封面

A組

書瑋的部分

階層式綱要

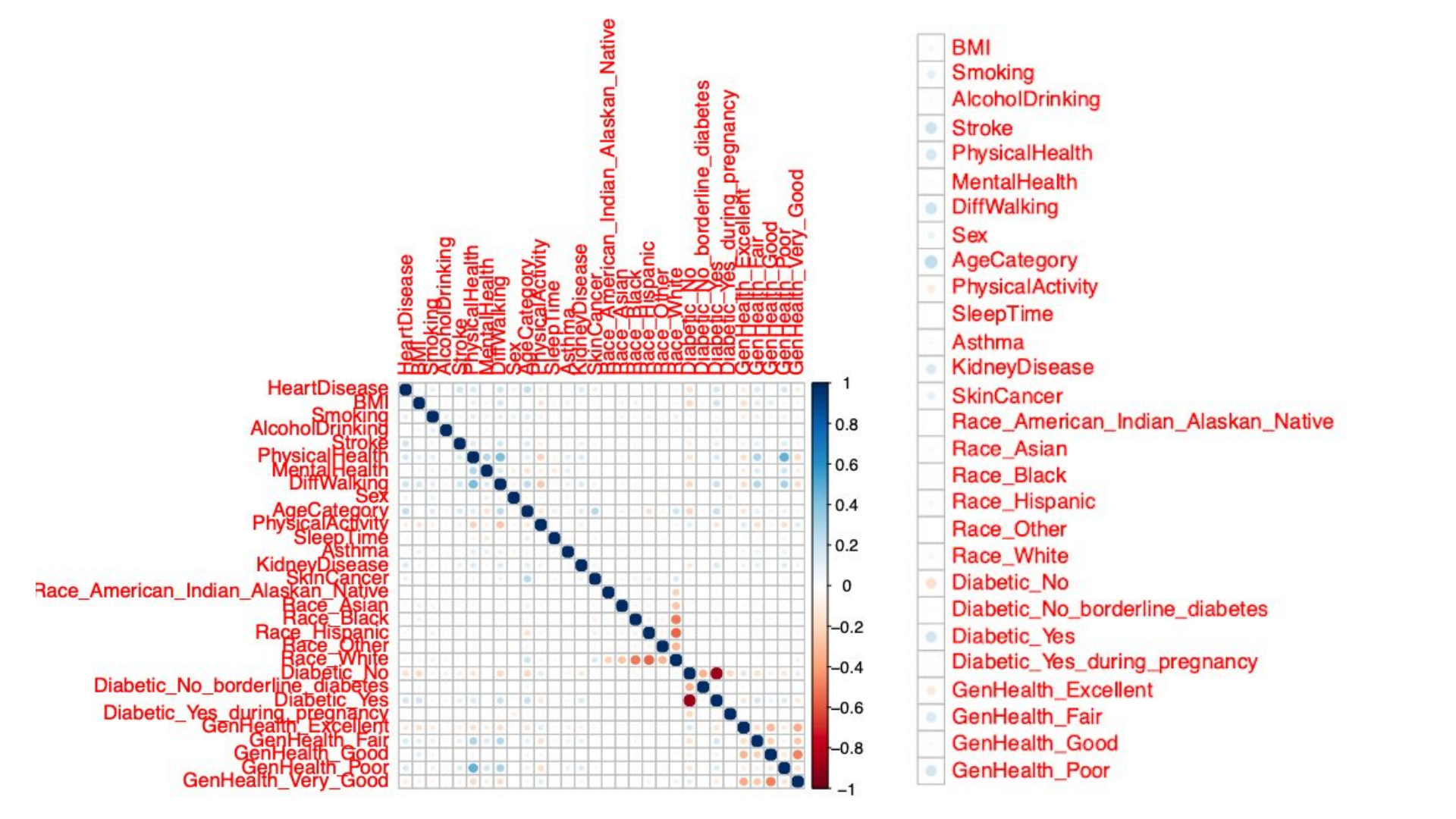
資料集的特徵相關性

Random Forest模型的成效

如何挑選好的Random Forest超參數

透過Random Forest 找出特徵的重要性、相似度

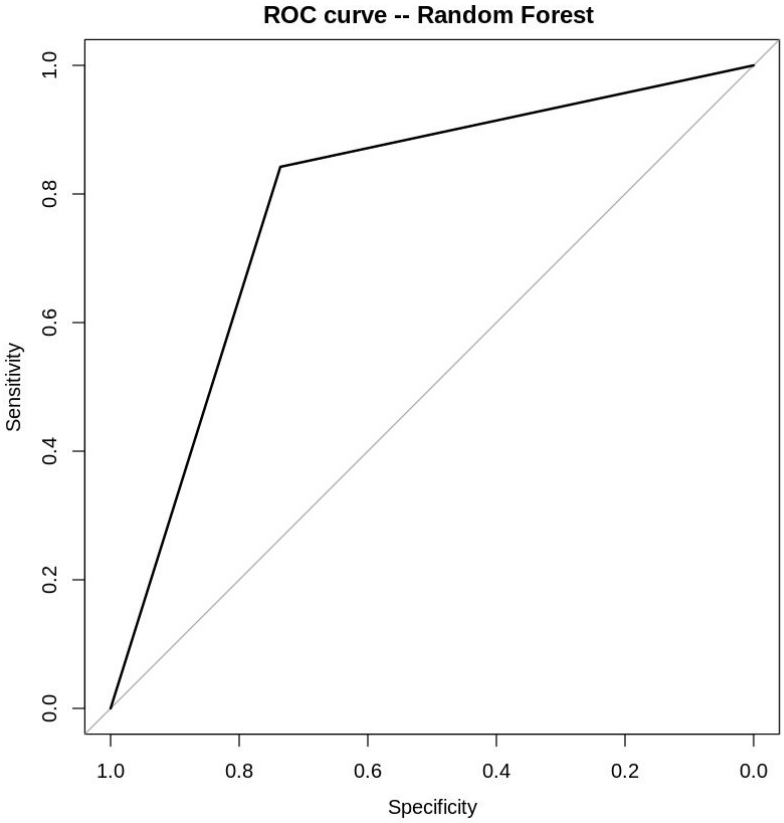
https://r-graph-gallery.com/index.html

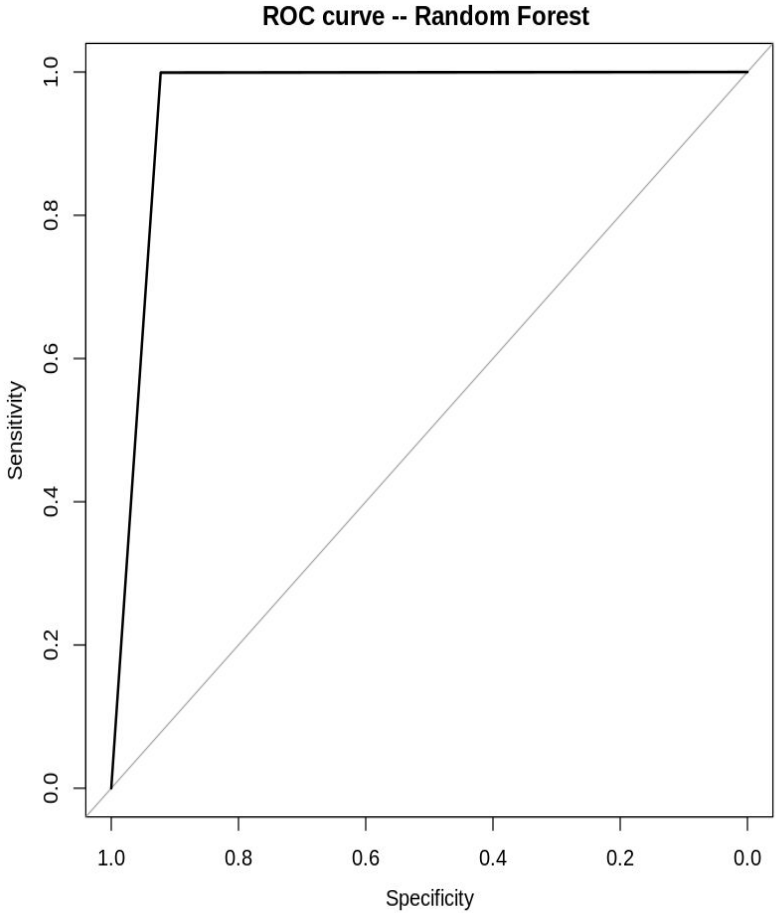

HeartDisease
BMI
Smoking
AlcoholDrinking
Stroke
PhysicalHealth
MentalHealth
DiffWalking
Sex
AgeCategory
PhysicalActivity
SleepTime
Asthma
KidneyDisease
SkinCancer
Race_American_Indian_Alaskan_Native
Race_Asian
Race_Black
Race_Hispanic
Race_Other
Race_White
Diabetic_No
Diabetic_No_borderline_diabetes
Diabetic_Yes
Diabetic_Yes_during_pregnancy
GenHealth_Excellent
GenHealth_Fair
GenHealth_Good
GenHealth_Poor
GenHealth_Very_Good
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
−0.6
−0.8
−1

model<-randomForest(as.factor(HeartDisease)~. ,data=train_data, ntree=15, **mtry=i**)

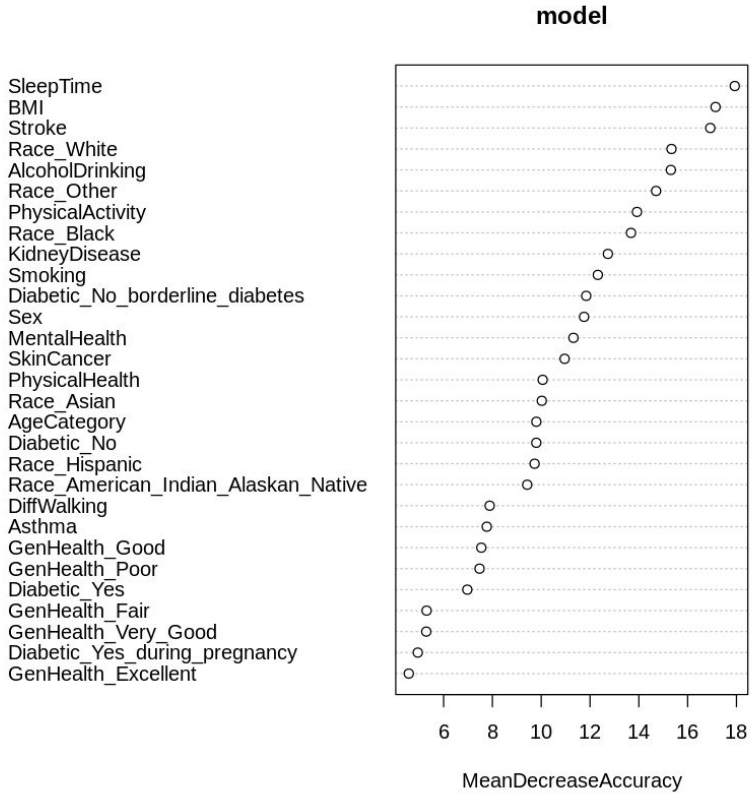
	error rate
10	0.1001042
11	0.08985848
12	0.08069532
13	0.07517755
14	0.06989365
15	0.06665733
16	0.06344449
17	0.06072315
18	0.05902768
19	0.05727703
20	0.05598322
21	0.05559717
22	0.05551134
23	0.05451079
24	0.05397596
25	0.05416524
26	0.05384128
27	0.05403395
28	0.0541914
29	0.05456589

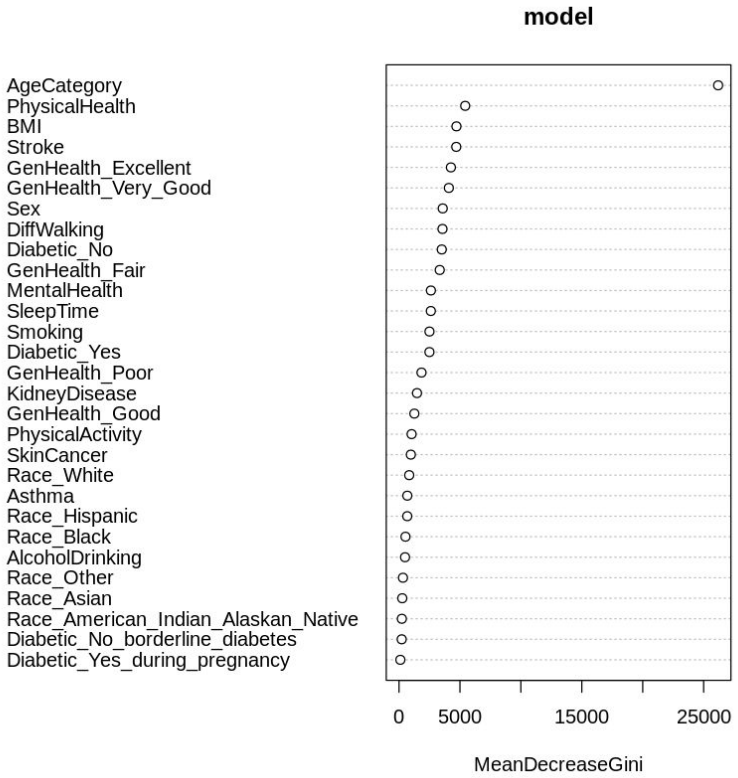
ntree=10, mtry = 3
[1] "accuracy : 0.788902875072971"
[1] "precision : 0.532009850963706"
[1] "recall : 0.5"
[1] "f1 : 0.515508500686217"
ntree=10, mtry = 26
[1] "accuracy : 0.960631932282545"
[1] "precision : 0.518344790914961"
[1] "recall : 0.5"
[1] "f1 : 0.509007160972699"


ROC curve -- Random Forest
Sensitivity
Specificity
1.0
0.8
0.6
0.4
0.2
0.0

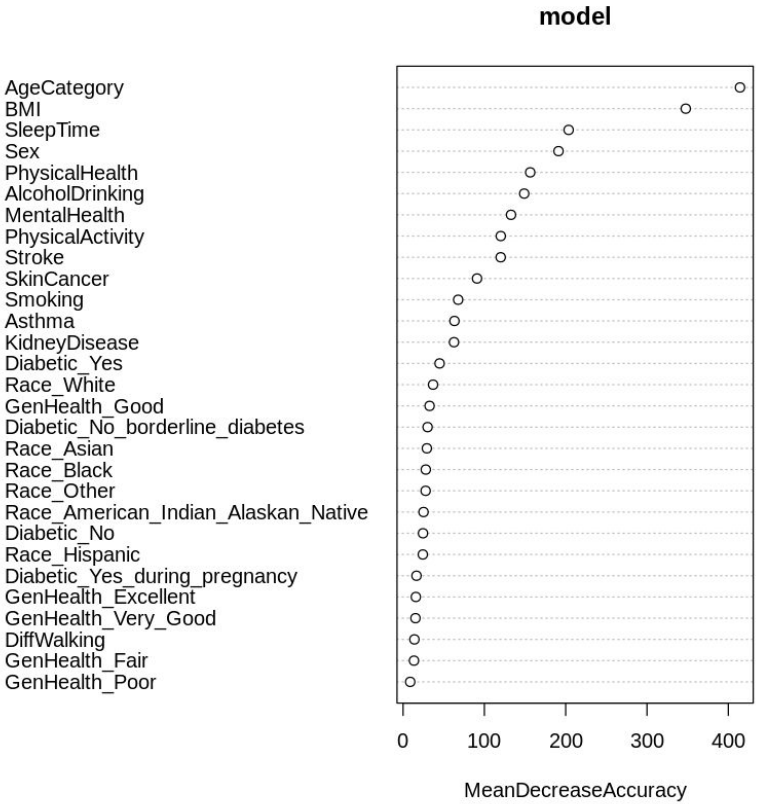

ROC curve -- Random Forest
Sensitivity
Specificity
1.0
0.8
0.6
0.4
0.2
0.0

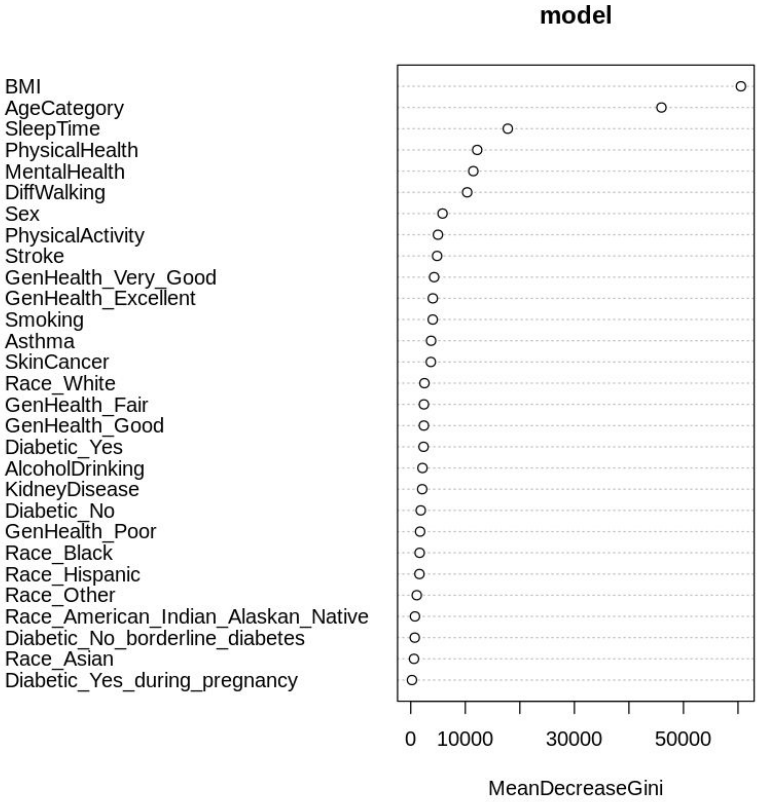
mtry = 3


model
SleepTime
BMI
Stroke
Race_White
AlcoholDrinking
Race_Other
PhysicalActivity
Race_Black
KidneyDisease
Smoking
Diabetic_No_borderline_diabetes
Sex
MentalHealth
SkinCancer
PhysicalHealth
Race_Asian
AgeCategory
Diabetic_No
Race_Hispanic
Race_American_Indian_Alaskan_Native
DiffWalking
Asthma
GenHealth_Good
GenHealth_Poor
Diabetic_Yes
GenHealth_Fair
GenHealth_Very_Good
Diabetic_Yes_during_pregnancy
GenHealth_Excellent
6
8
10
12
14
16
18
MeanDecreaseAccuracy


model
AgeCategory
PhysicalHealth
BMI
Stroke
GenHealth_Excellent
GenHealth_Very_Good
Sex
DiffWalking
Diabetic_No
GenHealth_Fair
MentalHealth
SleepTime
Smoking
Diabetic_Yes
GenHealth_Poor
KidneyDisease
GenHealth_Good
PhysicalActivity
SkinCancer
Race_White
Asthma
Race_Hispanic
Race_Black
AlcoholDrinking
Race_Other
Race_Asian
Race_American_Indian_Alaskan_Native
Diabetic_No_borderline_diabetes
Diabetic_Yes_during_pregnancy
0
5000
15000
25000
MeanDecreaseGini

mtry = 26


model
AgeCategory
BMI
SleepTime
Sex
PhysicalHealth
AlcoholDrinking
MentalHealth
PhysicalActivity
Stroke
SkinCancer
Smoking
Asthma
KidneyDisease
Diabetic_Yes
Race_White
GenHealth_Good
Diabetic_No_borderline_diabetes
Race_Asian
Race_Black
Race_Other
Race_American_Indian_Alaskan_Native
Diabetic_No
Race_Hispanic
Diabetic_Yes_during_pregnancy
GenHealth_Excellent
GenHealth_Very_Good
DiffWalking
GenHealth_Fair
GenHealth_Poor
0
100
200
300
400
MeanDecreaseAccuracy


model
BMI
AgeCategory
SleepTime
PhysicalHealth
MentalHealth
DiffWalking
Sex
PhysicalActivity
Stroke
GenHealth_Very_Good
GenHealth_Excellent
Smoking
Asthma
SkinCancer
Race_White
GenHealth_Fair
GenHealth_Good
Diabetic_Yes
AlcoholDrinking
KidneyDisease
Diabetic_No
GenHealth_Poor
Race_Black
Race_Hispanic
Race_Other
Race_American_Indian_Alaskan_Native
Diabetic_No_borderline_diabetes
Race_Asian
Diabetic_Yes_during_pregnancy
0
10000
30000
50000
MeanDecreaseGini

專案艱辛之處

1. **第一次接觸資料分析，也是第一次接觸 R，都是透過上課及作業學到的東西，加上不斷地Google及詢問ChatGPT，慢慢拼湊出最後的結果。**
2. 在對不平橫資料集的處理上，以及重要特徵的挑選上，因為不熟悉的關係，導致花了非常多的時間去嘗試。

引用資料

https://rpubs.com/jiankaiwang/rf
https://blog.csdn.net/ybdesire/article/details/120375089
https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-a7fa23d710

專案艱辛之處 & 引用資料(元亨)

1. 資料的不平衡:
   我們的資料集總共有30萬筆資料，包含約21萬筆健康人資料和2萬筆病患資料，在這種不平衡的情況下，Null Model傾向將資料分類為健康，導致準確率很高，但 Specificity(True Negitive Rage)和Sensitivity(True Positive Rate)卻極低。因此我們使用了 smote採樣平衡了健康人及病患的資料。一開始，我們也不知道該先切分 Training/Testing還是先smote，後來請教了老師後，才得知有很多人都在 這步驟錯誤地先smote才切資料集。
2. **程式的互相合作:**
   我們總共採用了3種(決策數、隨機森林、Xgboost)模型來分析資料，實際寫了 5份建模程式及1份視覺化程式，在這些Code中，每個人的Code pattern, 前處理, 套件選用等都不盡相同，讓我們在溝通、整合程式時費了一番功夫。

引用資料:

[RandomForest in R](https://ithelp.ithome.com.tw/articles/10303882?sc=iThelpR)

階層式綱要

資料前處理(smote前後差異)

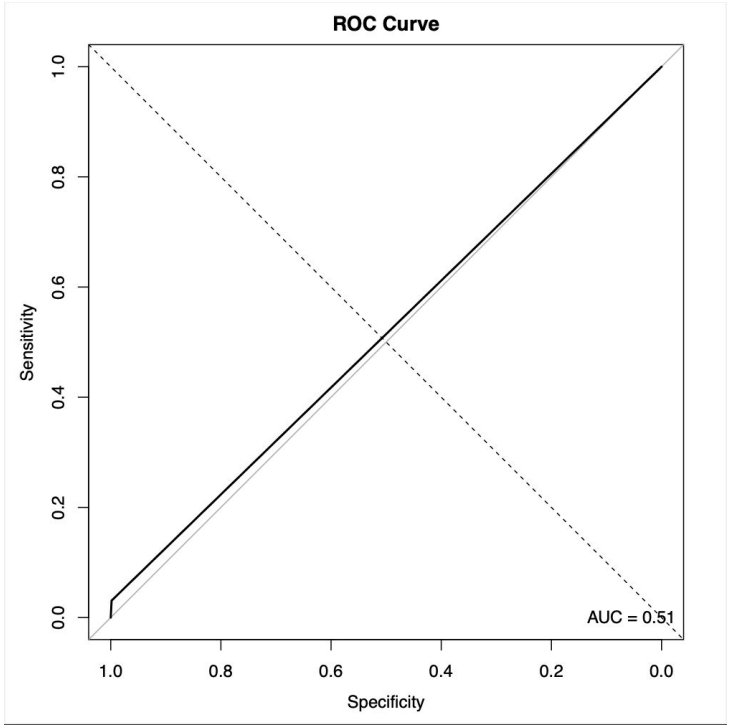
透過Random Forest 找出特徵的重要性、相似度

如何挑選好的Random Forest超參數

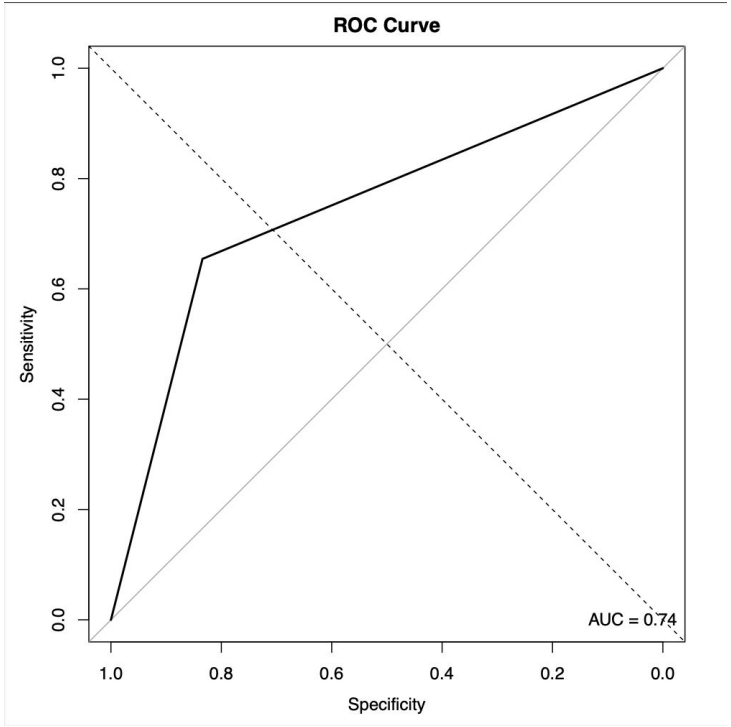
Smote前後差異(資料前處理測試)

將Training Data健康人與患者的10:1資料，經過smote過採樣轉為1.33:1的比例，實際AUC變化

Before Smote(29 feature)

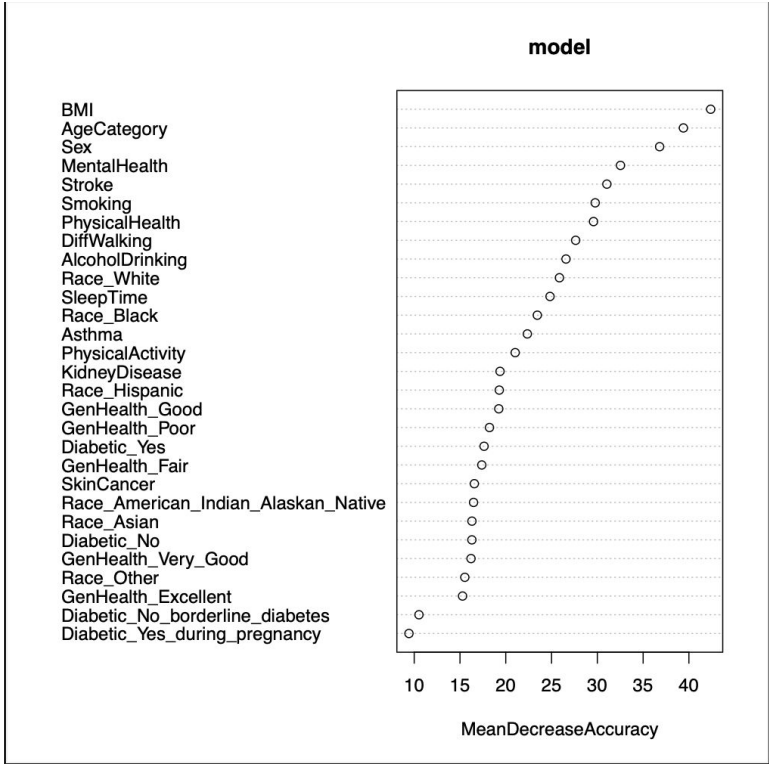


After Smote(29 feature)

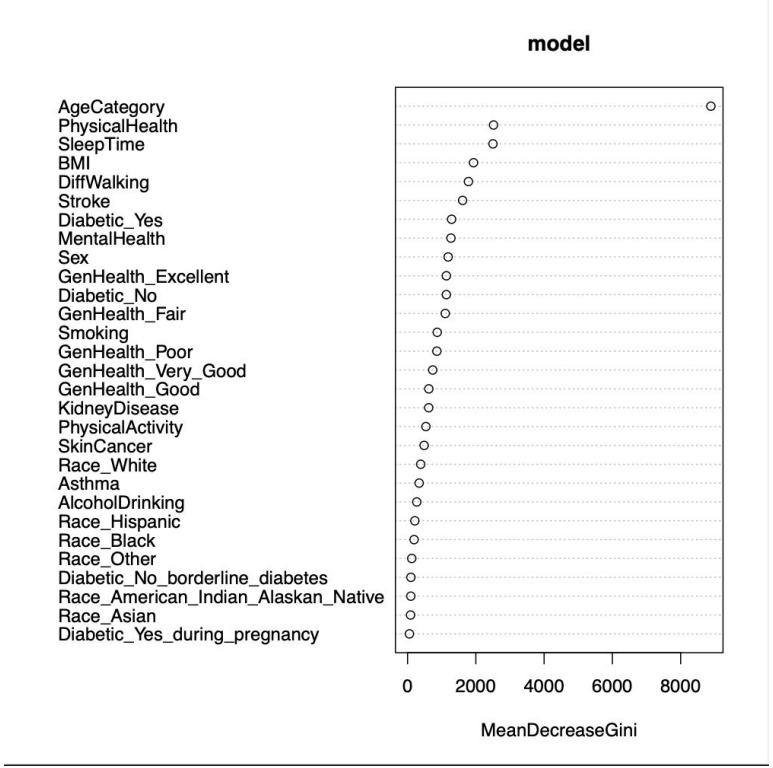


欄位重要性(從圖表中選出和決策樹一樣的13個特徵)

MeanDecreaseAccuracy: 刪除特徵後對模型的影響

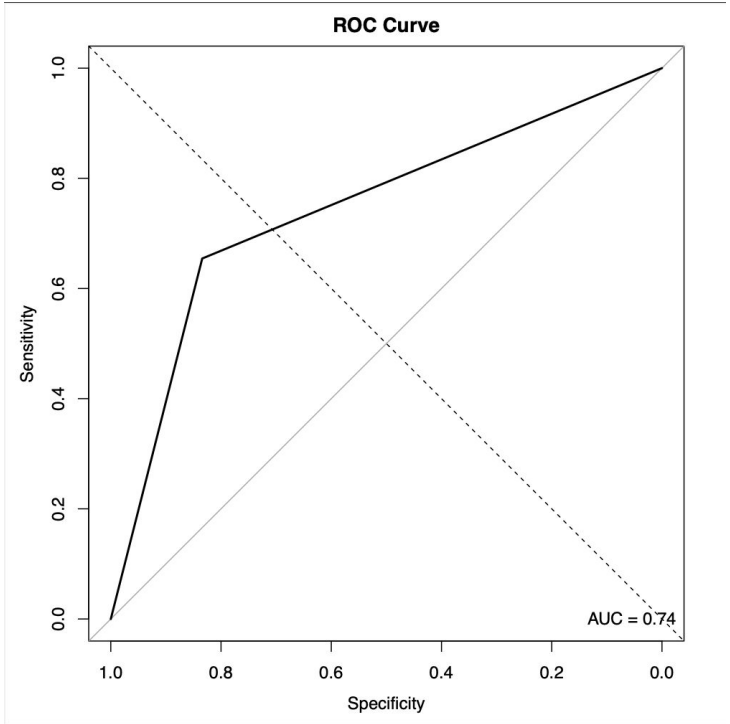


MeanDecreaseGini: 特徵對該模型的分類貢獻度

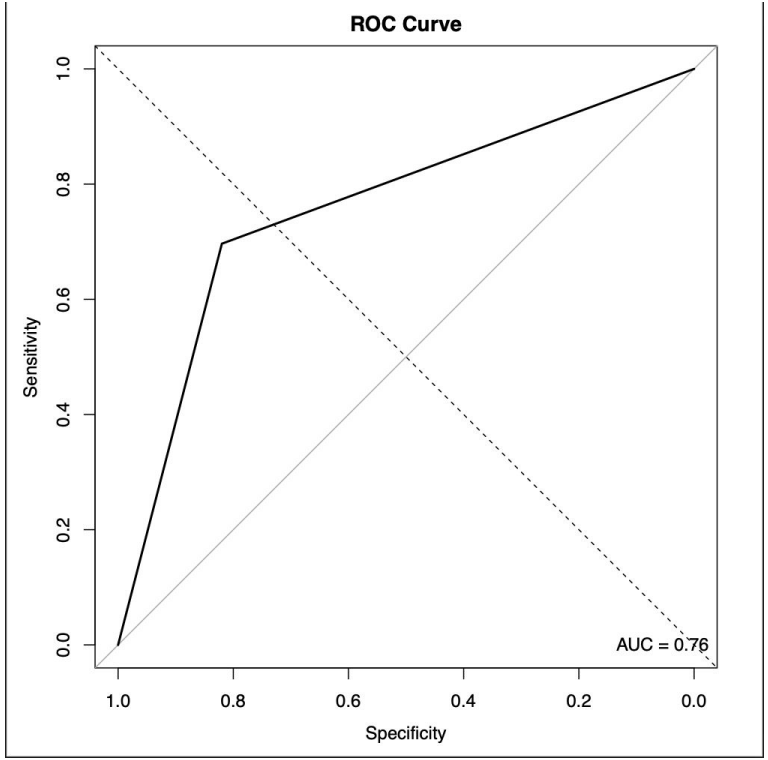


All Feature vs 13 Feature

All Feature



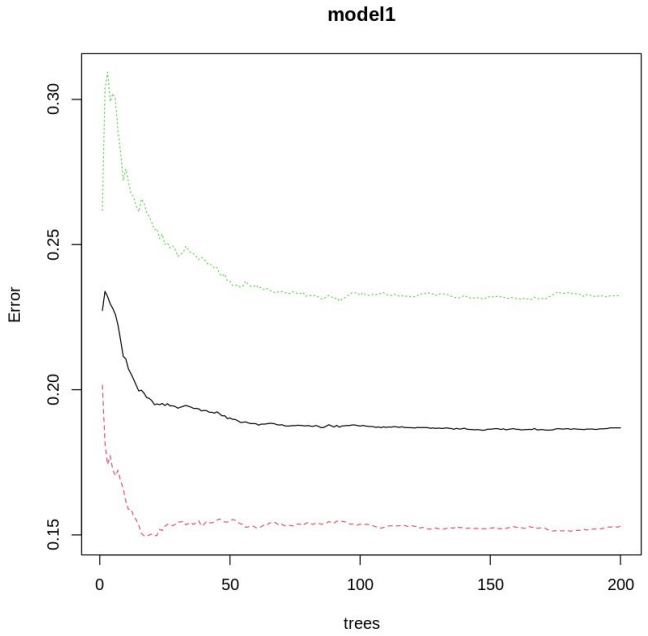
13 Feature



如何挑選好的Random Forest超參數(Random Forest參數挑選)

ntree：樹的數量

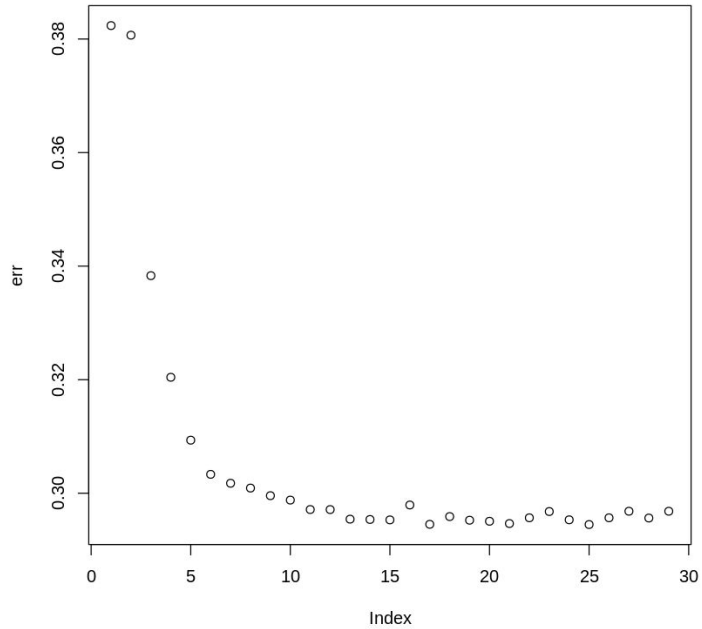
參數Default 500, 約在100左右就處於穩定



mtry：每棵樹使用的特徵數量

參數預設為sqrt(Feature),也就是5左右,

等於3時, 相差值最大 (實際測試3為最佳選擇)



如何挑選好的Random Forest超參數(Random Forest參數挑選)

(out of bag)OOB error rate : random forest 採樣時，沒被採樣到的資料

因此這些沒被選到的數據拿來進行預測評估，1 - (OOB data猜對數量 / (OOB data全部數量))

數值太多，只有貼部分
https://docs.google.com/spreadsheets/d/1IM2-4hOuFyP3FJt-YIv7QdDRmdFKmODZmsDmDpPo974/edit#gid=1549750515

0、1 欄位為null model 的error rate，0代表上一張投影片的紅線，1代表綠線

ntree	OOB	0	1
1	0.2272838	0.2015743	0.2617903
2	0.2338928	0.1815842	0.303601
3	0.2321145	0.1741547	0.3093591
4	0.2296104	0.1773268	0.2992522
5	0.2279823	0.1724328	0.3020781
6	0.225937	0.1704013	0.3000037
7	0.2222576	0.1722989	0.288872
8	0.2170802	0.1685793	0.2817696
9	0.2114995	0.1660718	0.2720123
10	0.2106179	0.1615557	0.2759899
11	0.2071853	0.1587754	0.2716994
12	0.2054948	0.1589637	0.2675014
13	0.2035805	0.156391	0.2664946

mtry	OOB error rate
1	0.3823684
2	0.3806774
3	0.3383198
4	0.3204217
5	0.3093427
6	0.3033233
7	0.301757
8	0.3009073
9	0.2995667
10	0.2987895
11	0.2971263
12	0.2971186
13	0.2954294
14	0.2953758
15	0.2953032

程式碼截圖

資料前處理, 篩選13個特徵:

```
train.index <- createDataPartition(df$HeartDisease, p = .7, list = FALSE)
train_df <- df[train.index,]
train_df <- smote(HeartDisease ~ ., data = train_df)
train_df <- na.omit(train_df)
selected_features = c('HeartDisease', 'BMI', 'Smoking', 'Stroke',
 'DiffWalking', 'Sex', 'AgeCategory', 'Asthma', 'KidneyDisease',
 'Race_Hispanic', 'GenHealth_Excellent', 'GenHealth_Good', 'GenHealth_Fair', 'GenHealth_Poor')
train_df <- train_df[, selected_features]
train_df$HeartDisease <- as.factor(train_df$HeartDisease)
```

建模:

```
model <- randomForest(HeartDisease ~ ., data=train_df, ntree=100, mtry=3, importance=TRUE)
```

此頁別刪

Code Usage

元亨: ***Rscript Jude_111971024.R --input ../data/heart_2020_cleaned.csv***

產出CSV: ***results/result.csv***
產出圖表1 (Roc Curve): ***results/roc_curve_after_smote.pdf***
產出圖表2 (參數重要性): ***results/varImpPlot.pdf***

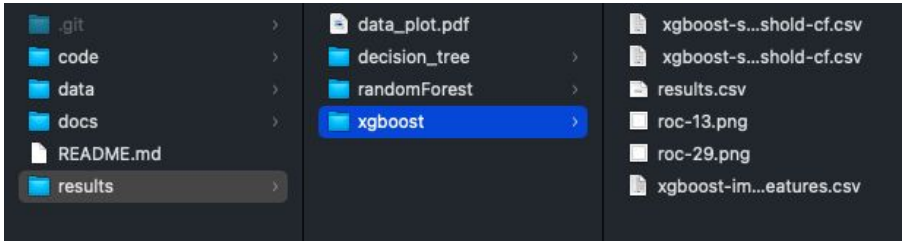
書瑋統整後的:

Code Usage

資料分析圖示
Rscript data_plot.R

主程式
Rscript main.R --d 2 --m 2
參數
--data_source or --d
1:讀取原資料，會處理dummy和smote
2:讀取已處理資料

--model or --m
1:decision tree (不支援data_source = 2)
2:random forest
3:xgboost



Feature	Accuracy	Precision	Sensitivity	Specificity	F1.Score	AUC
29Features	0.9	0.41	0.24	0.97	0.31	0.6
13Features	0.9	0.41	0.28	0.96	0.33	0.62

引用的套件

install.packages(‘randomforest’)

外部資料

https://rpubs.com/jiankaiwang/rf

[RandomForest in R](https://ithelp.ithome.com.tw/articles/10303882?sc=iThelpR)

正晏的部分

專案艱辛之處

1. 資料集不平均
   本次期末專案所使用的資料集，在二元分類中兩類分類比例達到 1:10的差距，導致在模型Training前需要調整比例
2. **對於不熟悉領域的探索**
   平時在於工作中並未使用到與 Data Science相關的技能，本次專案算是一個新領域的探索，在專案建立的過程中，除了需要時不時複習老師上課的 內容之外，還需要不斷地去學習，在資料集確定後就遇到了第一個難關，在不同的模型上，會針對其所需要的輸入去調整 Attribute，在建模的過程中，發現若不了解模型的原理，在參數的 值上也難以去調整。

簡報架構

- 為什麼使用XGBoost
- XGBoost在不同參數&資料處理的結果
  - 參數說明
  - Smote & No Smote
  - 效能評估
- XGBoost 重要參數

為什麼使用XGBoost

- XGBoost (eXtreme Gradient Boosting) 是一個boosting集成算法，主要思路為將成千上百的**決策樹模型**組合成一個高準確率的模型。
- Bagging vs Boosting : XGBoost 透過序列的方式生成樹，後面生成的術語前一棵樹相關。
- Bagging (隨機森林作法)、Boosting(XGBoost作法)

XGBoost參數設定

參數說明

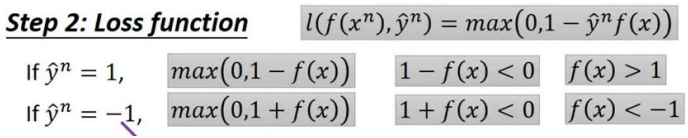
- max_depth: 樹的最大深度, 使用max_depth=6, 數值愈大模型擬合度越高。
- eta: 使用預設值0.3, 參數用於防止over fitting。
- eval_metric: error： 此為二進制分類錯誤率, model中default使用0.5來判斷, 計算式為#(wrong cases)/#(all cases)

- 學習任務參數

  - binary:logistic: 二元分類問題中邏輯回歸輸出機率
    考量實務面：要使用閥值(threshod)、產生ROC圖, 因此選用logistic。

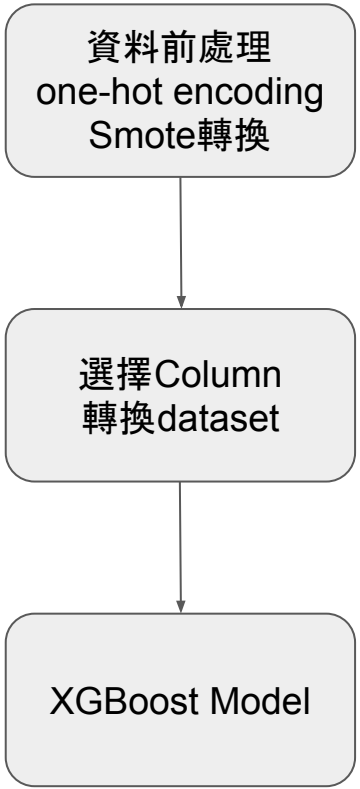
  - binary:hinge: 二元分類中使用hinge loss作為loss function進行分類, 輸出結果為0或1

  - binary:hinge: xgboost官方文件中簡單解釋為利用hinge loss來進行二元分類, 且輸出結果為0 or 1。該loss function主要目的為分類時候, 對於已經很確定預測結果的輸出, 也就是下方式子的f(x)>1 或是 f(x)<-1, 經過hinge後為0, 不希望這些非常肯定的預測結果影響我們分類器。而是針對預測結果輸出為 -1 ~ 1 之間, 無法分類分常明確的目標進行loss計算, 希望找到最小值, 著重於整體分類。



binary:hinge：基於loss function是hinge的SVM, model對每一筆預測資料輸出2元結果(1, 0)

XGboost使用Flow



train_matrix_selected	External pointer of class 'xgb.DMatrix'

DMatrix 是 XGBoost 使用的內部數據結構，它針對內存效率和訓練速度进行了優化。

```
> print(train_matrix_selected)
xgb.DMatrix  dim: 95855 x 13  info: label  colnames: yes
```

```r
# select 13 features
# "BMI+Smoking+Stroke+DiffWalking+Sex+AgeCa
# +GenHealth_Good+GenHealth_Poor"      You, 1 second ago • Uncommitted changes
features_selected <- c("BMI","Smoking","Stroke","DiffWalking","Sex","AgeCategory","Asthma",
"KidneyDisease","Race_Hispanic","GenHealth_Excellent","GenHealth_Fair","GenHealth_Good","GenHealth_Poor")
# features_selected <- c(1, 2, 4, 7, 8, 9, )
train_features_selected <- train_df[, features_selected]
test_features_selected <- test_df[, features_selected]
train_features <- as.matrix(train_features_selected)
train_labels <- as.matrix(factor(train_df[, 1]))

test_features <- as.matrix(test_features_selected)
test_labels <- as.matrix(test_df[, 1])

train_matrix_selected <- xgb.DMatrix(data = train_features, label = train_labels)
test_matrix_selected <- xgb.DMatrix(data = test_features)
```

欄位選擇
轉換xgboost
DMatrix格式

```r
params <- list(
  "objective" = "binary:logistic",
  "eval_metric" = "error",
  "max_depth" = 6,
  "eta" = 0.3,
  "nthread" = 4
)

model_xgboost_selected <- xgb.train(params = params, data = train_matrix_selected, nrounds = 100)
```

使用binary:logistic下不同threshold下的結果

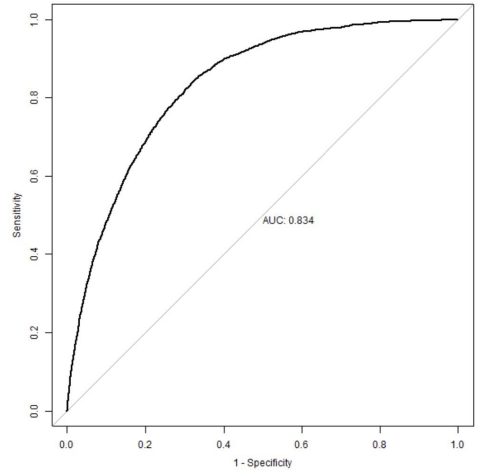
	threshold	sensitivity	specificity	F1
1	0.00	1.000000000	0.0000000	0.163362620
2	0.05	0.960004968	0.4110278	0.240221293
3	0.10	0.916532108	0.5503129	0.281035172
4	0.15	0.852937523	0.6491317	0.313189820
5	0.20	0.788970314	0.7198414	0.338719138
6	0.25	0.719165321	0.7789823	0.361118907
7	0.30	0.639920507	0.8252450	0.373144057
8	0.35	0.566761893	0.8646682	0.383864726
9	0.40	0.480934045	0.8972521	0.379682291
10	0.45	0.411377469	0.9216749	0.371675457
11	0.50	0.341696684	0.9423506	0.353689895
12	0.55	0.269531735	0.9600432	0.321100917
13	0.60	0.212520184	0.9721333	0.283747927
14	0.65	0.156253882	0.9815071	0.232231863
15	0.70	0.109054776	0.9886253	0.177966961
16	0.75	0.073903863	0.9934881	0.129587281
17	0.80	0.043721277	0.9966531	0.081115336
18	0.85	0.022233263	0.9985569	0.042879387
19	0.90	0.007949323	0.9995634	0.015703595
20	0.95	0.001490498	0.9999272	0.002974346

[1] "Max F1 0.383864726171448 at threshold 0.35"

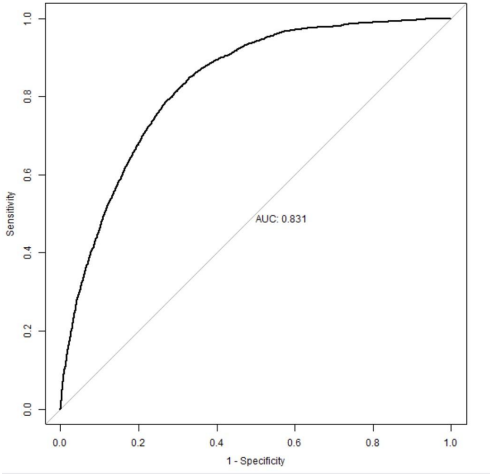
可以得出在threshold = 0.35時,
可以得到最大的F1 Score

XGBoost 結果

All Feature vs 13 Feature



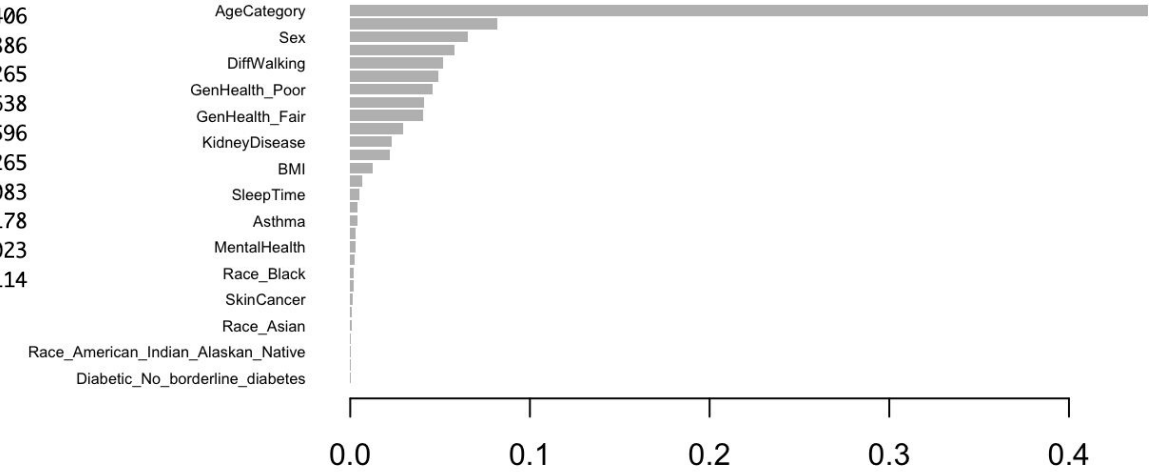
All Feature
AUC=0.834



13 Feature
AUC =0.831

XGBoost重要參數

Feature	Gain	Cover	Frequency
AgeCategory	0.4441518816	0.2048490163	0.141764406
Stroke	0.0816495303	0.0657671015	0.055583886
Sex	0.0652682038	0.0584358795	0.057332265
GenHealth_Excellent	0.0581343283	0.0384868896	0.034457638
DiffWalking	0.0516742155	0.0300137420	0.029649596
Diabetic_Yes	0.0492124508	0.0261172494	0.019669265
GenHealth_Poor	0.0460814248	0.0435628216	0.026517083
GenHealth_Very_Good	0.0408989579	0.0287238821	0.026954178
GenHealth_Fair	0.0405400615	0.0294886811	0.023166023
PhysicalHealth	0.0293500009	0.0422084196	0.057405114



元亨的部分

專案艱辛之處

1. 資料的不平衡:
   我們的資料集總共有 30萬筆資料，包含約21萬筆健康人資料和2萬筆病患資料，在這種不平衡的情況下，Null Model傾向將資料分類為健康，導致準確率很高，但 Specificity(True Negitive Rage)和 Sensitivity(True Positive Rate)卻極低。因此我們使用了 smote採樣平衡了健康人及病患的資料。一開始，我們也不知道該先切分 Training/Testing還是先smote，後來請教了老師後，才得知有很多人都在 這步驟錯誤地先smote才切資料集。
2. 程式的互相合作:
   我們總共採用了3種(決策數、隨機森林、Xgboost)模型來分析資料，實際寫了 5份建模程式及1份視覺化程式，在這些Code中，每個人的Code pattern, 前處理, 套件選用等都不盡相同，讓我們在溝通、整合程式時費了一番功夫。

資料前處理

1. 將資料轉為Numeric
2. Train和Valid比率切7:3
3. 將Train Data(約21萬筆資料)作smote, 將患病資料和未患病資料轉為約 1:1.33的比率(原先約1:10)
4. 將資料前處理模組化

分析結果(1)

未SMOTE

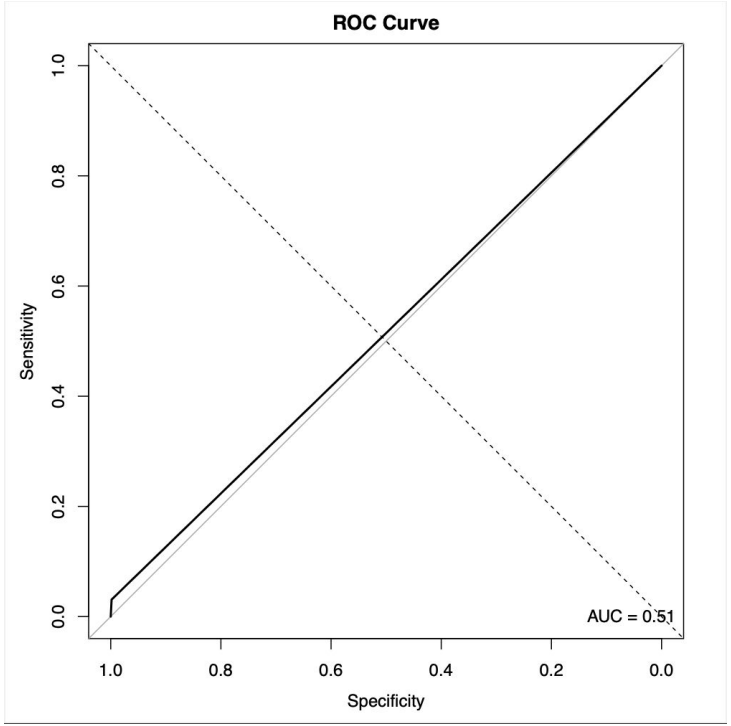
```
Before somte:

     0      1
204696  19162
After somte:

     0      1
204696  19162
Confusion Matrix:
   val_pred
        0     1
  0 87588   138
  1  7958   253
Accuracy:0.92
Precision:0.65
sensitivity(Recall):0.03
Specificity:1
F1 Score:0.06
Setting levels: control = 0, case = 1
Setting direction: controls < cases
AUC:0.51
Time takens:  2.702494  seconds.
```

已SMOTE:

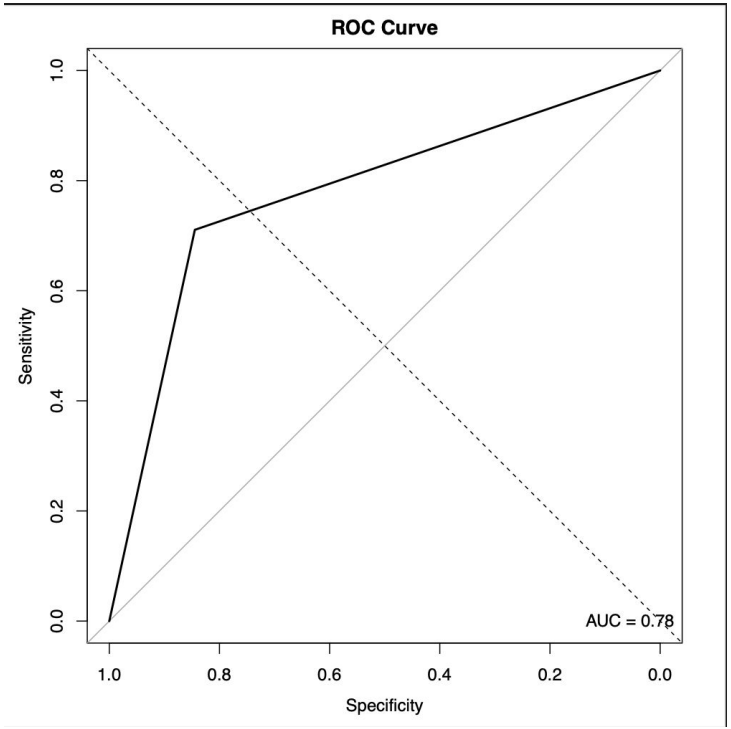
```
Before somte:

     0      1
204696  19162
After somte:

    0     1
76648 57486
Confusion Matrix:
   val_pred
        0     1
  0 73357 14369
  1  2941  5270
Accuracy:0.82
Precision:0.27
sensitivity(Recall):0.64
Specificity:0.84
F1 Score:0.38
Setting levels: control = 0, case = 1
Setting direction: controls < cases
AUC:0.74
Time takens:  4.666558  seconds.
```

分析結果(2)

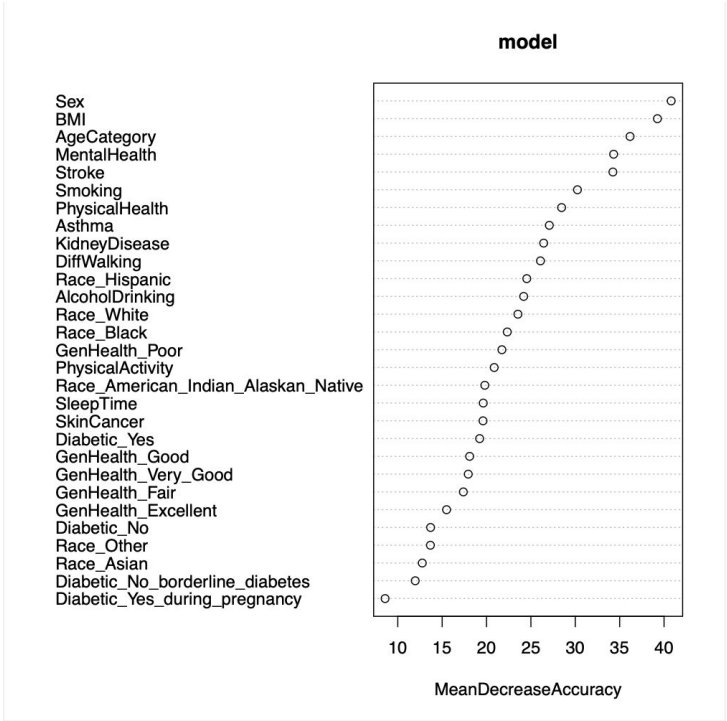
未SMOTE

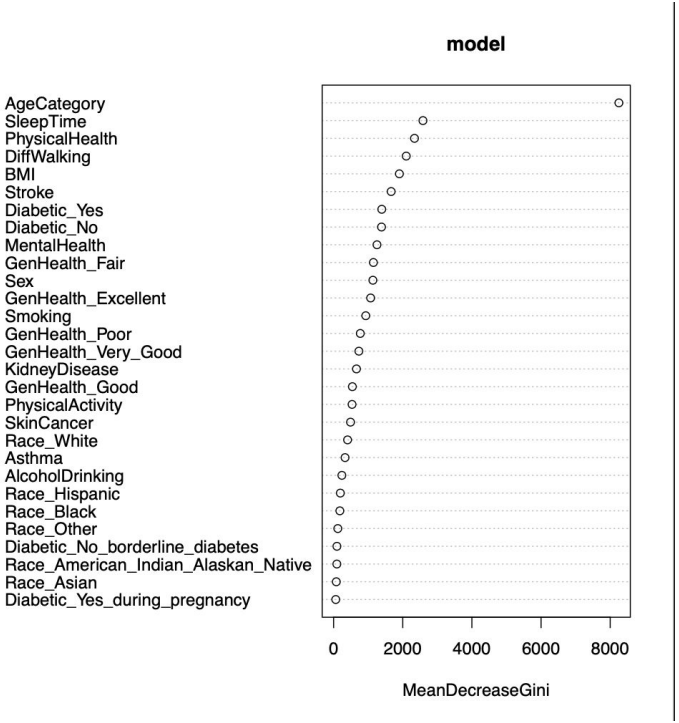


已SMOTE:



欄位重要性


model
Sex
BMI
AgeCategory
MentalHealth
Stroke
Smoking
PhysicalHealth
Asthma
KidneyDisease
DiffWalking
Race_Hispanic
AlcoholDrinking
Race_White
Race_Black
GenHealth_Poor
PhysicalActivity
Race_American_Indian_Alaskan_Native
SleepTime
SkinCancer
Diabetic_Yes
GenHealth_Good
GenHealth_Very_Good
GenHealth_Fair
GenHealth_Excellent
Diabetic_No
Race_Other
Race_Asian
Diabetic_No_borderline_diabetes
Diabetic_Yes_during_pregnancy
10
15
20
25
30
35
40
MeanDecreaseAccuracy


model
AgeCategory
SleepTime
PhysicalHealth
DiffWalking
BMI
Stroke
Diabetic_Yes
Diabetic_No
MentalHealth
GenHealth_Fair
Sex
GenHealth_Excellent
Smoking
GenHealth_Poor
GenHealth_Very_Good
KidneyDisease
GenHealth_Good
PhysicalActivity
SkinCancer
Race_White
Asthma
AlcoholDrinking
Race_Hispanic
Race_Black
Race_Other
Diabetic_No_borderline_diabetes
Race_American_Indian_Alaskan_Native
Race_Asian
Diabetic_Yes_during_pregnancy
0
2000
4000
6000
8000
MeanDecreaseGini

結論

在是否患有心臟病的分析中，Recall應比Precision重要

相比於原先患病與未患病的比率 (1:10), 雖然準確率下降 10%, 但Recall有顯著提高

階層式綱要(放書瑋投影片內)

Random Forest模型的成效 & 參數挑選

透過Random Forest 找出特徵的重要性、相似度

參考資料

[RandomForest in R](https://ithelp.ithome.com.tw/articles/10303882?sc=iThelpR)

宗佑的部分

(1)決策樹、xgboost模型測試，並且輸出confusion matrix以及ROC

(2)上述兩個模型皆測試使用smote 與不使用分析比較

(3)xgboost使用hinge進行預測，預測類別直接為0或1，並輸出confusion matrix。

(4)xgboost使用binary:logistic二分類邏輯回歸，得到預測類別的機率，並繪製importance matrix 以及決策樹。

xgboost importance matrix

	Feature	Gain	Cover	Frequency
1:	AgeCategory	0.3963859861	0.098128630	0.108067623
2:	BMI	0.0723353489	0.495949683	0.297881447
3:	Stroke	0.0703073358	0.025823177	0.021185534
4:	DiffWalking	0.0631028584	0.013315561	0.019901562
5:	Sex	0.0604583025	0.023259146	0.037021186
6:	GenHealth_Very_Good	0.0397905258	0.013742375	0.015407661
7:	Diabetic_Yes	0.0357454836	0.011907363	0.011983736
8:	GenHealth_Excellent	0.0351662227	0.020324929	0.008131821
9:	PhysicalHealth	0.0299906533	0.034584396	0.076824310
10:	GenHealth_Poor	0.0286423821	0.015333015	0.016263642
11:	GenHealth_Fair	0.0284400798	0.013786039	0.025037449
12:	SleepTime	0.0237249969	0.069364284	0.092017976
13:	MentalHealth	0.0188242512	0.028694214	0.076182324
14:	KidneyDisease	0.0182047581	0.012761860	0.016477637
15:	Smoking	0.0172359706	0.021936244	0.023539482
16:	GenHealth_Good	0.0136969951	0.007859916	0.018617590
17:	Race_White	0.0081791523	0.008379655	0.016905628
18:	Asthma	0.0075204233	0.010220116	0.017975605
19:	Diabetic_No	0.0064340072	0.005524867	0.013053713
20:	PhysicalActivity	0.0055503284	0.006615673	0.022897496
21:	SkinCancer	0.0044710271	0.005925147	0.015407661
22:	AlcoholDrinking	0.0032192894	0.009923763	0.008345816
23:	Race_Hispanic	0.0031114209	0.007711305	0.010485769
24:	Race_Black	0.0030704136	0.009317242	0.008131821
25:	Diabetic_No_borderline_diabetes	0.0015691831	0.004244621	0.005777873
26:	Race_American_Indian_Alaskan_Native	0.0015673495	0.006351062	0.005777873
27:	Race_Other	0.0013928564	0.004976199	0.005135887
28:	Race_Asian	0.0012912464	0.010017409	0.003209929
29:	Diabetic_Yes_during_pregnancy	0.0005711515	0.004022110	0.002353948

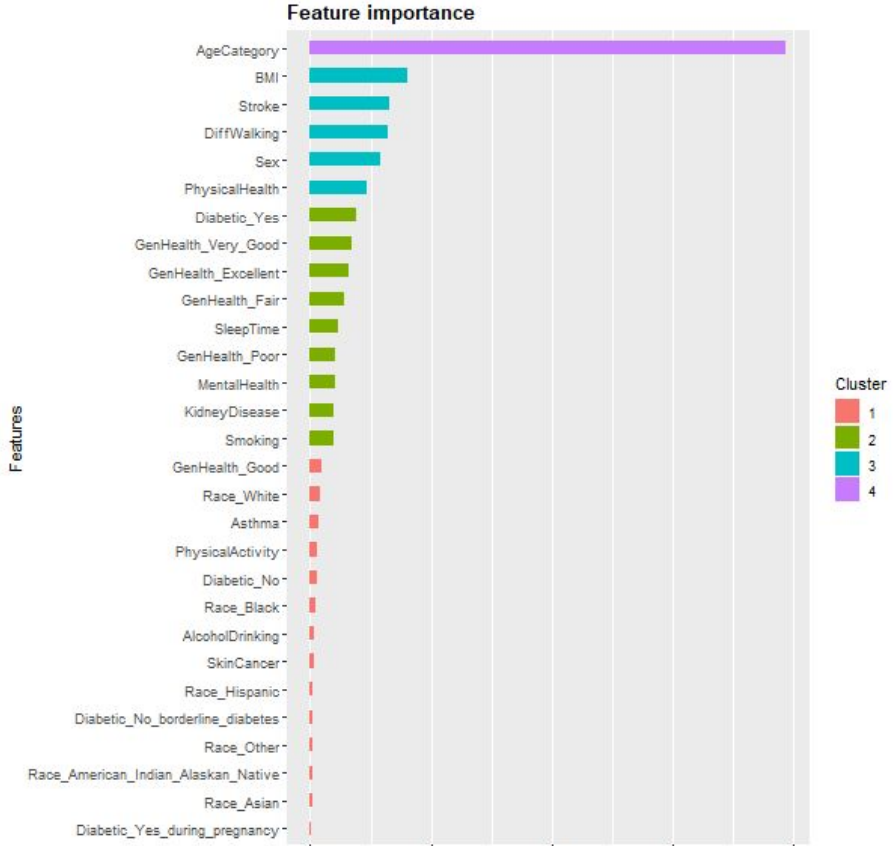
(1)Gain：使用該feature的所有分割的平均資訊增益（ average gain），也就是節點分裂時，該特徵帶來資訊增益（目標函數）優化的平均 值，是通過取每個feature對模型中每棵樹的貢獻來計算對應特徵對模型的相對貢獻。這個指標的 值越高，意味著它對生成預測更重要。
(2)Cover：
所有使用到 该feature的分割的平均覆盖率。

(3)Frequency:

频率是涉及特定特征的拆分的百分比，相 对于所做的每次拆分。您可以通过观察所有变量的频率总和为1来进行健全性检查

透過Gain可以發現Age這個features對於預測病人是否有心臟疾病相當重要。

xgboost importance matrix (kmeans 4 cluster)



透過上述Features中的Gain欄位進行kmeans分群。

初步分為四群結果如左圖。

可以看見Gain中最重要的為Age，之後為BMI、Stroke、DiffWalking、SEX、PhysicalHealth四個欄位。

xgboost 模型參數以及效能評估

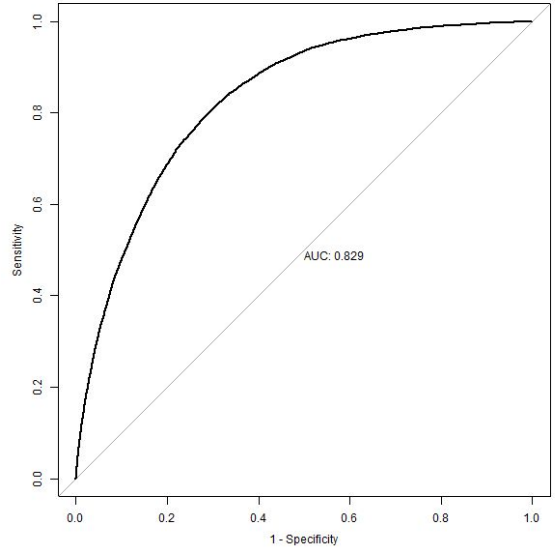
```
params <- list(
  "objective" = "binary:logistic",
  "eval_metric" = "error",
  "max_depth" = 6,
  "eta" = 0.3,
  "nthread" = 4
)
```

(1)Binary:logistic二分類邏輯回歸

(2)Eval_matrix: Error二進制分類錯誤率。對於預測，評估會將預測值大於 0.5 的實例視為正實例，將其他實例視為負實例

(3)最大深度6層

(3)學習率0.3



xgboost的AUC高達0.829，預測效果相當不錯。

xgboost 效能評估

threshold	sensitivity	specificity	F1
0.00	1.000000000	0.0000000	0.165724360
0.05	0.964049890	0.4207535	0.247317899
0.10	0.913426266	0.5586971	0.287396122
0.15	0.863536317	0.6520440	0.321804511
0.20	0.797138665	0.7178824	0.343774719
0.25	0.726705796	0.7740654	0.363219655
0.30	0.655539252	0.8196094	0.377641589
0.35	0.578503302	0.8589594	0.385857597
0.40	0.498532649	0.8922612	0.385969895
0.45	0.435069699	0.9180208	0.384939955
0.50	0.365370506	0.9386432	0.368479467
0.55	0.297872340	0.9546018	0.339464883
0.60	0.235143067	0.9683378	0.302644004
0.65	0.165810712	0.9796692	0.241970021
0.70	0.125458547	0.9864825	0.198895028
0.75	0.084372707	0.9917656	0.144563168
0.80	0.048789435	0.9955549	0.089231801
0.85	0.026045488	0.9979961	0.049789621
0.90	0.011371974	0.9994899	0.022374594
0.95	0.001100514	0.9999271	0.002196997

"Max F1 0.385969894916217 at threshold 0.4"

針對訓練好的xgboost模型進行不同threshold判斷Sensitivity, Specificity F1比較

本主題為探討心臟疾病預測，雖在診斷過程中準確預測出患者得到心臟病相較於將無病患者診斷為心臟病者更為重要，惟誤診為心臟病患者圖樣會降低病患信任，因此在兼具兩者情況下取 F1最大值為0.4。

專案艱辛之處

由於小組成員並無人從事資料科學相關行業，前期只能由小組成員一步一步針對老師上課 內容互相討論。

本課題遇到最大兩個困難處為

1.資料集不平衡

本主題資料集相當不平均，患者與非患者為 1比13，基本上使用null model全部猜測患者無病，準確性即高達九成多。然而這樣的猜測沒有太大意義，因此我們針對不平衡的資料瀏覽許多資料，最後透過 smote，降低樣本比例至1比4，提高sensitivity。

2.特徵挑選

原本資料集有18個欄位，但有些特徵的數 值為種類，例如膚色……，因此先進行轉換，轉換後欄位高達 29個。為此我們思考是否有些欄位重要較低，因此使用不同方式進行評估，最後挑選了 13個欄位，其最終auc結果與29個欄位相差不多。

資料前處理完成後，彼此分工針對不同模型進行測試，由於每個人撰寫程式碼的習慣不同，因此整合階段也額外花費許多時間。分組報告並非單獨個體，必須透由成員彼此討論、分工合作，才能有效解決問題。如果無法融入群體，只會拖累彼此，阻礙進度。所幸本小組最後還是非常感謝大家積極參與、熱烈討論，順利完成本次課題，令我不只在學科上獲得知識，也在人際互動受益良多。

昇豐的部分

(1) 範例：**資料前處理**，我想表達的階層式綱要如下：

1. 資料集的NA。
2. 資料集的特徵種類。
    2.1. 數值型、binary、類別型等等。
3. 資料集有病無病的不平衡狀況。

4. 為了建模需要，進行如何的前處理。
5. 未處理、前處理後的資料集差異對照。

原始資料集的分析問題：NA、重複、特徵類型、不平衡

```
> str(dataset)
'data.frame':   319,795 obs. of  18 variables:
 $ HeartDisease    : chr  "No" "No" "No" "No" ...
 $ BMI             : num  16.6 20.3 26.6 24.2 23.7 ...
 $ Smoking         : chr  "Yes" "No" "Yes" "No" ...
 $ AlcoholDrinking : chr  "No" "No" "No" "No" ...
 $ Stroke          : chr  "No" "Yes" "No" "No" ...
 $ PhysicalHealth  : num  3 0 20 0 28 6 15 5 0 0 ...
 $ MentalHealth    : num  30 0 30 0 0 0 0 0 0 0 ...
 $ DiffWalking     : chr  "No" "No" "No" "No" ...
 $ Sex             : chr  "Female" "Female" "Male" "Female" ...
 $ AgeCategory     : chr  "55-59" "80 or older" "65-69" "75-79" ...
 $ Race            : chr  "White" "White" "White" "White" ...
 $ Diabetic        : chr  "Yes" "No" "Yes" "No" ...
 $ PhysicalActivity: chr  "Yes" "Yes" "Yes" "No" ...
 $ GenHealth       : chr  "Very good" "Very good" "Fair" "Good" ...
 $ SleepTime       : num  5 7 8 6 8 12 4 9 5 10 ...
 $ Asthma          : chr  "Yes" "No" "Yes" "No" ...
```

target column, binary

numeric；4 個

binary；9 個

category；4 個

```
summary(duplicated(dataset))
Mode   FALSE    TRUE
logical  301,717   18,078

table(dataset$HeartDisease)
    No     Yes
292,422  27,373
```

為了建模需要，進行如何的前處理：

1. 特徵屬性有numeric、binary、category：
   應對方式：

特徵屬性	作法	說明
numeric	--	–
binary	轉成numeric	1. Yes->1 2. No->0
category	轉成numeric 擴展成dummy col	1. 特徵AgeCategory:chr "55-59" -> 中位數57 2. 特徵Race-> Race_White、Race_Black、... a. 移除特徵Race

2. 有18,078列重複：
   應對方式：去重，**319,795** - 18,078 = 301,717
3. 不平衡資料：
   應對方式：對去重後的訓練集資料取平衡，再建立模型；

資料集比例	沒病	有病
未取平衡後的比例	10	1
取平衡後的比例	1.3~4	1
備註	21萬取平衡，占比1.3； 10萬取平衡，占比4	--

切3個fold，使用決策樹	取平衡的訓練集	未取平衡的測試集	未取平衡的驗證集
建立模型	V	--	--
模型預測	V	V	V

未處理、前處理後的資料集差異對照：

	未處理	前處理後
資料筆數	319,795	301,717
不平衡的狀況	沒病　有病 292,422　27,373	沒病　有病 274,456　27,261
不平衡的比例	10:1	10:1
取平衡後的比例	--	訓練集資料呈現1.3~4:1
欄位數量(扣除target)	17	29
特徵屬性轉換示意		

$ HeartDisease : chr "No" "No" "No" "No" ...
$ **BMI** : **num** 16.6 20.3 26.6 24.2 23.7 ...
$ Smoking : chr "Yes" "No" "Yes" "No" …
$ AgeCategory : chr "55-59" "80 or older" "65-69" "75-79" ...
$ Race : chr "White" "White" "White" "White" ...

$ HeartDisease : chr "No" "No" "No" "No" ...
$ **BMI** : **num** 16.6 20.3 26.6 24.2 23.7 ...
$ Smoking : num 1 0 1 0 0 1 0 1 0 0 ...
$ AgeCategory : num 57 80 67 77 42 77 72 80 80 67 ...
$ Race_American_Indian_Alaskan_Native: int 0 0 0 0 0 0 0 0 0 0 ...
$ Race_Asian : int 0 0 0 0 0 0 0 0 0 0 ...
$ Race_Black : int 0 0 0 0 0 1 0 0 0 0 ...
$ Race_Hispanic : int 0 0 0 0 0 0 0 0 0 0 ...
$ Race_Other : int 0 0 0 0 0 0 0 0 0 0 ...
$ Race_White : int 1 1 1 1 1 0 1 1 1 1 ...

(1) 範例：**各模型的執行情況**，階層式綱要如下：

1. 介紹優劣

    1.1. 不建議用決策樹，

    1.2. 以AUC當作評選模型的指標

2. 建議的最終方案

    2.1. 介紹此方案

model	決策樹	隨機森林	XGBoost
k-fold/ training+test	k-fold, k = 3	training+test 70%+30%	training+test 70%+30%
smote	有使用smote, trainset=100,000	有使用smote, trainset=210,000	有使用smote, trainset=210,000
threshold(閥值)	--	--	基於F1得出 29 -> 0.4 13 -> 0.35
model 參數	minsplit = 5, minbucket = 5, cp = 0.001	ntree=100, mtry=3	objective = binary:logistic, eval_metric = error, max_depth = 6, eta = 0.3
feature selection	前處理後，產生29 feature。 基於glm，得出13 feature。		
Accuracy	29 -> 0.89 13 -> 0.89	29 -> 0.82 13 -> 0.77	29 -> 0.857 13 -> 0.834
Precision	29 -> 0.38 13 -> 0.38	29 -> 0.27 13 -> 0.23	29 -> 0.314 13 -> 0.288

各模型執行情況_1

model	決策樹	隨機森林	XGBoost
苦主	昇豐	元亨/書瑋	宗佑/正晏
k-fold/ training+test	k-fold, k = 3	training+test 70%+30%	
smote(處理不平衡)	smote(trainset=100,000) 沒病:有病 = 4:1	smote(trainset=210,000) 沒病:有病 = 1.3:1	
threshold(閥值)	--	--	基於F1得出 29 -> 0.4 13 -> 0.35
model 參數	minsplit = 5, minbucket = 5, cp = 0.001	ntree=100, mtry=3	objective = binary:logistic, eval_metric = error, max_depth = 6, eta = 0.3
feature selection	前處理後，產生**29 feature**。 基於**glm**，得出**13 feature**。		

各模型執行情況_2

model	決策樹		隨機森林		XGBoost	
Accuracy	**29 -> 0.89**	**13 -> 0.89**	**29 -> 0.82**	**13 -> 0.77**	**29 -> 0.857**	**13 -> 0.834**
Precision	**0.38**	**0.38**	**0.27**	**0.23**	**0.314**	**0.288**
Recall= Sensitivity= True positive rate	**0.34**	**0.33**	**0.65**	**0.74**	**0.506**	**0.576**
Specificity= True negative rate	**0.95**	**0.95**	**0.83**	**0.77**	**0.891**	**0.860**
F1	**0.35**	**0.35**	**0.38**	**0.36**	**0.388**	**0.384**
AUC	**0.64024**	**0.6357**	**0.74**	**0.76**	**0.833**	**0.830**

參考家銘老師的建議，**以AUC決定模型優劣，建議使用隨機森林、XGBoost處理**。

理由：資料集是不平衡(沒病：有病約10：1)，直觀預測沒病也有10/11的準確率，造成選用AUC以外的指標較無意義。

(1) 範例：**介紹最終建議方案**，階層式綱要如下：

2. 建議的最終方案

2.1. 特徵工程

1. chi-squre看看特徵跟output的顯著性

1.1. 未處理與前處理後的資料，其chi-saure test的比較：

未處理與前處理後的資料欄位們，都與HeartDisease有相關(dependent)。

2. 透過邏輯回歸(glm)，得出欄位重要性指標，評選出13個欄位

2.1. 由共線性檢查，13個欄位沒有共線性。

2.2. 介紹此方案

1. xgboost

2. random forest

chi-squre檢視欄位跟target HeartDisease的顯著性：

未處理，17個欄位

col_order	col_name	X-squared	df	p-value
2	BMI	6724.613578	3603	2.49E-192
3	Smoking	3713.815575	1	0
4	AlcoholDrinking	329.1041963	1	1.51E-73
5	Stroke	12390.18061	1	0
6	PhysicalHealth	9735.616088	30	0
7	MentalHealth	971.4189962	30	5.50E-185
8	DiffWalking	12953.23319	1	0
9	Sex	1568.808198	1	0
10	AgeCategory	19299.92039	12	0
11	Race	844.314886	5	2.99E-180
12	Diabetic	10959.86128	3	0
13	PhysicalActivity	3199.864826	1	0
14	GenHealth	21542.17736	4	0
15	SleepTime	2303.946242	23	0
16	Asthma	549.2855397	1	1.80E-121
17	KidneyDisease	6741.981347	1	0
18	SkinCancer	2784.787544	1	0

前處理後，29個欄位

col_order	col_name	X-squared	df	p-value
2	BMI	6210.709	3603	7.40E-143
3	Smoking	3296.317	1	0
4	AlcoholDrinking	397.3195	1	2.11E-88
5	Stroke	11433.4	1	0
6	PhysicalHealth	8597.503	30	0
7	MentalHealth	1065.589	30	7.13E-205
8	DiffWalking	11640.48	1	0
9	Sex	1671.667	1	0
10	AgeCategory	18912.37	12	0
11	PhysicalActivity	2643.152	1	0
12	SleepTime	1901.084	23	0
13	Asthma	386.3422	1	5.18E-86
14	KidneyDisease	6141.505	1	0
15	SkinCancer	2479.035	1	0
16	Race_American_Indian_Alaskan_Native	12.66774	1	0.000372019
17	Race_Asian	325.408	1	9.62E-73
18	Race_Black	63.58335	1	1.54E-15
19	Race_Hispanic	499.2888	1	1.36E-110
20	Race_Other	11.13859	1	0.0008455
21	Race_White	721.2408	1	7.19E-159
22	Diabetic_No	8310.73	1	0
23	Diabetic_No_borderline_diabetes	57.39683	1	3.56E-14
24	Diabetic_Yes	9658.298	1	0
25	Diabetic_Yes_during_pregnancy	72.56368	1	1.62E-17
26	GenHealth_Excellent	3867.477	1	0
27	GenHealth_Fair	6192.675	1	0
28	GenHealth_Good	304.1095	1	4.19E-68
29	GenHealth_Poor	8971.392	1	0
30	GenHealth_Very_Good	3049.853	1	0

- pseudo code：chisq.test(dataset$HeartDisease, dataset[,i], correct=FALSE)
- **p-value < 0.05, reject H0**，所有欄位都與HeartDisease有相關(dependent)

借助glm，得出欄位重要性、共線性：

pseudo code：

(1) preprocess(dataset)。

(2) suffle(dataset)。

(3) 取平衡(dataset[100000,])，去除na，

得出45,000筆，36000沒病：9000有病。

(4) model <- glm(HeartDisease~.), family="binomial"

, data=dataset)

13個欄位的VIF*數值皆小*，表示無共線性。

因此，**決定選取此13個欄位**。

```
> car::vif(model)
            BMI         Smoking          Stroke     DiffWalking
       1.108250        1.031146        1.021945        1.297409
            Sex     AgeCategory         Asthma   KidneyDisease
       1.059703        1.104021        1.053452        1.036308
 Race_Hispanic GenHealth_Excellent    GenHealth_Fair    GenHealth_Good
       1.024754        1.229081        1.623939        1.585377
GenHealth_Poor
       1.406402
```

欄位重要性，取>4，13個欄位

```
> caret::varImp(model)
                                      Overall
BMI                                4.4294187
Smoking                           13.0590904
AlcoholDrinking                    1.9100625
Stroke                            23.3059047
PhysicalHealth                     1.5127184
MentalHealth                       2.1510181
DiffWalking                        6.5815495
Sex                               25.4474306
AgeCategory                       47.6255661
PhysicalActivity                   1.3458511
SleepTime                          2.1629561
Asthma                             7.2590399
KidneyDisease                      9.7363807
SkinCancer                         1.7886560
Race_American_Indian_Alaskan_Native 0.4889626
Race_Asian                         3.5278293
Race_Black                         3.8891932
Race_Hispanic                      4.0550250
Race_Other                         1.3882326
Diabetic_No                        1.3356281
Diabetic_No_borderline_diabetes    1.0085721
Diabetic_Yes                       1.2487449
GenHealth_Excellent                7.0720475
GenHealth_Fair                    21.7952554
GenHealth_Good                    14.6508521
GenHealth_Poor                    20.6333198
```

XGBoost_1_參數與演算法選擇:

(1) 參數說明

1. max_depth:樹的最大深度, 使用max_depth=6。
   目的:數值愈大模型擬合度越高。
2. eta:又稱為learning_rate, 使用預設值0.3。
   目的:此參數用於防止over fitting。
3. eval_metric = error : 此為二進制分類錯誤率, 預設使用0.5來判斷。
   目的:評估每回迭代的分類效果, 公式「#(wrong cases) 除以 #(all cases)」

(2) 演算法選擇

1. binary:logistic: 羅吉斯回歸, model對每一筆預測資料輸出機率

   考量實務面, 選用logistic。理由:使用閥值(threshod)、產生ROC圖。

2. binary:hinge: 二元分類中使用hinge loss作為loss function進行分類,
   model對每一筆預測資料輸出2元結果(1, 0)

XGBoost_2_程式碼與使用流程:

資料前處理
one-hot encoding
Smote轉換

選擇13 Feature
轉換dataset成
DMatrix格式

XGBoost Model

```
train_matrix_selected | External pointer of class 'xgb.DMatrix'
> print(train_matrix_selected)
xgb.DMatrix  dim: 95855 x 13  info: label  colnames: yes
```

DMatrix 是 XGBoost 使用的內部資料結構，它會優化內存效率和訓練速度。

```r
features_selected <- c("BMI","Smoking","Stroke","DiffWalking","Sex","AgeCategory","Asthma",
"KidneyDisease","Race_Hispanic","GenHealth_Excellent","GenHealth_Fair","GenHealth_Good","GenHealth_Poor")
# features_selected <- c(1, 2, 4, 7, 8, 9, )
train_features_selected <- train_df[, features_selected]
test_features_selected <- test_df[, features_selected]
train_features <- as.matrix(train_features_selected)
train_labels <- as.matrix(factor(train_df[, 1]))

test_features <- as.matrix(test_features_selected)
test_labels <- as.matrix(test_df[, 1])

train_matrix_selected <- xgb.DMatrix(data = train_features, label = train_labels)
test_matrix_selected <- xgb.DMatrix(data = test_features)
```

```r
params <- list(
  "objective" = "binary:logistic",
  "eval_metric" = "error",
  "max_depth" = 6,
  "eta" = 0.3,
  "nthread" = 4
)

model_xgboost_selected <- xgb.train(params = params, data = train_matrix_selected, nrounds = 100)
```

XGBoost_3_threshold(閥值)選取方式：

	threshold	sensitivity	specificity	F1
1	0.00	1.000000000	0.0000000	0.163362620
2	0.05	0.960004968	0.4110278	0.240221293
3	0.10	0.916532108	0.5503129	0.281035172
4	0.15	0.852937523	0.6491317	0.313189820
5	0.20	0.788970314	0.7198414	0.338719138
6	0.25	0.719165321	0.7789823	0.361118907
7	0.30	0.639920507	0.8252450	0.373144057
8	0.35	0.566761893	0.8646682	0.383864726
9	0.40	0.480934045	0.8972521	0.379682291
10	0.45	0.411377469	0.9216749	0.371675457
11	0.50	0.341696684	0.9423506	0.353689895
12	0.55	0.269531735	0.9600432	0.321100917
13	0.60	0.212520184	0.9721333	0.283747927
14	0.65	0.156253882	0.9815071	0.232231863
15	0.70	0.109054776	0.9886253	0.177966961
16	0.75	0.073903863	0.9934881	0.129587281
17	0.80	0.043721277	0.9966531	0.081115336
18	0.85	0.022233263	0.9985569	0.042879387
19	0.90	0.007949323	0.9995634	0.015703595
20	0.95	0.001490498	0.9999272	0.002974346

[1] "Max F1 0.383864726171448 at threshold 0.35"

1. 採用13個Feature，基於binary:logistic搭配不同threshold，產生三種評估指標。

2. 已知資料集是不平衡(沒病：有病約10：1)，**成功檢測出患者有病才是重要的**。

3. 若一味降低threshold提高Precision，會擴大沒病者判定為有病的情況，此是不樂見的。

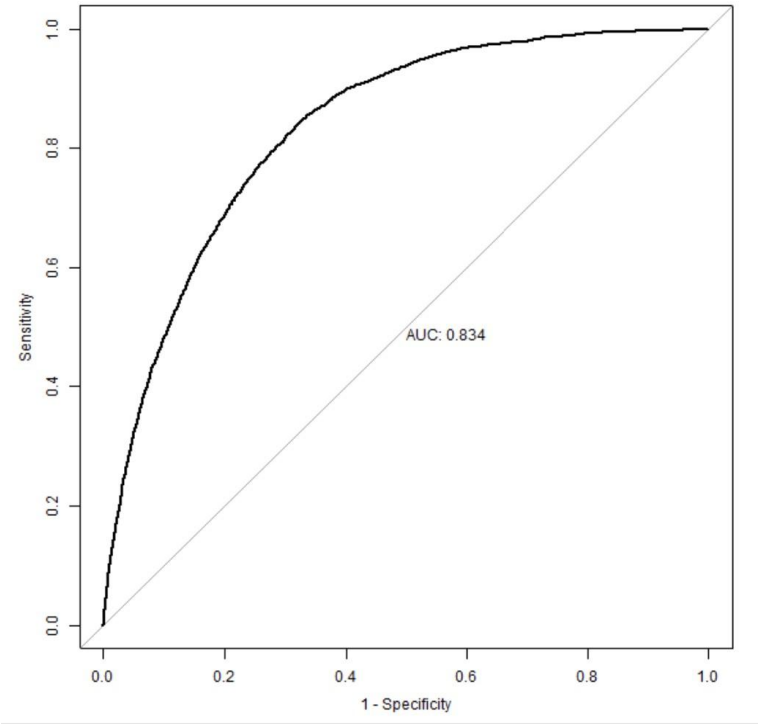
因此以綜合考量Precision and Recall的F1，

在F1有最大值 = 0.3838，採取threshold = 0.35進行建模。

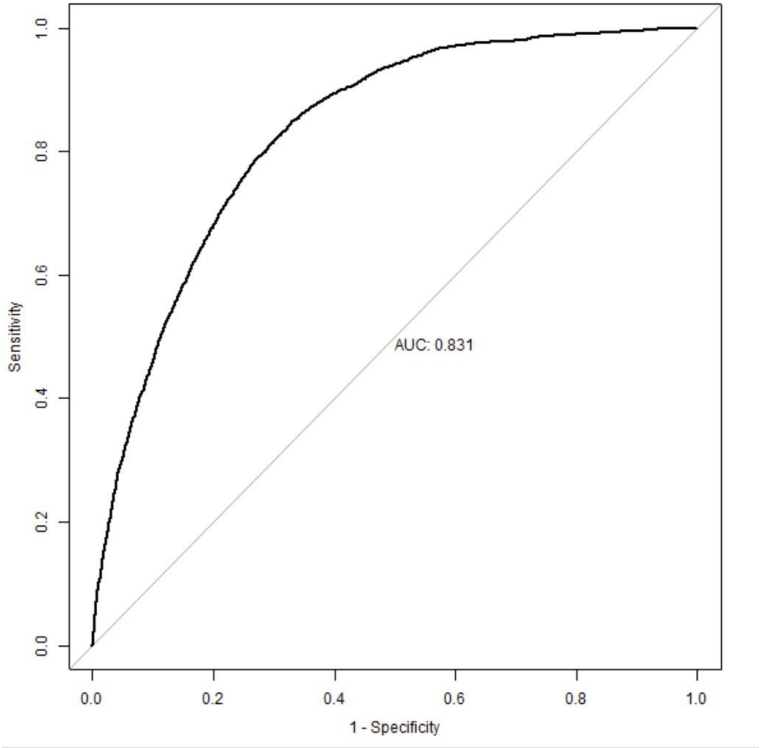
The F1 score

- $\frac{2}{\frac{1}{precision}+\frac{1}{recall}} = \frac{2*precision*recall}{precision+recall} = \frac{2*TP}{2*TP+FP+FN}$

XGBoost_4_建模後對test資料集的預測結果:



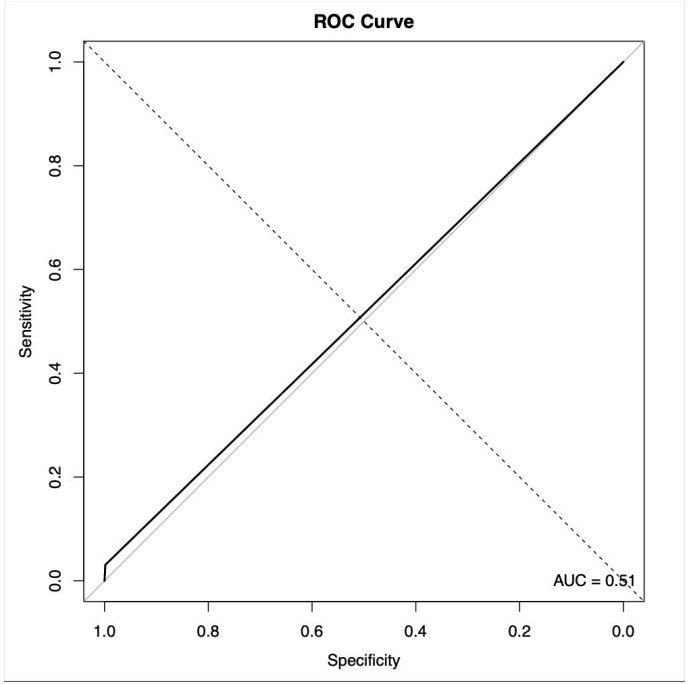
29 Feature
AUC = 0.834

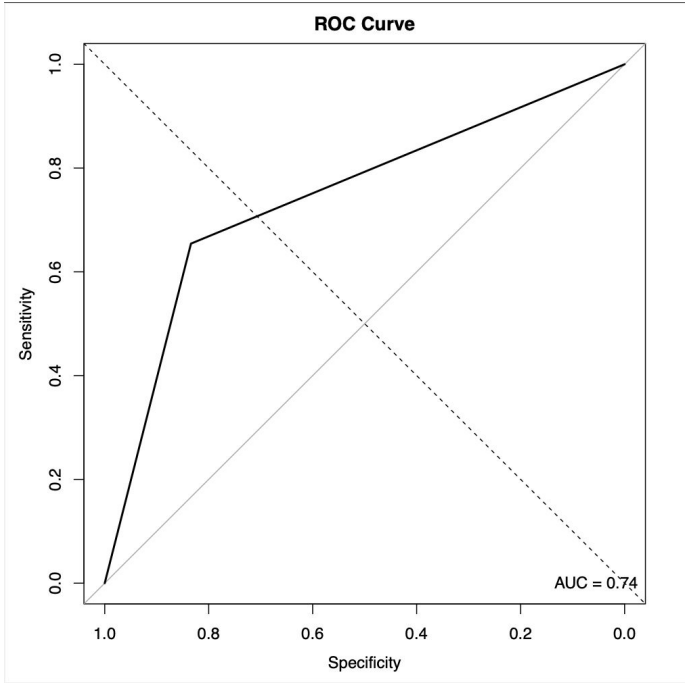


13 Feature
AUC = 0.831

test資料集是不平衡(沒病:有病約10:1)

隨機森林_1_AUC變化_使用smote處理不平衡資料集的必要性:





No Using smote

traing資料集是<u>不平衡</u>(沒病:有病約10:1)

29 Feature
AUC = 0.51

Using smote

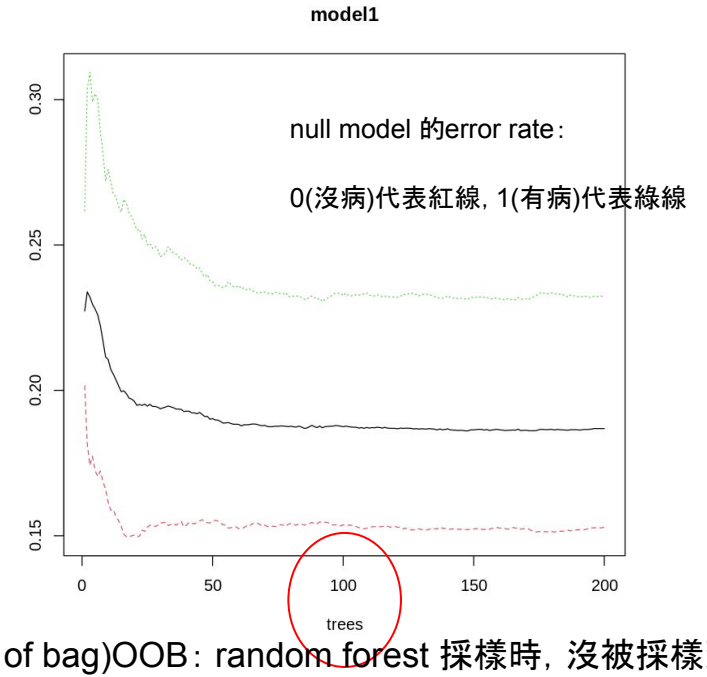
traing資料集是<u>平衡</u>(沒病:有病約1.3:1)

29 Feature
AUC = 0.74

隨機森林_2_超參數數值設定_1_圖示:

ntree:樹的數量

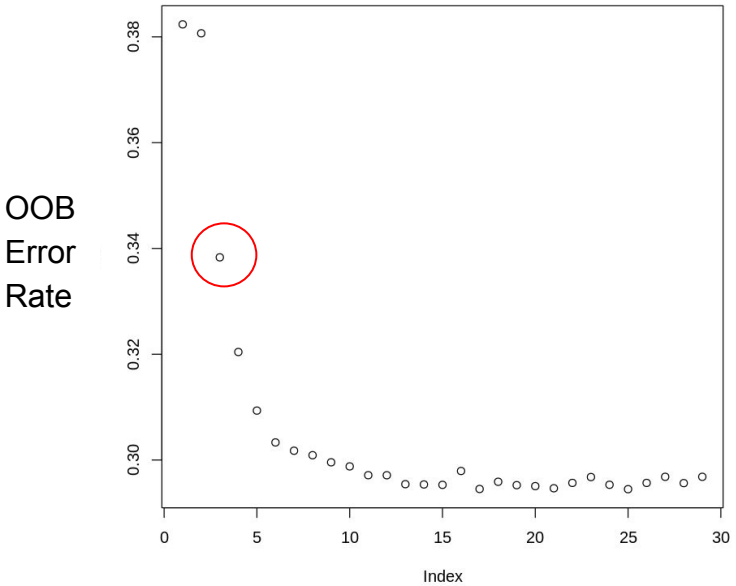
參數預設是500,, 數據觀察在100左右處於穩定



mtry:每棵樹使用的特徵數量

參數預設為sqrt(**29 Feature**), 也就是5左右;

等於3時, OOB Error Rate**降幅**最大, 因此設定為3。



(out of bag)OOB: random forest 採樣時, 沒被採樣到的資料。

利用沒被選到的資料進行模型的**驗證評估**, 得出OOB error rate = 1 - (OOB data猜對數量 / (OOB data全部數量))

隨機森林_2_超參數數值設定_2_數據佐證:

ntree:樹的數量

0、1 欄位為null model 的error rate, 0代表上一張投影片的紅線,1代表綠線

1	ntree	OOB	0	1
90	89	0.1875289	0.1541984	0.2319695
91	90	0.187171	0.1540679	0.2313085
92	91	0.1877302	0.1547594	0.2316912
93	92	0.1871263	0.1545637	0.2305431
94	93	0.1875065	0.1546811	0.2312737
95	94	0.1875885	0.1545246	0.2316738
96	95	0.1876705	0.1542897	0.2321783
97	96	0.1876854	0.1537287	0.2329611
98	97	0.1878942	0.1537548	0.2334134
99	98	0.1878569	0.1537157	0.2333786
100	99	0.1876631	0.1533765	0.2333786
101	100	0.1874842	0.1537287	0.2324914
102	101	0.1877227	0.1534548	0.2334134
103	102	0.187514	0.1536244	0.2327001
104	103	0.1874171	0.1536374	0.2324566
105	104	0.1873201	0.1532721	0.2327175
106	105	0.1873127	0.1531677	0.2328393
107	106	0.1870219	0.1528024	0.2326479
108	107	0.1872083	0.1527241	0.2331872
109	108	0.1869399	0.1523327	0.2330828
110	109	0.1872381	0.1525806	0.2334481
111	110	0.1870219	0.1527894	0.2326653
112	111	0.1871934	0.1530764	0.2326827
113	112	0.1871189	0.1531286	0.2324392
114	113	0.1873052	0.1531677	0.2328219
115	114	0.1872158	0.1530894	0.2327175
116	115	0.1870368	0.1531547	0.2322131
117	116	0.1872605	0.1533504	0.232474
118	117	0.1870145	0.1532591	0.2320217

更多數值

https://docs.google.com/spreadsheets/d/1IM2-4hOuFyP3FJt-YIv7QdDRmdFKmODZmsDmDpPo974/edit#gid=1549750515

mtry:每棵樹使用的特徵數量

2	mtry	error rate
12	1	0.3823684
13	2	0.3806774
14	3	0.3383198
15	4	0.3204217
16	5	0.3093427
17	6	0.3033233
18	7	0.301757
19	8	0.3009073
20	9	0.2995667
21	10	0.2987895
22	11	0.2971263
23	12	0.2971186
24	13	0.2954294
25	14	0.2953758
26	15	0.2953032
27	16	0.2979234
28	17	0.2945221
29	18	0.2958868
30	19	0.2952444
31	20	0.2950603
32	21	0.2946573
33	22	0.2956707
34	23	0.2967743
35	24	0.2953179
36	25	0.2944851
37	26	0.2956685

虛擬碼:

```
model <- randomForest(as.factor(HeartDisease)~. ,data=train_data, ntree=100, mtry=i)
```

隨機森林_3_程式碼:

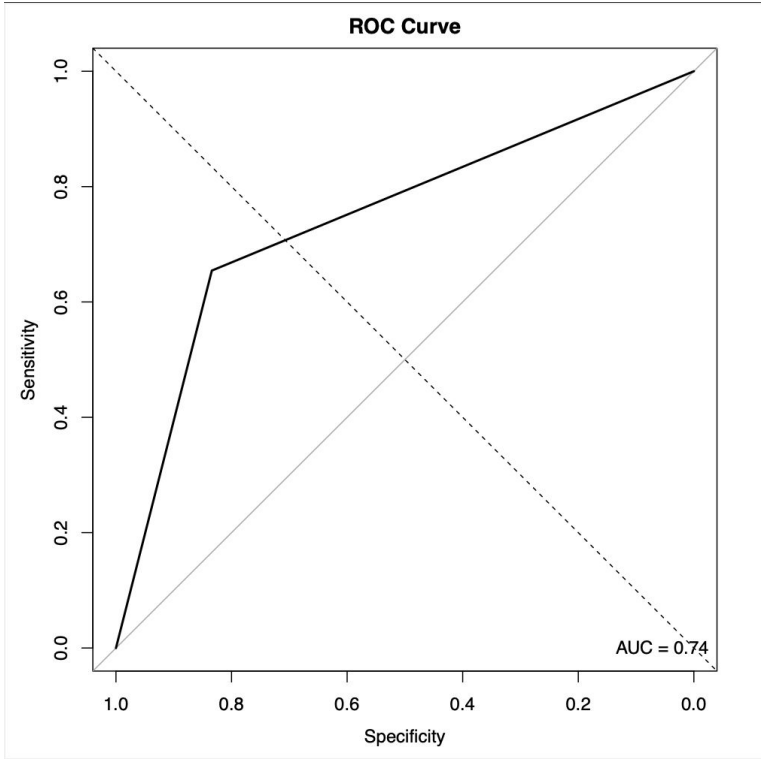
選用13個特徵:

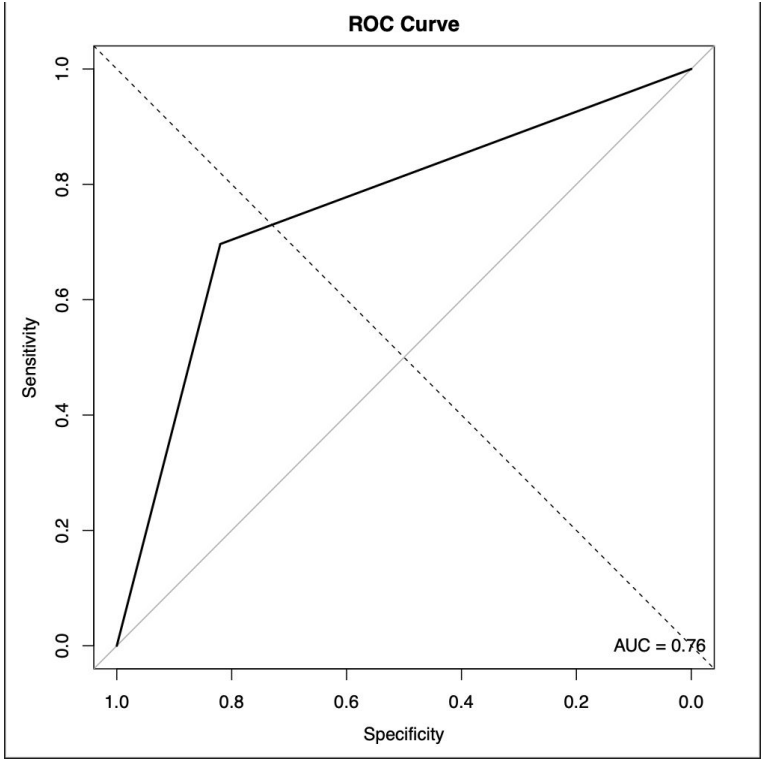
```
train.index <- createDataPartition(df$HeartDisease, p = .7, list = FALSE)
train_df <- df[train.index,]
train_df <- smote(HeartDisease ~ ., data = train_df)
train_df <- na.omit(train_df)
selected_features = c('HeartDisease', 'BMI', 'Smoking', 'Stroke',
 'DiffWalking', 'Sex', 'AgeCategory', 'Asthma', 'KidneyDisease',
 'Race_Hispanic', 'GenHealth_Excellent', 'GenHealth_Good', 'GenHealth_Fair', 'GenHealth_Poor')
train_df <- train_df[, selected_features]
train_df$HeartDisease <- as.factor(train_df$HeartDisease)
```

建模:

```
model <- randomForest(HeartDisease ~ ., data=train_df, ntree=100, mtry=3, importance=TRUE)
```

隨機森林_4_建模後對test資料集的預測結果:





29 Feature
AUC = 0.74

13 Feature
AUC = 0.76

test資料集是不平衡(沒病:有病約10:1)

再次重申評估與結論:

透過兩者的比對, 得出分析價值是:(1) 確定無相關的特徵**其實是重要的**;
(2)得出不具有共線性效果的特徵, 既**維持效能**又**提升**建立模型的**效率**。

原始資料集觀察			
統計直觀	Feature	No, 有病%	Yes, 有病%
有相關	**Smoking**	7.47	36.4
	Stroke	7.47	36.4
	AgeCategory	(50-54)5.45	22.6(80 up)
	KidneyDisease	7.77	29.3
	GenHeath	(Excellent)2.24	34.1(Poor)
	DiffWalking	6.3	22.6
無相關	**BMI**	27.26	28.34
	Sex(Gender)	10.6(Male)	6.69(Female)
	Asthma	8.1	11.5
	Race	3.3(Asian)	10.4(Indian)

特徵工程		
Feature	caret::varImp (glm_model)	car::vif (glm_model)
Race_Hispanic	4.0550250	1.024754
BMI	4.4294187	1.108250
DiffWalking	6.5815495	1.297409
GenHealth_Excellent	7.0720475	1.229081
Asthma	7.2590399	1.053452
KidneyDisease	9.7363807	1.036308
Smoking	13.0590904	1.031146
GenHealth_Good	14.6508521	1.585377
GenHealth_Poor	20.6333198	1.406402
GenHealth_Fair	21.7952554	1.623939
Stroke	23.3059047	1.021945
Sex(Gender)	25.4474306	1.059703
AgeCategory	47.6255661	1.104021
備註:glm的欄位重要性, 取>4, 13個欄位;VIF值皆 < 2		

DEMO說明

Feature	Accuracy	Precision	Sensitivity	Specificity	F1.Score	AUC
29Features	0.9	0.41	0.24	0.97	0.31	0.6
13Features	0.9	0.41	0.28	0.96	0.33	0.62

(1) Rscript語法

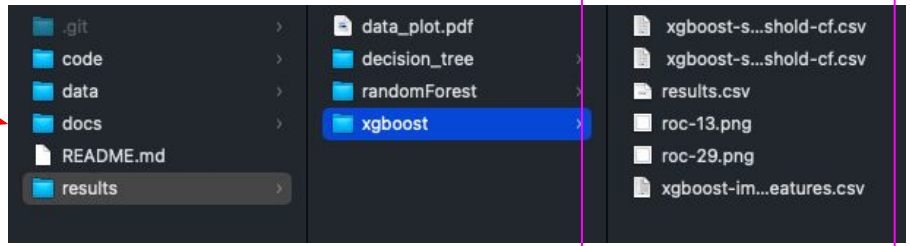
0. cd to code folder.
1. 產生資料分析圖示

Rscript data_plot.R

2. 執行主程式

Rscript main.R --d 2 --m 2

Rscript main.R --d 2 --m 3



(2)參數說明

--data_source or --d

1：讀取原資料，會處理dummy和smote

2：讀取已處理資料

--model or --m

1：decision tree (不支援data_source = 2，且會耗時3分鐘以上)

2：random forest

3：xgboost

隨機森林、XGBoost產生檔案交集：

1. result.csv：模型預測評估結果。
2. roc-13、roc-29：不同Freature的AUC表現。
3. ImportantFeature：兩個模型產生的欄位重要性。

(1) 範例：**專案甘苦談與專案引用的套件與外部資料**，階層式綱要如下：

1. 專案甘苦談

(1) 從「思想的巨人，行動的侏儒」中脫離，進行**行動力**變革。讀過不少ML、DS、統計的書，但工作既不是此領域、大學的統計也沒學好，因此這段期末報告的準備過程真的是**練習對抗的過程**；所幸考試後的課程會提點分析工具選用的訣竅外，老師也願意給予意見。組員也願意與信任地按照我仿照課堂教科書訂定的報告綱要準備。

(2) 期末主題是連續5個禮拜與老師討論後才決定。除了了解自己對資料分析的選題眼界不成熟之外，也理解DS專家看待資料集值得分析的考量角度。

(3) 會議討論都記載於google doc。雖然讓繁忙的組員感到煩，但卻也是累積、連結討論成果最有效的方式。從2023/04/21寄信組隊、4/27第一次討論開

(1) 範例：**專案甘苦談與專案引用的套件與外部資料**，階層式綱要如下：

2. 引用套件、外部資料

套件

#packages
#1. rpart / rpart.plot：decision tress
#2. corrplot：correlation among cols
#3. performanceEstimation：smote to tackle unbalanced dataset
#4. dplyr：tackele dataframe
#5. ROCR：roc curve and aus
#6. caret::varImp ：the important order of cols for target y
#7. car::vif ：check collinearity in cols

```
# --------------installed packages-----------------------
# install.packages('rpart.plot')
# install.packages('ggplot2')
```

外部資料：

kaggle dataset

(example)Hux.DA5030.Project | Kaggle
(example)STAT 451 Project | Kaggle
(example_explaindata)Heart Disease Scoring : Who
(example_explaindata)Heart Disease Prediction | Ka

smote

(performanceEstimation)imbalanced data - package

VIF

(glm_Variable Importance_VIF)How to Perform Logis
Statology
(multicollinearity)How to Fix in R: there are aliased c

AUC

專案艱辛之處

昇豐:從「思想的巨人，行動的侏儒」中脫離，進行**行動力**變革。讀過不少ML、DS、統計的書，但工作既不是此領域、大學的統計也沒學好，因此這段期末報告的準備過程真的是**練習對抗的過程**;所幸考試後的課程會提點分析工具選用的訣竅外，老師也願意給予意見。組員也願意與信任地按照我仿照課堂教科書訂定的報告綱要準備。

書瑋:第一次接觸資料分析，也是第一次接觸R，都是透過上課及作業學到的東西，加上不斷地Google及詢問ChatGPT，慢慢拼湊出最後的結果。

元亨:(程式的互相合作)我們總共採用了3種(決策樹、隨機森林、Xgboost)模型來分析資料，實際寫了5份建模程式及1份視覺化程式，在這些Code中，每個人的Code pattern, 前處理, 套件選用等都不盡相同，讓我們在溝通、整合程式時費了一番功夫。

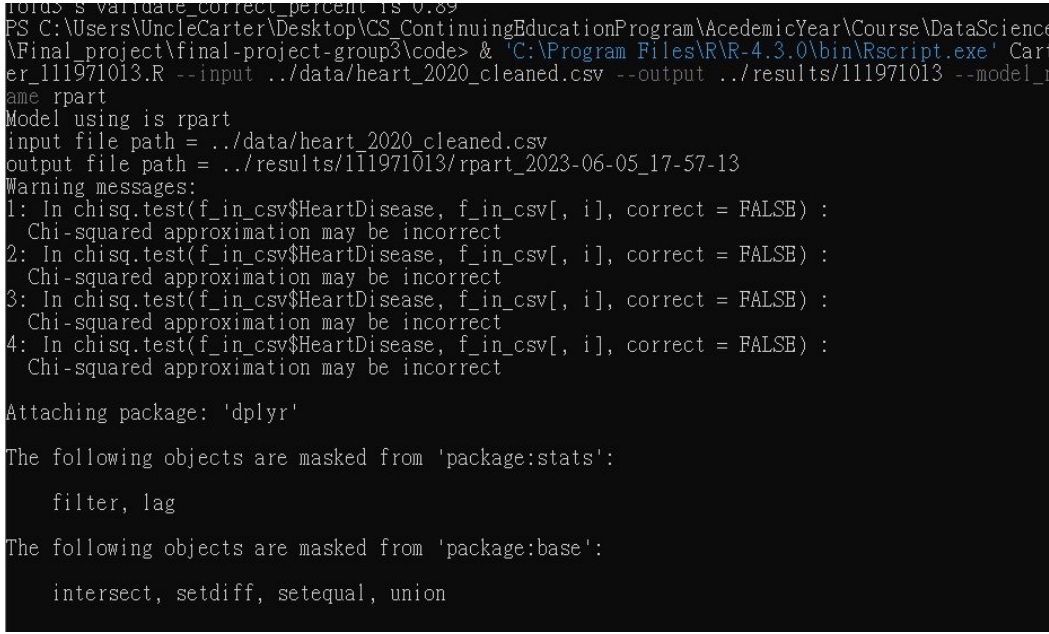
正晏:(對於不熟悉領域的探索)平時在於工作中並未使用到與Data Science相關的技能，本次專案算是一個新領域的探索，在專案建立的過程中，除了需要時不時複習老師上課的內容之外，還需要不斷地去學習，在資料集確定後就遇到了第一個難關，在不同的模型上，會針對其所需要的輸入去調整Attribute，在建模的過程中，發現若不了解模型的原理，在參數的值上也難以去調整。

宗佑:(資料集不平衡)本主題資料集相當不平均，患者與非患者為1比13，基本上使用null model全部猜測患者無病，準確性即高達九成多。然而這樣的猜測沒有太大意義，因此我們針對不平衡的資料瀏覽許多資料，最後透過smote，降低樣本比例至1比4，提高sensitivity。

弘軒：同學間的默契在磨合後增長，討論凝聚眾人智慧，分派任務考驗彼此信賴;專案合作的經驗啟發團隊合作，善用工具實現共同目標，迎接各種挑戰;至此深刻體會到老師設計本門課程的價值與其重要性。

(1) 範例: **DEMO**, 階層式綱要如下:

1. git clone : https://….
2. 執行的指令、terminal示意
   - 2.1. cd to repo folder
   - 2.2. Rscript.exe …
   - 2.3. 執行Rscript的示意畫面
3. what’s output, how to interprete.



B組

弘軒的部分

正晏的部分

結束

放在附件