智慧化企業整合

期中報告

以機器學習模型預測新冠肺炎病人一個月內之死亡率

第五組

110034551 高藝真

110030517 胡詠晴

110034544 賴筱庭

110034545 張芳綺

目錄

壹、 問題定義 3
一、 研究背景 3
二、 研究動機 3
三、 5W1H 3
四、 分析架構 4
貳、 資料預處理 5
一、 資料集描述 5
二、 資料欄位描述 5
三、 資料預處理 5
i. 資料整合 5
ii. 資料清理 6
iii. 空值處理 6
iv. 資料轉換 6
v. 不平衡資料的處理 6
參、 模型建構 7
一、 模型介紹與比較 7
i. 羅吉斯迴歸（Logistic Regression，LR） 7
ii. 支援向量機（Support Vector Machine，SVM） 8
iii. 隨機森林（Random Forest，RF） 9
二、 模型建構與參數調整 10
i. 羅吉斯迴歸（Logistic Regression，LR） 10
ii. 支援向量機（Support Vector Machine，SVM） 12
iii. 隨機森林（Random Forest，RF） 14
iv. 各模型重要特徵之比較 16
肆、 結果驗證 17
一、 混淆矩陣 17
二、 交叉驗證 17
伍、 研究結論 18
陸、 參考資料 18

壹、 問題定義

一、 研究背景

因應新冠肺炎(COVID-19)疫情日益嚴重，隨著確診者不斷增加，醫療資源的緊繃導致大眾對於醫療資源的分配愈發的重視。現今醫療體系資訊ｅ化，醫療資料量急劇增加，其中隱含極高之資訊價值，但相關研究的數量卻不多。且數據間具較高的相依關係，傳統的統計方法難以有效地擷取出有用的資訊，因此，我們藉由大數據分析的手法，挖掘出其中有價值的訊息。

二、 研究動機

在全球新冠肺炎(COVID-19)疫情不斷蔓延之下，全球各地學者都發揮各自專業，針對新冠肺炎(COVID-19)的議題進行研究，顯示全球專家學者對於新冠肺炎(COVID-19)的重視。新冠肺炎病毒的感染症狀多樣，院方難以憑病人的症狀診斷其死亡機率。因此，在醫療資源不足的前提下，我們希望能透過所學，利用巨量資料，以有效掌握更多資訊，並設計方法應對變異速度極快的病毒，提供院方更具有統計依據的方法，讓其進行呼吸器及病床等醫療資源的分配。

三、 5W1H

我們利用 5W1H 來定義問題：

表 1　5W1H 問題定義表

When	當呼吸器數量不足時
Where	醫院、科研中心
Who	醫療人員、科研中心分析人員
What	解決大量病理數據資料的分析
Why	使醫療資源能更有效的分配
How	資料分析、 機器學習(LR、SVM、RF)

四、 分析架構

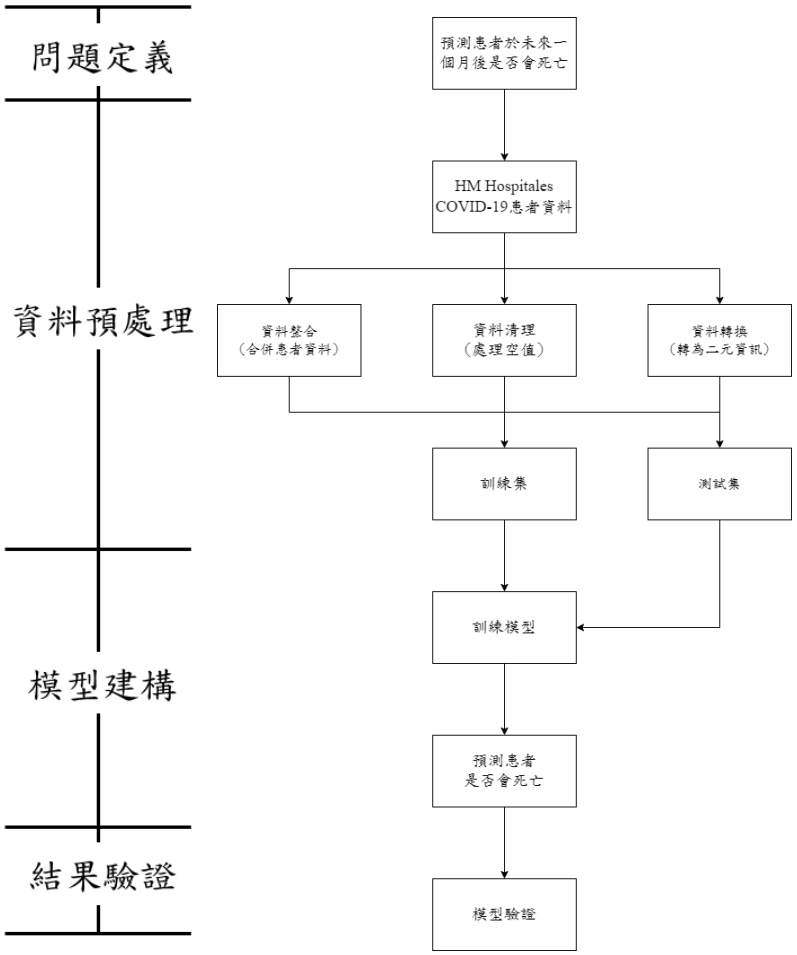

圖 1 分析架構圖

貳、 資料預處理

一、 資料集描述

本次分析預測所使用的資料為某間位於西班牙的醫療機構（HM Hospitales）所提供之公開資料，包含 2020 年 2 月至 6 月 COVID-19 患者的相關病理數據資料，描述確診患者各項檢查數據及死亡與否，共 1834 筆數據，48 個特徵項目，其中共有 18 個類別型欄位以及 32 個連續型欄位。而預測的目標（label）為死亡與否，若是 0 則代表在一個月內此病人仍然存活，若是 1 則代表此病人在一個月內死亡。

二、 資料欄位描述

表 2 資料欄位對照表

age	age on admission	**lab_sodium**	serum sodium
sex	sex	**lab_leukocyte**	leukocyte count
admission_datetime	date of admission	**lab_mean_platelet_volume**	mean platelet volume
ed_diagnosis	primary symptom	**lab_neutrophil**	neutrophil count
mechvent_fla	indicator for mechanical ventilation	**lab_alt**	serum alanine aminostransferase
vitals_temp_ed_first	body temperature at ED triage	**lab_ddimer**	serum d-dimer
vitals_sbp_ed_first	systolic blood pressure at ED triage	**lab_inr**	international normalized ratio
vitals_dbp_ed_first	diastolic blood pressure at ED triage		
vitals_hr_ed_first	heart rate at ED triage	**lab_mch**	serum mean corpuscular hemoglobin
vitals_spo2_ed_first	SpO2 at ED triage	**lab_creatinine**	serum creatinine
pmhx_diabetes	indicator for comorbidity: diabetes	**lab_mcv**	mean corpuscular volume
pmhx_hld	indicator for comorbidity: hyperlipidemia		
pmhx_htn	indicator for comorbidity: hypertension	**lab_aptt**	activated partial thromboplastin time
pmhx_ihd	indicator for comorbidity: ischemic heart disease	**lab_platelet**	platelet count
pmhx_ckd	indicator for comorbidity: chronic kidney disease	**lab_lymphocyte_percentage**	lymphocyte percentage
pmhx_copd	indicator for comorbidity: chronic obstructive pulmonary disease	**lab_glucose**	serum glucose
		lab_neutrophil_percentage	neutrophil percentage
pmhx_asthma	indicator for comorbidity: asthma	**lab_ldh**	lactate dehydrogenase
		lab_prothrombin_activity	prothrombin activity
pmhx_activecancer	indicator for comorbidity: cancer	**lab_urea**	serum urea
		lab_lymphocyte	lymphocyte count
pmhx_chronicliver	indicator for comorbidity: chronic liver disease	**lab_crp**	serum c-reative protein
		lab_rdw	red blood cell distribution width
pmhx_stroke	indicator for comorbidity: stroke	**lab_hemoglobin**	serum hemoglobin
		lab_rbc	red blood cell count
pmhx_chf	indicator for comorbidity: chronic heart failure	**lab_hct**	serum hematocrit
		lab_potassium	serum potassium
pmhx_dementia	indicator for comorbidity: dementia	**lab_ast**	serum aspartate aminotransferase

三、 資料預處理

在建模前需要先進行資料的預處理，我們依序進行了資料整合、資料清理、空值處理、資料轉換以及不平衡資料的處理。

i. 資料整合

因為此份資料的特徵值以及 label 值分屬兩個不同的檔案，因此我們依據患者的 ID 將此兩份資料合併，前 48 欄為病人的特徵值，最後一欄為病人的 label 值，即死亡與否。

ii. 資料清理

因為病人編號及時間非病理數據資料，為模型不需要使用到的資料，因此將 patient ID，admission_datetime 這兩欄無效欄位清除。

iii. 空值處理

此份資料有少許空值的情形，因此我們將類別型資料以眾數的方式補空值，連續型資料以中位數的方式來補空值。

iv. 資料轉換

此資料包含以文字敘述的類別型欄位（性別、症狀表現），需要轉換為二元資訊。

我們將性別欄位（兩種類別）轉換為 0 或 1，而症狀表現（五種類別）因為沒有大小順序之分，如果轉換成數字 0~4 會有失公正，因此我們採用 one hot encoding 的方式將每種症狀拆成一個獨立的特徵，以 1 或 0 的方式表示有或無。

v. 不平衡資料的處理

由於此份資料患者的存活人數較死亡人數多出許多，存活與死亡的比例將近 9：1，屬於不平衡資料集，為了避免預測失準，我們採用 Random Under Sampling 的方法隨機刪除 900 筆存活狀態為 0（活著）的數據，使存活與死亡的數據比例接近 4：1。

刪除 900 筆數據後剩下 934 筆資料，我們以 8：2 分割訓練集與測試集，訓練集共 747 筆，測試集共 187 筆。

參、 模型建構

一、 模型介紹與比較

i. 羅吉斯迴歸（Logistic Regression，LR）

羅吉斯迴歸的概念類似於線性迴歸（Linear Regression）分析，主要在探討依變數（Y）與自變數（X）之間的關係。線性迴歸中的依變數通常為連續型變數，但羅吉斯迴歸所探討的依變數主要為類別變數，特別是分成兩類的變數，例如：是或否、存在或不存在、死亡或存活……等，而自變數則沒有特別規定，可以是類別變數，也可以是連續變數。透過羅吉斯迴歸，我們預期可以找出類別型態的依變數和自變數之間的關係。

羅吉斯迴歸在運用上也需符合傳統迴歸分析的一般假設，也就是避免自變數之間共線性的問題，以及符合常態分配等的基本假設。因迴歸分析方法中，限制了依變數為連續型變數，若欲分析的變數非為連續型時則無法使用，此時可選擇羅吉斯迴歸來處理，並同時針對類別型變項計算勝算比（Odds Ratio）指標來判斷其對於依變數的影響強度。

勝算比公式如下：

$$\ln\frac{\mathrm{p}}{1-\mathrm{p}} = \mathrm{f(x)} = \beta_0 + \beta_1 X + \beta_2 X_2 + \beta_3 X_3 + \cdots + \beta_i X_i$$

對勝算比取對數後做線性迴歸，再經由 Sigmoid 函數的轉換將機率固定在 0 與 1 之間，因為函數嚴格遞增且中間斜率特別大之特性，有效將樣本事件的對應機率區分在趨近 0 或趨近 1 的位置，以達到二元分類的目的。

ii. 支援向量機（Support Vector Machine，SVM）

支援向量機是由 Vapnik 及其團隊所發展，近年來廣泛地被應用在二元分類的問題，例如疾病診斷、影像辨識等；其主要概念為，在一個高維度空間中，利用最小化統計風險尋找超平面（Hyper Plane），做為該二元資料的分類依據。超平面距離兩個類別的邊界（Margin）是最大值，而最靠近邊界的這些樣本點提供 SVM 最多的分類資訊，就叫做支持向量（Support Vector）。

能完全透過 SVM 分類數據的邊界稱為硬邊界（Hard-margin）；若無法完全分類數據的分類稱為軟邊界（Soft-margin），現實情況的資料通常都是以軟邊界為主。

超平面（Hyper Plane）

假設二元類別的資料可以找到一平面將資料一分為二，該平面就稱為「超平面」，在操作支援向量機的過程就是希望可以找出這個超平面，如圖所示，以二元資料分類問題為例，研究人員欲找出分割線將不同的資料分開，且該分割線距離此兩類之邊界越大越好，如此方可更清楚的判別資料屬於哪一個類別。

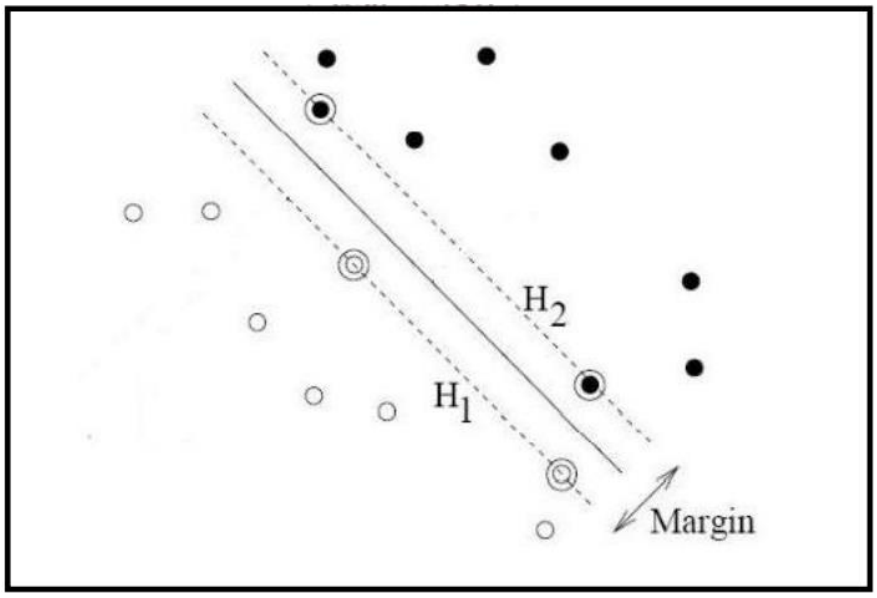

圖 2　超平面（Hyper Plane）示意圖

核心函數（Kernel Function）

當資料為線性時，可直接分割；若需要先將非線性資料進行分類，則可以利用核心函數改變其分布型態，作法是將輸入的資料由低維度經由核心函數轉換後投射到高維度中，原本無法分割的資料便可在高維度空間中進行分割。

iii. 隨機森林（Random Forest，RF）

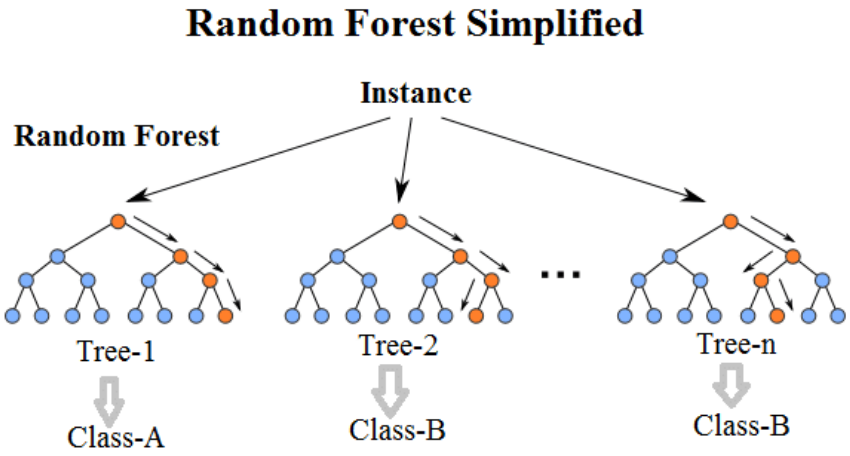

圖 3　隨機森林（Random Forest）示意圖

隨機森林的原理為結合多顆 CART（Classification and Regression tree）樹，並加入隨機分配的訓練資料，以大幅增進最終運算結果，並可以透過形成多棵具差異性的樹以進行 Ensemble Method，透過 Ensemble Method，結合多個「弱學習器」來建構一個「強學習器」，以產生不同的數據集，才可有多棵具差異性的 CART 樹。Ensemble Method 可使用方法如下：

1. Bagging（Bootstrap Aggregation）：
   「重新取樣原有數據用以產生新的數據，取樣的過程是均勻且可重複取樣」。使用 Bootstrap 即可從一組數據中生出多組數據集。此法會從訓練集中取出 K 個樣本，再從這 K 個樣本訓練出 K 個分類器（tree）。每次取出的 K 個樣本皆會再放回母體，因此這 K 個樣本之間會有部份資料重複，但因每顆樹的樣本還是不同，因此訓練出的分類器（tree）之間是還具有差異性，每個分類器的權重一致所以最後以投票方式（Majority vote）得到最終結果。
2. Boosting：
   與 Bagging 類似，但更強調對錯誤部份加強學習以提升整體的效率。是透過將舊分類器的錯誤資料權重提高，加重對錯誤部分的練習，訓練出新的分類器，這樣新的分類器就會學習到錯誤分類資料（misclassified data）的特性，進而提升分類結果。

隨機森林也可以看做將決策樹透過 Bagging 增強後的結果。

二、 模型建構與參數調整

接著，我們將針對三個模型分別進行模型建構以及參數調整。

i. 羅吉斯迴歸（Logistic Regression，LR）

模型建構：

我們首先使用他預設的參數建立了一個羅吉斯迴歸模型，並計算出 precision、recall、F1-score、accuracy，並且透過程式找出此模型的重要特徵。

```
Data_X = df_x[COD['all']]
Data_Y = df_y[Y]
Training_Set_X, Testing_Set_X, Training_Set_Y, Testing_Set_Y = train_test_split(Data_X, Data_Y, test_size = 0.2, shuffle = False)
sc=StandardScaler()
sc.fit(Training_Set_X)
Training_Set_X=sc.transform(Training_Set_X)
Testing_Set_X=sc.transform(Testing_Set_X)
model = LogisticRegression()
model.fit(Training_Set_X, Training_Set_Y)
pred_Y = model.predict(Testing_Set_X)
pred_train_Y = model.predict(Training_Set_X)# 預測訓練集 Training_Set_X
print(f'共{len(Data_Y.index)}筆: 訓練集{len(pred_train_Y)}, 測試集{len(pred_Y)}')
```

```
******Testing set*******
Precision: 0.7347
   Recall: 0.6667
       F1: 0.699
 Accuracy: 0.8342
```

圖 4 LR 程式碼截圖、以及各項效度之計算

```
#importance features
FI = {}
feat_importances = pd.Series(model.coef_.ravel(), index=COD['all'])
Testing_Result[Y] = re['pred_Y']
FI[Y] = feat_importances
show_FI(FI)
FI_name = show_FI_2(Y, FI)
```

	hospital_outcome
age	1.398893
lab_neutrophil_percentage	0.688699
lab_leukocyte	0.647015
lab_crp	0.445243
lab_urea	0.389462

圖 5 LR 程式碼截圖、以及模型之重要特徵

參數調整：

我們調整了三種參數，分別是 C、max_iter 和 penalty。C 越小則損失函數會越小，模型對損失函數的懲罰越重，正則化的效力越強；max_iter 為最大迭代次數；penalty 為正規化方法。我們先嘗試一次調整一個參數，再與原本的預設值做比較，發現 C 小一點準確率較高，penalty 選擇 l1 時準確率較高，迭代次數原本就已經收斂了因此更大也不會有差別。接著我們做了三種參數組合的比較，如下圖，可以發現第二種參數組合(Model 2)的表現最好。

表 3　個別參數調整比較表

C	precision	accuracy
1	0.7347	0.8342
100	0.7143	0.8235
0.1	0.7826	0.803

max_iter	precision	accuracy
100	0.7347	0.8342
500	0.7347	0.8342

max_iter	precision	accuracy
12	0.7347	0.8342
11	0.75	0.8396

表 4　三種參數組合比較表

LR Model	C	max_iter	penalty	result
Model 1 (預設)	1.0	100	l2	Precision: 0.7347 Recall: 0.6667 F1: 0.699 Accuracy: 0.8342
Model 2	0.1	100	l2	Precision: 0.7826 Recall: 0.6667 F1: 0.72 Accuracy: 0.8503
Model 3	0.1	100	l1	Precision: 0.7317 Recall: 0.5556 F1: 0.6316 Accuracy: 0.8128

ii. 支援向量機（Support Vector Machine，SVM）

模型建構：

我們首先使用他預設的參數建立了一個 SVM 模型，並計算出 precision、recall、F1-score 、accuracy，並且透過程式找出此模型的重要特徵。

```
Data_X = df_x[COD['all']]
Data_Y = df_y[Y]
X_train, X_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.2, random_state=1)
sc = StandardScaler()
sc.fit(X_train)
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
#fit model
model = SVC()
model.fit(X_train_std, y_train.values)
pred_Y = model.predict(Testing_Set_X)
pred_train_Y = model.predict(Training_Set_X)# 預測訓練集 Training_Set_X
print(f'共{len(Data_Y.index)}筆: 訓練集{len(pred_train_Y)}, 測試集{len(pred_Y)}')
```

```
Precision: 0.9111
   Recall: 0.7593
       F1: 0.8283
 Accuracy: 0.9091
```

圖 6 SVM 程式碼截圖、以及各項效度之計算

```
#importance features
FI = {}
feat_importances = pd.Series(model.coef_.ravel(), index=COD['all'])
Testing_Result[Y] = re['pred_Y']
FI[Y] = feat_importances
show_FI(FI)
FI_name = show_FI_2(Y, FI)
```

	hospital_outcome
lab_leukocyte	1.016098
age	0.844514
lab_neutrophil_percentage	0.461286
lab_ast	0.408425
lab_rbc	0.292921

圖 7 SVM 程式碼截圖、以及模型之重要特徵

參數調整：

我們調整了兩種參數，分別是 C 和 kernel。C 越大對誤分類的懲罰增大，訓練集測試時準確率高，但泛化能力弱；kernel 為核函數。我們先嘗試一次調整一個參數，再與原本的預設值做比較，發現 C 越大準確率越高，kernel 選擇 rbf 時準確率較高。接著我們做了三種參數組合的比較，如下圖，可以發現第二種參數組合（Model 2）的表現最好。

表 5　個別參數調整比較表

C	precision	accuracy
1	0.9111	0.9091
100	0.9412	0.9519
0.1	0.9048	0.893

kernel	precision	accuracy
rbf	0.9111	0.9091
linear	0.7818	0.877

表 6　三種參數組合比較表

LR Model	C	kernel	result
Model 1 (預設)	1.0	rbf	Precision: 0.9111 Recall: 0.7593 F1: 0.8283 Accuracy: 0.9091
Model 2	100	rbf	Precision: 0.9412 Recall: 0.8889 F1: 0.9143 Accuracy: 0.9519
Model 3	100	linear	Precision: 0.7736 Recall: 0.7593 F1: 0.7664 Accuracy: 0.8663

iii. 隨機森林（Random Forest，RF）

模型建構：

我們首先使用他預設的參數建立了一個 Random Forest 模型，並計算出 precision、recall、F1-score、accuracy，並透過程式找出此模型的重要特徵。

```
import math
from sklearn.ensemble import RandomForestClassifier
Data_X = df_x[COD['all']]
Data_Y = df_y[Y]
Training_Set_X, Testing_Set_X, Training_Set_Y, Testing_Set_Y = train_test_split(Data_X, Data_Y, test_size = 0.2, shuffle = False)
Testing_Result = {}    #預測結果(值)
model = RandomForestClassifier()
model.fit(Training_Set_X, Training_Set_Y)
pred_Y = model.predict(Testing_Set_X)
pred_train_Y = model.predict(Training_Set_X)#  預測訓練集  Training_Set_X
print(f'共{len(Data_Y.index)}筆:  訓練集{len(pred_train_Y)},  測試集{len(pred_Y)}')
```

```
Precision: 0.7692
   Recall: 0.5556
       F1: 0.6452
 Accuracy: 0.8235
```

圖 8　RF 程式碼截圖、以及各項效度之計算

```
#importance features
FI = {}
feat_importances = pd.Series(model.feature_importances_, index=Training_Set_X.columns)
Testing_Result[Y] = re['pred_Y']
FI[Y] = feat_importances
show_FI(FI)
FI_name = show_FI_2(Y, FI)
```

	hospital_outcome
age	0.131575
lab_urea	0.060701
vitals_spo2_ed_first	0.056227
lab_neutrophil_percentage	0.049156
lab_creatinine	0.045734

圖 9　RF 程式碼截圖、以及模型之重要特徵

參數調整：

我們調整了三種參數，分別是 n_estimators、bootstrap 和 criterion。n_estimators 是弱學習器的最大迭代次數；bootstrap 代表是否有放回的採樣；criterion 為 CART 樹做劃分時對特徵的評價標準。我們先嘗試一次調整一個參數，再與原本的預設值做比較，發現。n_estimators 大一點準確率會稍微高一點點，bootstrap 選擇 True 時準確率較高；criterion 選擇使用 gini 係數時準確率會稍微高一點。接著我們做了三種參數組合的比較。如下表，可以發現第二種參數組合（Model 2）的表現較好，但三者差異不大。

表 7　個別參數調整比較表

n_estimators	precision	accuracy
10	0.7692	0.8235
100	0.7805	0.8342
500	0.7857	0.8396

bootstrap	precision	accuracy
True	0.7692	0.8235
False	0.7111	0.8128

citerion	precision	accuracy
Gini	0.7692	0.8235
Entropy	0.7561	0.8235

表 8　三種參數組合比較表

LR Model	n_estimators	booststrap	citerion	result
Model 1 (預設)	10	True	Gini	Precision: 0.7692 Recall: 0.5556 F1: 0.6452 Accuracy: 0.8235
Model 2	500	True	Gini	Precision: 0.775 Recall: 0.5741 F1: 0.6596 Accuracy: 0.8289
Model 3	100	True	Entropy	Precision: 0.7368 Recall: 0.5185 F1: 0.6087 Accuracy: 0.8075

iv. 各模型重要特徵之比較

表格為各模型中第一至第五重要的特徵，其中在三個模型都較為重要特徵的有年齡（Age）、嗜中性白血球（lab_neutrophil_percentage）。

表 9 各模型重要特徵表

		Models		
		LR	SVM	RF
Important Feature	1	Age (年齡)	lab_ leukocyte (白血球)	Age (年齡)
	2	lab_neutrophil_ percentage (嗜中性白血球)	Age (年齡)	lab_urea (尿素氮)
	3	lab_ leukocyte (白血球)	lab_neutrophil_ percentage (嗜中性白血球)	Vitals_spo2_ed_first (血氧飽和度)
	4	lab_crp (C 反應蛋白)	lab_ast 血清麩胺酸苯醋酸 (轉氨基酵素)	lab_lymphocyte_ percentage (淋巴球)
	5	lab_urea (尿素氮)	lab_rbc (紅血球計數)	lab_neutrophil_ percentage (嗜中性白血球)

肆、 結果驗證

一、 混淆矩陣

我們用混淆矩陣的四個指標—precision、recall、F1- score、accuracy 來驗證這三個模型的效度，混淆矩陣的介紹如下：

	Positive	negative
Predict positive	True positive (TP)	False positive (FP)
Predict negative	False negative (FN)	True negative (TN)

(1) Accuracy：$\frac{TP+TN}{TP+TN+FP+FN}$；模型準確率

(2) Precision：$\frac{TP}{TP+FP}$；模型正確率

(3) Recall：$\frac{TP}{TP+FN}$；實際死往且預測得死亡者(回想率)

(4) F1 score：$\frac{2*TP}{2*TP+FP+FN}$；

(2)(3)的調和平均數，越接近1模型越穩健

下表為這三個模型在較佳參數配置下之各項效度比較（交叉驗證後之平均）。結果顯示這 3 個模型的準確率都超過 80%，而 RF 的準確率稍微比其他兩者高一點，但其 recall 較低；而 LR 和 SVM 各項指標的結果都較均衡。

表 10　三模型在較佳參數配置下之各項效度比較表

Model	Precision	Recall	F1-score	Accuracy
LR	69.69%	59.83%	64.24%	81.69%
SVM	69.32%	56.8%	62.23%	80.94%
RF	73.86%	54.13%	62.36%	82.12%

二、 交叉驗證

我們也使用 k 折交叉驗證的方法來驗證我們模型的穩定性。我們將 934 筆資料分成五份，每次輪流取其中的四分當訓練集，一份當測試集，共會進行五次，這樣能讓我們更客觀的去檢視這三個模型的準確率和穩定性。結果以折線圖顯示，圖中紅色的是 Random Forest（RF），綠色是 Logistic Regression（LR），藍色是 SVM。在右上方的 Recall 圖中，可以看到 LR 的 Recall 一直都是最好的，而三個模型的準確率一直都很穩定，都在 0.8 左右。值得注意的是，三個模型在 Precision 的表現變異都比較大。

伍、 研究結論

因 COVID-19 之病患存活者仍為多數，因此本研究有針對不平衡資料進行處理，對於 LR、SVM、RF 三個模型而言，Recall 和 F1 score 皆有效提升。而針對研究中嘗試的三個模型中，由結果可見 RF 之準確率稍微優於 SVM 和 LR。而我們也於研究中提供各模型之顯著特徵，以協助人為對患者進行診斷時可做為判斷之參考。而模型交叉驗證的結果顯示出三種模型之穩定性及其準確率都有不錯的表現，可見利用分類模型對患者的存活率進行判斷是可行的，我們可利用本研究之模型協助院內醫生對呼吸器的分配進行決策，以提高生存機會高之患者獲得呼吸器的機會，使醫療資源不足的情況下也不會錯過搶救易存活患者的機會。

陸、 參考資料

羅吉斯迴歸分析(Logistic regression, logit model)-統計說明與 SPSS 操作

機器學習-邏輯回歸(Logistic Regression)

sklearn.svm.SVC 參數說明.台部落

Sklearn-RandomForest 隨機森林參數及實例.台部落

ML 入門（十七）隨機森林(Random Forest).程式設計之旅

徐仙陽（2011）。應用支援向量機建構半導體黃光升溫製程之 FDC 系統。國立清華大學工業工程與工程管理學系