

```
from sklearn.metrics import confusion_matrix

mat = confusion_matrix(y_test,y_pred)

sns.set(font_scale = 1.5)
sns.heatmap(mat.T,square = True,annot = True,fmt = 'd',cbar = False)

plt.xticks(fontsize = 20)
plt.yticks(fontsize = 20)
plt.ylabel('true label')
plt.xlabel('predicted label');
```



模型評估

- 由以上資料可得下列精確率的大小：
- Voting(DTree、KNN、SVM) ≈ KNN > SVM ≈ LogisticRegression > Gradient_boosting
- 因此Voting(DTree、KNN、SVM)是反而最好的預測模型。

```
acc = accuracy_score(y_test, y_pred)
print("不同演算法Voting(DTree、KNN、SVM)的accuracy: ",acc)
```

```
不同演算法Voting(DTree、KNN、SVM)的accuracy:  0.9682306940371457
```

```
y_pred = knn.predict(x_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy: ",acc)
```

```
Accuracy:  0.9682306940371457
```

```
y_pred = model.predict(x_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy: ",acc)
```

```
Accuracy:  0.967741935483871
```

```
y_pred = logreg.predict(x_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy: ",acc)
```

```
Accuracy:  0.967741935483871
```

```
acc = accuracy_score(y_test, y_pred)
print("accuracy: ",acc)
```

```
accuracy:  0.9667644183773216
```

問題說明及困境

- 對於MLP模型的使用滿多問題的，可能也是在報告前才學，使用了許多方式卻無法使用，也查閱了許多資料，看起來是tensorflow跟keras相容性問題，也嘗試了網路上的許多解法，像是解除安裝numpy再重新安裝、更新numpy等，但依舊沒辦法解答。
- 其他的話幾乎照搬作業語上課內容上傳就行，還有其實有點不知道PPT該做什麼，或許直接將講解的部分以及大部分的內容直接放在ipynb檔案內就行了。

本學期心得

- 束家安：我認為教了許多東西，但結束OpenCV的部分後可視化有點變太深，像是我這種大三的老屁股都聽不大明白，難度偏高，也可能是因為我python偏弱比較難懂。希望助教可以改成教一部份就讓同學進行操作，或丟出範例程序讓同學更好理解，如果沒有錄影或拍照，若不明白就只能自己找資料看別人怎麼寫了，畢竟太長一段程序也難以全部背起來。

本學期心得

- 溫宏岳：當初會選這門課就是想去了解AI到底在做什麼，經過了一學期感覺後半比較有內容，也或許是因為開給資工系大一的關係吧。還有，就如同東家安所說，我也希望助教可以教一部份就讓同學進行操作那一部分，雖然我只學過anaconda的python，但也過了滿久沒用的，再加上我是輔系的沒認識多少人，上課大部分也只能靠自己拍下來或上網去研究。

組員分工項目

- S07353064 束家安：口頭報告、PPT(主要)、MLP模型執行
- S07240018 溫宏岳：書面資料、程式碼編寫(除了MLP模型)、PPT(檢查與補充)、資料查找

謝謝聆聽~~