智慧化企業整合

Project 2

心臟病發生原因預測模型之建構與分類

指導老師：

邱銘傳 教授

第一組

組員：

109034402 陳光源

109034553 莊婉琦

109034570 林溥鈞

第 1 章 研究主題說明 2
【5W1H】 2
第 2 章 資料分析與前處理 2
資料來源 2
資料欄位 3
資料分析 3
標準化資料集 10
資料樣本區分 11
第 3 章 預測模型設定 11
決策樹演算法 11
高斯貝氏分類器 13
羅吉斯回歸模型 16
羅吉斯回歸建模 17
羅吉斯回歸超參數調整 18
混淆矩陣模型評估 20
ROC/AUC 模型評估 21
KNN (K Nearest Neighbor) 22
KNN Model 23
調整參數 23
評估 KNN 的準確率 24
混淆矩陣 25
第 4 章 模型效度驗證 25
混淆矩陣 25
模型於訓練及與測試及效度比較 27
第 5 章 結論與未來展望 27
第 6 章 參考資料 27

第 1 章研究主題說明

今年有許多藝人因為心臟相關的疾病相繼離世，令人感到相當惋惜。心臟疾病一直是蟬聯國人的前十大死因之一，十年來的排名位居第二名且不曾降低過。心臟病常常是突發性、沒有預警的，它最可怕的是在靜態或休息狀態下毫無症狀的，而是好發在活動或刺激後才會發作，單靠一般的健康檢查是很難發現病狀的，故我們需要更進一步的檢查或追蹤才能及早發現及早治療及預防。我們想藉由 Kaggle 上心臟資料庫裡面所記錄的相關身理特徵，建立預測模型分類導致心臟疾病的相關因子，提供院方判斷需要特別注意的患病原因。

【5W1H】

What?	分類罹患心臟病的身理特徵
Who?	具有相關症狀的患者、心臟科醫生
When?	心臟科醫生問診時
Where?	心臟科醫生醫療紀錄系統
Why?	心臟病需透過詳細的檢查才能被發現，常常一發作就導致死亡
How?	機器學習、資料分析、決策樹分析

透過機器學習訓練出來的模型、決策樹分析，我們期望提供此預測系統，除了協助心臟科醫生更精準、精確的判斷患者罹患心臟病的可能重要原因，也預防醫生誤判的可能。

第 2 章資料分析與前處理

資料來源

我們採用 Kaggle 網站內蒐集的"Heart Dease UCI"資料庫數據資料，作為本次建立模型的資料來源。其中共計有 303 筆資料，14 筆資料欄位中記錄身理的相關病徵。

資料欄位

首先釐清各欄位資料所代表的資訊及其意涵，方便後續資料處理與模型分析得以順利進行，資料欄位共有下列 14 欄:

欄位	定義	資料型別
age	年齡	連續型
sex	性別	0 = 女生 1 = 男生
cp	胸部疼痛類型	1 = 典型心絞痛 2 = 非典型心絞痛 3 = 非心絞痛 4 = 無症狀的
trestbps	靜息血壓	連續型 (unit: mmHg)
chol	膽固醇	連續型 (unit: mg/dl)
fbs	空腹血糖	1 = True (> 120mg/dl) 2 = False (<= 120mg/dl)
restecg	靜息心電圖測量	0 = 正常 1 = ST-T波異常 2 = 可能或確定左心室肥大
thalach	達到的最大心率	連續型

欄位	定義	資料型別
exang	運動誘發心絞痛	1 = Yes 2 = No
oldpeak	運動引起的S-T下降相較於休息	連續型 & float
slope	最高S-T段的斜率	1 = 向上 2 = 平坦 3 = 向下
ca	主要血管	0 - 3
thal	地中海貧血 (說明: 一種先天性貧血)	3 = 正常 6 = 固定缺陷 7 = 可逆缺陷
target	心臟病	0 = No 1 = Yes

表一、資料欄位說明表

資料分析

接著，將檔案以 dataframe 匯入，並
利用 pandas 的

1. head()方法來檢查前幾項數據

```
df = pd.read_csv('heart.csv')
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

2. shape() 查看整筆數據欄位與行數
3. groupby(‘target’) 檢查有無心臟病分布數量

```
[5] print('Heart Disease data set dimension: {}'.format(df.shape))
    df.groupby('target').size()

    Heart Disease data set dimension: (303, 14)
    target
    0    138
    1    165
    dtype: int64
```

4. info() 檢查有沒有缺值(Non-Null)與資料類別(Dtype)

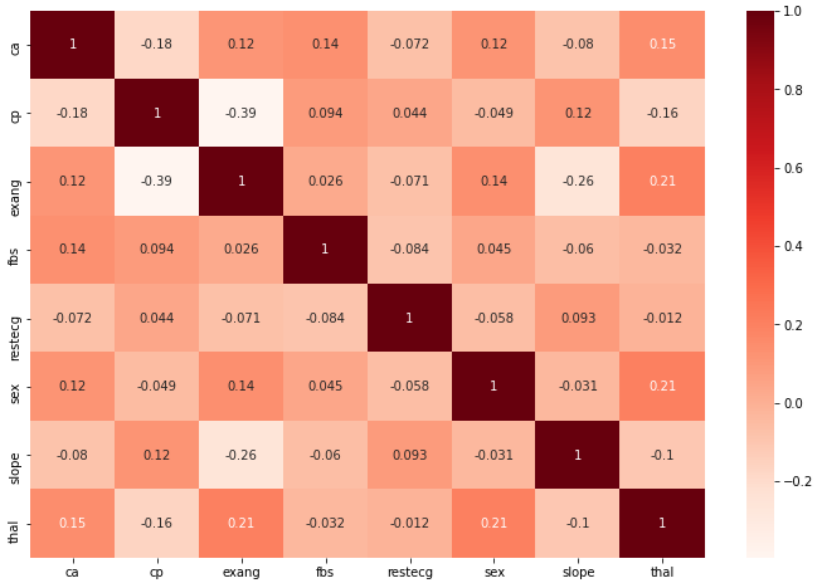
```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

結果發現

1. 資料一共有 303 筆資料與 14 個欄位。
2. 資料無缺失。無心臟病(target = 0) 有 138 筆、有心臟病 (traget = 1)有 165 筆。
3. 資料類別除了一欄為浮點數外，其他為整數型但有連續與離散之分，所依後續資料前處裡需要標準化 (Standardization)和資料編碼 (Encoding)。

因為資料無缺失值、異常值或重複值，所以在資料前處理部分不需要進行資料清洗、資料補值或資料刪除等動作。所以可以進行下一步資料欄位間的關聯性分析。先將資料中的離散型的資料取出 (col_dis)，再以 seaborn 提供之 sns 為離散型資料欄位繪製熱力圖，檢查各欄為間的相關性，此筆擷取出的離散型的資料 (col_dis)在後續資料編碼時也會用到。

```
col_dis = ['ca', 'cp', 'exang', 'fbs', 'restecg', 'sex', 'slope', 'thal']
```



圖一、以熱力圖顯示離散型欄位間相關性

發現與目標欄位(target)有正相關的欄位有:

1. 胸部疼痛類型(cp: 0.43): 有發生過心絞痛的人較易有心臟病
2. 達到的最大心率(thalach: 0.42): 心率越高較亦有心臟病
3. 達到的最大心率(slope: 0.35): 心電圖最高 S-T 段的斜率越大越亦有心臟病

亦可發現其他欄位間的關聯性，如運動引起的 S-T 下降(oldpeak) 與運動誘發心絞痛(exang)有正相關(0.39)，為合理的關聯等。

```
df[['cp', 'target']].groupby(['cp'],  as_index=False).mean().sort_values(by='target', ascending=False)
```

```
df[['thal', 'target']].groupby(['thal'], as_index=False).mean().sort_values(by='target', ascending=False)
```

```
df[['slope', 'target']].groupby(['slope'],  as_index=False).mean().sort_values(by='target', ascending=False)
```

特徵值和與其有高度正相關欄位間關係	說明	視覺化 0: 無心臟病，1: 有心臟病

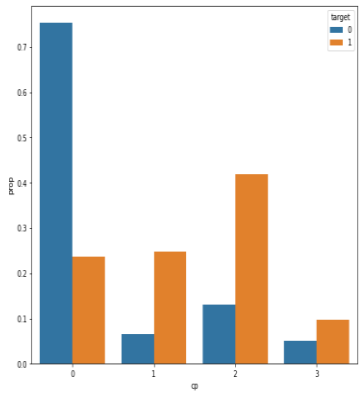
<table>
<tr>
<td>

	cp	target
1	1	0.820000
2	2	0.793103
3	3	0.695652
0	0	0.272727

胸部疼痛類型

</td>
<td>

- 非典型心絞痛具有心臟病的機率為 82.00%。
- 非心絞痛具有心臟病的機率為 79.31%。
- 無症狀具有心絞痛的機率為 69.57%。
- 典型心絞痛具有心絞痛的機率為 27.27%。

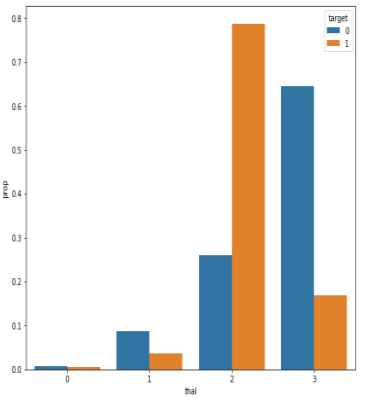
</td>
<td>  </td>
</tr>
<tr>
<td>

	thal	target
2	2	0.783133
0	0	0.500000
1	1	0.333333
3	3	0.239316

地中海貧血

</td>
<td>

- 地中海貧血中可逆缺陷有心臟病機率為 78.31%。
- 地中海貧血中正常有心臟病機率為 50.00%。
- 地中海貧血中固定缺陷有心臟病的機率為 33.33%。

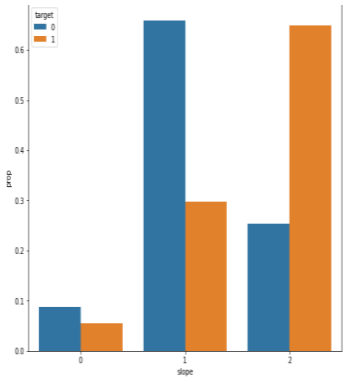
</td>
<td>  </td>
</tr>
<tr>
<td>

	slope	target
2	2	0.753521
0	0	0.428571
1	1	0.350000

最高 S-T 段的斜率

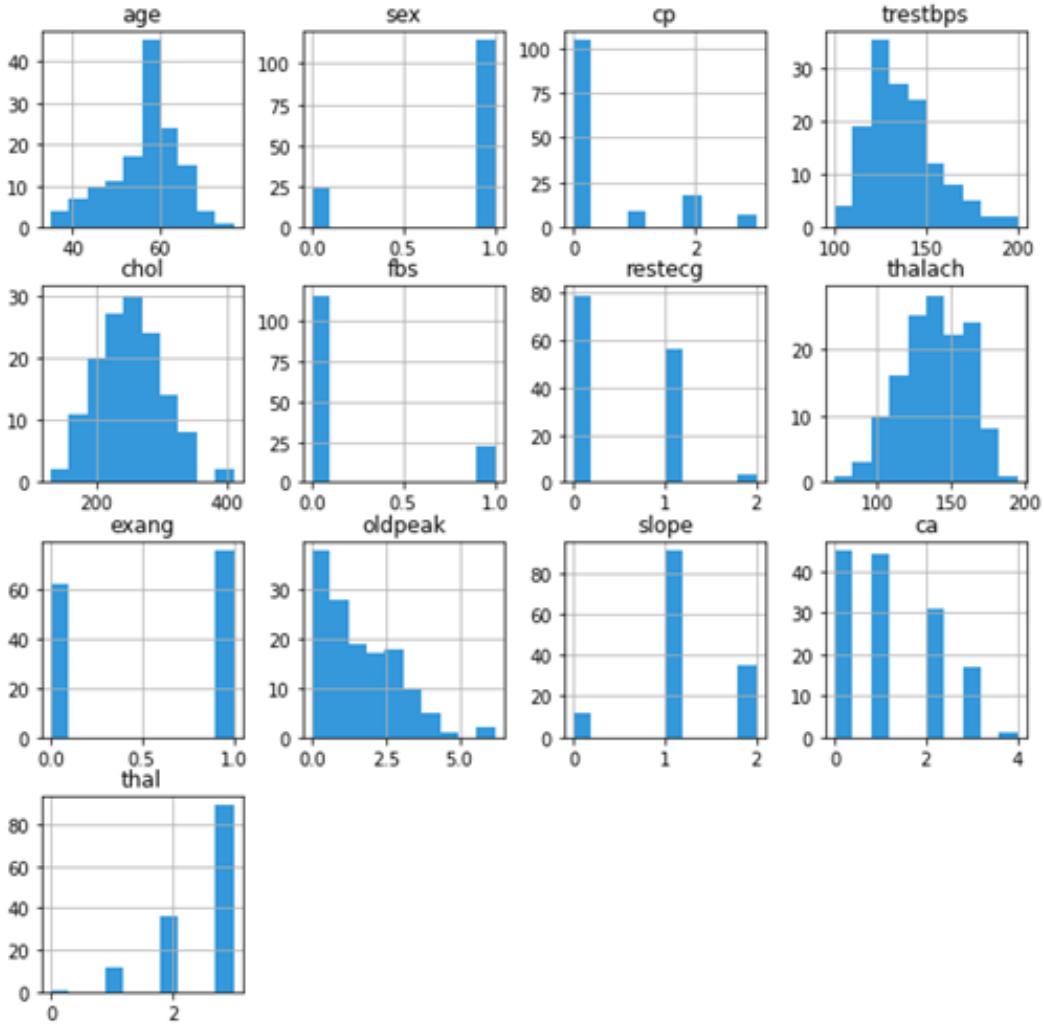
</td>
<td>

- 最高 S-T 段的斜率中途形向下有心臟病機率為 75.35%。
- 最高 S-T 段的斜率中途形向上有心臟病機率為 75.35%。
- 最高 S-T 段的斜率中途形平坦有心臟病機率為 75.35%。

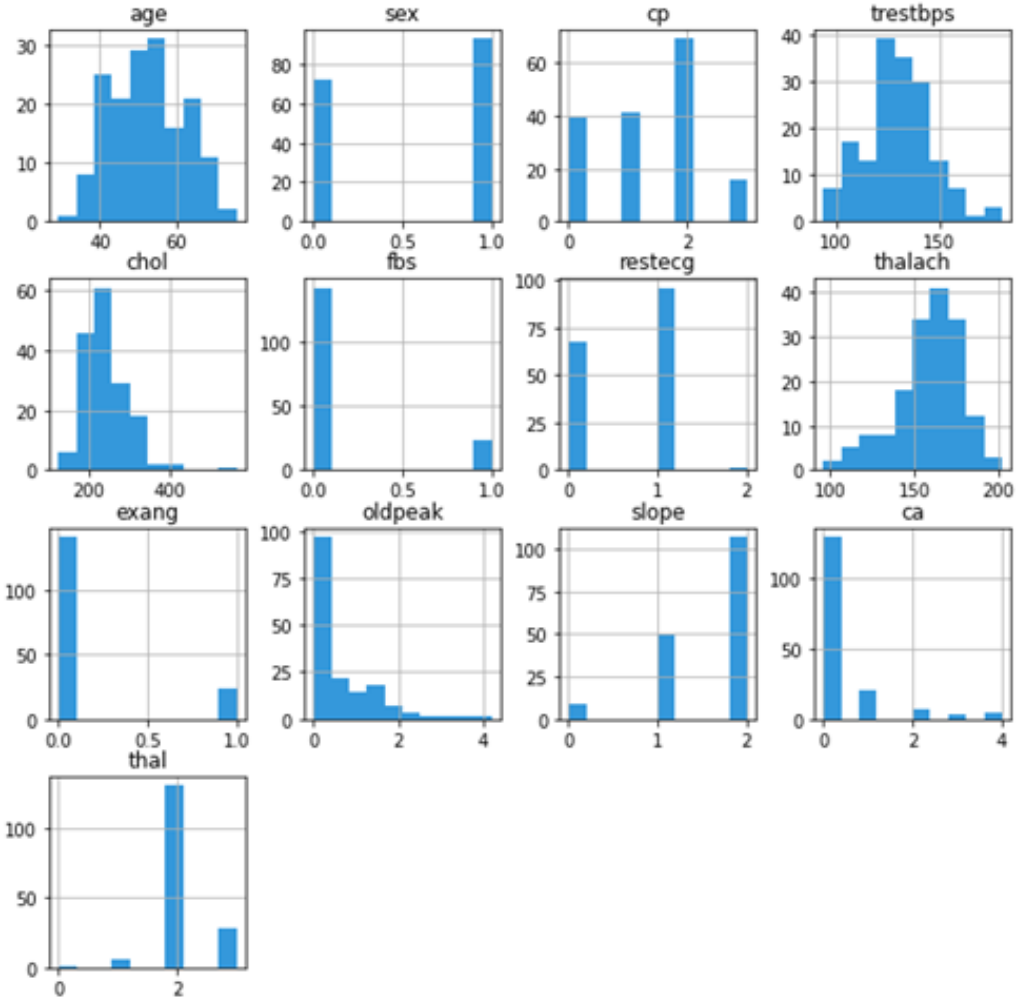
</td>
<td>  </td>
</tr>
</table>

表二、根據熱力圖結果，繪製與心臟病有高度相關的特徵值所占比率

對於連續型資料欄位我們利用 pandas 的 groupby()視覺化工具幫助查看各欄位在有無心臟病時的資料分佈情形。



圖二、無心臟病(target = 0)與各變數間的資料分佈圖



圖三、有心臟病(target = 1)與各變數間的資料分佈圖

Encoding 離散型欄位

```
df = pd.get_dummies(df, columns=col_dis, drop_first=True)
```

資料分析小結

以 panda 所提供的 profiling 套件總結資料分析結果並以網頁方式呈現。

```
pip install pandas-profiling
```

```
import pandas_profiling as pp
pro = pp.ProfileReport(df)
```

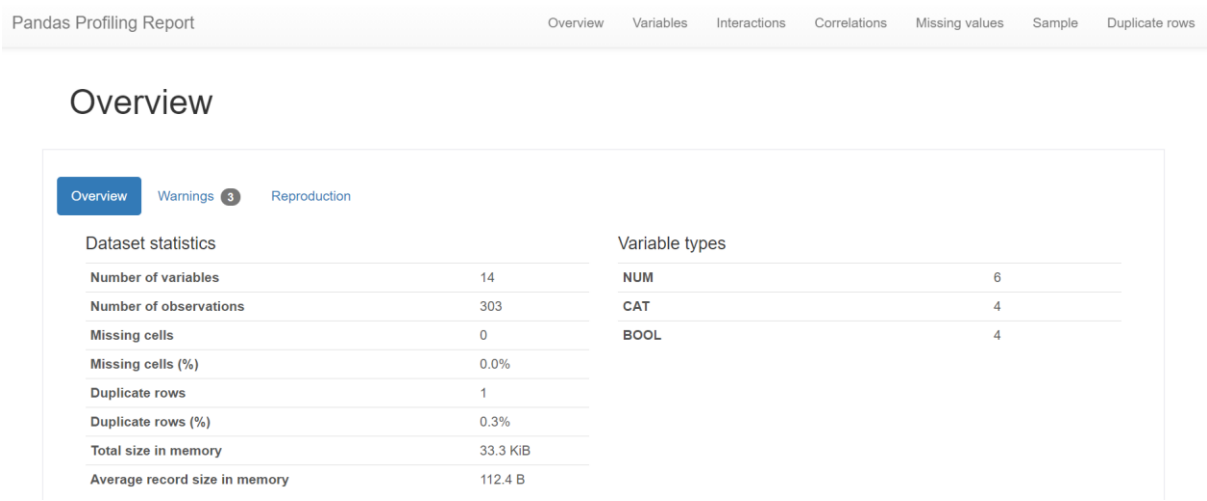
```
pro.to_file('iie_group1_output.html')
```

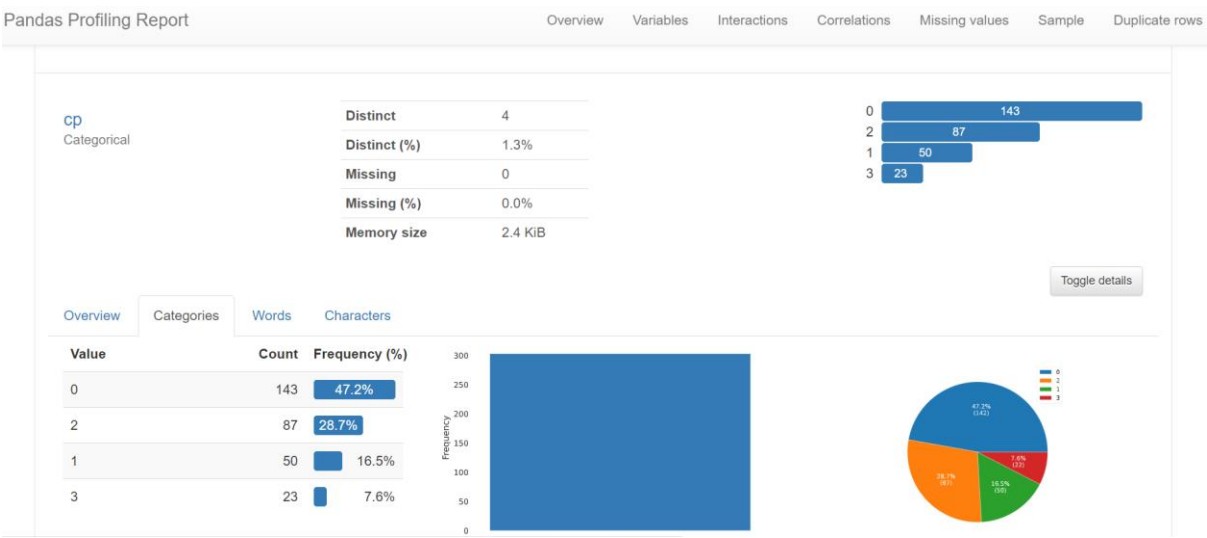
Summarize dataset: 100% 28/28 [00:09<00:00, 2.94it/s, Completed]

Generate report structure: 100% 1/1 [00:06<00:00, 6.02s/it]

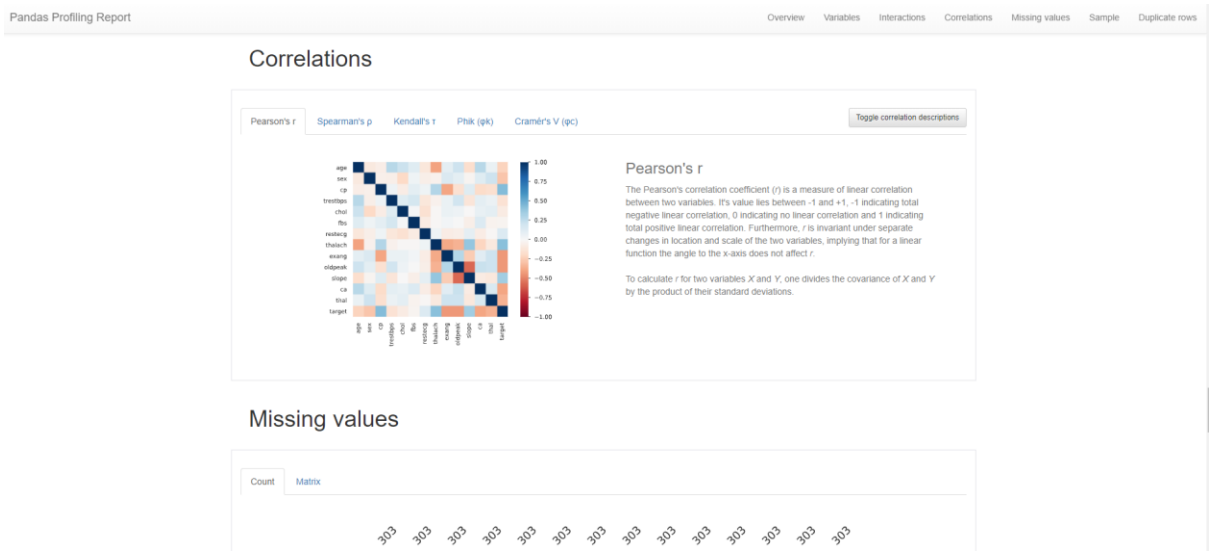
Render HTML: 100% 1/1 [00:01<00:00, 1.23s/it]

Export report to file: 100% 1/1 [00:00<00:00, 17.93it/s]





Profiling 套件還提供個特徵值間的交互作用與熱力圖。



經過一連串資料分析與視覺化後，繼續根據其特性進行標準化與資料編碼等資料前處理步驟。

標準化資料集

以 sklearn 的資料前處理提供的 StandartScaler 進行資料標準化。 為了避免後續模型訓練時特徵值大的資料欄位影響其他特徵值，將所有資料標準化；使其變異數為 0，標準差為 1。

```
from sklearn.preprocessing import StandardScaler
df_std = pd.DataFrame(df, columns=df.columns)
X_std = df_std.drop(['target'], axis=1)
y_std = df_std['target']
df_std = StandardScaler().fit_transform(df_std)
```

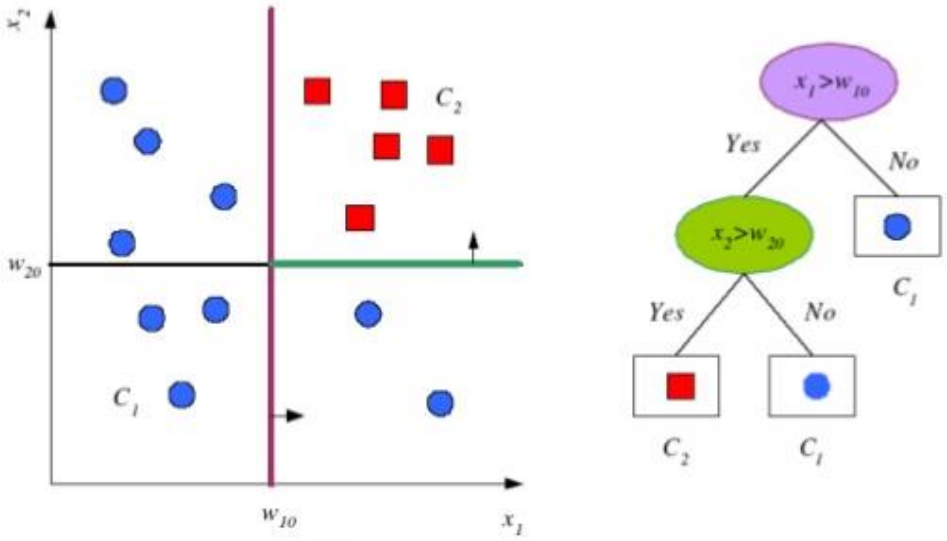
資料樣本區分

以 sklearn 為資料選擇所提供的 train_test_split()將資料分為特徵值與目標；特徵值為模型中的 x 值，目標值為模型中的 y 值。標準化後，將資料分為訓練集與測試集，統一後續模型使用之訓練與測試資料。

```
# define feature & target
X = df.loc[:, df.columns != 'target']
y = df['target']
# Spliting dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10)
```

第 3 章預測模型設定

決策樹演算法



圖四、決策樹演算法是意圖

首先選用監督式學習的決策樹原因為決策樹模型的訓練，不需要冗長的資料前處理例如標準化等步驟，所以較易實現與訓練。基於其可解釋性非領域專業人員也能夠從輸出結果的樹圖解讀模型預測結果，是可解釋性較高的模型。模型預測時在內部節點用某一特徵值自動進行 if – then – else 判斷，根據判斷結果決定進入哪個分支

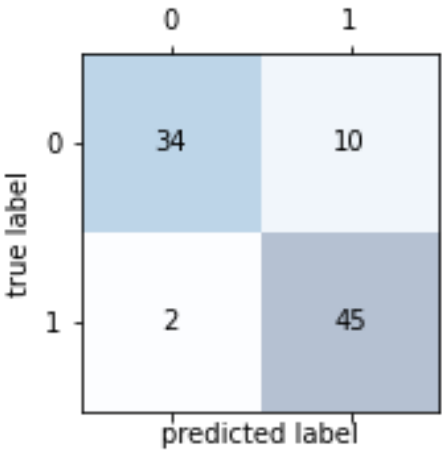
節點，直到到達葉節點處，得到分類結果。我們利用 分枝準則選擇墒(entropy) 、樹深度(max_depth)為 3 與 random_state 為 101 來訓練模型。

```
# 用 X_train,Y_train 訓練決策樹模型
clf_dt = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=101).fit(X_train,y_train)
print('決策樹 BEFORE Standardization', '\n')
# 計算模型的準確度
print('Accuracy on trianing set:  {:.2f}%'.format(clf_dt.score(X_train, y_train)*100))

# 用訓練出來的模型，對 X_test 進行預測
clf_dt.predict(X_test)

# 計算 X_test 預測結果的準確率
print('Accuracy on testing set: {:.2f}% '.format(clf_dt.score(X_test,y_test)*100))
```

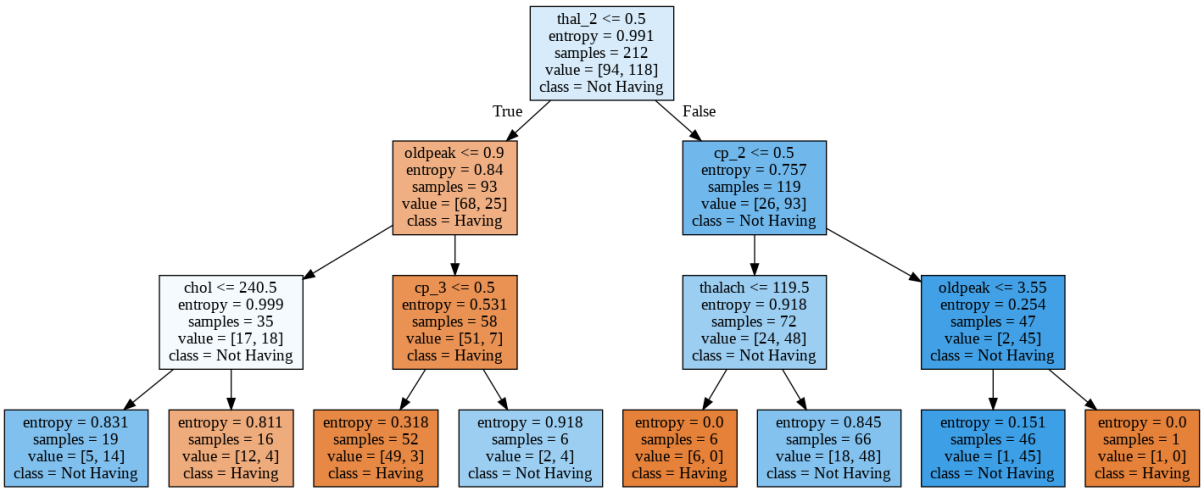
```
決策樹 BEFORE Standardization

Accuracy on trianing set:  84.43%
Accuracy on testing set: 82.42%
Scores of 5-fold cross validation:  [0.81967213 0.81967213 0.70491803 0.7        0.8       ]
Average score from 5-fold cross validation: 76.89%
Confusion Matrix:
 [[33 11]
 [ 5 42]]
```

```
決策樹 AFTER Standardization

Accuracy on trianing set: 84.43%
Accuracy on testing set: 82.42%
Scores of 5-fold cross validation:  [0.81967213 0.81967213 0.70491803 0.7        0.8       ]
Average accuracy from 5-fold cross validation: 76.89%
Confusion Matrix:
 [[33 11]
 [ 5 42]]
```

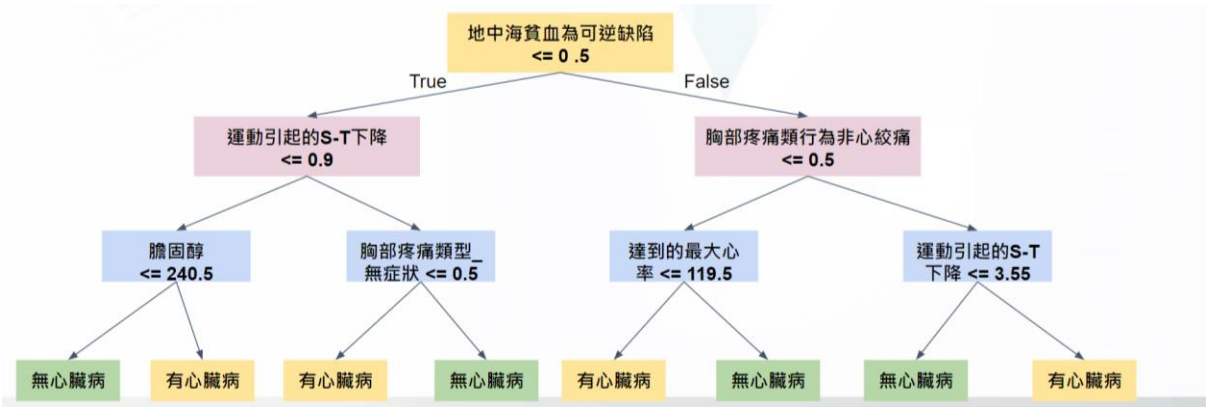
決策樹模型在訓練集準確率為 84.43%，在測試集準確率為 82.42%。經過 5-fold cross validation，可以得到決策樹分類模型平均分數為 76.89%。這可以證明模型泛化能力(Generalization)，也就是對沒有見過的資料集可以有 76.89% 準確率的預測結果，據混淆矩陣可以看出有 11 位沒有心臟病的人被確診有心臟病，知道決策樹對沒有心臟病的分析能力較低；但對有心臟病的分析僅有 5 位有心臟病的被預測出沒有心臟病。我們發現決策樹在標準化前後的模型訓練與測試效度都相同，符合決策樹的定義(不要求資料標準化)。



圖五、決策樹混淆矩陣



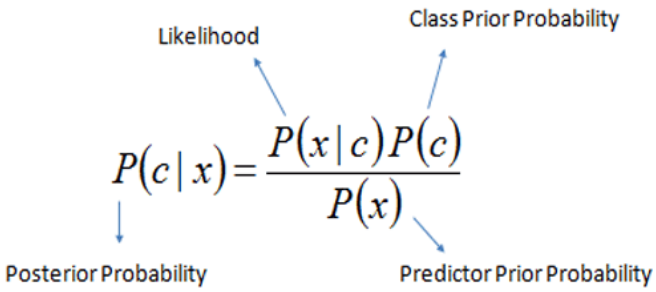
圖六、決策樹模型輸出結果



圖七、決策樹圖形文字敘述

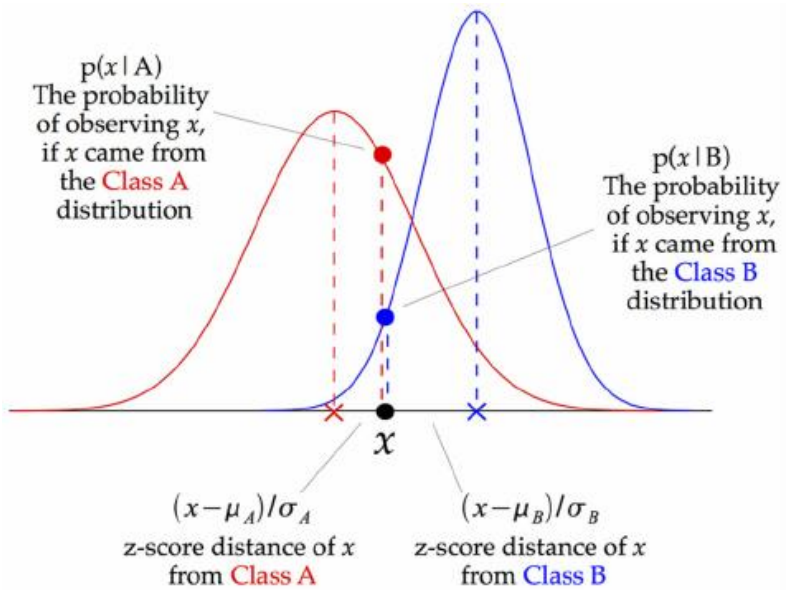
高斯貝氏分類器

貝氏定理說明在已知條件下，某件事情發生的機率。我們已知資料提供了 13 欄特徵值，所以貝氏定理可以訓練模型如何在這些已知的條件下更準確的預測有無心臟病（目標值）。貝氏定理還能在較少的資料下訓練模型，且資料的連續與離散型類別與資料間關聯性對模型訓練影響不大。



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

因分析的資料中有連續與離散型資料，所以我們選擇高斯貝式分類器來做為第二個訓練模型。高斯貝式分類器藉由假設連續型資料為常態分配的情況下，利用樣本的資料的標準差及變異數來計算機率。



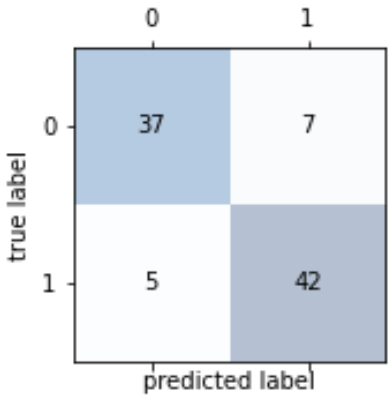
圖八、貝氏定理分類示意圖

```python
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb.fit(X_train, y_train)
print('貝氏 BEFORE Standardization', '\n')
print('Accuracy on trianing set: {:.2f}%'.format(nb.score(X_train, y_train)*100))

acc = nb.score(X_test, y_test)*100
print("Accuracy on testing set: {:.2f}%".format(acc))
```

```
貝氏 BEFORE Standardization

Accuracy on trianing set: 82.55%
Accuracy on testing set: 86.81%
Scores of 5-fold cross validation:  [0.80327869 0.8852459  0.80327869 0.76666667 0.73333333]
Average accuracy from 5-fold cross validation: 79.84%
Confusion Matrix:
 [[37  7]
 [ 5 42]]
```

```
貝氏 AFTER Standardization

Accuracy on trianing set: 82.55%
Accuracy on testing set: 86.81%
Scores of 5-fold cross validation:  [0.80327869 0.8852459  0.80327869 0.76666667 0.73333333]
Average accuracy from 5-fold cross validation: 79.84%
Confusion Matrix:
 [[37  7]
 [ 5 42]]
```

高斯貝氏分類模型在訓練集的準確率為 82.55%，在測試集的準確率為 86.81%決。經過 5-fold cross validation，可以得到高斯貝氏分類模型平均分數為 79.84%，稍微高於決策樹平均結果。同樣可以證明模型泛化能力(Generalization)，對沒有見過的資料集可以有 79.84% 準確率的預測結果，據混淆矩陣可以看出有 7 位沒有心臟病的人被確診有心臟病，知道決策樹對沒有心臟病的分析能力較低；但對有心臟病的分析僅有 5 位有心臟病的被預測出沒有心臟病，結果與決策樹相同。



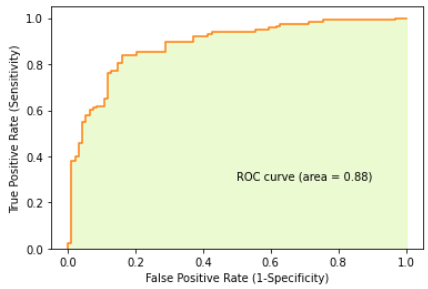
圖九、高斯貝氏混淆矩陣

高斯貝氏 ROC 視覺化

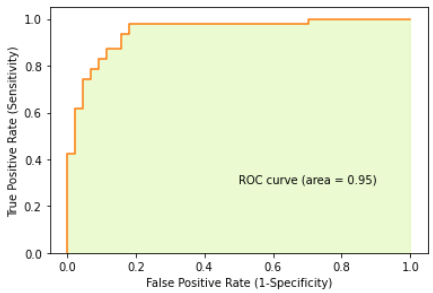
```
transformed = pd.DataFrame(StandardScaler().fit_transform(X_train))

pipeline = make_pipeline(StandardScaler(), GaussianNB())
pipeline.fit(X_train, y_train)
```

```
transformed = pd.DataFrame(StandardScaler().fit_transform(X_test))

pipeline = make_pipeline(StandardScaler(), GaussianNB())
pipeline.fit(X_test, y_test)
```

Receiver Operating Characteristic Plot for training



Receiver Operating Characteristic Plot for testing



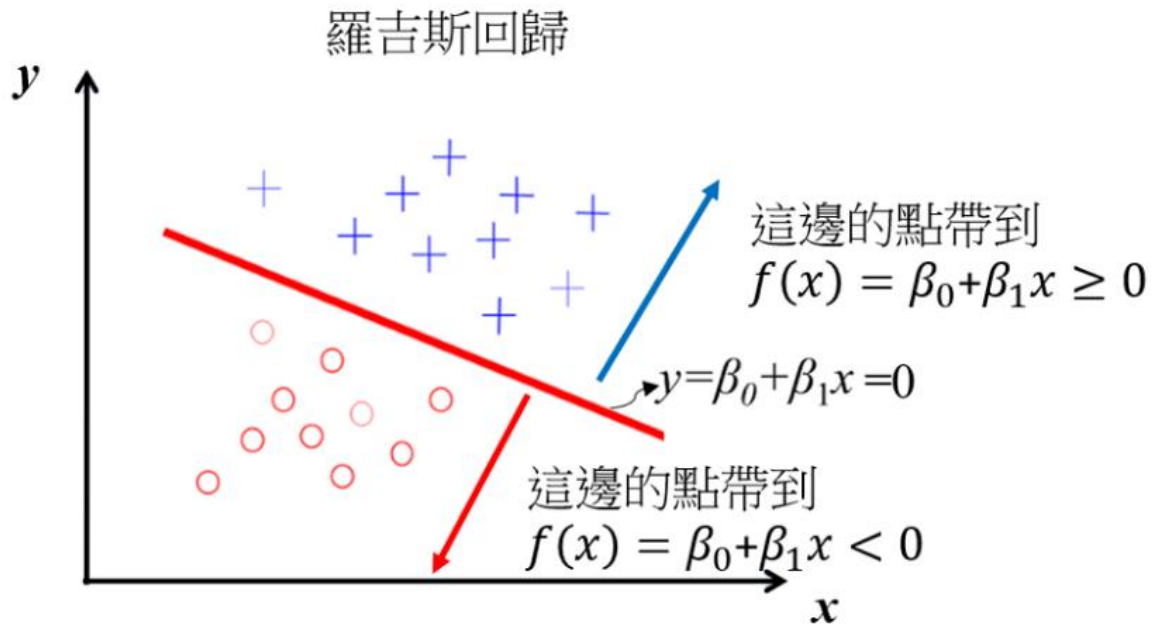
高斯貝氏模型的測試集具有較好的準確率所以進一步查看其預測能力。從 ROC 圖可以看出因兩張圖型 AUC > 0.8，所以訓練與測試集都具有優良的鑑別力(excellent discrimination)。

羅吉斯回歸模型

羅吉斯回歸是用線性回歸的輸出來判斷這個資料屬不屬於 target(二分類問題: target 和 non-target)，或是多分類問題。羅吉斯回歸的做法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)。

$$\sigma\left(f(x)\right)=\begin{cases}0 & f(x)<0\\ 1 & f(x)\geq 0\end{cases}$$

但越靠近 0 的位置，理論上越容易分類錯誤，所以加上一個對數函數，使輸出更有彈性，羅吉斯回歸所用到的對數函數是 Sigmoid 函數。



圖十、羅吉斯回歸理論示意圖

$$\sigma\left(f(x)\right)=\frac{1}{1+e^{-f(x)}}=\frac{e^{f(x)}}{1+e^{f(x)}}$$

Sigmoid 函數

羅吉斯回歸建模

使用 sklearn.linear_model 中的 Logistic Regression，超參數使用 C、class_weight、max_iter、solver、tol。

- C：C 參數與懲罰系數成反比，C 值越小，則正規化效果越強
- class_weight **權重參數** ：調節樣本比例
- solver **優化算法** ： liblinear ,sag , lbfgs , newton-cg
- max_iter ： 迭代次數
- tol：残差收斂條件，即迭代的連續兩次之間残差小于 tol 就停止

```
from  sklearn.linear_model    import  LogisticRegression
lr=LogisticRegression(    C=50.0,
                          class_weight='balanced',
                          max_iter=100,
                          solver='liblinear',
                          tol=0.01   )
```

參數值	優化方式
liblinear	坐標軸下降來迭代，優化損失函數
lbfgs	利用損失函數二階導數矩陣來迭代，優化損失函數
newton-cg	lbfgs 的另一種方法，也是利用損失函數二階導數矩陣來迭代
sag	隨機平均梯度下降，每次迭代僅用一部份樣本計算梯度適合大樣本

表三、solver 優化算法

羅吉斯回歸超參數調整

調整參數有 c、max_ iter、solver、 tol

超參數調整

超參數 C	結果	決策
c = 1	0.769	
c = 50	0.791	✔
c = 100	0.791	

超參數 max_ iter	結果	決策
100	0.791	✔
500	0.791	
1000	0.791	
50	0.791	

超參數 solver	結果	決策
liblinear	0.791	✔
sag	0.782	
lbfgs	0.791	✔
newton-cg	0.773	

超參數 tol	結果	決策
0.0001	0.791	✔
0.001	0.791	
0.01	0.780	
0.1	0.769	

最終選擇 c =50、max_iter =100、solver =liblinear、tol =0.0001 準確率 0.791。然後這樣調整參數有一個問題，就是會忽略參數組合性，所以新增一種 GridSearchCV 的方法，此方法可以把想要調整的參數做組合，透過各種組合搭配找出好的參數組合，最後找出準確率為 0.791。而 GridSearchCV 和手動調整參數雖然準確率都是 0.791，但 GridSearchCV 的 FN 比較少，不會讓原本有心臟病的人誤以為沒有，而錯過急救黃金時間，所以這方面我們覺得 GridSearchCV 的 model 比較好。

超參數調整

```
[204] from sklearn.model_selection import GridSearchCV
      import warnings
      warnings.filterwarnings("ignore")

      # 參数设置
      params = {'C':[   1,  50,  100],
                'max_iter':[50,  100,500,  1000],
                'class_weight':['balanced',  None],
                'tol':[0.0001,0.001,0.01,0.1],
                'solver':['liblinear','sag','lbfgs','newton-cg']
               }
      lr = LogisticRegression()
      clf = GridSearchCV(lr, param_grid=params, cv=10)
      clf.fit(x_train_nor,y_train)
```

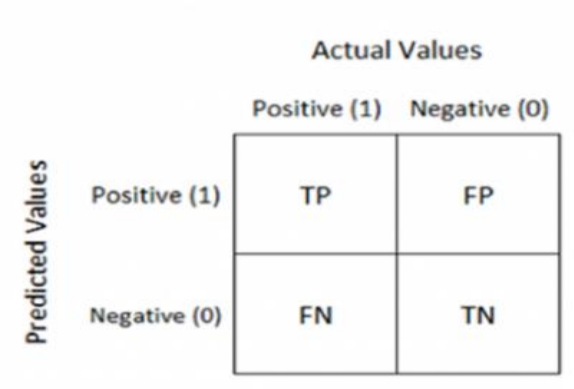
超參數	數值	結果	決策
c	1	0.791	✔
max_iter	100		
tol	0.0001		
solver	lbfgs		

```
train_accuracy: 0.8113207547169812

test_accuracy: 0.7912087912087912

[[-0.31948254 -0.42793986  0.85542465  0.164883    0.10222755 -0.87524607]]
[0.24768291]
```

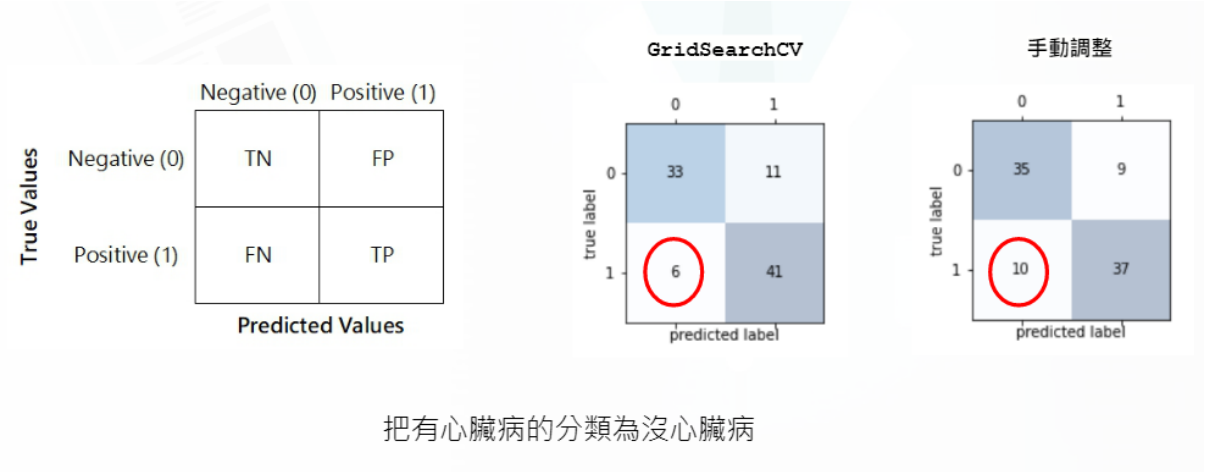
混淆矩陣模型評估

評估訓練出來的模型成效好不好，我們使用混淆矩陣，分辨模型在分類上的準確率，有四個分類指標。

- True Positive (TP)「真陽性」:真實情況是「有」，模型說「有」的個數。
- True Negative(TN)「真陰性」:真實情況是「沒有」，模型說「沒有」的個數。
- False Positive (FP)「偽陽性」:真實情況是「沒有」，模型說「有」的個數。
- False Negative(FN)「偽陰性」:真實情況是「有」，模型說「沒有」的個數。



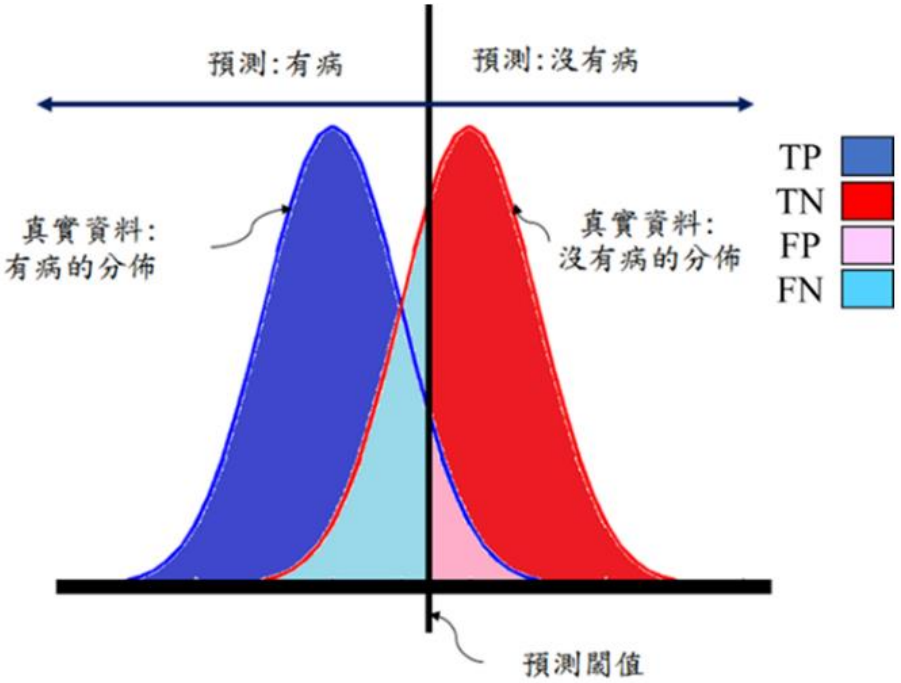
而羅吉斯模型的混淆矩陣 TP =41　TN =33　FP =11　FN = 6，有 11 個人沒有心臟病卻被分類為有心臟病，這點是我們必須做調整的，但如果調整 FP 時要注意會不會讓原本有心臟病的人反而沒有被預測出來。



圖十一、羅吉斯回歸混淆矩陣調參數後比較

ROC/AUC 模型評估

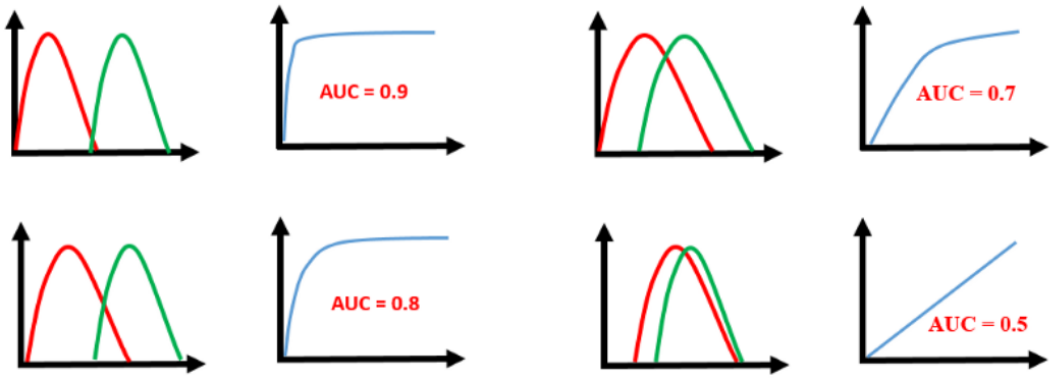
使用得到的混淆矩陣去計算 ROC/AUC，ROC 曲線用來評估二元分類問題的一個方法，會設定閾值(threshold)來控制 FP 和 FN，FP 和 FN 變化基本上靈敏性(sensitivity)和特異性(specificity)也是跟著變，且 FP 和 FN 說的是個數，靈敏性和特異性是百分比的指標。而曲線下的面積(Area Under Curve; AUC)來判別 ROC 曲線的鑑別力。而我們的羅吉斯回歸模型 AUC 為 0.82，效果不錯。



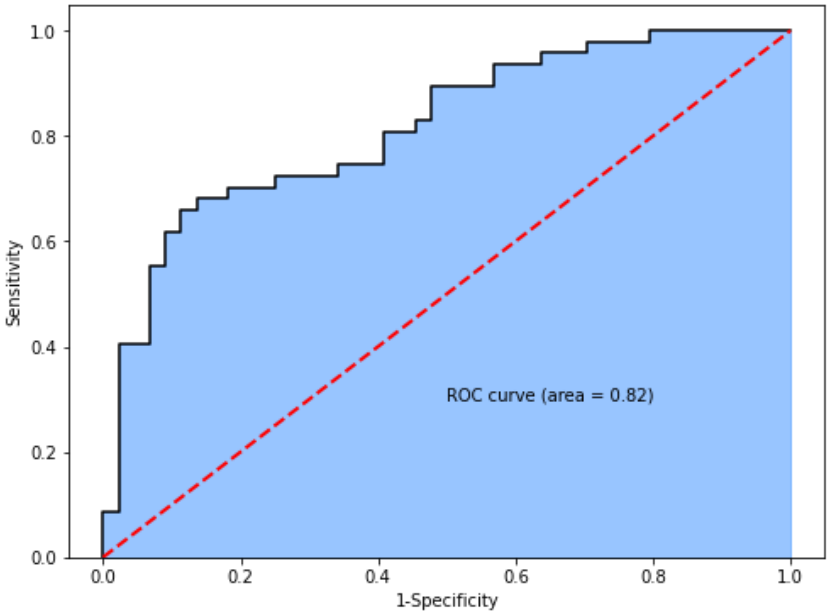
圖十二、FN、FP、threshold 關係圖

$$\text{Recall} = \text{or Sensitivity} = \frac{TP}{TP + FN} \quad \text{(真陽率)}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \quad \text{(假陽率)}$$



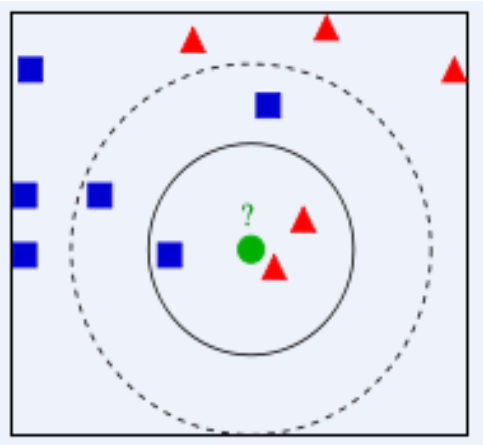
圖十三、各 AUC 比較圖



圖十四、羅吉斯回歸 AUC = 0.82

KNN (K Nearest Neighbor)

KNN 是一種用於分類的無母數統計方法。 K 代表參數計算的鄰近單元數量，為一個沒有類別標籤的向量，會被歸類為最接近該點的 k 個樣本點中最頻繁使用的一類。藉由改變 KNN 的 neighbor 數(k)來訓練模型，找出在不同 k 時，對於模型準確度的影響並找出最佳值。其優點為: KNN 是所有機器學習演算法最簡單的一種，無論是分類還是回歸，衡量鄰居的權重都非常有用。



由上圖說明，當模型要對測試樣本(綠點)做預測分類時，若 k=3，該樣本就被分類為紅色類別；當 k=5 時，測試樣本就會被分類為藍色類別。所以 k 值的選擇會影響演算法的結果。

KNN Model

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=101)
```

```
from sklearn.neighbors import KNeighborsClassifier
def knnprediction(k, x_train, y_train, x_test):
        knn = KNeighborsClassifier(n_neighbors = k)
        knn.fit(x_train, y_train)
        pred = knn.predict(x_test)
        return pred

        #        f1 = result['1']['f1-score']
#performance_data= {'f1-score':round(f1, 2),'accuracy':round(accuracy, 2)}

knn_pred = knnprediction(12, x_train, y_train, x_test)
```

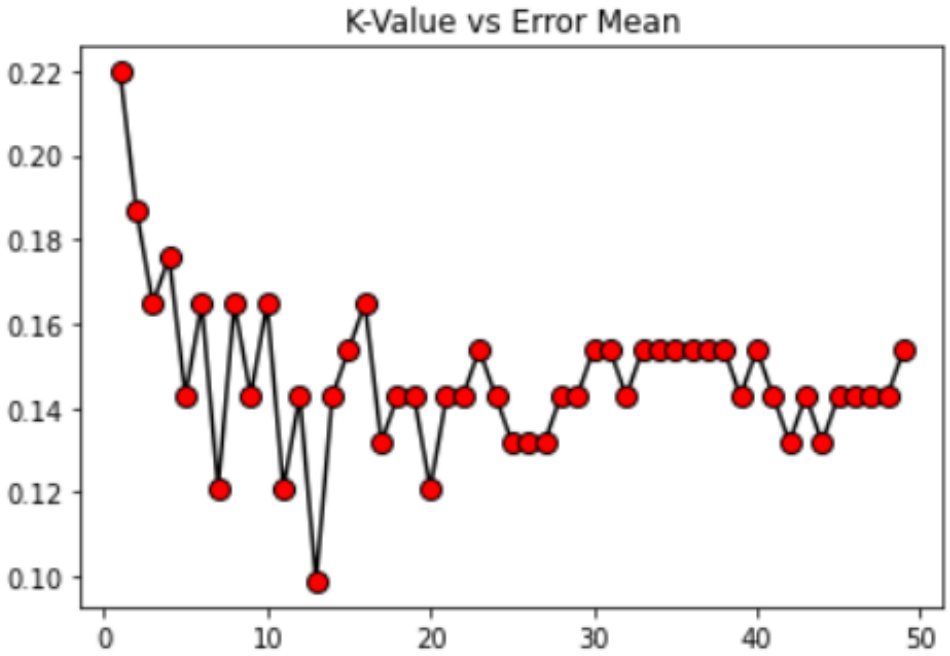
建立 KNN 模型必須定義鄰居數 k，我們定義的 k 值大小會決定用來進行分類的範圍，選擇較大的 k 值時，進行預測的鄰域較大，可以減少學習的估計誤差，但是學習的近似誤差會增大;反之，選擇較小的 k，進行預測的鄰域較小，其優點是學習誤差會減小，但是估計誤差會增大。故我們要選擇一個最佳的 k 值，使其測試單元組到 k 個單元的距離為最小。

調整參數

```
error_mean = []
for k in range(1, 50):
        pred_i = knnprediction(k, x_train, y_train, x_test)
        error_mean.append(np.mean(pred_i != y_test))

error_mean

min(error_mean)

for i in range(0,49):
    if error_mean[i] == min(error_mean):
        print(i)

print(error_mean[12])
```

利用 for 迴圈找出最佳的 k 值，使 error_mean 為最小，也就是說，我們愈分類的樣本點跟訊聯及樣本點的平均距離為最小。

```
plt.plot(range(1,50), error_mean, color='black', marker='o', markerfacecolor='r', markersize=8)
plt.title('K-Value vs Error Mean')
plt.show()
```



圖十五、K 值與 error mean 比較

其中可以看出 k=12 的時候其 Error Mean 為最小，故 k 我們設定為 12 讓模型進行預測、分類。

評估 KNN 的準確率

```
from sklearn.metrics import classification_report
def evaluate(prediction,y_test):
        result = classification_report(y_test,prediction,output_dict=True)
        accuracy = result['accuracy']
        performance_data= {'accuracy':(accuracy)}
        return performance_data

knn_pr = evaluate(knn_pred,y_test)
print(knn_pr)
```

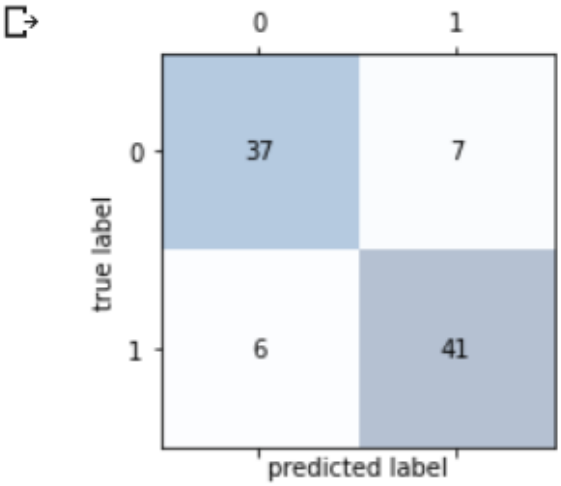
```
{'accuracy': 0.8571428571428571}
```

```
print(knn.score(x_train,y_train))
```

```
0.8443396226415094
```

混淆矩陣

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, knn_pred)
fig, ax = plt.subplots(figsize=(3, 3))
ax.matshow(cm, cmap=plt.cm.Blues, alpha=0.3)
for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
                ax.text(x=j, y=i, s=cm[i,j], va='center', ha='center')
plt.xlabel('predicted label')
plt.ylabel('true label')
plt.show()
```



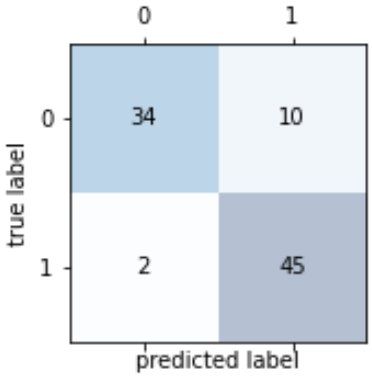
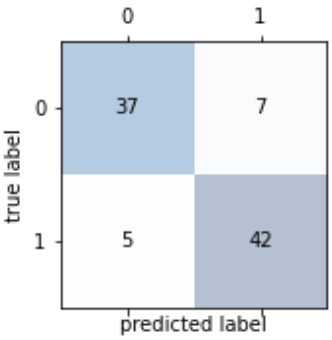
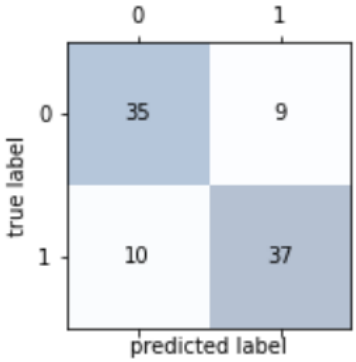
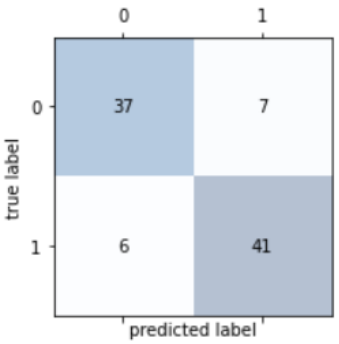
圖十六、KNN 混淆矩陣

KNN 模型的混淆矩陣為 TP =41, TN =37, FP =7 FN = 6，其中 7 筆資料是沒有心臟病卻被分類為有心臟病，6 筆資料為有心臟病卻預測沒有心臟病，我們必須針對 FP、FN 作探討及調整。

第 4 章模型效度驗證

混淆矩陣

Model	混淆矩陣 Confusion Matrix

Decision Tree	
Gaussian Bayes	
Logistic Regression	
KNN	

表四、個模型混淆矩陣比較

模型於訓練及與測試及效度比較

Model	train_accuracy	test_accuracy
Decision Tree	0.826	0.868
Gaussian Bayes	0.844	0.824
Logistic Regression	0.811	0.791
KNN	0.844	0.857

表五、模型效度比較

第 5 章結論與未來展望

本篇研究建立 4 種預測模型，針對心臟病的身理特徵做分類。根據我們預測模型的準確率來看，4 種模型都具有泛化能力，另外也發現決策樹在訓練資料集的表現上是最好的，若未來能夠將此模型應用在醫院中，輔助醫師判斷罹患心臟病的重要可能原因，除了減少誤判的可能和提升辨識效率。而未來我們也想透過這類型的模型分析，應用在其他疾病，透過熱力圖找出良好的特徵值，刪去不必要的因子，提升其他疾病預測的準確率和提早發現潛在危險因子。

未來可在院方的網站或 APP 設置心臟病的查詢功能，讓民眾能自行輸入個人的資料後，透過預測模型判斷是否罹患心臟病，若有再進一步到醫院進行診斷及治療，如此一來就能減少醫療的浪費，民眾也能初步的掌握自己的健康狀況。

第 6 章參考資料

https://www.kaggle.com/ronitf/heart-disease-uci
https://www.kaggle.com/neisha/heart-disease-prediction-using-logistic-regression
https://heartbeat.fritz.ai/understanding-the-mathematics-behind-naive-bayes-ab6ee85f50d0
https://www.kaggle.com/cdabakoglu/heart-disease-classifications-machine-learning
https://pyecontech.com/2020/02/27/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92%E9%A6%96%E9%83%A8%E6%9B%B2-%E8%B2%9D%E6%B0%8F%E5%88%86%E9%A1%9E%E5%99%A8-bayesian-classifier