機器學習概論一期末報告

作者：

數學二 412210010 施柏均

資工二 412210037 林威岑

113 年 1 月 3 日

目錄

1. Introduction

- 主題：**Predict Diabetes 預測糖尿病**
- 介紹：我們將透過 8 個與糖尿病相關的真實變數資料去預測病人是否會得糖尿病，也就是資料中的變數"Outcome"，Outcome 中的 0 代表沒得糖尿病，1 代表有得糖尿病。

  我們會使用羅吉斯迴歸 GLM、線性判別分析 LDA、 K-近鄰演算法 KNN(K-Nearest Neighbor)、主成分分析 PCA、決策樹 Dicision tree、隨機森林 Random Forest、支持向量機 SVM，配適出上述 7 個模型後並比較哪個模型擁有最好的預測能力。在這裡我們將處理後的資料隨機抽樣一半去做訓練資料集，剩下的一半則做為測試資料集。

- 資料來源：

  This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.2
  From the data set in the (.csv) File We can find several variables, some of them are independent (several medical predictor variables) and only one target dependent variable (Outcome).
  （https://www.kaggle.com/datasets/whenamancodes/predict-diabities/data）
- 原始資料維度：768*9
- 處理後（刪除 0 資料）的資料維度：392*9
- 變數資料：
  - Pregnancies(times)： To express the Number of pregnancies
  - Glucose(mg/DL)： To express the Glucose level in blood
  - BloodPressure (mm Hg) ： To express the Blood pressure measurement
  - SkinThickness(mm)： To express the thickness of the skin
  - Insulin(μ U/ml)： To express the Insulin level in blood
  - BMI(kg/m^2)： To express the Body mass index
  - DiabetesPedigreeFunction： To express the Diabetes percentage(根據家族糖尿病史推算得糖尿病的機率)

- Age(year) ： To express the age
- Outcome ： To express the final result 1 is Yes and 0 is No

▲ Outcome 是我們要預測的結果（1：有糖尿病 / 0：無糖尿病）

2. 解釋模型

2-1. Linear Regression

第一次配適

可以先看第一次所有變數配適出的線性回歸，也就是

Outcome = [-1.103e+00] + [Pregnancies* 1.295e-02] + ...

可以觀察結果我們發先 Glucose、BMI、DiabetesPedigreeFunction、Age 是很重要的變數，而其中 Gloucose 是非常重要的變數，而我們再拿這其中幾個再跟其他的幾個再去做配適。

```
## (Intercept)              -1.103e+00  1.436e-01  -7.681 1.34e-13 ***
## Pregnancies               1.295e-02  8.364e-03   1.549  0.12230
## Glucose                   6.409e-03  8.159e-04   7.855 4.07e-14 ***
## BloodPressure             5.465e-05  1.730e-03   0.032  0.97482
## BMI                       9.325e-03  3.901e-03   2.391  0.01730 *
## DiabetesPedigreeFunction  1.572e-01  5.804e-02   2.708  0.00707 **
## Age                       5.878e-03  2.787e-03   2.109  0.03559 *
## SkinThickness             1.678e-03  2.522e-03   0.665  0.50631
## Insulin                  -1.233e-04  2.045e-04  -0.603  0.54681
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3853 on 383 degrees of freedom
## Multiple R-squared:  0.3458, Adjusted R-squared:  0.3321
## F-statistic:  25.3 on 8 and 383 DF,  p-value: < 2.2e-16
```

第二次配適

這是第二次配適出來的線性回歸，也就是

Outcome = [-1.1183739] + [Age*0.0088972] + [Glucose*0.0061314] + [DiabetesPedigreeFunction* 0.1529780]

可以看出 Glucose、BMI、DiabetesPedigreeFunction、Age 非常重要，他的 R-squared 比第一次配適出來的還要高，所以每一個變數的重要性都提升了。

```
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.1183739  0.1164806  -9.601  < 2e-16 ***
## Age                       0.0088972  0.0020341   4.374 1.57e-05 ***
## Glucose                   0.0061314  0.0006882   8.909  < 2e-16 ***
## BMI                       0.0103822  0.0028591   3.631  0.00032 ***
## DiabetesPedigreeFunction  0.1529780  0.0574792   2.661  0.00810 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.385 on 387 degrees of freedom
## Multiple R-squared:  0.3398, Adjusted R-squared:  0.333
## F-statistic:  49.8 on 4 and 387 DF,  p-value: < 2.2e-16
```

2-2. 羅吉斯迴歸 GLM

第一次配適

最開始，我們先將所有變數都丟下去羅吉斯迴歸模型中配適，我們可以從下圖的執行結果觀察到只有變數"Glucose", “BMI”, “Age”在此模型中佔有顯著的重要性，尤其是 Glucose 血糖的資料！接著我們將變數篩選成這些重要的變數再重新配適一次模型。

```
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -1.104e+01  1.844e+00  -5.985 2.17e-09 ***
## Pregnancies               -6.846e-02  7.533e-02  -0.909   0.3635
## Glucose                    3.261e-02  7.669e-03   4.253 2.11e-05 ***
## BloodPressure              1.397e-02  1.868e-02   0.748   0.4546
## SkinThickness              1.429e-02  2.526e-02   0.565   0.5717
## Insulin                   -2.441e-04  2.089e-03  -0.117   0.9070
## BMI                        8.917e-02  4.124e-02   2.162   0.0306 *
## DiabetesPedigreeFunction   7.012e-01  6.649e-01   1.055   0.2916
## Age                        4.730e-02  2.594e-02   1.823   0.0683 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## AIC: 189.41
```

第二次配適

這是我們第二次配適的結果，我們可以觀察到當變數刪減到只剩重要的變數時，其他變數的重要性也能更顯著地被觀察出來，而觀察 AIC 數值也可以發現的確有下降。

```
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.082459   1.529839  -6.591 4.38e-11 ***
## Glucose       0.033218   0.006333   5.246 1.56e-07 ***
## BMI           0.111617   0.031846   3.505 0.000457 ***
## Age           0.038158   0.018840   2.025 0.042830 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## AIC: 182.06
```

預測結果

所以就將第二個配適的模型做為我們最終的羅吉斯配適模型：

Outcome = -10.0825 + 0.0332*Glucose + 0.1116*BMI + 0.0382*Age

我們同樣是透過提前分好的訓練集資料去訓練模型，接著再用測試集資料去預測結果並計算出模型的正確率。我們可以觀察到此模型的**正確率為 79%**。

```
## glm.pred   0   1
##        0 117  31
##        1  10  38

## 正確率

## [1] 0.7908163
```

2-3. 線性判別分析 LDA

從 linear regression 知道 Glucose、BMI、DiabetesPedigreeFunction、Age 這些去做分類，而這個配適出來的模型:

Outcome= Age* 0.02786757+ Glucose* 0.02601238+ BMI* 0.07485925+ DiabetesPedigreeFunction* 0.46170142

```
## Call:
## lda(Outcome ~ Age + Glucose + BMI + DiabetesPedigreeFunction,
##     data = Diabetes_data, subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6887755 0.3112245
##
## Group means:
##        Age  Glucose      BMI DiabetesPedigreeFunction
## 0 28.97037 112.4296 31.49556                0.4506741
## 1 35.13115 147.6066 36.00820                0.5709180
##
## Coefficients of linear discriminants:
##                                 LD1
## Age                      0.02786757
## Glucose                  0.02601238
## BMI                      0.07485925
## DiabetesPedigreeFunction 0.46170142
```

預測結果

我們同樣是透過提前分好的訓練集資料去訓練模型，接著再用測試集資料去預測結果並計算出模型的正確率。我們可以觀察到此模型的**正確率為 78%**。

```
## lda3.class   0   1
##          0 115  31
##          1  12  38
```

```
## 正確率
```

```
## [1] 0.7806122
```

2-4. K-近鄰演算法 KNN

KNN 演算法的概念是將每個樣本中與其周圍最近的 k 個鄰居劃分為一類。

在這邊我們分別使用了三組變數組合去跑 k=1~100 的所有 KNN 模型，讓我們快速找到 k 在 100 內的最佳 k，使得我們的模型擁有最高的準確率。其中三個模型分別是由

a. 全部的變數：Diabetes_data[train, -9]

b. 重要的變數：cbind(Glucose, BMI, DiabetesPedigreeFunction, Age)[train,]

c. 再篩選後的變數： cbind(Glucose, DiabetesPedigreeFunction, Age)[train,]

下去配飾的 KNN 模型。會這樣選是因為若是使用了無關的變數加進去選鄰居，會使的分類的結果變差，所以我們第一個選擇使用全部的變數下去作配適模型，可以讓我們做比較，第二個我們則是選擇前面分析中較常出現的重要性變數下去配適，接著第三個我們則是針對第二組的重要變數再進一步做篩選，我們測試過好幾次，發現當排除 BMI 的變數資料後會擁有更高的準確率。

從結果我們可以得知經過篩選後的 c 模型在 k=11 時會擁有最高的**正確率 80.10%**。這顯示在此模型下 Glucose、DiabetesPedigreeFunction 以及 Age 是分類鄰居的最佳指標，會使得我們預測測試集資料擁有最高的正確率。

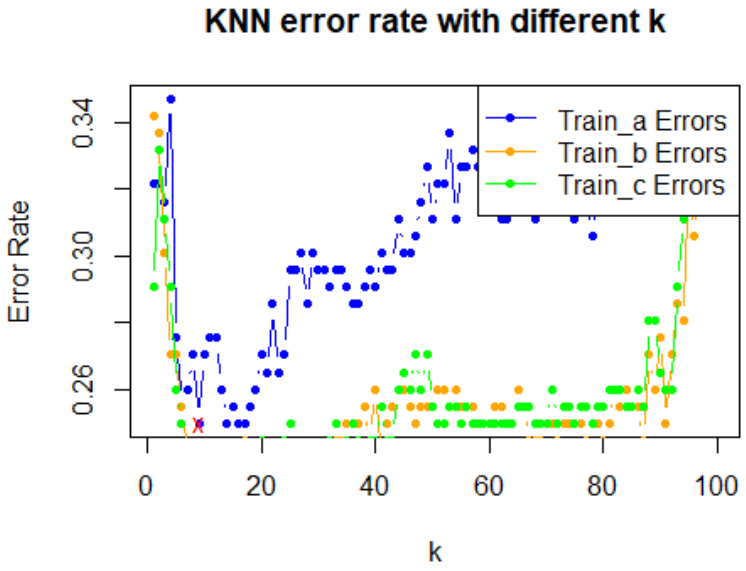
```
# 輸出最佳的train k 和test k，以及其所對應的正確率
c(best_train_a_k,best_train_b_k,best_train_c_k)
```

```
## [1]  9 10 11
```

```
c(round(1-little_train_a,digits=5),round(1-little_train_b,digits=5),round(1-
little_train_c,digits=5))
```

```
## [1] 0.75000 0.78571 0.80102
```

我們也可以觀察到隨著 k 越來越大，無論哪個變數組合的誤差都開始變高，因為當我們選太多個鄰居作為邊界，就會導致分類線太簡單；而若是 k 太小，我們可以看到每個變數的組合在一開始都有超級高的誤差，這是因為過度擬合了資料，分得太多太細，而使得分界線太複雜，所以誤差很高。



2-5. 主成分分析 PCA

PCA 分析結果

我們透過 PCA 主成分分析將資料保留住變異性最大的部分，也就是資料最精華的部分。

我們這邊就先探討前三個主成分，首先我們可以從第一主成分 PC1 觀察到在 Glucose 和 Outcome 中擁有最高的 loading，從我們前面觀察變數的重要性也大致可以同意這個觀點，在這邊我們可以形容 PC1 這筆資料在衡量「血糖相關指標」的屬性。所以若是 PC1 分數越高的人，通常代表血糖也越高，有更高的機率罹患糖尿病。

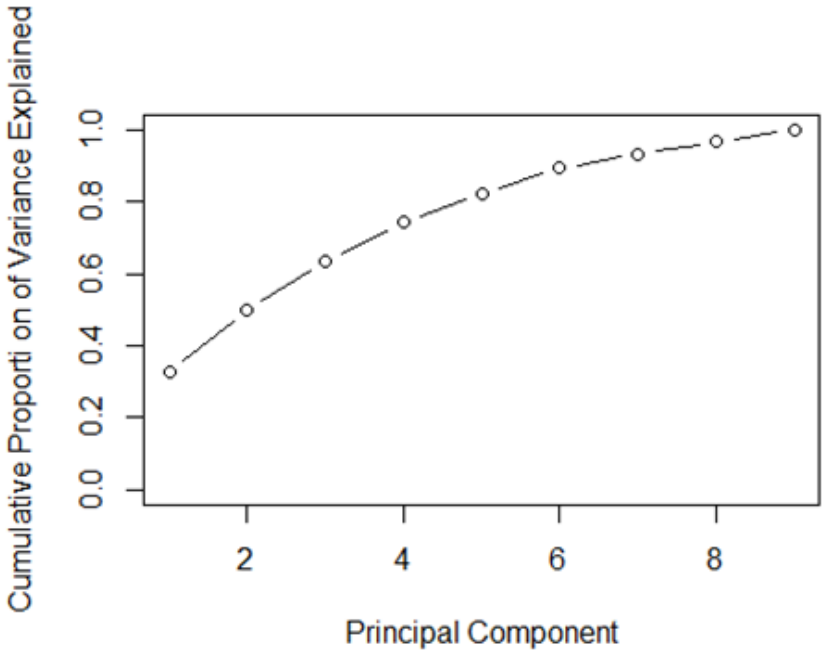
接著探討 PC2 的主成分，我們可以觀察到這筆資料著重的點在於 BMI 跟 SkinThickness，其擁有較高的 loading，並且和 Pregnancies、Age 呈現負相關，我們可以把 PC2 這筆資料想像成是在衡量「肥胖程度與年齡的相關指標」的屬性。所以若是 PC2 分數越高的人，通常代表 BMI 也就是肥胖指數和皮膚厚度越高，並且年齡越小，懷孕次數較少的人。

最後探討 PC3 的主成分，我們可以觀察到 Insulin 和 Glucose 擁有較高的負相關，以及和 BloodPressure 有較高的正相關，我們可以形容 PC3 這筆資料在形容「血壓、血糖與胰島素的相關指標」。也就是說 PC3 分數越高的人，代表其擁有較高的血壓，但是胰島素和血糖都偏低。

```
##                      PC1         PC2        PC3         PC4
## Pregnancies     0.2861619 -0.54048074  0.2600234 -0.18854762
## Glucose         0.4174054 -0.04390767 -0.4194181  0.25906169
## BloodPressure   0.2822264 -0.01607912  0.4345492  0.29956591
```

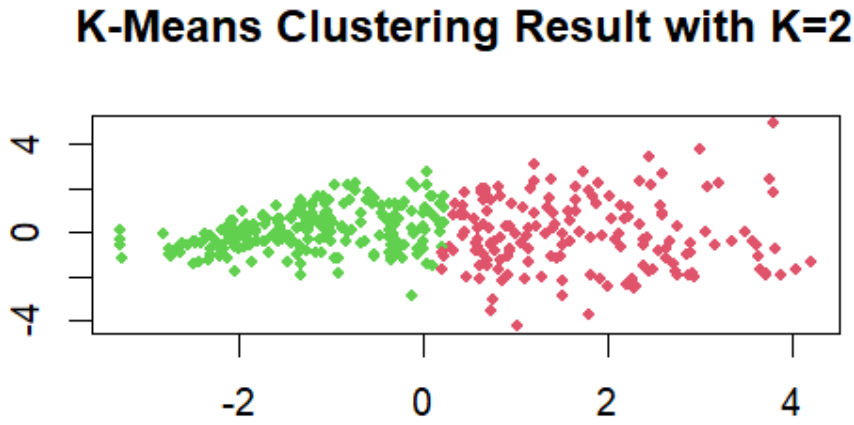
```
## SkinThickness             0.3333287  0.42666496  0.3323389 -0.08920747
## Insulin                   0.3375709  0.07236187 -0.4916387  0.31711763
## BMI                       0.3234157  0.51692606  0.2817824  0.03002165
## DiabetesPedigreeFunction 0.1672021  0.18723663 -0.2575468 -0.81903172
## Age                       0.3773315 -0.46387809  0.1739967 -0.13893589
## Outcome                   0.4052615 -0.03101504 -0.1986934 -0.08976693
##                                  PC5         PC6         PC7         PC8
## Pregnancies               0.24955642  0.11832785 -0.09735567 -0.21736288
## Glucose                  -0.06139077 -0.06728382  0.69533182 -0.29050231
## BloodPressure            -0.77534260  0.07799010 -0.02984687  0.14738893
## SkinThickness             0.38349232  0.06377869  0.32248505  0.56208037
## Insulin                   0.09000243  0.50275840 -0.47226034  0.21963675
## BMI                       0.14166125  0.01500220 -0.25337161 -0.66619419
## DiabetesPedigreeFunction -0.37873608  0.22789381  0.03650428 -0.01968351
## Age                       0.11553573  0.16504285  0.02644896  0.06421974
## Outcome                  -0.02779195 -0.79947196 -0.33598956  0.18476289
##                                  PC9
## Pregnancies               0.62435959
## Glucose                   0.06816477
## BloodPressure             0.10800026
## SkinThickness             0.13182123
## Insulin                   0.08000558
## BMI                      -0.13993679
## DiabetesPedigreeFunction  0.05240235
## Age                      -0.73989772
## Outcome                   0.01760479
```

我們也可以從主成分的累積解釋變異程度圖觀察到，PC1 可以解釋約 35%的變異程度，而隨著主成分越多，大概累積到 PC6 的時候可以解釋約 90%資料的變異程度，也就意味著我們可以保留大部分的原始資料，而不會損失太多資料。



將 PCA 主成分作為 K-means 的訓練集資料配適模型

而我們可以發現當我們使用前三個主成分作為訓練集資料下去做 K-means 時擁有最好的結果，以下是我們將資料分兩群的結果，可以看的出來區分的界線還算明顯，分得很好。



模型預測結果

從結果我們可以看到此模型擁有目前看到最高的**正確率 83.93%**。

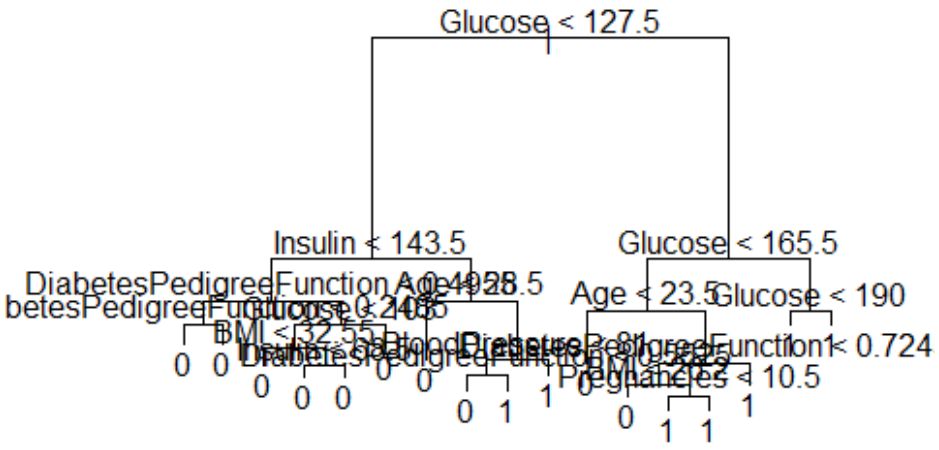
```
## result   0   1
##      0 214  15
##      1  48 115

## 正確率

## [1] 0.8392857
```

2-6. 決策樹 Dicision tree

這是一個最容易簡單判斷模型分類的一個模型，由下圖可知他是用 Glucose、 Insulin、 DiabetesPedigreeFunction、 BMI、 Age、 BloodPressure、 Pregnancies 這剩下的變數去做，並且這棵樹的節點有十七個。

```
## [1] "Glucose"                  "Insulin"
## [3] "DiabetesPedigreeFunction" "BMI"
## [5] "Age"                      "BloodPressure"
## [7] "Pregnancies"
## Number of terminal nodes:  17
```

配適出來如下圖



2-7. 隨機森林 Random Forest

隨機森林透過結合多個隨機生成的決策樹和隨機挑選的變數來提升模型準確率和穩定性。

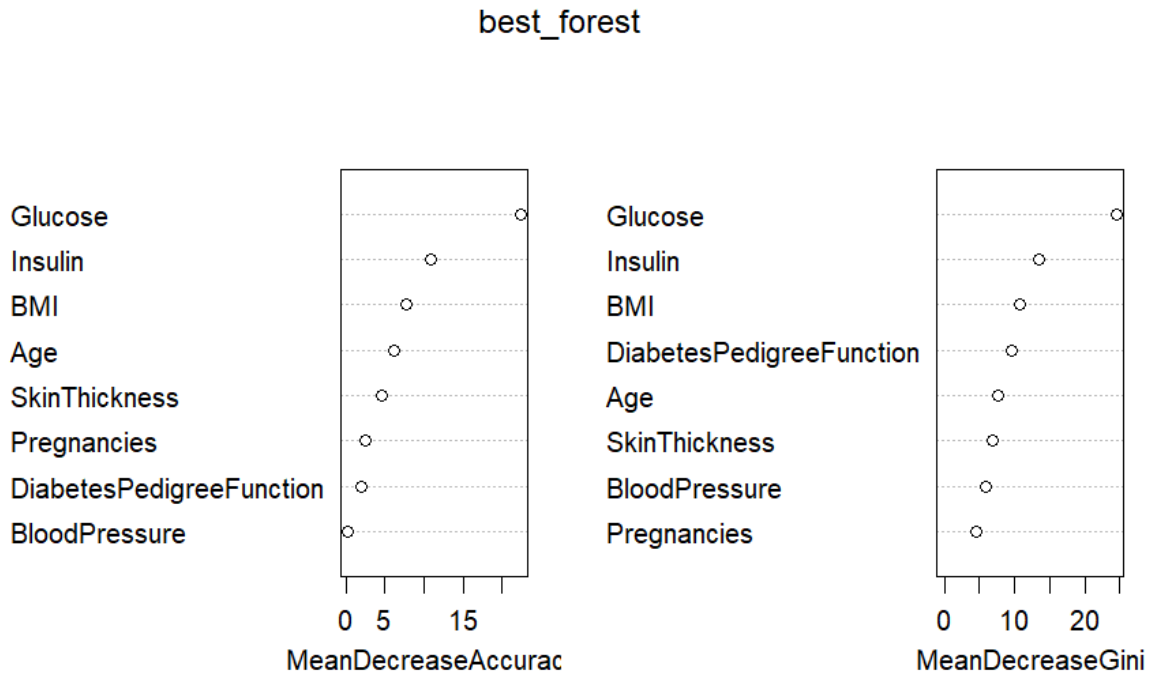
在這邊我使用迴圈跑過所有的 mtry 可能值，也就是隨機挑選的變數數量從 1~8 中找到一個準確率更高的隨機森林模型，理論上在 sqrt(8)=2.83 附近會有最佳的模型，而我們在 mtry=5 的時候找到最佳值，**正確率是 77.55%**。

```
## best_forest_pred   0   1
##                0 116  33
##                1  11  36

## 正確率

## [1] 0.7755102
```

以下是此最佳隨機森林模型的內容：

```
## Call:
## randomForest(formula = Outcome ~ ., data = Diabetes_data, mtry = x,
## importance = TRUE, subset = train)
##               Type of random forest: classification
##                     Number of trees: 500
## No. of variables tried at each split: 5
##
##        OOB estimate of  error rate: 22.45%
## Confusion matrix:
##     0  1 class.error
## 0 120 15   0.1111111
## 1  29 32   0.4754098
```

以下是在此最佳隨機森林模型下的變數重要性圖表，我們可以看出 Glucose 對模型的預測擁有非常重大的影響，再來是 Insulin 和 BMI 及 Age 擁有較顯著的影響，而若是將不重要的資料刪除後應該會有更好的模型表現。



2-8. 支持向量機 SVM

SVM 找到一條最佳分割平面，使得 margin 越遠越好，也就是兩個分類資料離越遠越好。

我們在這邊使用 tune 找到模型在給定參數下的最佳組合，結果是當 cost=1 並且當 gamma=0.5 時會有最小的誤差。

```
## - best parameters:
##  cost gamma
##     1   0.5
## - best performance: 0.2713158
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.3126316  0.1158944
## 2  1e+00   0.5 0.2713158  0.1154664
## 3  1e+01   0.5 0.3078947  0.1433453
## 4  1e+02   0.5 0.3078947  0.1433453
## 5  1e+03   0.5 0.3078947  0.1433453
## 6  1e-01   1.0 0.3126316  0.1158944
## 7  1e+00   1.0 0.3071053  0.1000389
## 8  1e+01   1.0 0.3073684  0.1372452
## 9  1e+02   1.0 0.3073684  0.1372452
## 10 1e+03   1.0 0.3073684  0.1372452
## 11 1e-01   2.0 0.3126316  0.1158944
## 12 1e+00   2.0 0.3126316  0.1158944
## 13 1e+01   2.0 0.3073684  0.1115367
```

```
## 14 1e+02   2.0 0.3073684  0.1115367
## 15 1e+03   2.0 0.3073684  0.1115367
## 16 1e-01   3.0 0.3126316  0.1158944
## 17 1e+00   3.0 0.3126316  0.1158944
## 18 1e+01   3.0 0.3126316  0.1158944
## 19 1e+02   3.0 0.3126316  0.1158944
## 20 1e+03   3.0 0.3126316  0.1158944
## 21 1e-01   4.0 0.3126316  0.1158944
## 22 1e+00   4.0 0.3126316  0.1158944
## 23 1e+01   4.0 0.3126316  0.1158944
## 24 1e+02   4.0 0.3126316  0.1158944
## 25 1e+03   4.0 0.3126316  0.1158944
```

而我們用此參數配適出的模型配適出來的**正確率為 56.69%**。

```
##     pred
## true  0  1
##    0 96 31
##    1 48 21

## 正確率

## [1] 0.5969388
```

3. 結論

配適模型	模型正確率
羅吉斯迴歸 GLM	0.7908163
線性判別分析 LDA	0.7806122
K-近鄰演算法 KNN	0.80102
主成分分析 PCA	0.8392857
決策樹 Dicision tree	-
隨機森林 Random Forest	0.7755102
支持向量機 SVM	0.5969388

綜合上述模型的結果比較，我們發現主成分分析（PCA）模型的正確率最高，達到 83.93%，使用的是前三個主成分作為訓練資料進行 K-means 分群分析。這說明 PCA 在降低資料維度的同時，能夠有效保留變數之間的主要資訊，提升模型的分類準確度。

另外，KNN 演算法也展現了不錯的表現（80.10%），而羅吉斯迴歸（GLM）和線性判別分析（LDA）表現則相近，分別為 79.08% 和 78.06%，隨機森林模型的正確率為 77.55%，然而支持向量機（SVM）的正確率僅為 59.69%，這可能與我們的資料特徵分布或模型的參數調整不夠理想有關。

在變數影響性分析方面，我們發現 **Glucose（葡萄糖濃度）** 是影響預測結果的關鍵變數。根據不同模型的變數重要性排序和主成分負荷量來看，Glucose 在大多數模型中皆佔有重要的權重。因此，可以推測 **血糖濃度** 在疾病預測中扮演關鍵角色。

4. 程式碼

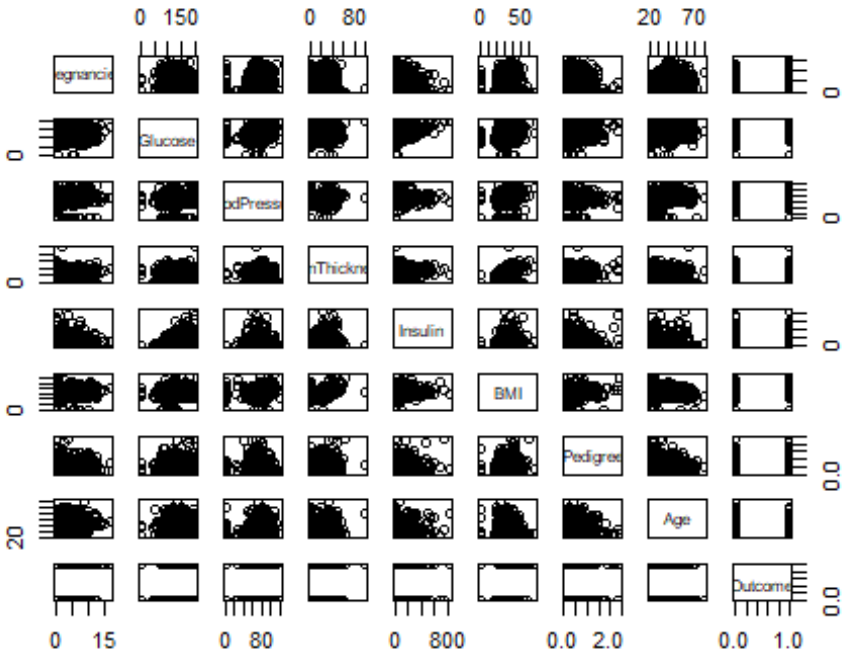
Topic:Diabetes Prediction

Import csv data

```
# 設定檔案的路徑
file_path = "C:/Users/USER/Documents/diabetes.csv"

# 載入 readr 套件
library(readr)

## Warning: 套件 'readr' 是用 R 版本 4.4.2 來建造的

# 使用 read_csv 匯入資料
Diabetes_data = read_csv(file_path)

## Rows: 768 Columns: 9
## ── Column specification ────────────────────────────────────────────────
──────
## Delimiter: ","
## dbl (9): Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI,
 D...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.

# 檢查資料
print("Original dimesions")

## [1] "Original dimesions"

dim(Diabetes_data)

## [1] 768   9

names(Diabetes_data)

## [1] "Pregnancies"              "Glucose"
## [3] "BloodPressure"            "SkinThickness"
## [5] "Insulin"                  "BMI"
## [7] "DiabetesPedigreeFunction" "Age"
## [9] "Outcome"

summary(Diabetes_data)

##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
```

```
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

```
# 觀察資料之間的關係
pairs(Diabetes_data)
```




```
cor(Diabetes_data)
```

```
##                           Pregnancies    Glucose BloodPressure SkinThicknes
s
## Pregnancies                1.00000000 0.12945867    0.14128198   -0.0816717
```

```
7
## Glucose                   0.12945867 1.00000000    0.15258959    0.0573278
9
## BloodPressure             0.14128198 0.15258959    1.00000000    0.2073705
4
## SkinThickness            -0.08167177 0.05732789    0.20737054    1.0000000
0
## Insulin                  -0.07353461 0.33135711    0.08893338    0.4367825
7
## BMI                       0.01768309 0.22107107    0.28180529    0.3925732
0
## DiabetesPedigreeFunction -0.03352267 0.13733730    0.04126495    0.1839275
7
## Age                       0.54434123 0.26351432    0.23952795   -0.1139702
6
## Outcome                   0.22189815 0.46658140    0.06506836    0.0747522
3
##                              Insulin        BMI DiabetesPedigreeFunction
## Pregnancies              -0.07353461 0.01768309              -0.03352267
## Glucose                   0.33135711 0.22107107               0.13733730
## BloodPressure             0.08893338 0.28180529               0.04126495
## SkinThickness             0.43678257 0.39257320               0.18392757
## Insulin                   1.00000000 0.19785906               0.18507093
## BMI                       0.19785906 1.00000000               0.14064695
## DiabetesPedigreeFunction  0.18507093 0.14064695               1.00000000
## Age                      -0.04216295 0.03624187               0.03356131
## Outcome                   0.13054795 0.29269466               0.17384407
##                                 Age    Outcome
## Pregnancies               0.54434123 0.22189815
## Glucose                   0.26351432 0.46658140
## BloodPressure             0.23952795 0.06506836
## SkinThickness            -0.11397026 0.07475223
## Insulin                  -0.04216295 0.13054795
## BMI                       0.03624187 0.29269466
## DiabetesPedigreeFunction  0.03356131 0.17384407
## Age                       1.00000000 0.23835598
## Outcome                   0.23835598 1.00000000
```

```
# 資料前處理(刪除等於0的欄位，但不包含懷孕變數中的0以及判斷結果的0)
Diabetes_data = Diabetes_data[rowSums(Diabetes_data[, -c(1,9)] == 0)==0,]
print("dimesions after delete rows which has 0\n")
```

```
## [1] "dimesions after delete rows which has 0\n"
```

```
dim(Diabetes_data)
```

```
## [1] 392   9
```

```
# 處理完後，再引入資料
attach(Diabetes_data)
knitr::opts_chunk$set(echo = TRUE)
```

Spilt half data into training set & testing set

```
set.seed(1)
train = sample(1:nrow(Diabetes_data), nrow(Diabetes_data)/2)
test = rep(1:nrow(Diabetes_data))
test = test[-train]
```

Lec1.Linear Regression

```
lm.fit = lm(Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesP
edigreeFunction + Age + SkinThickness + Insulin, data = Diabetes_data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
##     BMI + DiabetesPedigreeFunction + Age + SkinThickness + Insulin,
##     data = Diabetes_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.07966 -0.25711 -0.06177  0.25851  1.03750
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -1.103e+00  1.436e-01  -7.681 1.34e-13 ***
## Pregnancies                1.295e-02  8.364e-03   1.549  0.12230
## Glucose                    6.409e-03  8.159e-04   7.855 4.07e-14 ***
## BloodPressure              5.465e-05  1.730e-03   0.032  0.97482
## BMI                        9.325e-03  3.901e-03   2.391  0.01730 *
## DiabetesPedigreeFunction   1.572e-01  5.804e-02   2.708  0.00707 **
## Age                        5.878e-03  2.787e-03   2.109  0.03559 *
## SkinThickness              1.678e-03  2.522e-03   0.665  0.50631
## Insulin                   -1.233e-04  2.045e-04  -0.603  0.54681
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3853 on 383 degrees of freedom
## Multiple R-squared:  0.3458, Adjusted R-squared:  0.3321
## F-statistic:  25.3 on 8 and 383 DF,  p-value: < 2.2e-16
```

```
lm.fit = lm(Outcome ~ Pregnancies + Glucose + BMI +DiabetesPedigreeFunction,
data = Diabetes_data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Outcome ~ Pregnancies + Glucose + BMI + DiabetesPedigreeFunct
ion,
##     data = Diabetes_data)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -1.17701 -0.25751 -0.07474  0.26297  1.03755
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.0109077  0.1120866  -9.019  < 2e-16 ***
## Pregnancies               0.0256428  0.0062174   4.124 4.55e-05 ***
## Glucose                   0.0065590  0.0006655   9.856  < 2e-16 ***
## BMI                       0.0110866  0.0028731   3.859 0.000133 ***
## DiabetesPedigreeFunction  0.1658826  0.0575870   2.881 0.004191 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.386 on 387 degrees of freedom
## Multiple R-squared:  0.3364, Adjusted R-squared:  0.3295
## F-statistic: 49.04 on 4 and 387 DF,  p-value: < 2.2e-16
```

```
lm.fit = lm(Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesP
edigreeFunction + Insulin, data = Diabetes_data)
summary(lm.fit)
```

```
##
## Call:
## lm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
##     BMI + DiabetesPedigreeFunction + Insulin, data = Diabetes_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.13946 -0.26093 -0.07272  0.25821  1.05359
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.059e+00  1.405e-01  -7.539 3.42e-13 ***
## Pregnancies               2.499e-02  6.369e-03   3.924 0.000103 ***
## Glucose                   6.723e-03  8.068e-04   8.333 1.41e-15 ***
## BloodPressure             7.110e-04  1.710e-03   0.416 0.677814
## BMI                       1.087e-02  3.044e-03   3.571 0.000401 ***
## DiabetesPedigreeFunction  1.693e-01  5.798e-02   2.920 0.003711 **
## Insulin                  -9.223e-05  2.048e-04  -0.450 0.652722
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3868 on 385 degrees of freedom
## Multiple R-squared:  0.337,  Adjusted R-squared:  0.3267
## F-statistic: 32.62 on 6 and 385 DF,  p-value: < 2.2e-16
```

```
 lm.fit = lm(Outcome ~ Age + Glucose + BMI + DiabetesPedigreeFunction, data =
 Diabetes_data)
  summary(lm.fit)
```

```
##
## Call:
## lm(formula = Outcome ~ Age + Glucose + BMI + DiabetesPedigreeFunction,
##     data = Diabetes_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.11846 -0.26145 -0.07393  0.27717  1.04901
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)              -1.1183739  0.1164806  -9.601  < 2e-16 ***
## Age                       0.0088972  0.0020341   4.374 1.57e-05 ***
## Glucose                   0.0061314  0.0006882   8.909  < 2e-16 ***
## BMI                       0.0103822  0.0028591   3.631  0.00032 ***
## DiabetesPedigreeFunction  0.1529780  0.0574792   2.661  0.00810 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.385 on 387 degrees of freedom
## Multiple R-squared:  0.3398, Adjusted R-squared:  0.333
## F-statistic:  49.8 on 4 and 387 DF,  p-value: < 2.2e-16
```

```
knitr::opts_chunk$set(echo = TRUE)
```

Lec2.use logistic regression

```
# fitting glm model with all parameters
glm.fit = glm(Outcome~Pregnancies + Glucose + BloodPressure + SkinThickness +
 Insulin + BMI + DiabetesPedigreeFunction + Age, data=Diabetes_data, subset=t
rain, family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Outcome ~ Pregnancies + Glucose + BloodPressure +
##     SkinThickness + Insulin + BMI + DiabetesPedigreeFunction +
##     Age, family = binomial, data = Diabetes_data, subset = train)
##
## Coefficients:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)              -1.104e+01  1.844e+00  -5.985 2.17e-09 ***
## Pregnancies              -6.846e-02  7.533e-02  -0.909   0.3635
## Glucose                   3.261e-02  7.669e-03   4.253 2.11e-05 ***
## BloodPressure             1.397e-02  1.868e-02   0.748   0.4546
## SkinThickness             1.429e-02  2.526e-02   0.565   0.5717
## Insulin                  -2.441e-04  2.089e-03  -0.117   0.9070
## BMI                       8.917e-02  4.124e-02   2.162   0.0306 *
## DiabetesPedigreeFunction  7.012e-01  6.649e-01   1.055   0.2916
## Age                       4.730e-02  2.594e-02   1.823   0.0683 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 243.07  on 195  degrees of freedom
## Residual deviance: 171.41  on 187  degrees of freedom
## AIC: 189.41
##
## Number of Fisher Scoring iterations: 5
```

```
# fitting again with selected important parameters
glm.fit.1 = glm(Outcome~ Glucose + BMI + Age, data=Diabetes_data, subset=trai
n, family=binomial)
summary(glm.fit.1)
```

```
##
## Call:
## glm(formula = Outcome ~ Glucose + BMI + Age, family = binomial,
##     data = Diabetes_data, subset = train)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -10.082459   1.529839  -6.591 4.38e-11 ***
## Glucose       0.033218   0.006333   5.246 1.56e-07 ***
## BMI           0.111617   0.031846   3.505 0.000457 ***
## Age           0.038158   0.018840   2.025 0.042830 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 243.07  on 195  degrees of freedom
## Residual deviance: 174.06  on 192  degrees of freedom
## AIC: 182.06
##
## Number of Fisher Scoring iterations: 5
```

```
# show result model coefficient
summary(glm.fit.1)$coef
```

```
##                 Estimate  Std. Error   z value     Pr(>|z|)
## (Intercept) -10.08245946 1.529838981 -6.590536 4.382404e-11
## Glucose       0.03321775 0.006332572  5.245539 1.558262e-07
## BMI           0.11161699 0.031845680  3.504934 4.567211e-04
## Age           0.03815754 0.018839859  2.025362 4.283016e-02
```

```
# predict testing data
glm.probs = predict(glm.fit.1,Diabetes_data[test,],type="response")
glm.pred = rep(0,nrow(Diabetes_data)/2)
glm.pred[glm.probs>0.5] = 1
```

```r
# show result of prediction & accuracy
table(glm.pred, Outcome[test])
```

```
##
## glm.pred   0   1
##        0 117  31
##        1  10  38
```

```r
mean(glm.pred==Outcome[test])
```

```
## [1] 0.7908163
```

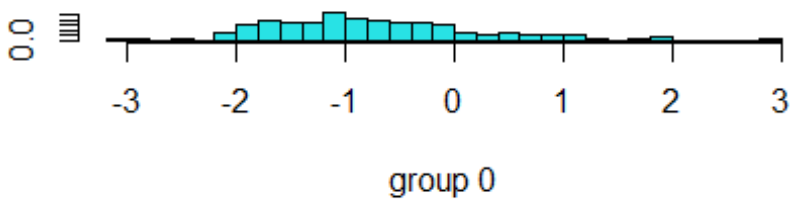
```r
knitr::opts_chunk$set(echo = TRUE)
```

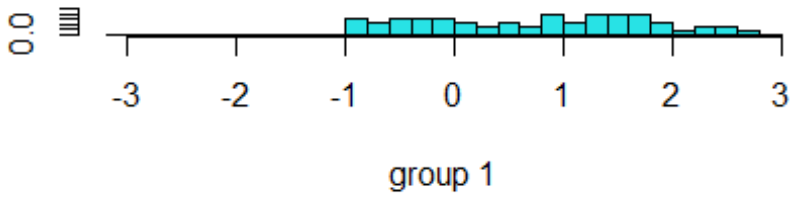
Lec3.LDA

```r
library(MASS)
lda1.fit = lda(Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + Diabet
esPedigreeFunction, data = Diabetes_data, subset = train)
lda1.fit
```

```
## Call:
## lda(Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesPedigr
eeFunction,
##     data = Diabetes_data, subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6887755 0.3112245
##
## Group means:
##   Pregnancies  Glucose BloodPressure      BMI DiabetesPedigreeFunction
## 0    3.066667 112.4296      69.34815 31.49556                 0.4506741
## 1    4.163934 147.6066      75.54098 36.00820                 0.5709180
##
## Coefficients of linear discriminants:
##                                 LD1
## Pregnancies              0.01984221
## Glucose                  0.02788976
## BloodPressure            0.01419150
## BMI                      0.06666257
## DiabetesPedigreeFunction 0.50711705
```

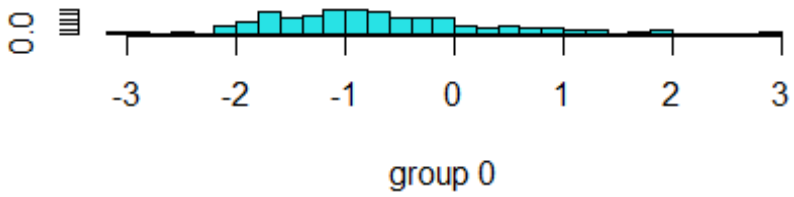
```r
plot(lda1.fit)
```



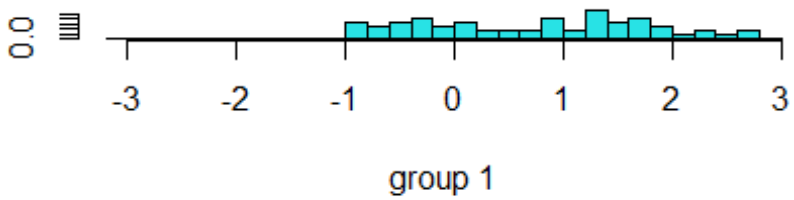



```
lda1.pred = predict(lda1.fit,Diabetes_data[test,])
lda1.class = lda1.pred$class

# show result of prediction & accuracy
table(lda1.class, Outcome[test])

##
## lda1.class   0   1
##          0 117  32
##          1  10  37

mean(lda1.class == Outcome[test])

## [1] 0.7857143

lda2.fit = lda(Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + Diabet
esPedigreeFunction + Insulin, data = Diabetes_data, subset = train)
lda2.fit

## Call:
## lda(Outcome ~ Pregnancies + Glucose + BloodPressure + BMI + DiabetesPedigr
eeFunction +
##     Insulin, data = Diabetes_data, subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6887755 0.3112245
##
```

```
## Group means:
##   Pregnancies  Glucose BloodPressure      BMI DiabetesPedigreeFunction  In
sulin
## 0    3.066667 112.4296      69.34815 31.49556                 0.4506741 12
8.5111
## 1    4.163934 147.6066      75.54098 36.00820                 0.5709180 21
3.0164
##
## Coefficients of linear discriminants:
##                                    LD1
## Pregnancies                0.0192794919
## Glucose                    0.0283883702
## BloodPressure              0.0140799340
## BMI                        0.0672568521
## DiabetesPedigreeFunction   0.5198510234
## Insulin                   -0.0002382696

plot(lda2.fit)
```



group 0



group 1

```
lda2.pred = predict(lda2.fit,Diabetes_data[test,])
lda2.class = lda2.pred$class

# show result of prediction & accuracy
table(lda2.class, Outcome[test])

##
## lda2.class   0   1
```

```
##          0 117  33
##          1  10  36
```

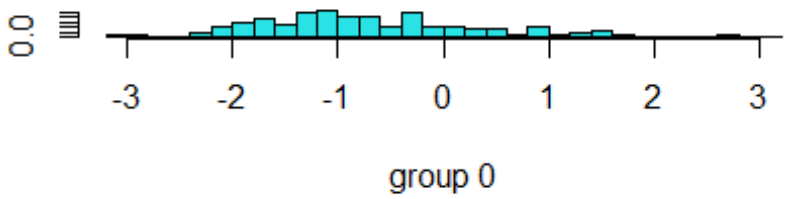
```
mean(lda2.class == Outcome[test])
```

```
## [1] 0.7806122
```

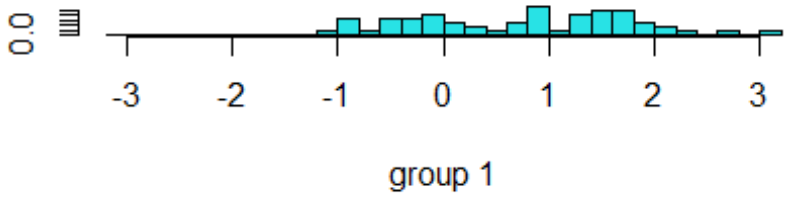
```
lda3.fit = lda(Outcome ~ Age + Glucose + BMI + DiabetesPedigreeFunction, data
 = Diabetes_data, subset = train)
lda3.fit
```

```
## Call:
## lda(Outcome ~ Age + Glucose + BMI + DiabetesPedigreeFunction,
##     data = Diabetes_data, subset = train)
##
## Prior probabilities of groups:
##         0         1
## 0.6887755 0.3112245
##
## Group means:
##        Age  Glucose      BMI DiabetesPedigreeFunction
## 0 28.97037 112.4296 31.49556                0.4506741
## 1 35.13115 147.6066 36.00820                0.5709180
##
## Coefficients of linear discriminants:
##                                 LD1
## Age                      0.02786757
## Glucose                  0.02601238
## BMI                      0.07485925
## DiabetesPedigreeFunction 0.46170142
```

```
plot(lda3.fit)
```



group 0



group 1

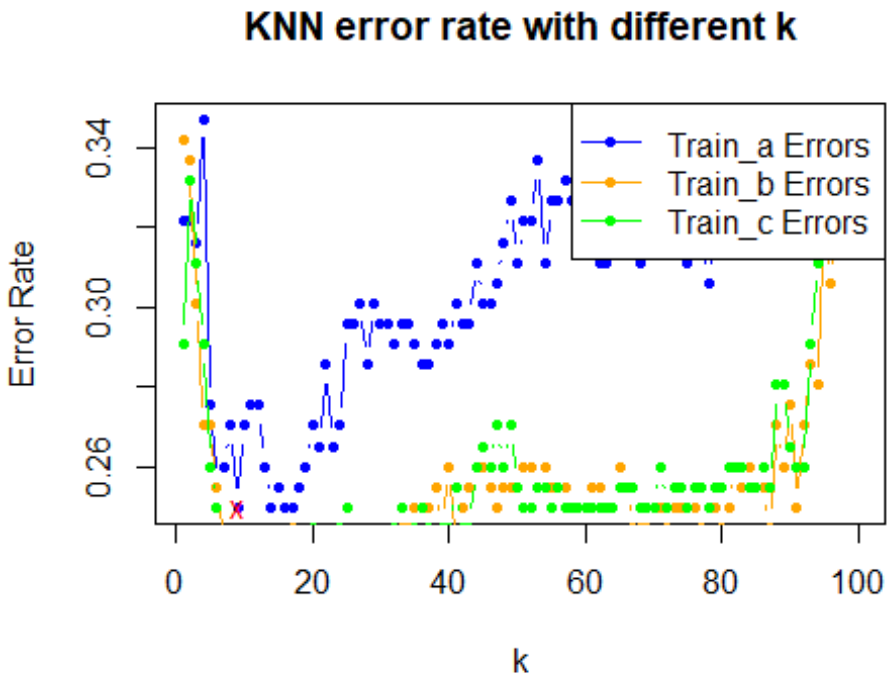
```
lda3.pred = predict(lda3.fit,Diabetes_data[test,])
lda3.class = lda3.pred$class

# show result of prediction & accuracy
table(lda3.class, Outcome[test])

##
## lda3.class   0   1
##          0 115  31
##          1  12  38

mean(lda3.class == Outcome[test])

## [1] 0.7806122

knitr::opts_chunk$set(echo = TRUE)
```

Lec4.use k-nearest neighbor(KNN)

跑迴圈 k = 1~100，找出最佳的 k

```
# import function
library(class)

# 設定訓練資料和測試資料 (a:全部的參數 / b:重要的參數 / c:再刪除 BMI)
train_a = Diabetes_data[train,-9]
test_a = Diabetes_data[test,-9]
```

```r
train_b = cbind(Glucose, BMI, DiabetesPedigreeFunction, Age)[train,]
test_b = cbind(Glucose, BMI, DiabetesPedigreeFunction, Age)[test,]

train_c = cbind(Glucose, DiabetesPedigreeFunction, Age)[train,]
test_c = cbind(Glucose, DiabetesPedigreeFunction, Age)[test,]

# 初始化
train_a.err = rep(0,100)
train_b.err = rep(0,100)
train_c.err = rep(0,100)
little_train_a = 1
little_train_b = 1
little_train_c = 1
best_train_a_k = 0
best_train_b_k = 0
best_train_c_k = 0

# 計算預測及誤差，還有找出最佳k
for(x in 1:100){
  set.seed(1)
  train_a.pred = knn(train_a , test_a , Outcome[train] , k=x)
  train_b.pred = knn(train_b , test_b , Outcome[train] , k=x)
  train_c.pred = knn(train_c , test_c , Outcome[train] , k=x)
  train_a.err[x] = mean(train_a.pred != Outcome[test])
  train_b.err[x] = mean(train_b.pred != Outcome[test])
  train_c.err[x] = mean(train_c.pred != Outcome[test])
  if(little_train_a>train_a.err[x]){
    little_train_a = train_a.err[x]
    best_train_a_k = x
  }
  if(little_train_b>train_b.err[x]){
    little_train_b = train_b.err[x]
    best_train_b_k = x
  }
  if(little_train_c>train_c.err[x]){
    little_train_c = train_c.err[x]
    best_train_c_k = x
  }
}
# 畫圖
plot(1:100 , train_a.err , type="b" , col="blue" ,main="KNN error rate with d
ifferent k" , xlab="k" , ylab="Error Rate" , pch=20)
points(1:100 , train_b.err , type="b" , col="orange" , pch=20)
points(1:100 , train_c.err , type="b" , col="green" , pch=20)
points(best_train_a_k, little_train_a , type="b" , col="red" , pch="x")
points(best_train_b_k , little_train_b , type="b" , col="red" , pch="x")
points(best_train_c_k , little_train_c , type="b" , col="red" , pch="x")
legend("topright" , legend = c("Train_a Errors","Train_b Errors","Train_c Err
ors") , col=c("blue","orange","green") , pch=20 , lty=1)
```



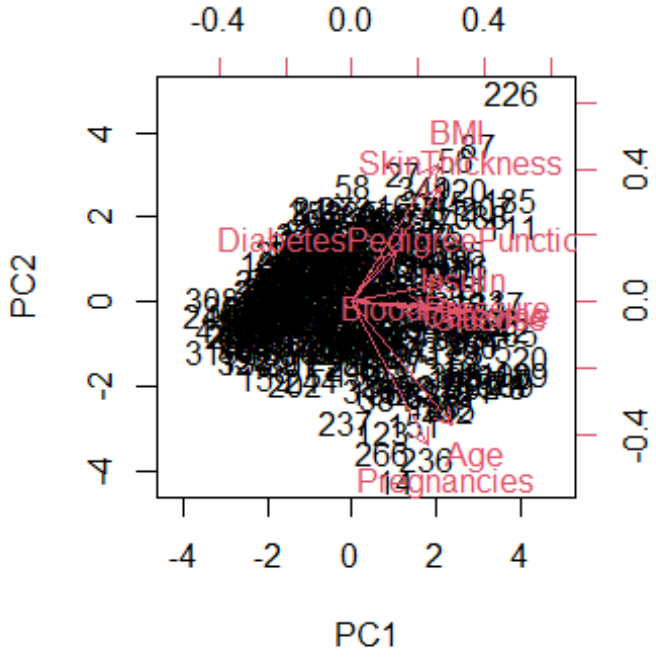

```
# 輸出最佳的train k 和test k，以及其所對應的正確率
c(best_train_a_k,best_train_b_k,best_train_c_k)

## [1]  9 10 11

c(round(1-little_train_a,digits=5),round(1-little_train_b,digits=5),round(1-l
ittle_train_c,digits=5))

## [1] 0.75000 0.78571 0.80102

knitr::opts_chunk$set(echo = TRUE)
```

Lec5.PCA

```
pr.out=prcomp(Diabetes_data,scale=TRUE)
names(pr.out)

## [1] "sdev"     "rotation" "center"   "scale"    "x"

pr.out$rotation

##                                PC1         PC2        PC3         PC4
## Pregnancies              0.2861619 -0.54048074  0.2600234 -0.18854762
## Glucose                  0.4174054 -0.04390767 -0.4194181  0.25906169
## BloodPressure            0.2822264 -0.01607912  0.4345492  0.29956591
## SkinThickness            0.3333287  0.42666496  0.3323389 -0.08920747
## Insulin                  0.3375709  0.07236187 -0.4916387  0.31711763
## BMI                      0.3234157  0.51692606  0.2817824  0.03002165
## DiabetesPedigreeFunction 0.1672021  0.18723663 -0.2575468 -0.81903172
```

```
## Age                          0.3773315 -0.46387809  0.1739967 -0.13893589
## Outcome                      0.4052615 -0.03101504 -0.1986934 -0.08976693
##                                     PC5         PC6         PC7         PC8
## Pregnancies                  0.24955642  0.11832785 -0.09735567 -0.21736288
## Glucose                     -0.06139077 -0.06728382  0.69533182 -0.29050231
## BloodPressure               -0.77534260  0.07799010 -0.02984687  0.14738893
## SkinThickness                0.38349232  0.06377869  0.32248505  0.56208037
## Insulin                      0.09000243  0.50275840 -0.47226034  0.21963675
## BMI                          0.14166125  0.01500220 -0.25337161 -0.66619419
## DiabetesPedigreeFunction    -0.37873608  0.22789381  0.03650428 -0.01968351
## Age                          0.11553573  0.16504285  0.02644896  0.06421974
## Outcome                     -0.02779195 -0.79947196 -0.33598956  0.18476289
##                                     PC9
## Pregnancies                  0.62435959
## Glucose                      0.06816477
## BloodPressure                0.10800026
## SkinThickness                0.13182123
## Insulin                      0.08000558
## BMI                         -0.13993679
## DiabetesPedigreeFunction     0.05240235
## Age                         -0.73989772
## Outcome                      0.01760479
```

```
biplot(pr.out, scale=0)
```




```
pr.out$scale
```

```
##               Pregnancies                   Glucose             BloodPressure
##                 3.2114245                30.8607806                12.4960916
##             SkinThickness                   Insulin                       BMI
##                10.5164239               118.8416898                 7.0276592
## DiabetesPedigreeFunction                       Age                   Outcome
##                 0.3454880                10.2007765                 0.4714014
```

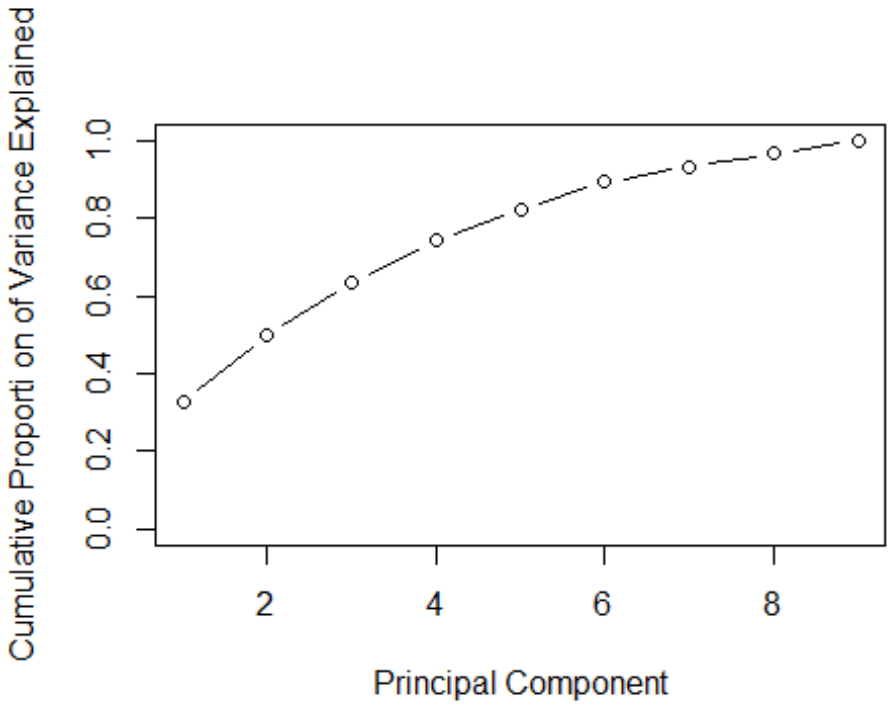
```
pr.out$sdev
```

```
## [1] 1.7136649 1.2481719 1.1091720 0.9796174 0.8486866 0.7996861 0.5963075
## [8] 0.5510200 0.5445494
```

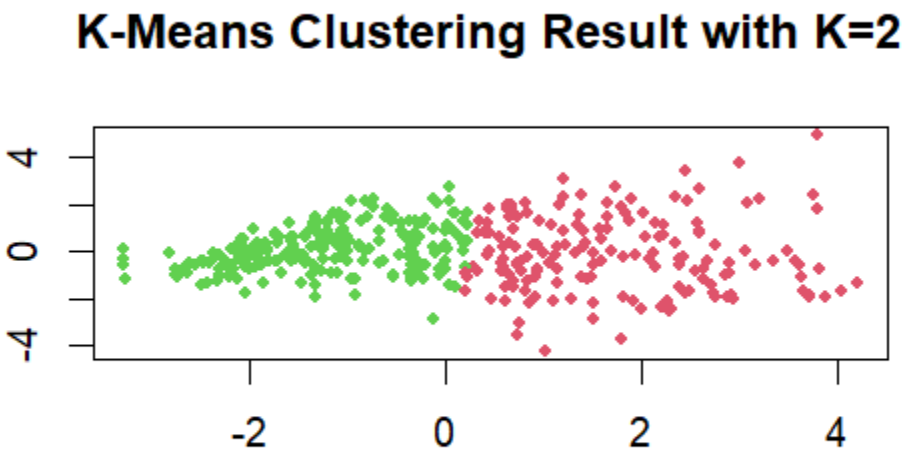
```
pr.var = pr.out$sdev^2
pve = pr.var/sum(pr.var)
pve
```

```
## [1] 0.32629417 0.17310368 0.13669583 0.10662782 0.08002989 0.07105532 0.03
950918
## [8] 0.03373589 0.03294822
```

```
plot(cumsum(pve), xlab="Principal Component", ylab="Cumulative Proporti on of
 Variance Explained", ylim=c(0,1),type='b')
```



```r
# 將PC1,PC2 作為要被分群的資料
train_pca = pr.out$x[,1:3]
km.out = kmeans(train_pca,2,nstart=30)
plot(train_pca , col=(km.out$cluster+1) , main="K-Means Clustering Result wit
h K=2" , xlab="" , ylab="" , pch=20 , cex=1)
```

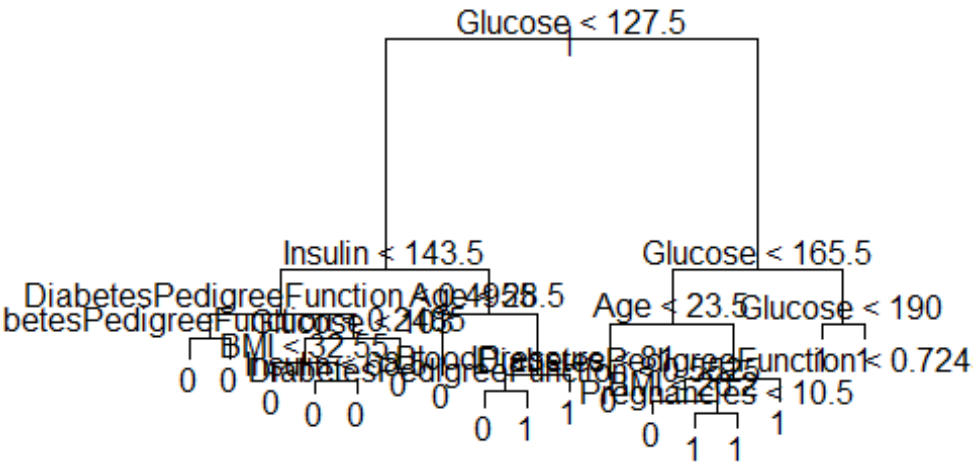



```r
result = rep(1,length(km.out$cluster))
result[km.out$cluster==2] = 0

# show the result and accuracy
table(result, t(Diabetes_data["Outcome"]))

##
## result   0   1
##      0 214  15
##      1  48 115

mean(result == t(Diabetes_data["Outcome"]))

## [1] 0.8392857

knitr::opts_chunk$set(echo = TRUE)
```

Lec11.Dicision tree (Classification tree)

```r
library(tree)

## Warning: 套件 'tree' 是用 R 版本 4.4.2 來建造的

Diabetes_data$Outcome = as.factor(Diabetes_data$Outcome)
tree.d = tree(Outcome~.-Outcome, Diabetes_data)
summary(tree.d)
```

```
##
## Classification tree:
## tree(formula = Outcome ~ . - Outcome, data = Diabetes_data)
## Variables actually used in tree construction:
## [1] "Glucose"                  "Insulin"
## [3] "DiabetesPedigreeFunction" "BMI"
## [5] "Age"                      "BloodPressure"
## [7] "Pregnancies"
## Number of terminal nodes:  17
## Residual mean deviance:  0.6294 = 236 / 375
## Misclassification error rate: 0.148 = 58 / 392

plot(tree.d)
text(tree.d,pretty=0)
```




```
knitr::opts_chunk$set(echo = TRUE)
```

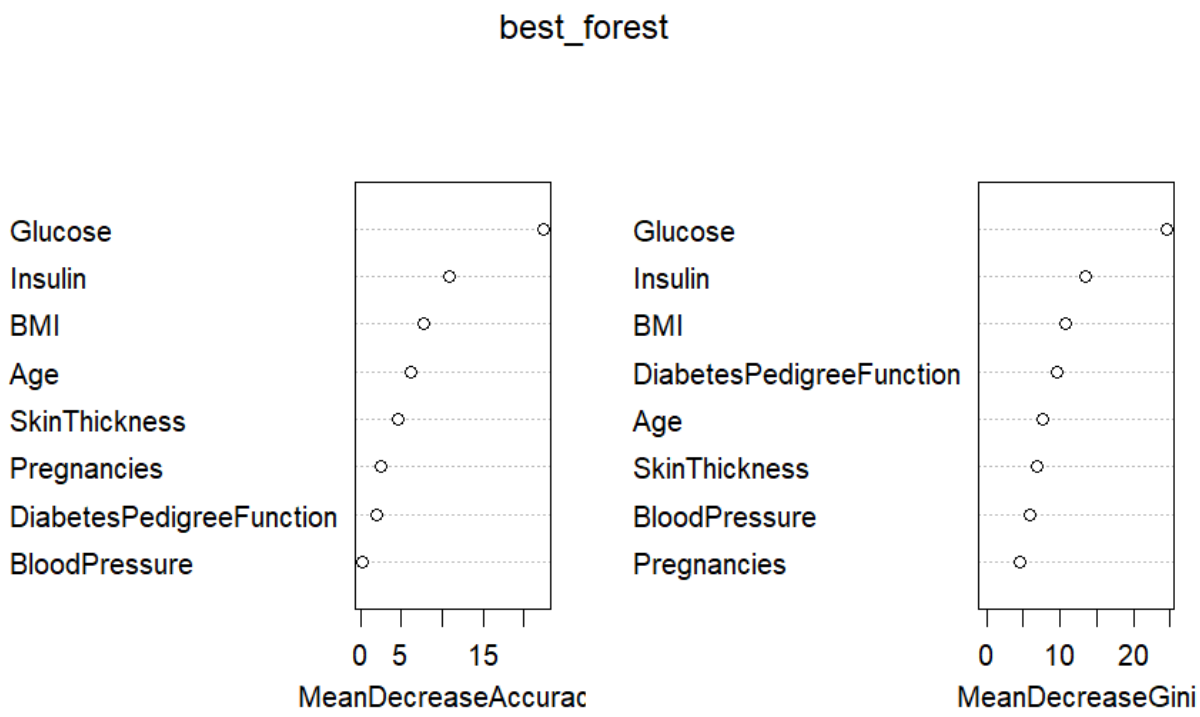
Lec11.Random Forest

```
# import functions
library(randomForest)

## Warning: 套件 'randomForest' 是用 R 版本 4.4.2 來建造的

## randomForest 4.7-1.2

## Type rfNews() to see new features/changes/bug fixes.
```

```r
Diabetes_data$Outcome = as.factor(Diabetes_data$Outcome)

best_x = 0
best_forest = 0
best_forest_pred = 0
best_forest_acc = 0

# build a forest with training set
for(x in 1:8){
  Diabete_forest = randomForest(Outcome~., data=Diabetes_data, subset = trai
n, mtry=x, importance=TRUE)
  # predict testing data
  Diabete_forest_pred = predict(Diabete_forest, newdata=Diabetes_data[test,
 ])
  accur_forest = mean(Outcome[test]==Diabete_forest_pred)
  if(accur_forest > best_forest_acc){
    best_x = x
    best_forest = Diabete_forest
    best_forest_pred = Diabete_forest_pred
    best_forest_acc = accur_forest
  }
}

Best_forest

##

## Call:
## randomForest(formula = Outcome ~ ., data = Diabetes_data, mtry = x,      #
# importance = TRUE, subset = train)
##               Type of random forest: classification
##                     Number of trees: 500
## No. of variables tried at each split: 5
##
##        OOB estimate of  error rate: 22.45%
## Confusion matrix:
##     0  1 class.error
## 0 120 15   0.1111111
## 1  29 32   0.4754098
best_x

## [1] 5

importance(best_forest)

##                                0          1 MeanDecreaseAccuracy
## Pregnancies             5.209779 -3.9645770            2.5510697
## Glucose                13.930817 18.0659369           22.4067909
## BloodPressure          -1.548222  2.6431990            0.2360875
## SkinThickness           3.736586  2.9358896            4.6014575
## Insulin                 4.532332 11.7465439           10.9214549
```

```
## BMI                          6.720604  4.3479663            7.7778515
## DiabetesPedigreeFunction     2.292963  0.5420898            1.9294765
## Age                          5.805903  2.2142898            6.2447697
##                           MeanDecreaseGini
## Pregnancies                       4.543178
## Glucose                          24.617193
## BloodPressure                     5.939702
## SkinThickness                     6.961102
## Insulin                          13.453945
## BMI                              10.707598
## DiabetesPedigreeFunction          9.548912
## Age                               7.700717
```

```r
varImpPlot(best_forest)
```




```r
# show the predict result
confusion_matrix = table(best_forest_pred, Outcome[test])
confusion_matrix
```

```
##
## best_forest_pred   0   1
##                0 116  33
##                1  11  36
```

```r
best_forest_acc
```

```
## [1] 0.7755102
```

```r
knitr::opts_chunk$set(echo = TRUE)
```

Lec14.SVM

```r
# import function
library(e1071)
```

```
## Warning: 套件 'e1071' 是用 R 版本 4.4.2 來建造的
```

```r
Diabetes_data$Outcome = as.factor(Diabetes_data$Outcome)

tune.out = tune(svm, Outcome~.,data=Diabetes_data[train,], kernel="radial", r
anges=list(cost=c(0.1,1,10,100,1000), gamma=c(0.5,1,2,3,4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     1   0.5
##
## - best performance: 0.2713158
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1  1e-01   0.5 0.3126316  0.1158944
## 2  1e+00   0.5 0.2713158  0.1154664
## 3  1e+01   0.5 0.3078947  0.1433453
## 4  1e+02   0.5 0.3078947  0.1433453
## 5  1e+03   0.5 0.3078947  0.1433453
## 6  1e-01   1.0 0.3126316  0.1158944
## 7  1e+00   1.0 0.3071053  0.1000389
## 8  1e+01   1.0 0.3073684  0.1372452
## 9  1e+02   1.0 0.3073684  0.1372452
## 10 1e+03   1.0 0.3073684  0.1372452
## 11 1e-01   2.0 0.3126316  0.1158944
## 12 1e+00   2.0 0.3126316  0.1158944
## 13 1e+01   2.0 0.3073684  0.1115367
## 14 1e+02   2.0 0.3073684  0.1115367
## 15 1e+03   2.0 0.3073684  0.1115367
## 16 1e-01   3.0 0.3126316  0.1158944
## 17 1e+00   3.0 0.3126316  0.1158944
## 18 1e+01   3.0 0.3126316  0.1158944
## 19 1e+02   3.0 0.3126316  0.1158944
## 20 1e+03   3.0 0.3126316  0.1158944
## 21 1e-01   4.0 0.3126316  0.1158944
## 22 1e+00   4.0 0.3126316  0.1158944
## 23 1e+01   4.0 0.3126316  0.1158944
```

```
## 24 1e+02   4.0 0.3126316  0.1158944
## 25 1e+03   4.0 0.3126316  0.1158944

true=Outcome[test]
pred=predict(tune.out$best.model, newx=Diabetes_data[test,])

table(true, pred)

##     pred
## true  0  1
##    0 96 31
##    1 48 21

mean(pred == true)

## [1] 0.5969388

knitr::opts_chunk$set(echo = TRUE)
```