資料探勘期末報告

應用機器學習預測生存率：以心臟衰竭患者為例

碩管一甲：周彥廷，學號：M09218001 ，授課老師：劉弘一 老師

目錄

第一章：研究目的與動機

健康議題廣為大眾所關注，108 年台灣國人非事故十大死因依序為(1)惡性腫瘤(癌症)(2)心臟疾病(3)肺炎(4)腦血管疾病(5)糖尿病(6)事故傷害(7)慢性下呼吸道疾病(8)高血壓性疾病(9)腎炎腎病症候群及腎病變(10)慢性肝病及肝硬化，排名順位與 107 年相同[1]。

癌症雖然長年位居十大死因之首，但衛福部在 2019 年的調查顯示，心臟疾病為台灣第 2 大死因，當年度有近 2 萬人死於心臟疾病並顯示心臟衰竭 5 年死亡率高於很多癌症，台灣約有 36 萬名心臟衰竭病患，然而患者卻鮮少知道自身患有心臟衰竭疾病[2]。

本研究利用 UCI 機器學習庫中的心力衰竭臨床記錄數據集（https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records），嘗試透過一系列資料探勘分析與建立機器學習演算法，探究心臟衰竭患者與死亡之間關係，並由資料視覺化手段呈現分析之結果。

第二章：資料探勘

資料來源 UCI Machine learning

```
data<-read.csv("https://archive.ics.uci.edu/ml/machine-learning-databases/00519/heart_failure_clinical_records_dataset.csv"",header=TRUE,sep = ",",na.strings=" ")
```

2.1 基本資料探勘

當讀取資料後，使用下列函數查看資料維度、欄位資料型態與統計值，可以了解資料集為 299×13，並掌握有些欄位為類別變數，另為數值變數。

```
#資料屬性查詢函數
dimnames(data1)     # 查看資料維度名稱 顯示 1 至 299
length(data1)       # 查看資料長度 顯示 13
dim(data1)          # 查看資料維度 299 13
class(data1)        # 查看資料類型 "tbl_df"     "tbl"       "data.frame"
table(data1$sex)    # 以 sex 為例 顯示 0_105 筆，1_194 筆
head(data1)         #查看前六筆資料
str(data1)          #查看資料結構
summary(data1)      #查看資料統計值(最大值、最小值、平均值....等統計值)
```

使用 names()函數可以查看資料欄位名稱，顯示有 13 欄位如下：

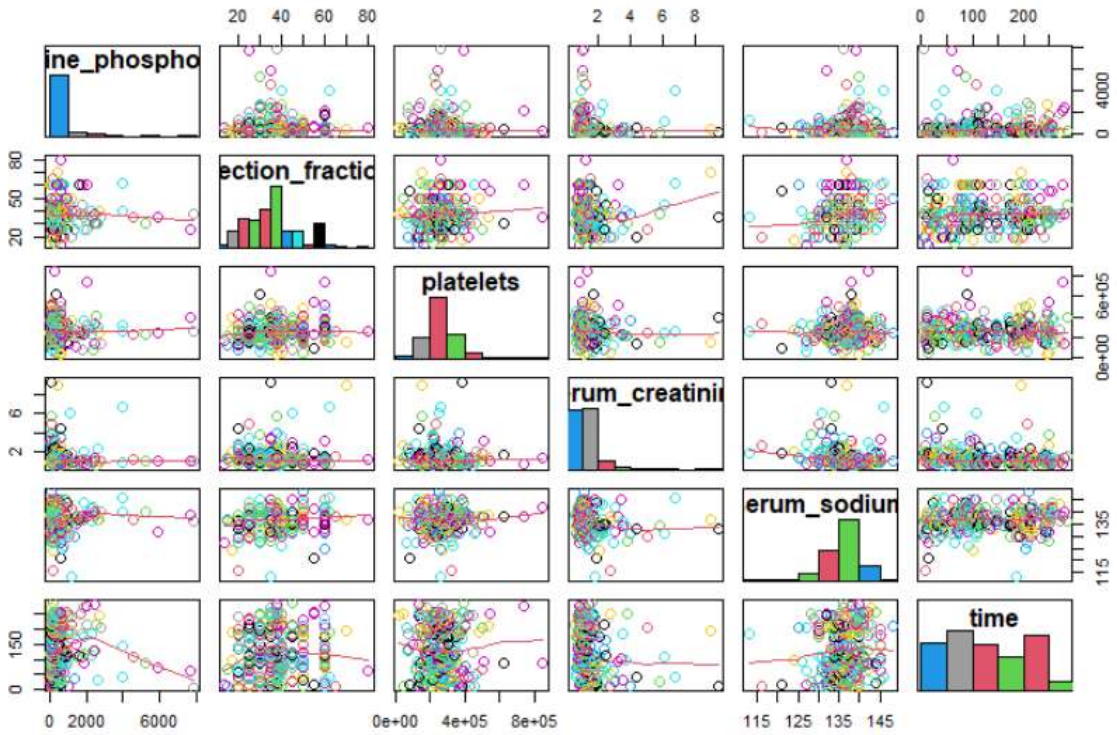
```
names(data1)   #顯示所有欄位名稱

##  [1] "age"                  "anaemia"
##  [3] "creatinine_phosphokinase" "diabetes"
##  [5] "ejection_fraction"        "high_blood_pressure"
##  [7] "platelets"                "serum_creatinine"
##  [9] "serum_sodium"             "sex"
## [11] "smoking"                  "time"
## [13] "DEATH_EVENT"
```

```
#單變數探索
dim(data1)      #資料維度
attributes(data1) #資料屬性
tail(data1)     #最後六筆資料
```

2.2.1 資料可視化

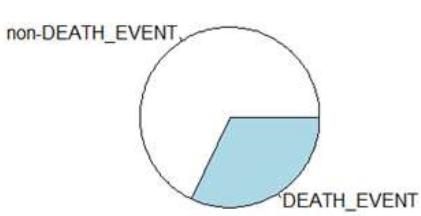
對資料進行可視化可了解數據分布狀態、是否有離群值與評估資料是否可用，透過以下使用 hist()函數可以繪製直方圖，在此將其他特徵與 age 進行散佈圖矩陣，可快速了解資料分佈狀況。

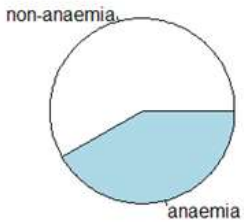
```
panel.hist = function( x, ...) {
  usr = par("usr"); on.exit(par(usr))
  par(usr = c(usr[1:2], 0, 1.5) )
  h = hist(x, plot = FALSE)
  breaks = h$breaks; nB = length(breaks)
  y = h$counts; y = y/max(y)
  rect(breaks[ -nB], 0, breaks[ -1], y, ...)
}
pairs( newdata[ c(3,5,7,8,9,12)], col = as.integer(newdata[ , 1]) + 1,
       panel = panel.smooth,
       diag.panel = panel.hist ,
       cex.labels = 1.5,
       cex = 1.5, font.labels = 2)
```



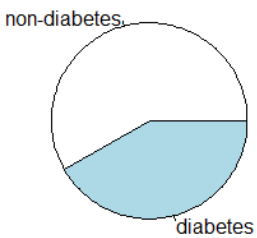
使用 pie 函數繪製圓餅圖可以了解類別資料的比例(以死亡數與貧血)。

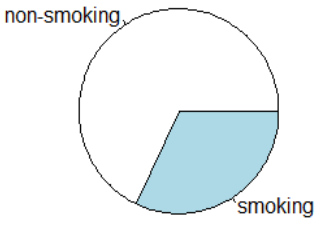
```
table(data1$DEATH_EVENT)
##
##   0   1
## 203  96
lbls <- c(non-DEATH_EVENT", "DEATH_EVENT")
pie(table(data1$DEATH_EVENT),labels = lbls)

table(data1$anaemia)
##
##   0   1
## 170 129
lbls <- c("non-anaemia", "anaemia")
pie(table(data1$diabetes),labels = lbls)
```






```
table(data1$diabetes)
lbls1 <- c("non-diabetes", "diabetes")
pie(table(data1$diabetes),labels = lbls1)

table(data1$smoking)
lbls2 <- c("non-smoking", "smoking")
pie(table(data1$smoking),labels = lbls2)
```



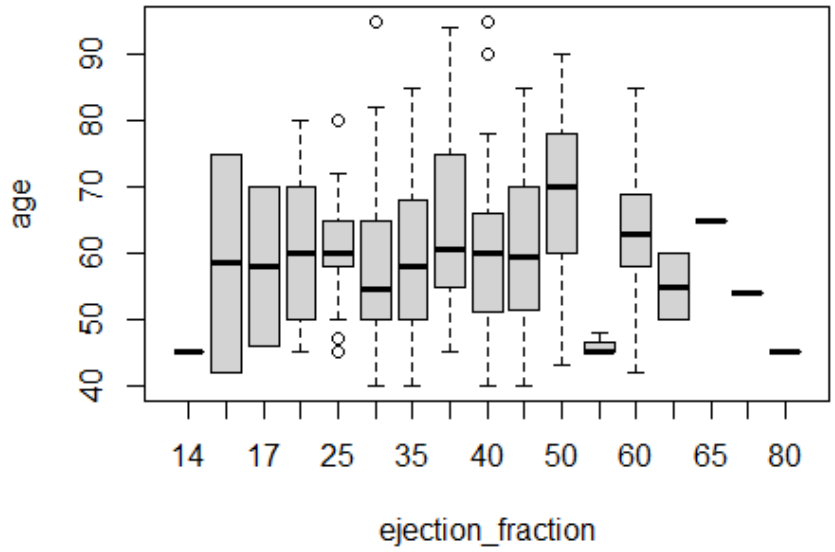


此處開始探討各欄位間關係，以本研究為例 DEATH_EVENT 是要預測的欄位，因此使用 cov 和 cor 兩個函數，令其他 12 欄位與 DEATH_EVENT 欄位求出共變異數(Covariance)和相關係數(Correlation Coefficient)。
共變異數可以了解各特徵欄位與 DEATH_EVENT 相互影響程度，相關係數可以觀察到 time(-0.526)和 erum_creatinine(0.294)呈現最大負相關與最大正相關。

```
#多變數探索
#共變異數
cov(data1, data1$DEATH_EVENT)

##                             [,1]
## age                    1.411454e+00
## anaemia                1.537564e-02
## creatinine_phosphokinase  2.846447e+01
## diabetes               -4.489237e-04
## ejection_fraction      -1.486667e+00
## high_blood_pressure     1.774371e-02
## platelets              -2.247619e+03
## serum_creatinine        1.423744e-01
## serum_sodium           -4.028192e-01
## sex                    -9.651860e-04
## smoking                -2.760881e-03
## time                   -1.912766e+01
## DEATH_EVENT             2.187156e-01

#相關係數
cor(data1, data1$DEATH_EVENT)

##                             [,1]
## age                    0.253728543
## anaemia                0.066270098
## creatinine_phosphokinase  0.062728160
## diabetes               -0.001942883
## ejection_fraction      -0.268603312
## high_blood_pressure     0.079351058
## platelets              -0.049138868
## serum_creatinine        0.294277561
## serum_sodium           -0.195203596
## sex                    -0.004316376
## smoking                -0.012623153
## time                   -0.526963779
## DEATH_EVENT             1.000000000
```

2.2.2 資料可視化(多變數)

age(正相關係數第二高)與射血分數(ejection_fraction)繪製盒鬚圖顯示兩者數據分佈狀態。




```
boxplot(age~ ejection_fraction, data=data1)
```

age 與 serum_creatinine 繪製散佈圖顯示兩者數據分佈狀態。

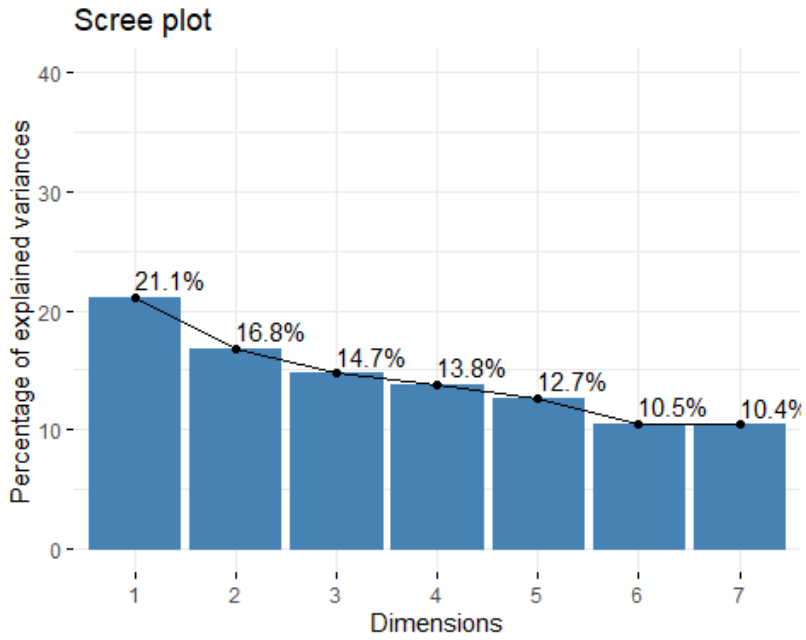
2.3 主成分分析(Principal Component Analysis, PCA)

簡介：PCA 是由 1901 年由統計學家卡爾·皮爾森所提出，是一非監督式學習與透過降維(Dimension reduction)內特徵擷取(Feature extraction)的一種方法，目的是將資料的整體維度減少，例如：(將重要變數適當的給予較大的權重，不重要的變數給予較小的權重)，使整體效能不會差異太大甚至提升效能[3,4]。

本案例經過 PCA 結果顯示第一主成分為 ejection_fraction，第二主成分為 age，第三主成分為 serum_sodium

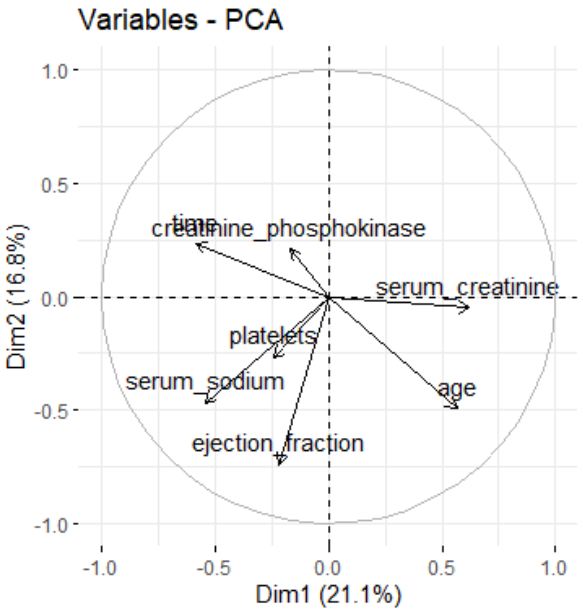
將資料集中數值型資料取出，並使用 factoextra 套件值型 PCA 分析

```
#PCA 主成分分析
library(factoextra)

pca_data <- subset(data1, select = c(-2,-4,-6,-10,-11,-13))
pca_w <- prcomp(x=pca_data, center=TRUE, scale=TRUE)
eig.val <- get_eigenvalue(pca_w)
fviz_eig(pca_w, addlabels = TRUE, ylim = c(0, 40))  #Scree plot
```

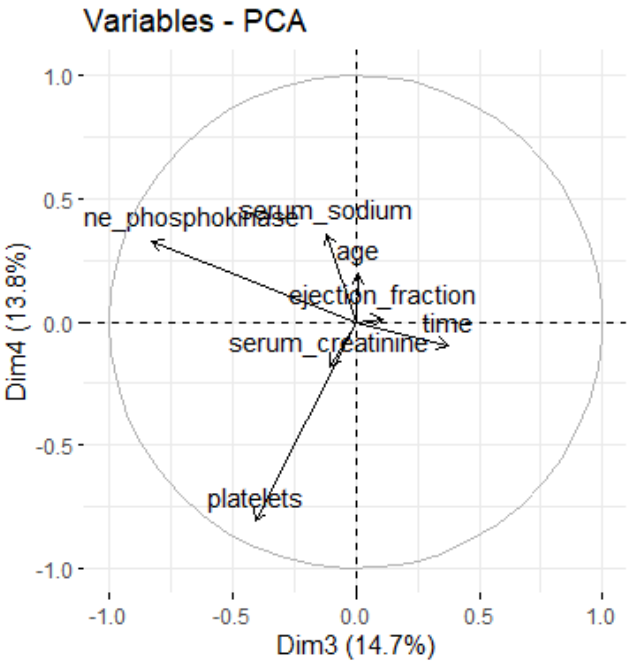



```
var$coord[,c(1:4)] #變量和主成分（PC）之間的相關性
```

```
##                          Dim.1       Dim.2       Dim.3       Dim.4
## age                  0.5646122 -0.49022298  0.007913901  0.19471817
## creatinine_phosphokinase -0.1675268  0.21022842 -0.826979846  0.32871011
## ejection_fraction       -0.2172326 -0.73889076  0.108274741  0.01277376
## platelets               -0.2419625 -0.26757736 -0.409218900 -0.80697109
## serum_creatinine         0.6214609 -0.04954616 -0.103159985 -0.17916082
## serum_sodium            -0.5432998 -0.46592218 -0.119702387  0.35643083
## time                    -0.5836063  0.23233891  0.375987635 -0.09875816
```

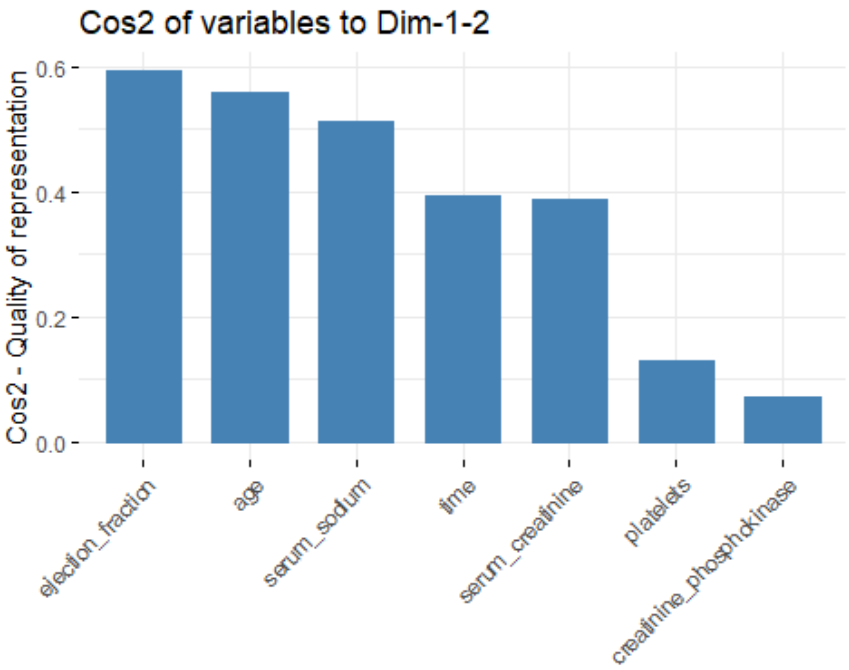



```
fviz_pca_var(pca_w, col.var="black")  #二維主成分圖( Dim1 和 Dim2)
```

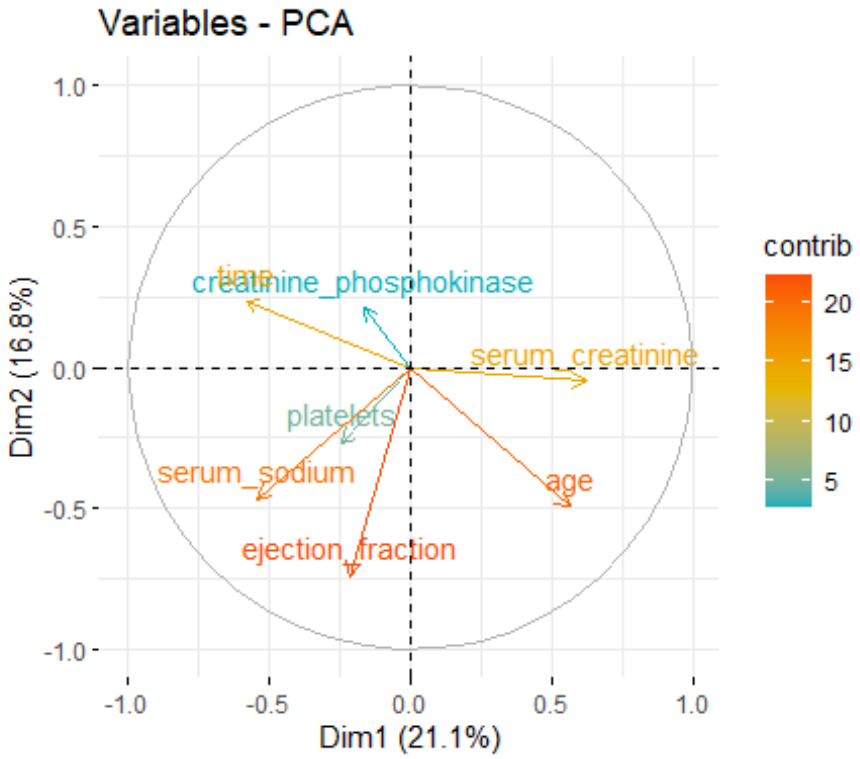



```
fviz_pca_var(pca_w, col.var="black", axes=c(3, 4)) #(Dim3 和 Dim4)
```

```
fviz_cos2(pca_w, choice = "var", axes = 1:2)
```




```
fviz_pca_var(pca_w, col.var = "contrib",
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07") )
```



2.4 關聯分析(Association Rules)

簡介：最早 Agrawal et al.(1993)提出，目的從蒐集的龐大交易資料中，發掘隱含於商品間的關聯性，以瞭解消費者購買行為與產品銷售關係[5,6]。

- 支持度(support)：衡量關聯規則的顯著性
- 信賴度(confidence)：衡量關聯規則的正確性
- 增益(lift)：衡量關聯規則的資訊價值

```
#資料整理
library(dplyr)
require(arules)
data2 <- dplyr::mutate(data1,
                       SEX = ifelse(sex==0,"WOMAN","MEN"),
                       HIGH_BLOOD = ifelse(high_blood_pressure==0,"NO","YES"),
                       DIABETES = ifelse(diabetes==0,"NO","YES"),
                       ANAEMIA = ifelse(anaemia==0,"NO","YES"),
                       SMOKE = ifelse(smoking==0,"NO","YES"),
                       DEATH=ifelse(DEATH_EVENT==0,"No","Yes"))
data3 <- subset(data2, select = c(-1:-13))
require(arules)

#建立"是否存活"的規則 & 調整衡量指標之篩選門檻
rule3 <- apriori(data3, parameter=list(minlen=3, conf=0.7),
                 appearance = list(default="lhs", rhs=c("DEATH=No", "DEATH=Yes")
                 ))
```

```r
#使用 inspect 函數查看 rule3

inspect(rule3)

summary(rule3)

## set of 35 rules
##
## rule length distribution (lhs + rhs):sizes
##  3  4  5
## 13 18  4
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   3.000   4.000   3.743   4.000   5.000
##
## summary of quality measures:
##     support         confidence        coverage          lift
##  Min.   :0.1037   Min.   :0.7037   Min.   :0.1338   Min.   :1.036
##  1st Qu.:0.1237   1st Qu.:0.7165   1st Qu.:0.1622   1st Qu.:1.055
##  Median :0.1472   Median :0.7333   Median :0.2040   Median :1.080
##  Mean   :0.1556   Mean   :0.7388   Mean   :0.2120   Mean   :1.088
##  3rd Qu.:0.1672   3rd Qu.:0.7592   3rd Qu.:0.2324   3rd Qu.:1.118
##  Max.   :0.2776   Max.   :0.8085   Max.   :0.3779   Max.   :1.191
##      count
##  Min.   :31.00
##  1st Qu.:37.00
##  Median :44.00
##  Mean   :46.51
##  3rd Qu.:50.00
##  Max.   :83.00
##
## mining info:
##   data ntransactions support confidence
##  data3          299     0.1        0.7

sort.rule <- sort(rule3, by="support")  #根據 support 大小排序 rules
sort.rule
## set of 35 rules
inspect(sort.rule)
subset.matrix <- as.matrix(is.subset(x=sort.rule, y=sort.rule)) #比對某項規則是否為其他規則的子集(subset)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA  #下三角元素改為 NA（便於辨識）
redundant <- colSums(subset.matrix, na.rm=T) >= 1 #辨識多餘的 column (若有一個以上的 TRUE)
sort.rule <- sort.rule[!redundant] #移除多餘的 column
inspect(sort.rule)  #顯示關聯規則指標
##      lhs                          rhs        support   confidence coverage
## [1]  {HIGH_BLOOD=NO,ANAEMIA=NO}   => {DEATH=No} 0.2775920 0.7345133  0.3779264
## [2]  {ANAEMIA=NO,SMOKE=NO}        => {DEATH=No} 0.2541806 0.7037037  0.361204
```

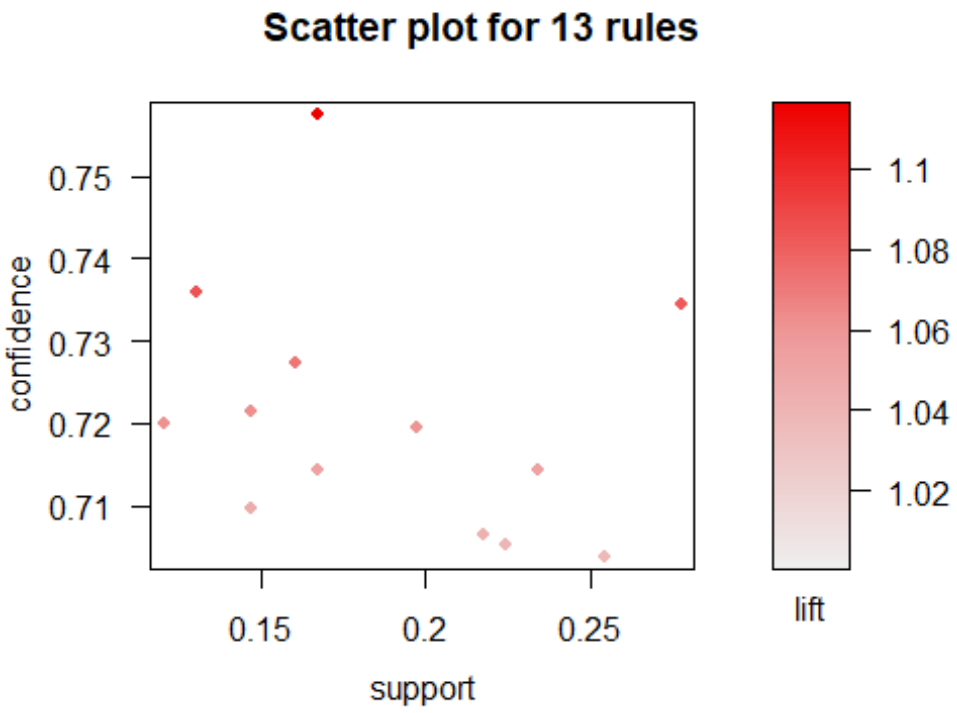
```
## [3]  {DIABETES=NO,ANAEMIA=NO}     => {DEATH=No} 0.2341137 0.7142857  0.32775
92
## [4]  {DIABETES=YES,SMOKE=NO}      => {DEATH=No} 0.2240803 0.7052632  0.317725
8
## [5]  {SEX=MEN,SMOKE=YES}          => {DEATH=No} 0.2173913 0.7065217  0.3076923
## [6]  {HIGH_BLOOD=NO,DIABETES=YES} => {DEATH=No} 0.1973244 0.7195122  0.27
42475
## [7]  {HIGH_BLOOD=NO,SMOKE=YES}    => {DEATH=No} 0.1672241 0.7575758  0.220
7358
## [8]  {SEX=MEN,DIABETES=YES}       => {DEATH=No} 0.1672241 0.7142857  0.2341137
## [9]  {DIABETES=NO,SMOKE=YES}      => {DEATH=No} 0.1605351 0.7272727  0.220735
8
## [10] {ANAEMIA=NO,SMOKE=YES}       => {DEATH=No} 0.1471572 0.7096774  0.20735
79
## [11] {SEX=WOMAN,HIGH_BLOOD=NO}    => {DEATH=No} 0.1471572 0.7213115  0.20
40134
## [12] {SEX=WOMAN,ANAEMIA=NO}       => {DEATH=No} 0.1304348 0.7358491  0.1772
575
## [13] {SEX=WOMAN,DIABETES=NO}      => {DEATH=No} 0.1204013 0.7200000  0.1672
241
##      lift     count
## [1]  1.081869 83
## [2]  1.036490 76
## [3]  1.052076 70
## [4]  1.038787 67
## [5]  1.040640 65
## [6]  1.059774 59
## [7]  1.115838 50
## [8]  1.052076 50
## [9]  1.071205 48
## [10] 1.045288 44
## [11] 1.062424 44
## [12] 1.083837 39
## [13] 1.060493 36
```

```
require(arulesViz) #關聯規則的視覺化套件
```

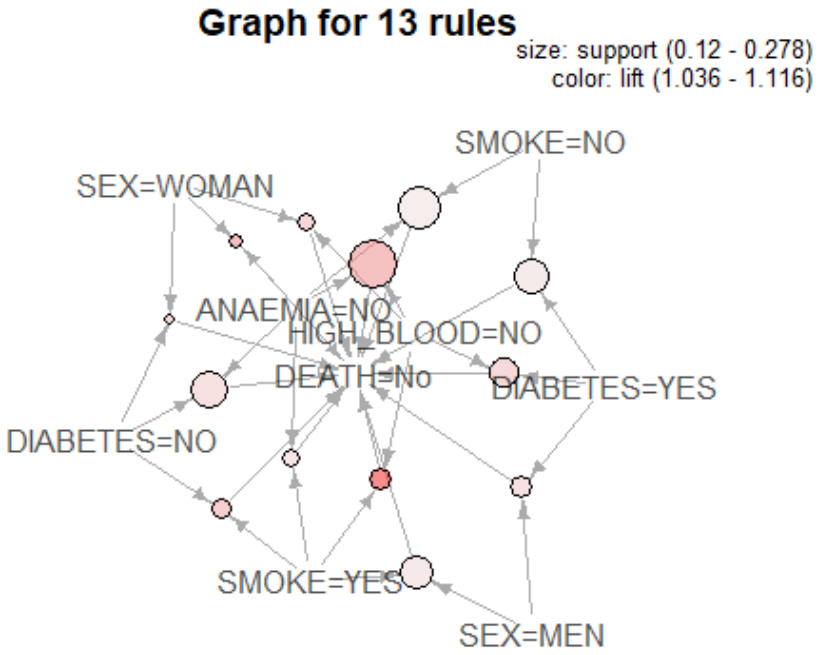
```
## Loading required package: arulesViz
```

```
## Warning: package 'arulesViz' was built under R version 4.0.4
```

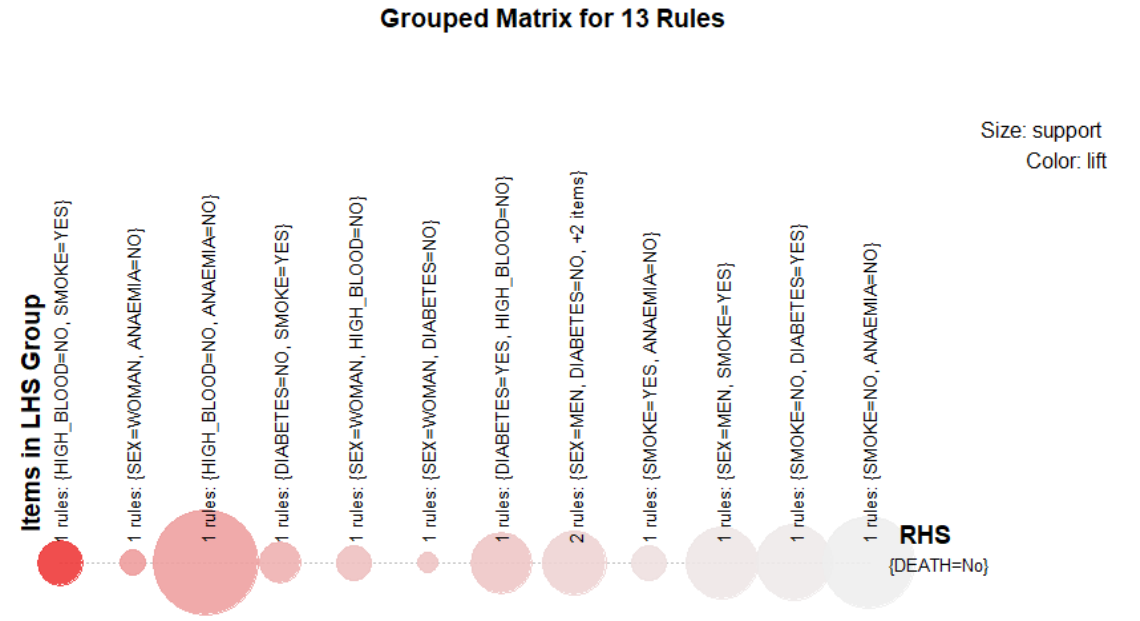
```
plot(sort.rule)
```




```
plot(sort.rule, method="graph")
```



```
plot(sort.rule, method="grouped")
```



結語：透過視覺化可快速掌握在特定條件下計算關聯分析之結果，並按照 lift 與 count 將結果由高至低將關聯規則列出(此處條件 conf = 0.7 並依據 support 排序) 。

第三章：研究方法

3.1 演算法選擇

本次資料集問題概念是二元分類問題，演算法選擇(包含課堂所學)，使用 spot checking，透過快速建立模型與比較表現效能，挑選出適合訓練模型。

3.1.1 決策樹(Decision Tree)

簡介：決策樹根據分枝準則進行特徵劃分探索並推導規則進行預測[7]。

常見的分枝準則：1.資訊增益 (information gain)，2.Gini 係數 (Gini index)

3.1.2 隨機森林(Random Forest)

簡介：為集成 (Ensemble)方式之一，方式 是將多個模型的結果組合在一起，透過投票或是加權的方式得到最終結果，隨機森林的每一棵樹在生成過程中，都是隨機使用一部份的訓練資料與特徵代表每棵樹都是用隨機的資料訓練而成的 [8]。

3.1.3 貝氏分類(Bayesian Classifier)

簡介：藉由資料中分析屬性與反應變數之間的機率模型，根據貝氏定理(Bayes' theorem)來更新資訊以推理判斷樣本資料歸屬的類別，作為分類和推論的依據[9]。

3.1.4 羅吉斯迴歸(Logistic Regression)

簡介：羅吉斯迴歸為概率型非線性迴歸模型，是研究二分類觀察結果與一些影響因素之間關係的一種多變量分析方法。通常的問題是，研究某些因素條件下某個結果是否發生[10]。

3.1.5 正則化迴歸(Regularized Regression)

簡介：正則化(Regularizer)允許給予各層參數在最佳化的過程中給予懲罰項，解決模型過度配適(overfitting)造成預測時失真情況[11]。

3.1.6 支持向量機(Support Vector Machine, SVM)

簡介：SVM 是一種監督式學習方法，主要可用於分類（classification）或迴歸（regression）類型的問題，SVM 的學習目的在於找到具有最大邊緣的超平面（maximum marginal hyperplane）以達到有效分類[12]。

```
#對 DEATH 欄位進行處理

Tdata<- dplyr::mutate(data1,DEATH=ifelse(data1$DEATH_EVENT==0,"No","Yes"))

newdata <- subset(Tdata, select = -13)

library (caret)

#進行 spot checking

#針對下列 6 種演算法進行初步比較，並 set up 10-fold cross validation

# set up 10-fold cross validation procedure

set.seed(7)

control <- trainControl(method="cv", number=10)

# 1.train Classification and Regression Trees

fit.rpart <- train(DEATH~., data=newdata, method="rpart", metric="Accuracy",
preProc=c("center", "scale"),trControl=control)

# 2.train Random Forest

fit.rf <- train(DEATH~., data=newdata, method="rf", metric="Accuracy", preProc=c("center",
"scale"),trControl=control)

# 3.train Naive Bayes

fit.nb <- train(DEATH~., data=newdata, method="nb",
metric="Accuracy",preProc=c("center", "scale"), trControl=control)

# 4.train Logistic Regression

fit.glm <- train(DEATH~., data=newdata, method="glm", metric="Accuracy",
preProc=c("center", "scale"), trControl=control)
```

```
# 5.train Regularized Regression
fit.glmnet <- train(DEATH~., data=newdata,  method="glmnet", metric="Accuracy",
preProc=c("center", "scale"), trControl=control)
# 6.train Support Vector Machine
fit.svmRadial <- train(DEATH~., data=newdata, method="svmRadial",
metric="Accuracy", preProc=c("center", "scale"), trControl=control)
# collect resamples using resamples
results <- resamples(list(CART=fit.rpart, RF=fit.rf, NB=fit.nb, LR=fit.glm,
RR=fit.glmnet, SVM=fit.svmRadial))
summarize differences between modes
summary(results)
```

```
Call:
summary.resamples(object = results)

Models: CART, RF, NB, LR, RR, SVM
Number of resamples: 10

Accuracy
          Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
CART 0.7586207 0.7948276 0.8032258 0.8191509 0.8583333 0.9000000    0
RF   0.7586207 0.7948276 0.8500000 0.8389359 0.8698925 0.9333333    0
NB   0.7241379 0.7666667 0.8032258 0.7958176 0.8318966 0.8666667    0
LR   0.6896552 0.8000000 0.8166667 0.8285910 0.8927419 0.9333333    0
RR   0.7586207 0.8000000 0.8166667 0.8321617 0.8666667 0.9032258    0
SVM  0.7000000 0.7586207 0.8166667 0.8021542 0.8666667 0.8709677    0
```

結語：比較表中顯示 **rf** 演算法有最高正確率(Accuracy)，因此後續模型優化選擇優先使用 rf。

3.2 建模與訓練

```
# 利用 createDataPartition()函數將資料進行切分比例(訓練資料 0.7/測試資料 0.3)
set.seed(7)
trainIndex <- createDataPartition(newdata$DEATH, p = 0.3, list = FALSE )
Train <- newdata[ trainIndex,] #0.7_訓練資料
Test <- newdata[-trainIndex,] #0.3_測試資料
```

```
#資料前處理同樣使用'center', 'scale'
preProValues <- preProcess(Train, method = c('center', 'scale'))
preProValues
traindata <- predict(preProValues, Train) #訓練資料
testdata <- predict(preProValues, Test)  #測試資料

#建立隨機森林 rf 模型與檢視效能
#使用交叉驗證法(k-folds cross-validation)
control <- trainControl(method='cv', number=10)
set.seed(7)
rf.model <- caret::train(DEATH~., data=traindata, method='rf',
                                                  metric='Accuracy', trControl=control)
pred_rf<-predict(rf.model, newdata=testdata)
#建立矩陣並顯示模型效能指標
confusionMatrix_rf<-confusionMatrix(pred_rf,
                                                    factor(testdata$DEATH),
                                                dnn = c('Prediction', 'Reference'),
                                                      mode = 'everything')
```

CV = 10

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  126  25
       Yes  16  42

               Accuracy : 0.8038
                 95% CI : (0.7434, 0.8554)
    No Information Rate : 0.6794
    P-Value [Acc > NIR] : 4.16e-05

                  Kappa : 0.5331

 Mcnemar's Test P-Value : 0.2115

            Sensitivity : 0.8873
            Specificity : 0.6269
         Pos Pred Value : 0.8344
         Neg Pred Value : 0.7241
              Precision : 0.8344
                 Recall : 0.8873
                     F1 : 0.8601
             Prevalence : 0.6794
         Detection Rate : 0.6029
   Detection Prevalence : 0.7225
      Balanced Accuracy : 0.7571

       'Positive' Class : No
```

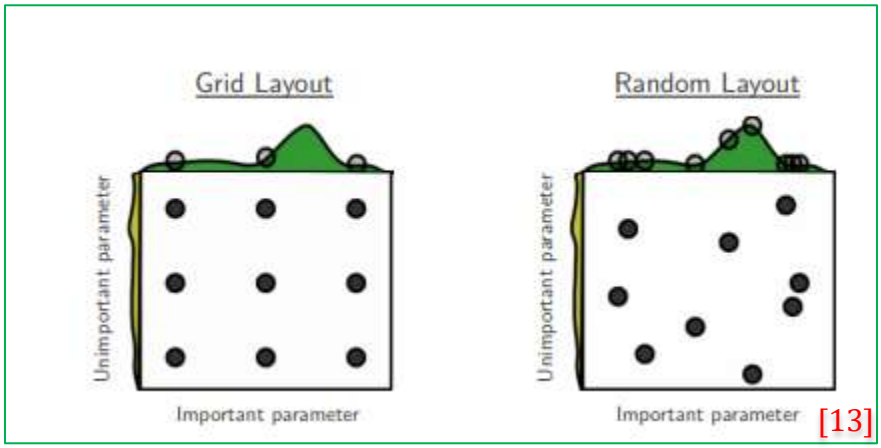
CV = 5

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  118  18
       Yes  24  49

               Accuracy : 0.799
                 95% CI : (0.7382, 0.8512)
    No Information Rate : 0.6794
    P-Value [Acc > NIR] : 8.089e-05

                  Kappa : 0.5493

 Mcnemar's Test P-Value : 0.4404

            Sensitivity : 0.8310
            Specificity : 0.7313
         Pos Pred Value : 0.8676
         Neg Pred Value : 0.6712
              Precision : 0.8676
                 Recall : 0.8310
                     F1 : 0.8489
             Prevalence : 0.6794
         Detection Rate : 0.5646
   Detection Prevalence : 0.6507
      Balanced Accuracy : 0.7812

       'Positive' Class : No
```

結語：發現在此案例中 CV 設定次數為 10 有較好正確率 0.8038 與 F1 值 0.8601，因此後續 cv 值使用 10。

3.3：調整超參數(Hyper-parameter Tuning)

超參數調整策略示意圖如下[13]：



[13]

超參數調整策略：

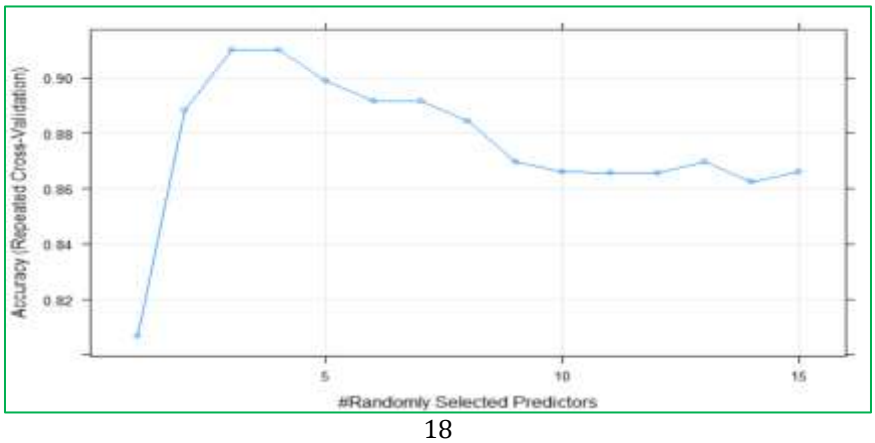
1.網格搜索調參(TuneGrid)：給定參數特定範圍並窮舉所有交叉驗證試驗可能之超參數值組合，進而求出最佳解。

2.隨機搜索調參(TuneLength)：與網格調參不同，針對每一超參數隨機值特定數量的隨機組合進行搜尋，適合超參數的搜索空間很大時使用。

評估電腦計算運行時間與資料集大小，優化策略採用 TuneGrid 方式進行調參。

TuneGrid

```
library(randomForest)
library(mlbench)
library(e1071)
control <- trainControl(method='repeatedcv', repeats = 3, number=10, search = "grid")
glmnet_grid <- expand.grid(alpha = 0:1, lambda = seq(0.0001, 1, length = 100))
tunegrid <- expand.grid(.mtry = (1:15))

rf_tunegrid <- train(DEATH~.,  data= traindata,  method='rf',  metric='Accuracy',
                                                          tuneGrid=tunegrid,
                                                            trControl=control)
plot(rf_tunegrid)
```

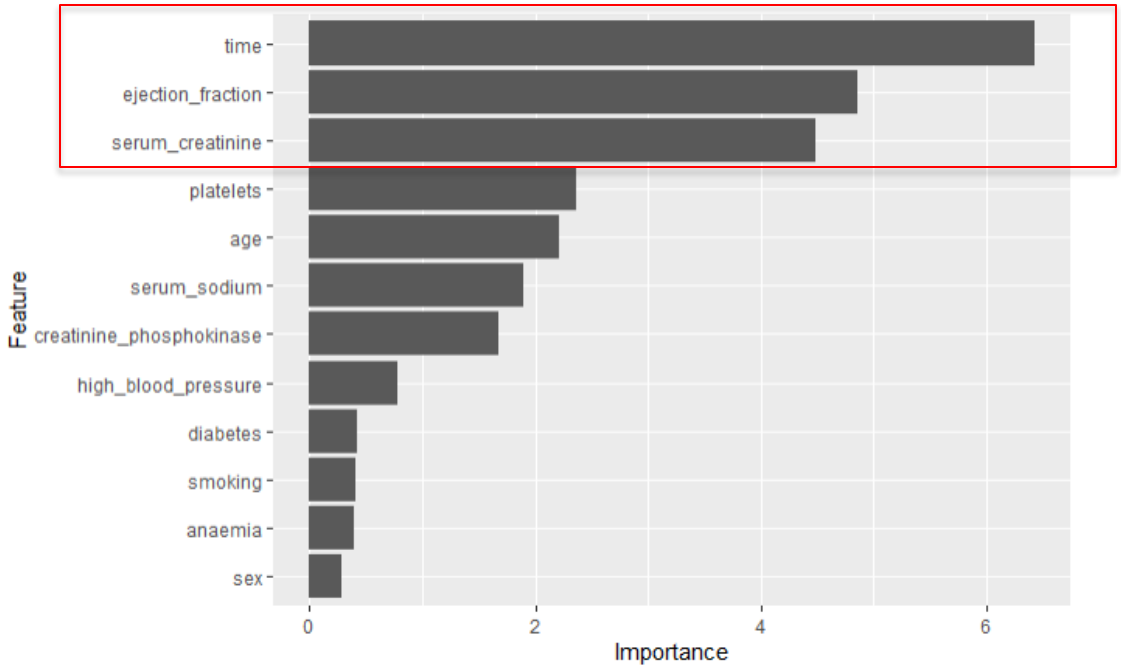


```
control <- trainControl(method='repeatedcv',  number=10, repeats=3,  search='grid')

#create tunegrid
modellist <- list()
tunegrid <- expand.grid(.mtry = 4) #mrty 選擇 4

#train with different ntree parameters
for (ntree in c(1000,1500,2000,2500,3000)){
  set.seed(7)
  rf_tunefinal <- train(DEATH~., data = traindata, method = 'rf', metric = 'Accuracy',
                                                                  tuneGrid = tunegrid,
                                                                   trControl = control,
                                                                         ntree = ntree)
  key <- toString(ntree)
  modellist[[key]] <- rf_tunefinal
}
rf_tunefinal
results <- resamples(modellist)
summary(results)
pred_tunefinal<-predict(rf_tunefinal,newdata=testdata)

confusionMatrix_rf_tunefinal <- confusionMatrix(pred_tunefinal,
                                factor(testdata$DEATH),  dnn = c('Prediction', 'Reference'),
                                                                      mode = 'everything')
confusionMatrix_rf_tunefinal
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  123  17
       Yes  19  50

               Accuracy : 0.8278
                 95% CI : (0.7696, 0.8763)
    No Information Rate : 0.6794
    P-Value [Acc > NIR] : 9.575e-07

                  Kappa : 0.6077

 Mcnemar's Test P-Value : 0.8676

            Sensitivity : 0.8662
            Specificity : 0.7463
         Pos Pred Value : 0.8786
         Neg Pred Value : 0.7246
              Precision : 0.8786
                 Recall : 0.8662
                     F1 : 0.8723
             Prevalence : 0.6794
         Detection Rate : 0.5885
   Detection Prevalence : 0.6699
      Balanced Accuracy : 0.8062

       'Positive' Class : No
```

4.2 結語：網格調參正確率為 0.8278 與未調參時 0.8038 相比提升 0.024。

```
#重要變數列表
importance <- varImp(fit.rf, scale = FALSE)
ggplot(importance)
```



重要變數列表顯示訓練模型與死亡相關前三項特徵為 1.追蹤天數(time) 2.射血分數(ejection_fraction) 3. 血清肌酐(serum creatinine)。

第五章：結論

本研究按照課程所學依序從資料基本處理(前處理)→可視化(了解數據分佈)→PCA 分析→關聯分析至建模訓練，接續嘗試使用 TuneGrid 改善隨機森林模型效能，最後得出正確率為 0.8278。

整理實驗結果如下：

1.cv 的 number 設定 5 和 10 會相差 0.0048。

2.本實驗使用 TuneGrid 方式進行超參數調整提升正確率 0.024，其他重要效能指標整理如下表所示：

Model	Accuracy	Precision	Recall	F1
rf	0.8038	0.8344	0.8873	0.8601
rf_TuneGrid	0.8278	0.8786	0.8662	0.8723

3.重要變數列表與 Ahmad et al.(2017)皆顯示射血分數(ejection_fraction)與血清肌酐(serum_creatinine)為促成疾病的關鍵因素，會使心力衰竭患者死亡的風險增加，另吸煙習慣、罹患糖尿病和患者性別此三項特徵與文獻所述低相關結果一致 [14]。

4.透過機器學習方式建立一具有 82.78%正確率以射血分數(ejection_fraction)與血清肌酐(serum_creatinine)作為主要判斷心臟衰竭患者是否會死亡演算法。

5.PCA 分析結果由第 8 頁**圖 Cos of variables to Din-1-2** 與重要變數列表皆顯示射血分數與血清肌肝為重要特徵。
6.關聯分析由第 13 頁**圖 Grouped Matrix for 13 Rules** 中同時考量 lift 最高與 supprt 最大規則顯示存活病患沒有罹患高血壓與貧血，此關聯規則與重要變數列表預測死亡低相關特徵互相印證。

參考文獻

[1]衛生福利部 https://www.mohw.gov.tw/cp-16-54482-1.html

[2]作者/邱淑宜/康健編輯部,2020/10/07
https://www.commonhealth.com.tw/article/82909

[3] Ch.5 主成分分析(Principal Component Analysis, PCA)課堂投影片

[4]主成分分析的原理 https://web.ntpu.edu.tw/~ccw/statmath/M_pca.pdf

[5] Ch.6 關聯規則(Association Rules)課堂投影片

[6] https://sites.google.com/a/gm.pu.edu.tw/rshiny-analysis/applications/association

[7] Ch.7 決策樹(Decision Tree)課堂投影片

https://rpubs.com/skydome20/R-Note6-Apriori-DecisionTree

[8] 百日機器學習馬拉松 https://www.cupoy.com/event/ml100/mission/1586225294223

[9] Ch.8 貝氏分類法、貝氏網路課堂投影片

[10] Logistic 迴歸原理及公式推導 https://www.itread01.com/content/1541769487.html

[11]正則化迴歸 https://www.itread01.com/content/1525847895.html

[12]支持向量機 Ch.9 支持向量機課堂投影片

[13] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2).

[14] Ahmad, T., Munir, A., Bhatti, S. H., Aftab, M., & Raza, M. A. (2017). Survival analysis of heart failure patients: A case study. PloS one, 12(7), e0181001.