

# 題目 Topic：透過機器學習預測心臟病患風險

第一組 組員：許幸羽、吳億暄、賴亭璇

## 一. 摘要 Abstract

心臟病(Heart disease)是全球死亡的主因之一，部分致病因素與後天的生活習慣息息相關。隨著醫療技術提升，全球死亡率有下降的趨勢，但由於不正常的作息與錯誤的飲食習慣，導致近年死亡數甚至有上漲的趨勢[1]。心臟病可以通過心電圖、斷層掃描等專業技術精準判斷，但結合健康的社會決定因素(Social determinants of health, SDOH)可以更好的預測疾病風險，提早發現提早治療，達到降低死亡率的功能。

在本文中，我們使用 Kaggle 中關於心血管疾病的 70,000 項統計數據，包含 11 個 features，選用五種被廣泛使用的機器學習分類器，分別是決策樹(Decision tree)、線性回歸(Linear regression)、支持向量機(Support vector machine)、最近鄰居(k-nearest neighbor)、XGBoost 及分群(Cluster)。建立多種心臟病預測模型，並分別計算各種分類器的準確率，尋找最佳的預測模型，幫助人們警示與面對心臟疾病。

## 二. 引言、背景 Introduction

### 1. 心臟病

心臟病是一種心血管疾病(CVD)，是全球第一大死因，每年造成全球約 1700 萬人死亡，約佔全球所有死亡人數的 30%。[2]可能導致心臟病的幾個因素包括不健康的飲食、生活作息不正常、缺乏身體活動、其他疾病（如：高血壓、糖尿病）以及過度吸菸與飲酒。心臟病可以通過心電圖、斷層掃描等專業技術精準判斷，但結合健康的社會決定因素(Social determinants of health, SDOH)可以更好的預測疾病風險，我們可以通過養成良好的日常生活習慣來減少誘發因子，如減少飲食中的鹽分和油脂、食用水果和蔬菜、定期運動、停止吸菸以及適量飲酒，都有助於降低心臟病發生的風險。多數患者因在晚期才發現疾病，來不及治療而導致死亡，因此我們確實需要一個能夠預測心臟病存在的方式，提供患者及早發現、治療的機會，降低死亡風險；不僅如此，精準醫療意識抬頭，能夠將個人資料庫透過模型預測，輔佐醫師診斷，更大程度確保患者能被正確的診斷並接受最適當的治療方式。

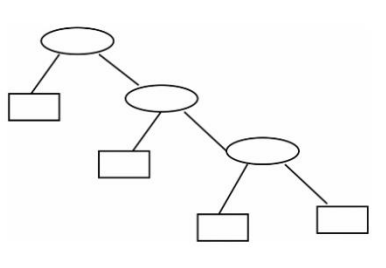
### 2. 機器學習(Machine learning)

機器學習是人工智慧(Artificial Intelligence, AI)的一個分支，用於檢驗數據，透過過往數據與資料找尋其中規則，建立系統以幫助預測未來發生事件的機率。近年來，AI 應用於醫療行業的發展熱度不斷提升，致力於預

測疾病風險、發病率等。機器學習即是訓練電腦從資料中學習，透過演算法找出大量資料中的關聯性，並隨著經驗累積而進行改善，最後得以分析出最佳決策或預測。[3]在本研究中，我們將使用決策樹(Decision tree)、線性回歸(Linear regression)、支持向量機(Support vector machine)、最近鄰居(k-nearest neighbor)及分群(Cluster)等五種分類模型來進行預測，並找出結果最佳的分類器，以下介紹這些分類器。

### 2.1 決策樹(Decision tree)

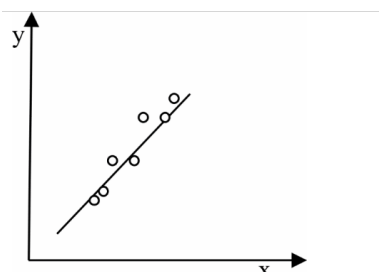
為一種監督式學習(Supervised learning)，每一層根據不同特徵進行分類，最後得到一種樹形結構(如圖一)。希望建構出熵值下降最快的樹。



圖一 決策樹模型

### 2.2 線性回歸(Linear regression)

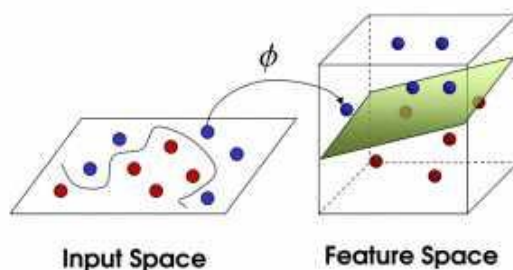
為監督式學習，觀察自變數  $x$  和應變數  $y$  的關係，以線性方程式表示。(如圖二)



圖二 線性回歸圖形

### 2.3 支持向量機(Support vector machine)

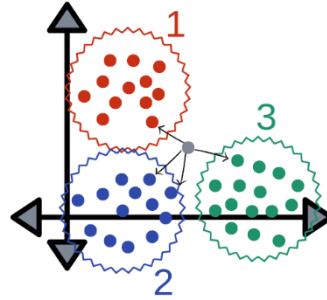
是監督式學習，尋找超平面(hyperplane)使得平面與所有點距離之間為最大值，藉此平面來進行分群。(如圖三)



圖三 支持向量機圖形

#### 2.4 最近鄰居法(k-nearest neighbor)

是一種監督式學習，輸入包含特徵空間中的  $k$  個最接近的訓練樣本， $k$  個最近鄰居中最常見的分類決定了賦予該物件的類別。圖四即為  $k=3$  的範例圖形。



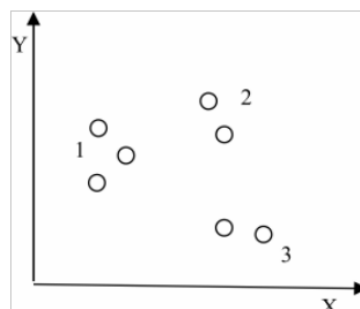
圖四  $k=3$  之最近鄰居法範例

#### 2.5 XGBoost

是一種以結合 Bagging 和 Boosting 的集成式學習法。保有 Gradient Boosting 的做法，每一棵樹是相互聯的，希望後生成的樹能夠修正前面的樹犯的錯。採用特徵隨機採樣的技巧，因此在每棵樹的生成中不會每一次都拿全部的特徵參與建模。擁有更高的精確性與靈活性，常見於 Kaggle 競賽。

#### 2.6 分群(Cluster)

屬於非監督式學習，在各個屬性進行評分，依評分結果將類似屬性的物件放於同一子集，達到最高群內相似度與最低群間相似度。(如圖五)



圖五 分群圖形

### 三. 資料集 Dataset

資料來源：[Cardiovascular Disease dataset | Kaggle](#)

本次研究資料為心血管疾病的資料，共 70,000 筆病人資料，藉由 11 個 features，我們可以對其有無心血管疾病做分析及預測患病機率等。所有病人資料皆在其接受健康檢查的同一時間量測的。其中 features 的種類分為三種：

- (1)客觀個人資訊(Objective)：factual information
- (2)醫院檢查資訊(Examination)：results of medical examination
- (3)主觀患者資訊(Subjective)：information given by the patient

詳細 feature 資料如表一：

表一

Name	Types	Short name	unit
Age	Objective	age	int(days)
Height	Objective	height	int (cm)
Weight	Objective	weight	float (kg)
Gender	Objective	gender	categorical code
Systolic blood pressure	Examination	ap_hi	int
Diastolic blood pressure	Examination	ap_lo	int
Cholesterol	Examination	cholesterol	1, 2, 3
Glucose	Examination	gluc	1, 2, 3
Smoking	Subjective	smoke	binary
Alcohol intake	Subjective	alco	binary
Physical activity	Subjective	active	binary

詳細 target 資料如表二：

表二

Name	Types	Short name	unit
Presence or absence of cardiovascular disease	Target Variable	cardio	binary

資料簡圖如表三：

表三

#id	#age	#gender	...	#alco	#active	#cardio
1	18393	2	...	0	1	0
2	20228	1		0	1	1
3	18857	1		0	0	1
...	...	...	...	...	...	...

## 四. 方法 Method

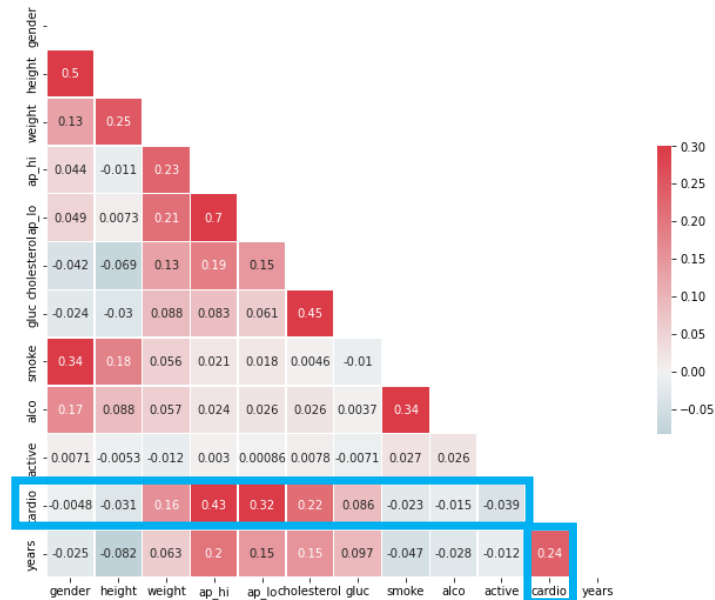
首先，由於以天數計算年齡的方法比較不直觀，所以將天數除以 365 來估算大約的年齡，命名新的 feature 為「years」。

接著，對於獲得的數據先進行前處理，去除不合理數據。可以看到數據庫中最小身高為 55cm、最小體重 10kg。但最小年齡是 29 歲，這不是個常見的身材數據，因此我們選擇去除身高及體重範圍在小於 2.5% 以及大於 97.5% 的 data。

另外，也存在 Systolic blood pressure 小於 Diastolic blood pressure 等錯誤，我們同樣去除血壓範圍在小於 2.5% 以及大於 97.5% 的 data。

最後剩餘 60142 個 data，並將這些數據以 70% : 30% 的切割成 training 和 test，進行機器學習的分析。

圖六為我們的數據所做出的 Heat map，可以看出 11 種 features 與 cardio 的相關性，可以判斷不同實驗條件下不同 feature 的表達模式。



圖六 本文使用心血管疾病資料之 Heat map

## 五. 結果 Results

### 5.1 決策樹(Decision tree)

使用 python 的 sklearn 中的 DecisionTreeClassifier 分類器，透過修改 max\_depth 調整，找出較適合的分支數，我們選擇使用 max\_depth = 5 的 model，獲得 training 的 accuracy score = 0.727、而 test 的 accuracy score = 0.723。(如圖七)我們在建模後，另外使用 matplotlib.pyplot 畫出在不同 depth number 的 accuracy score，(如圖八)發現最高 accuracy score 和我們選用的 max\_depth = 5 的 model 剛好相符。

A

```
[76] my_tree_1 = DecisionTreeClassifier(max_depth=5)

my_tree_1.fit(X_train, y_train)
y_pred = my_tree_1.predict(X_train)

from sklearn.metrics import accuracy_score
print(accuracy_score(y_train, y_pred))

0.7272144231454429

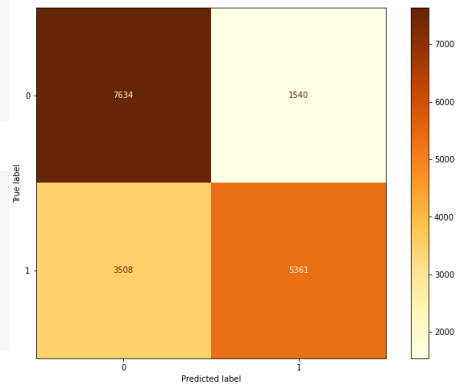
[72] my_tree_1 = DecisionTreeClassifier(max_depth=5)

my_tree_1.fit(X_train, y_train)
y_pred = my_tree_1.predict(X_test)

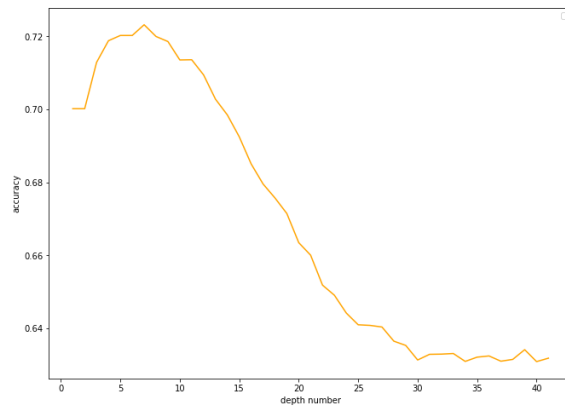
from sklearn.metrics import accuracy_score
print(accuracy_score(y_test, y_pred))

0.7233276062739012
```

B



圖七 (A) Decision tree 在  $\text{max\_depth} = 5$  時，所算出的 training (上)及 test (下)的 accuracy score (B) test 在 Decision tree 下的 confusion matrix



圖八 在 Decision tree model 中不同 depth number 下的 accuracy

## 5.2 線性回歸(Linear regression)

使用 python 中的 LinearRegression，截距及每一項的係數如下圖。

LinearRegression( fit\_intercept = True, normalize = 'deprecated', n\_jobs = None, positive = False )。

- fit\_intercept: 是否計算截距

- normalize: 是否正規化

- n\_jobs: 計算的工作數量，預設為 1

- positive: 係數是否必為正

由圖九可以看到 root mean squared error 為 0.44，誤差偏大，因此推論以線性迴歸分析效果不好。

```
[11] regressor.score(x_train,y_train)

0.2270705496477472

[12] #To retrieve the intercept:
print(regressor.intercept_)

#For retrieving the slope:
print(regressor.coef_)

-1.831425301159936
[-0.00373267 -0.00110905  0.00249349  0.01247047  0.00203118  0.09811223
 -0.02311895 -0.03140465 -0.04466457 -0.05058326  0.01001558]

Mean Absolute Error: 0.38683258609023735
Mean Squared Error: 0.19140121338092195
Root Mean Squared Error: 0.4374942438260439
```

圖九 Linear regression 模型下的相關參數與 root mean squared error

### 5.3 支持向量機(Support vector machine)

用 python 的 svm.SVC()建模，並利用 GridSearch 搜索參數，並結合 10-fold cross-validation 做交叉驗證。由於 polynomial kernel 較適合用在分布有線性趨勢的資料集上，因此選用 radial basis function kernel，將資料投射到更高維的空間去找 hyperplane。

```
parameters = {'C': [1.0,10,100,1000,10000], 'gamma': [0.1,0.001,10,100], 'kernel': ['rbf']}
svm = svm.SVC()
GS_CV = GridSearchCV(svm,parameters,cv=10)
GS_CV.fit(x_train, y_train)
```

圖十 SVM 模型所選用的參數

得到最佳參數為 C=1000，gamma=100。(圖十)其中 C 是懲罰係數，也就是對誤差的寬容度。C 高，說明越不能容忍出現誤差，所建的模型可能 generalization ability 變差。gamma 是選擇 rbf kernel 後的自帶參數，決定了數據投影到高維空間的分布，gamma 越大，支持向量越少，影響模型的訓練與預測速度。用訓練出來的模型分別預測 training dataset 及 testing dataset 得到的準確度皆為 75。

A

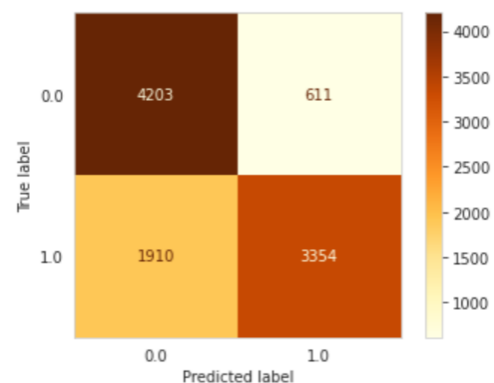
```
Best module parameters:
{'C': 1000, 'gamma': 100, 'kernel': 'rbf'}
Best score in 10 fold cross-validation:
0.750868258589874
```

	precision	recall	f1-score	support
0.0	0.69	0.87	0.77	19431
1.0	0.85	0.64	0.73	20877
accuracy			0.75	40308
macro avg	0.77	0.76	0.75	40308
weighted avg	0.77	0.75	0.75	40308

	precision	recall	f1-score	support
0.0	0.69	0.87	0.77	4814
1.0	0.85	0.64	0.73	5264
accuracy			0.75	10078
macro avg	0.77	0.76	0.75	10078
weighted avg	0.77	0.75	0.75	10078

B



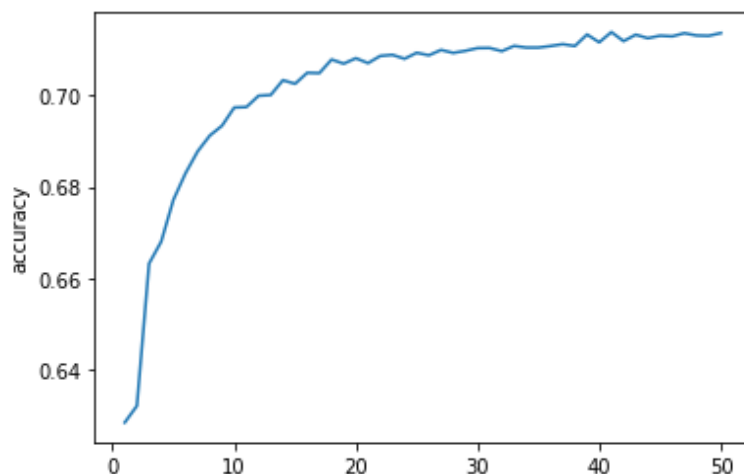
圖十一 SVM 所算出的 training (上)及 test (下)的 accuracy score (B) test 在 SVM 的 confusion matrix

#### 5.4 最近鄰居(k-nearest neighbor)

使用 python 中的 kNeighborClassifier 建模，k = 10 之後準確率上升幅度趨於平緩，(如圖十二)train 和 test 準確度分別是 0.74 和 0.70。(圖十三)

KNeighborsClassifier(n\_neighbors = 10, \*, weights = 'distance', algorithm = 'auto', leaf\_size = 30, p = 2, metric = 'minkowski' )

- n\_neighbors = 鄰居數
- weights = 算權重的方式
- algorithm = 最近鄰居的演算法，預設 auto 會由 model.fit 尋找最佳
- leaf\_size = BallTree 或 KDTree 演算法需建樹，size 越大需越多記憶體
- metric = 樹所建立出的距離矩陣



圖十二 KNN 在 training data 中不同 K 值模型的 AC 值

A

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
#建模型
knnModel = KNeighborsClassifier(n_neighbors=10)
knnModel.fit(x_train,y_train)
train_score = knnModel.score(x_train,y_train)
#valid_score = cross_val_score(knnModel,x_train,y_train,cv=5,scoring = 'accuracy')
test_score = knnModel.score(x_test,y_test)
print(train_score)
#print(valid_score.mean())
print(test_score.mean())
```

0.7420603814817454  
0.6975558388294629

B

7417	1797
3806	5023

圖十三 KNN 所算出的 training (上)及 test (下)的 accuracy score (B) test 在 KNN 的 confusion matrix

#### 5.5 XGBoost

用 python 的 XGBClassifier()建模，並利用 RandomizedSearchCV 搜索參數，(如圖十四)並結合 10-fold cross-validation 做交叉驗證，其結果如圖十五。



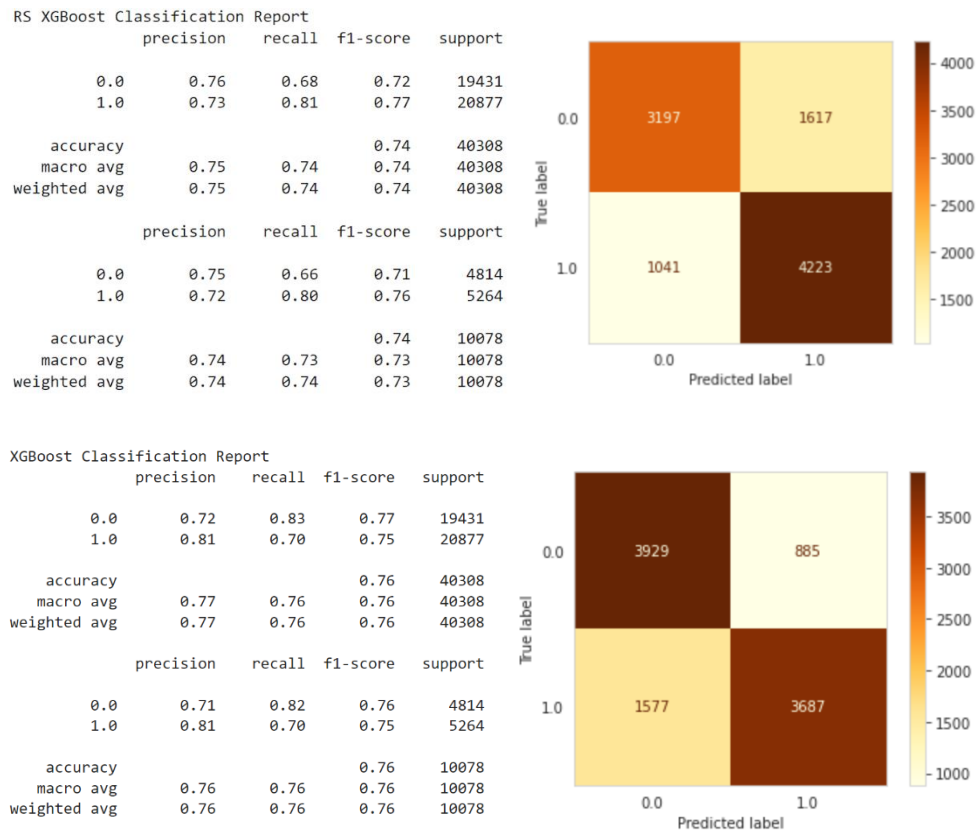
```

xgb = XGBClassifier()
# create hyperparameter grid
param_grid_xgb = {
    'learning_rate': [0.08],
    'max_depth': [4],
    'min_child_weight': [2, 3],
    'n_estimators': [125, 150],
    'scale_pos_weight': [1.5, 1.7]
}

RS_xgb = RandomizedSearchCV(xgb, param_grid_xgb, n_jobs= 3, scoring= 'recall', random_state=42)
RS_xgb.fit(x_train, y_train)

```

圖十四 XGBoost 模型所選用的參數



圖十五 XGBoost 所算出的 training (左上)及 test (左下)的 accuracy score (B) test 在 XGBoost 的 confusion matrix

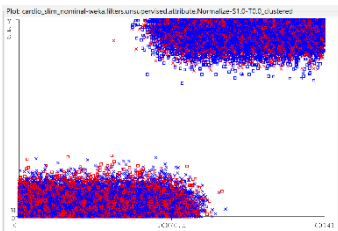
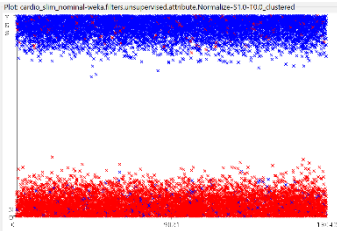
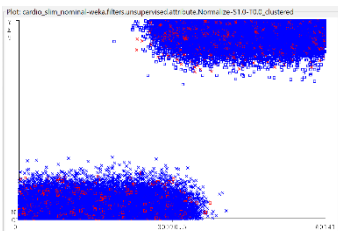
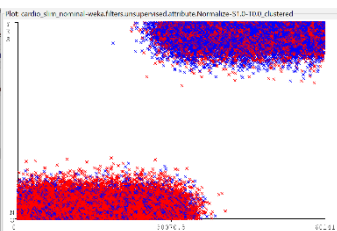
可以發現當 XGBoost 參數為 `base_score = 0.5`, `colsample_bylevel = 1`, `colsample_bytree = 1`, `gamma = 0`, `learning_rate = 0.1`, `max_delta_step = 0`, `max_depth = 10`, `min_child_weight = 1`, `missing = None`, `n_estimators = 100`, `nthread = -1`, `objective = 'binary:logistic'`, `reg_alpha = 0`, `reg_lambda = 1`, `scale_pos_weight = 1`, `seed = 0`, `silent = True`, `subsample = 1` 時，建模的表現最佳。

## 5.6 分群(Cluster)

用 **Weka** 中不同的 cluster model，K means、Canopy、EM 與 Fatherfirst 去建模，cluster mode 選用 classes to clusters evaluation，下方為其分類結果、incorrectly

clustered instances 以及 visualize cluster assignment(Instance-number, cardio)，結果統整於表四。

表四

	With 2 clusters :	
Method	K means	Canopy
Weka result	<pre> 0      1  &lt;-- assigned to cluster 20020 10759   NO 19234 10129   YES  Cluster 0 &lt;-- NO Cluster 1 &lt;-- YES </pre>	<pre> 0      1  &lt;-- assigned to cluster 27880 2899   NO 22947 6416   YES  Cluster 0 &lt;-- NO Cluster 1 &lt;-- YES </pre>
Incorrectly clustered instances	29993.0 49.8703 %	25846.0 42.975 %
Visualize cluster assignment		
Method	Fatherfirst	EM
Weka result	<pre> 0      1  &lt;-- assigned to cluster 29108 1671   NO 26251 3112   YES  Cluster 0 &lt;-- NO Cluster 1 &lt;-- YES </pre>	<pre> 0      1  &lt;-- assigned to cluster 3631 27148   NO 3094 26269   YES  Cluster 0 &lt;-- YES Cluster 1 &lt;-- NO </pre>
Visualize cluster assignment		
Incorrectly clustered instances	27922.0 46.4268 %	29900.0 49.7157 %

我們也有嘗試將 cluster 的數量往上調，發現其 incorrectly clustered instances 不降反升，因此用 cluster = 2 去分群。看上表我們可以發現在 Weka 中的 Canopy 表現最佳，K means 與 EM 的表現相對較差，EM 又被認為是 soft K-means。結果

推測為 Cynopy 在資料集較大的時候，因為其演算法的選擇教不嚴謹，表現較佳的同時也節省了時間。

另外還有選用 **fuzzy-c-means**，以 python 中的 fcmeans 的 FCM()建模，找出分群的 center 並回傳分群結果且用 Silhouette score 分析其表現。(圖十六)

```
from fcmeans import FCM
my_model = FCM(n_clusters=6) # we use two cluster as an example
## X, numpy array. rows:samples columns:features

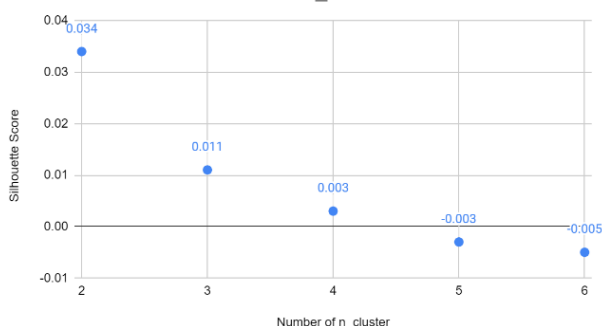
my_model.fit(D)
center = my_model.centers
print(center)

y_pred_train = my_model.predict(D)
print(y_pred_train)

score = silhouette_score(X, y_pred_train, metric='euclidean')
#
print('Silhouette Score with 6 cluster : %.3f' % score)
```

圖十六 fuzzy-c-means 程式碼

Silhouette Score vs. Number of n\_cluster



Silhouette Score with 2 cluster : 0.034  
 Silhouette Score with 3 cluster : 0.011  
 Silhouette Score with 4 cluster : 0.003  
 Silhouette Score with 5 cluster : -0.003  
 Silhouette Score with 6 cluster : -0.005

圖十七 silhouette score 跟 cluster 數量的關係圖

我們可以看出來當 cluster 數量增加時 Silhouette Score 離 1 越遠。因此我們拿 cluster 為 2 做進一步的分析。

得到兩群在不同 feature 中的中心點座標(如圖十八)

```
Index(['years', 'age', 'gender', 'height', 'BMI', 'weight', 'ap_hi', 'ap_lo',
      'cholesterol', 'gluc', 'smoke', 'alco', 'active', 'cardio'],
      dtype='object')
[[5.84260280e+01 2.13252702e+04 1.33650767e+00 1.64135826e+02
  2.75261818e+01 7.39886323e+01 1.27821048e+02 8.19235857e+01
  1.42993530e+00 1.26432618e+00 7.57074201e-02 4.71607547e-02
  8.00406634e-01 5.74972310e-01]
[4.63978569e+01 1.69353260e+04 1.36812892e+00 1.65151067e+02
  2.66945138e+01 7.27031463e+01 1.22979196e+02 7.98166187e+01
  1.24703273e+00 1.16156117e+00 1.00341208e-01 5.85375234e-02
  8.09481586e-01 3.75151401e-01]]
[1 0 1 ... 0 0 0]
Silhouetter Score: 0.034
```

圖十八 以 fuzzy-c-means 模型所得出兩群在不同 feature 中的中心點座標

fuzzy-c-means 為 soft cluster，可以接受的錯誤分群比例較大，但 cluster 的結果也沒有很好，Silhouette score 為 0.034，可能我們的資料集分布、關聯並不適合用 unsupervised 的分法來分析。

## 六. 討論 Discussion

由於我們選用的模型包含監督式學習與非監督式學習，甚至還含有集成式學習，因此無法直接將所有模型依照 AC 值來進行排名，但是可以從數據明顯看出，屬於集成式學習的 XGBoost 無論是 training 和 test 都表現最好，這個結果符合我們的預期，也在在證明集成式學習能更優化模型。

Model	training AC	test AC
Decision tree	73%	72%
Regression	27%	25%
SVM	75%	75%
KNN	74%	70%
RS-XGBoost	74%	74%
XGBoost	76%	76%
Fuzzy-C Means	39%	36%

表五 各模型的 training AC 及 test AC 統整比較表

因此，我們選用 **XGBoost** 來製作可以通過輸入自己的生理數值來進行心臟病預測的小程式應用，我們使用的程式 code 是將 XGBoost 的模型利用 pkl 存起來，再將使用者輸入的資料讀取、正規化，最後用 C++ Builder 呼叫 python 檔案對資料進行預測並輸出結果。(如圖十九)

(A)

```

double years = StrToFloat(Form1->Edit10->Text);
//double age = StrToFloat(Form1->Edit1->Text);
double gender = StrToFloat(Form1->Edit2->Text);
double height = StrToFloat(Form1->Edit3->Text);
double BMI = StrToFloat(Form1->Edit11->Text);
double weight = StrToFloat(Form1->Edit4->Text);
double ap_hi = StrToFloat(Form1->Edit5->Text);
double ap_lo = StrToFloat(Form1->Edit6->Text);
double cholesterol = StrToFloat(Form1->Edit7->Text);
double gluc = StrToFloat(Form1->Edit8->Text);
double smoke = StrToFloat(Form1->Edit9->Text);
double alco = StrToFloat(Form1->Edit13->Text);
double active = StrToFloat(Form1->Edit12->Text);

data = new int [13];
data[0] = (years-53.338798)/6.746864; //data[1] = (age-19468.719979)/2460.510296
data[2] = (gender-1.347311)/0.476120; data[3] = (height-164.554854)/6.830174;
data[4] = (BMI-27.170628)/4.404588; data[5] = (weight-73.426805)/11.614806;
data[6] = (ap_hi-125.770526)/13.761847; data[7] = (ap_lo-81.046307)/8.239157;
data[8] = (cholesterol-1.350953)/0.670076; data[9] = (gluc-1.220229)/0.567607;
data[10] = (smoke-0.085631)/0.279820; data[11] = (alco-0.051877)/0.221781;
data[12] = (active-0.803648)/0.397241;

```

(B)

Cardiovascular\_disease\_prediction

Do you have cardiovascular disease?

Years : 50

Gender : 2

Height : 168

Weight : 62

BMI : 21.96

Systolic blood pressure : 110

Diastolic blood pressure : 80

Cholesterol level : 1

Glucose level : 0

Smoke : 0

Activities : 1

Alcohol : 0

Predict My Possible Result

Please enter info as following.  
 ///  
 Years: your ages  
 Gender: 1 - women, 2 - men  
 Height: in cm  
 Weight: in kg  
 BMI: in kg/(m^2)  
 Systolic blood pressure measured  
 Diastolic blood pressure measured  
 Cholesterol level: 1: normal, 2: above normal, 3: well above normal  
 Glucose level: 1: normal, 2: above normal, 3: well above normal  
 Smoke: 1 - smoke, 2 - no  
 Activity: do sport or not; 1 - yes, 2 - no  
 Alcohol: 0 - not drink, 1 - drink  
 ///  
 Result:  
 Congratulations! You are healthy right now.  
 But you still have chance to get cardiovascular disease in thw future.  
 Keep good habits and be carefull.

圖十九 A)小程序應用程式碼 (B) 小程序應用範例圖示

## 七. 組員分工 Assignment of Responsibility

1. 建構模型： Decision tree - 吳億暄  
 Linear regression - 許幸羽  
 SVM - 賴亭璇、許幸羽  
 KNN - 許幸羽  
 Cluster - 賴亭璇  
 XGBoost - 賴亭璇
2. 模型優化 - 許幸羽
3. 書面報告 - 吳億暄
4. 報告、PPT 統整、小型預測程式 - 賴亭璇

## 八. 學習心得

### [許幸羽]

在這次的報告中練習了很多實作的經驗，理論是一回事，實際操作又是另一個層面的問題。像是在試著用 knn，一開始用 weka 跑，但似乎因為我們的資料 feature 太多，七萬筆資料檔案也太大，跑很久遲遲無法出現結果，因此後來用 python 自己建模。但自己建模最大的問題就在要餵給模型對的資料型態，才知道原來處理數據也不是想像中的這麼容易，相較之下，有了 sklearn，建立模型反倒是輕鬆簡易。不論是哪一種模型，成功餵給模型對的 training dataset 後，都要開始想著如何讓模型有更好的預測結果。在過程中不停尋找解決問題的方法，上網搜尋看看別人做出甚麼結果，試試看用不同參數或算法。例如用 grid search 找到最佳參數，又例如 linear regression 還能用 lasso 或 ridge 模型正規化回歸來優化模型，避免過度擬合；或是看到別人用甚麼方式呈現模型分類結果，除呈現 accuracy 之外，有直接畫圖或者畫 confusion matrix 等等。看著模型的準確度能變好真的是一件很有成就感的事。非常感謝另外兩位組員的幫忙與統整，謝謝億暄將書面報告整理好，謝謝亭璇的報告內容非常完整清晰，兩位都太給力了，我們常常約時間一起討論進展，在過程中相互討論也學到很多，真的很感謝他們。

### [吳億暄]

在這堂課上第一次接觸機器學習，SVM、MatLab 之類的對我來說都是陌生的知識，雖然助教有一一講解不同軟體的使用方法，但從不曾聽聞到馬上精熟到能做出一個完整的 project 還是不容易。負責 decision tree 的我本來想用 weka 來操作，但每一個過程的選擇太多，當使用其中一個 filter 時，就會開始質疑自己：「選擇這個 filter 後 AC 值又更高是因為什麼緣故？」「這真的是可以在這種 model 中使用的 filter 嗎？」，所以最後還是回到相對比較熟悉的 python 來製作 model。真的很感謝兩位組員的幫助，每一週的討論以及解惑，讓我可以了解每種模型的差異，要如何選擇參數，變動那些係數能更優化模型。不得不說實作真的是最好的導師，在課程上懵懵懂懂的內容通過專題漸漸清晰，也對於老師說機器學習常見的問題更感同身受，從一開始連題目都想不到，頭昏腦脹地聽著組員討論各種資料前處理的差異，到看著模型一個一個做出來，雖然我的貢獻沒有很多，還有很多需要精進的地方，但還是在這次專題學到很多，因為不清楚機器學習的論文會把甚麼資訊寫上去，所以看了一些範例。也很謝謝老師對於我們的報告的指正，非常喜歡他用了淺顯易懂的濃縮洗衣精來形容監督與非監督的差異，讓我很快就意識到我們的問題，也希望這份報告能展現我們這學期在這堂課所學的一切。

## [賴亭璇]

我覺得這次的報告有讓我把上課所學的東西盡量呈現出來，除此之外為了複習一些觀念我還找了一些其他人推薦的機器學習的書來看，包括 Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems，還有 Bishop: Pattern Recognition and Machine Learning。雖然沒辦法全部都翻閱完，但是我經過這次的報告我找到了許多自我學習的資源。除了上述兩本書以外，還有 YouTube 頻道教各式各樣的機器學習 model 概念，例如 StatQuest with Josh Starmer。在實作時也從 kaggle 上學了許多方法，像是為了不要 garbage-in-garbage-out，一開始 training dataset 的 processing 其實相當重要。課堂上沒有特別教方法，但自己學的時候反而印象更深刻，像是怎麼分析、處理缺值、去除 outliers 讓資料分布好一點等等。還有因為成效看起來沒有差很多而沒有使用的，跟學姊學到了 one-hot encoding 等等。為了不讓自己的小腦袋忘記這些方法，我還有自己做筆記，記下在課堂外學習到的新知。為了不浪費報告的篇幅，我把自己的筆記放在 google drive：  
<https://docs.google.com/document/d/1YG-4RIKJPIR-0wMs0EAXvLBu-QNGvw7gWRXJpWYqFqY/edit?usp=sharing>。

我們組每次的討論，發現問題，都讓我更進一步學習到許多知識。因為跟兩位學姊比起來我在機器學習方面真的是剛入手的菜鳥等級。我許多不懂的地方都會問她們兩個。例如我不太懂為什麼 KNN 鄰居數跟 training accuracy 的關係圖形在鄰居數增加到一定程度以後雖然已經趨於平緩了，但還是有起起伏伏的小曲折。學姊們就說可能是因為 10-fold cross validation 的取樣都是隨機取，所以應該跟取樣的資料分布有關，因此即便在 over training 的狀態下 training 的 accuracy 還是會有起起伏伏的樣子。像是這樣的狀況還有很多，真的很感謝她們兩位。

雖然沒辦法從 KNN 到 ANN 沒辦法全部都複習到，但是經由討論我們也可以幫彼此複習自己負責的 model，節省時間之外又學習到了很多技巧與知識，讓我覺得用這次的報告作為學期的收尾真的很棒！

## 九. 參考資料 References

- [1] Lopez, A. D., & Adair, T. (2019). Is the long-term decline in cardiovascular-disease mortality in high-income countries over? Evidence from national vital statistics. **International Journal of Epidemiology**. doi:10.1093/ije/dyz143
- [2] WHO (2011/06/11). **Cardiovascular diseases (CVDs)**. from: [https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-\(cvds\)](https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds))

[3] A. Singh and R. Kumar, "Heart Disease Prediction Using Machine Learning Algorithms," 2020. **International Conference on Electrical and Electronics Engineering (ICE3)**, 2020, pp. 452-457, doi: 10.1109/ICE348803.2020.9122958.