以機器學習模型
預測新冠肺炎病人一個月內之死亡率

Project 2　group 5

張芳綺、賴筱庭、高藝真、胡詠晴

OUTLINE

01 問題定義

資料集描述
分析架構

問題定義(1/3)

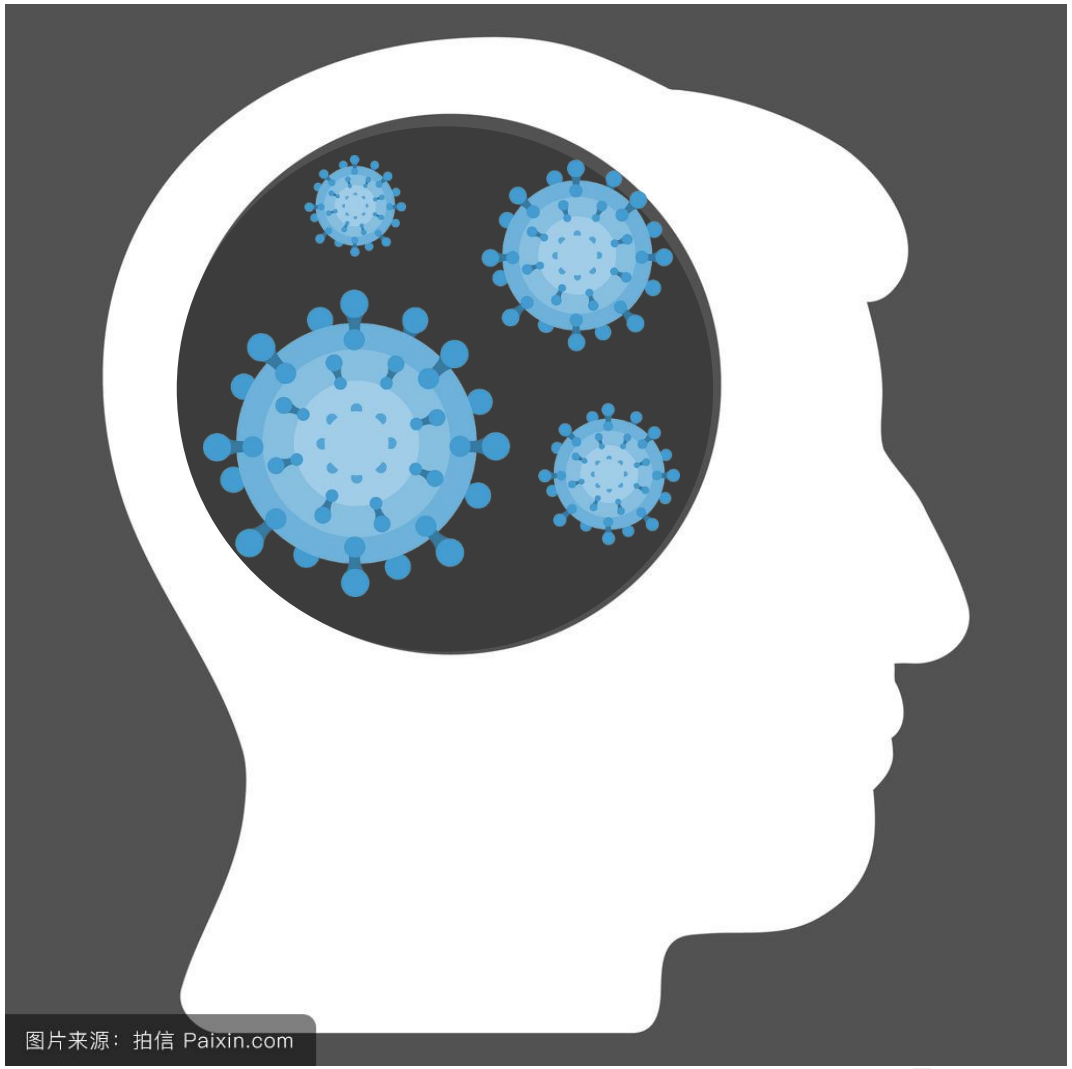
研究背景

- 因疫情日益嚴重的情況，隨著確診者不斷的增加，醫療資源的分配獲得大眾的重視。
- 醫療體系e化，醫療資料量急劇增加，其中隱含極高之資訊價值。
- 醫療資料大量收集，但相關研究的數量卻不多。
- 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，藉由大數據分析的手法，挖掘出其中有價值的訊息。

問題定義(2/3)

研究動機

- 在covid-19疫情日益嚴重的現在，希望透過利用大量資料的分析，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。
- 病毒症狀多樣，難以憑病人的症狀診斷其死亡機率。
- 醫療資源不足，院方希望能分配呼吸器給生存率較高的患者。



問題定義(3/3)

WHY

使資源能夠有效的分配

WHAT

解決大量病理數據資料分析

WHERE

醫院、科研中心

WHEN

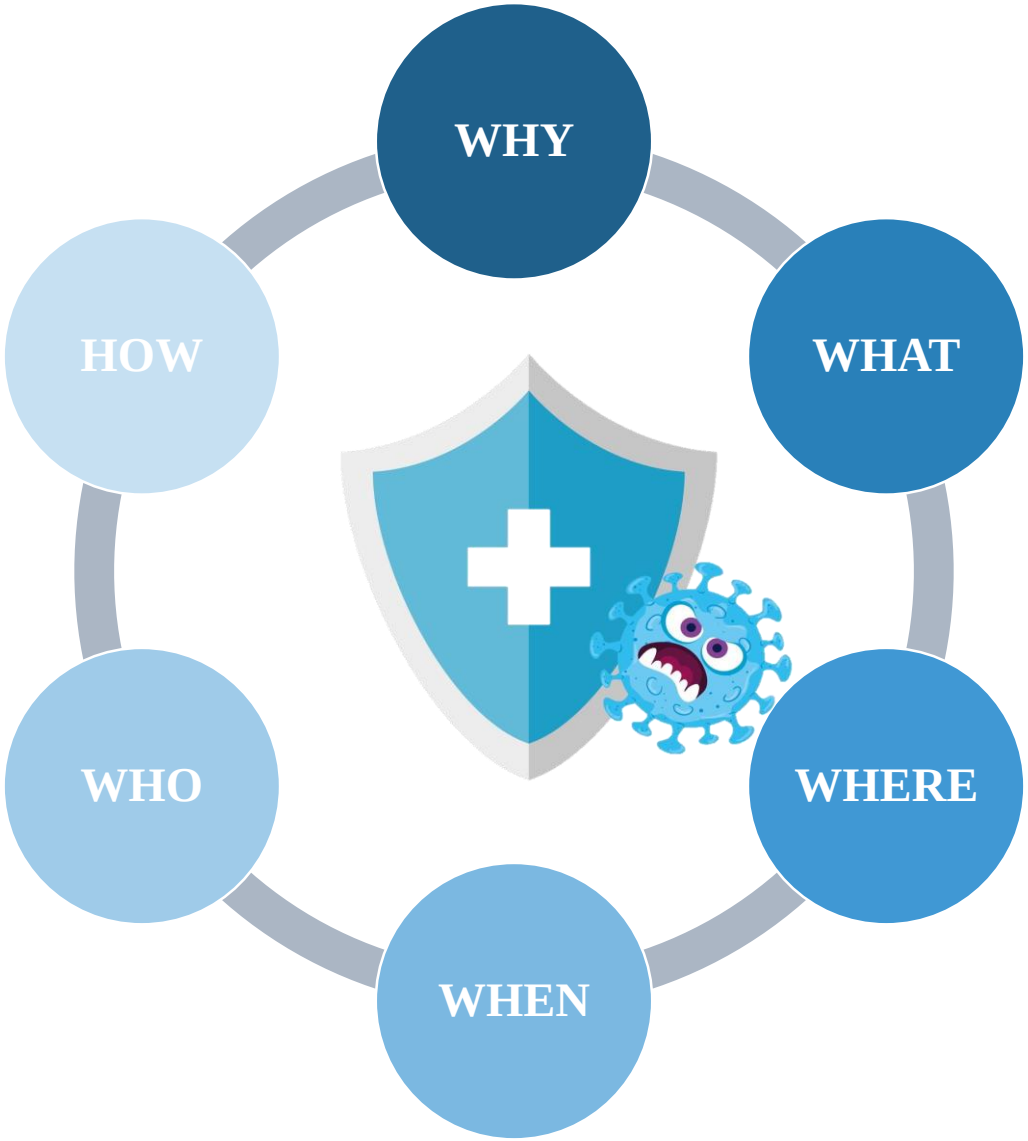
當呼吸器數量不足時

WHO

醫療人員、科研中心人員

HOW

資料分析、 機器學習(LR、SVM、RF)



資料集描述

- 取得方式
  - 某間位於西班牙的醫療機構(HM Hospitales)提供之公開資料。
- 資料背景
  - 此為2020年2月至6月covid-19患者的相關數據資料，裡面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。
- 資料欄位
  - 1834筆數據 (2020/2/5 ~ 2020/6/10)
  - 18個類別型欄位
  - 32個連續型欄位

  預測目標：死亡與否

  0 →活著

  1 →死亡

資料集欄位描述

age	age on admission
sex	sex
admission_datetime	date of admission
ed_diagnosis	primary symptom
mechvent_fla	indicator for mechanical ventilation
vitals_temp_ed_first	body temperature at ED triage
vitals_sbp_ed_first	systolic blood pressure at ED triage
vitals_dbp_ed_first	diastolic blood pressure at ED triage
vitals_hr_ed_first	heart rate at ED triage
vitals_spo2_ed_first	SpO2 at ED triage
pmhx_diabetes	indicator for comorbidity: diabetes
pmhx_hld	indicator for comorbidity: hyperlipidemia
pmhx_htn	indicator for comorbidity: hypertension
pmhx_ihd	indicator for comorbidity: ischemic heart disease
pmhx_ckd	indicator for comorbidity: chronic kidney disease
pmhx_copd	indicator for comorbidity: chronic obstructive pulmonary disease
pmhx_asthma	indicator for comorbidity: asthma
pmhx_activecancer	indicator for comorbidity: cancer
pmhx_chronicliver	indicator for comorbidity: chronic liver disease
pmhx_stroke	indicator for comorbidity: stroke
pmhx_chf	indicator for comorbidity: chronic heart failure
pmhx_dementia	indicator for comorbidity: dementia

lab_sodium	serum sodium
lab_leukocyte	leukocyte count
lab_mean_platelet_volume	mean platelet volume
lab_neutrophil	neutrophil count
lab_alt	serum alanine aminostransferase
lab_ddimer	serum d-dimer
lab_inr	international normalized ratio
lab_mch	serum mean corpuscular hemoglobin
lab_creatinine	serum creatinine
lab_mcv	mean corpuscular volume
lab_aptt	activated partial thromboplastin time
lab_platelet	platelet count
lab_lymphocyte_percentage	lymphocyte percentage
lab_glucose	serum glucose
lab_neutrophil_percentage	neutrophil percentage
lab_ldh	lactate dehydrogenase
lab_prothrombin_activity	prothrombin activity
lab_urea	serum urea
lab_lymphocyte	lymphocyte count
lab_crp	serum c-reative protein
lab_rdw	red blood cell distribution width
lab_hemoglobin	serum hemoglobin
lab_rbc	red blood cell count
lab_hct	serum hematocrit
lab_potassium	serum potassium
lab_ast	serum aspartate aminotransferase

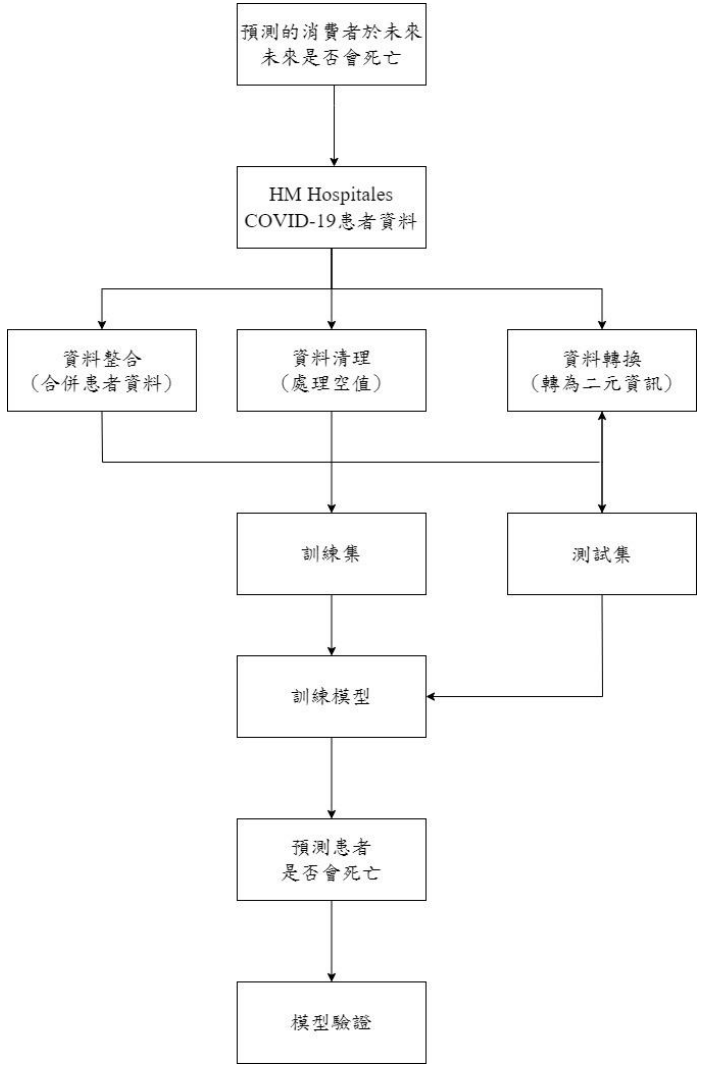
分析架構

問題定義

資料預處理

模型建構

結果驗證



02 資料預處理

資料預處理

- **資料整合：依據患者的ID合併資料的特徵值及label值**
- **資料轉換：將敘述的類別欄位轉換為二元資訊，如性別、症狀表現**
  - ✓性別：使用Label encoding將性別欄位轉換為0或1
  - ✓症狀表現：症狀欄位有五種類別,使用one hot encoding將每種症狀拆成一個獨立特徵

```
#將性別使用Label  encoding轉換，將症狀用one  hot  encoding轉換
label_encoder=preprocessing.LabelEncoder()
encoded_sex=label_encoder.fit_transform(df_x['sex'])
df_x['sex']=encoded_sex
data_dum  =  pd.get_dummies(df_x['ed_diagnosis'])
df_dia=pd.DataFrame(data_dum)
df_x=df_x.drop(columns=['ed_diagnosis'])
df_x=pd.concat([df_dia,df_x],  axis=1)
df_x.to_csv(  'clean_data.csv'    ,  index=False  )
```

資料預處理

- **資料清理：**
  - **清除無效欄位：**patient ID, admission_datetime
  - **處理空值：**類別型資料—補眾數；連續型資料—補中位數

```
#用中位數或眾數填補空直
values={}
L=[]
for  i  in  df_x:
        L.append(i)
vital_list=L[7:12]
pmhx_list=L[12:24]
lab_list=L[24:]
for  i  in  vital_list:
        values[i]=np.nanmedian(df_x[i])
for  i  in  pmhx_list:
        values[i]=stats.mode(df_x[i])[0][0]
for  i  in  lab_list:
        values[i]=np.nanmedian(df_x[i])
df_x=df_x.fillna(value=values)
df_x.to_csv(  'modified_data.csv',index=False  )
```

- 不平衡資料的處理 (Imbalance data)
  - **Random Under Sampling**
    - ✓ 隨機刪除900筆存活狀態為0(活著)數據，使存活和死亡的數據比例接近4:1
  - **刪除900筆數據後，以8:2分割訓練集與測試集：**
    - ✓剩下共934筆資料，訓練集747筆, 測試集187筆

```
#down  sampling(刪除900筆)
idx_list=[]
random.seed(a=42,  version=2)
Y  =  'hospital_outcome'
count_0  =  0;  count_1  =  1
for  i  in  df_y[Y]:
        if  i==0:  count_0  +=1
        else:  count_1  +=1
print(f'0:  {count_0},  1:  {count_1},  ratio:{round(count_0/count_1,  4)}')
delete  =  900  #count_1*4
print(f'共{len(df_y.index)}筆')
for  idx,i  in  enumerate(df_y['hospital_outcome']):
        if  i==0:
                idx_list.append(idx)
idx_list=random.sample(idx_list,  delete)
df_x.drop(idx_list,inplace=True)
df_y.drop(idx_list,inplace=True)
count_0  =  0;  count_1  =  1
```

```
for  i  in  df_y[Y]:
        if  i==0:  count_0  +=1
        else:  count_1  +=1
print(f'0:  {count_0},  1:  {count_1},  ratio:{round(count_0/count_1,  4)}')
print(f'共{len(df_y.index)}筆')
```

```
0: 1577, 1: 258, ratio:6.1124
共1834筆
0: 677, 1: 258, ratio:2.624
共934筆
```

03 模型建構

模型介紹與比較
重要特徵

- **Logistic Regression (LR)**
  - **Binomial LR: models the probability of occurrence of the success one of the two classes**
  - **Linear combination of predictors is used to fit a Logit transformation**

$$Ln\left[\frac{\hat{\pi}_i}{(1-\hat{\pi}_i)}\right] = w_o + w_1 X_{1i} + \cdots + \beta w_n X_{ni}$$

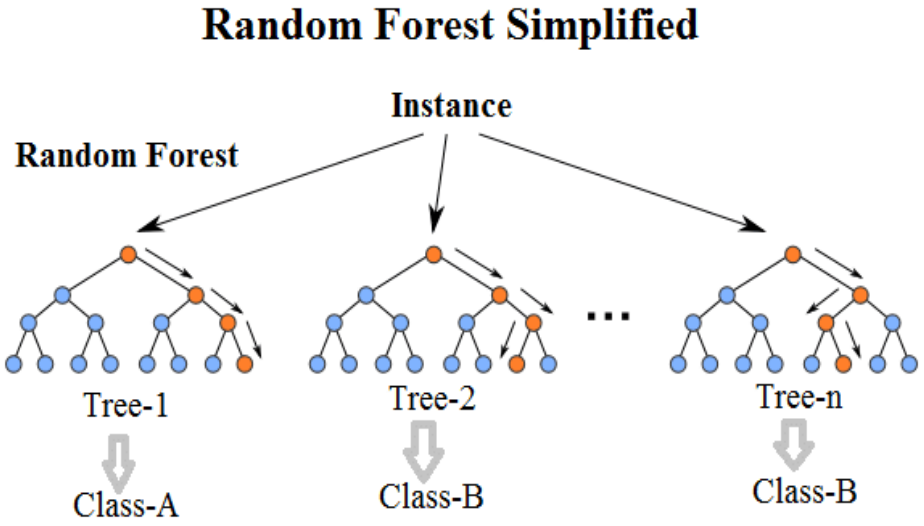
$$estimated\ probability\ \hat{\pi}_i = \frac{e^{w_o + w_1 X_{1i} + \cdots + w_n X_{ni}}}{1 + e^{w_o + w_1 X_{1i} + \cdots + w_n X_{ni}}},$$

$$\begin{cases} if\ > 0.5\ (or\ other\ user\ predefined\ threshold\ value): success\ group \\ else: failure\ group \end{cases}$$

- **Support Vector Machine (SVM)**
  - **Developed by Vapnik and his group in former AT&T Bell Labotories**
  - **Important topic in machine learning and pattern recognition**
  - **"hyper-plane classifier": using a linear hyper-plane which maximizes the distance between two class to create a classifier**

$$w'\emptyset(x) + b = 0 \begin{cases} x: vector\ of\ predictor \\ \emptyset: nonlinear\ feature\ function \\ w: weight \\ b: bias\ offset \end{cases}$$

- **Random Forest (RF)**
  - **Bagging (Bootstrap aggregation):**
    - Split and build decision trees: each time a random sample of m predictors is chosen as split candidates from the full set of p predictors
    - Final prediction is made by voting (classification) / averaging (regression)
  - **Boosting:**
    - Learn from misclassified data to improve final prediction
    - "iteratively learning weak classifiers with respect to a distribution and adding them to a final strong classifier"



Source:
https://medium.com/chung-yi/ml%E5%85%A5%E9%96%80-%E5%8D%81%E4%B8%83-%E9%9A%A8%E6%A9%9F%E6%A3%AE%E6%9E%97-random-forest-6afc24871857
https://en.wikipedia.org/wiki/Random_forest

- **模型選擇**
  - **因資料的分析結果為預測患者是否會死亡，屬於二元分類的問題，故選擇適合此種問題類型的相關模型，並進行比較。**

方法	type	優點	缺點
Logistic Regression	二元分類器	適用於連續和類別性自變數，且計算效率優於SVM和Random Forest	不能很好地處理大量多類特徵或變數
SVM	二元分類器	可以解決高維問題，即大型特徵空間，且可解決小樣本下機器學習問題	當訓練大規模數據時效率較低，且對非線性問題沒有通用解決方案，有時候很難找到一個合適的核函數
Random Forest	多元分類器	能處理大量多類特徵的資料，且抗過擬合能力較強	對於少量少類特徵的預測準度不是很好，且可能有很多相似的決策樹，掩蓋真實結果

Logistic Regression Model (LR)
羅吉斯回歸模型(預設參數)

```
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
```

```
#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

Logistic Regression Model (LR)
羅吉斯回歸模型

調整參數	解釋
C	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	最大迭代次數
penalty	正規化方法

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

C	precision	accuracy
1	0.7347	0.8342
100	0.7143	0.8235
0.1	0.7826	0.8503

max_iter	precision	accuracy
100	0.7347	0.8342
500	0.7347	0.8342

penalty	precision	accuracy
l2	0.7347	0.8342
l1	0.75	0.8396

Support Vector Machine Model (SVM)
支持向量機模型(預設參數)

```
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)}')
```

```
#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)
```

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

	hospital_outcome
lab_leukocyte	1.016098
age	0.844514
lab_neutrophil_percentage	0.461286
lab_ast	0.408425
lab_rbc	0.292921

Support Vector Machine Model (SVM)
支持向量機模型

調整參數	解釋
C	懲罰係數。C越大對錯誤分類的懲罰增大，訓練集測試時準確率高，但泛化能力弱
kernel	核函數

SVM 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

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

Random Forest Model(RF)
隨機森林模型(預設參數)

```
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
```

```
#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

Random Forest Model(RF)
隨機森林模型

調整參數	解釋
n_estimators	弱學習器的最大迭代次數
bootstrap	是否有放回的採樣
criterion	CART樹做劃分時對特徵的評價標準

RF Model	n_estimators	bootstrap	criterion	
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 4	500	True	entropy	Precision: 0.7368 Recall: 0.5185 F1: 0.6087 Accuracy: 0.8075

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

criterion	precision	accuracy
gini	0.7692	0.8235
entropy	0.7561	0.8235

- 各模型之重要特徵

		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)

04 結果驗證

混淆矩陣：準確率(Precision)、召回率(Recall)、F1-score、Accuracy

交叉驗證

- **Confusion matrix**

	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：$\frac{TP}{TP+FN}$；實際死亡且預測得死亡者(回想率)

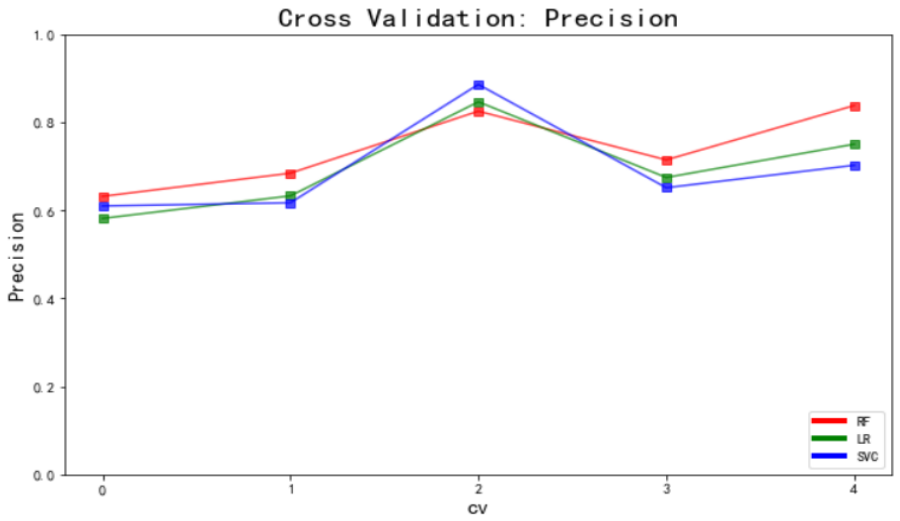
(4) F1 score：$\frac{2*\mathrm{TP}}{2*TP+FP+FN}$；

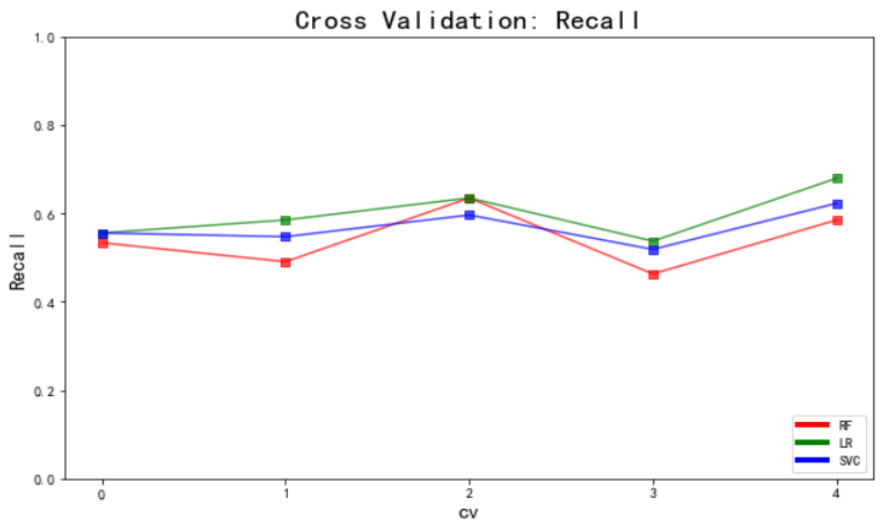
(2)(3)的調和平均數，越接近1模型越穩健

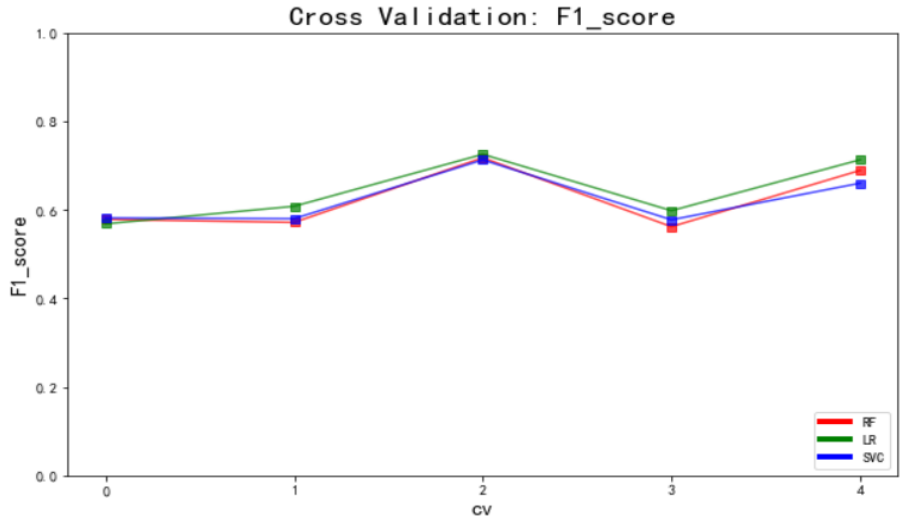
Model	Precision	Recall	F1-score	Accuracy
LR	69.69%	59.83%	64.24%	81.69%
SVM	69.32%	56.8%	62.23%	80.94%
RF	73.86%	54.13%	62.36%	82.12%

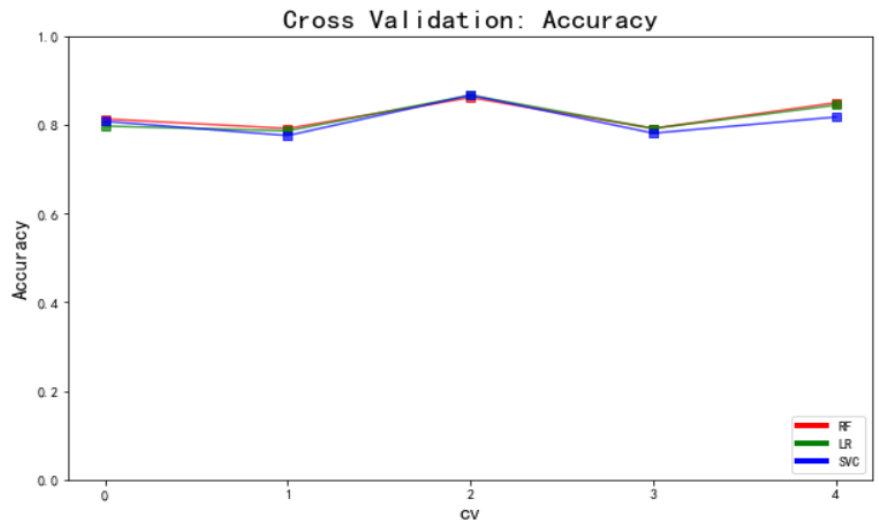
▲ 三模型在較佳參數配置下之各項效度比較(交叉驗證後之平均)

■對各模型進行交叉驗證以驗證模型的穩定性









05 研究結論

研究結論

- 因Covid-19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。
- 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、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。
- 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較不會錯過搶救易存活患者的機會。

Thank You

Project 2　group 5

張芳綺、賴筱庭、高藝真、胡詠晴