

Case Studies in Data Mining with R

曾意儒 Yi-Ju Tseng

Chung Gung University

2020/05/18

Reviews

R and RStudio

R : Core (engine)



Source

R and RStudio

RStudio : IDE (dashboard)



Source

R and R Packages

R : Core (iPhone)



R and R Packages

R Packages : APP



Source

Install and Use Packages

- Install

```
install.packages("tidymodels")  
library(tidymodels)
```

- Use

```
install.packages("tidymodels")  
library(tidymodels)
```

```
## — Attaching packages ————— tidymodels 0.1.0 —
```

```
## ✓ broom      0.5.6      ✓ recipes      0.1.12  
## ✓ dials      0.0.6      ✓ rsample      0.0.6  
## ✓ dplyr      0.8.5      ✓ tibble       3.0.1  
## ✓ ggplot2    3.2.1      ✓ tune         0.1.0  
## ✓ infer      0.5.1      ✓ workflows    0.1.1  
## ✓ parsnip    0.1.1      ✓ yardstick    0.0.6  
## ✓ purrr      0.3.3
```

```
## — Conflicts ————— tidymodels_conflicts() —  
## x purrr::discard() masks scales::discard()
```

Pipe %>%

- 處理資料流的方法
- 將資料（輸出）丟到後方函數中

Get Started - Data Import

建模範例資料 - 載入

- mlbench套件內附的PimaIndiansDiabetes2資料集
- Outcome為是否有糖尿病 **diabetes**欄位
- 可輸入?PimaIndiansDiabetes2查看所有欄位變數

```
library(mlbench)  
data("PimaIndiansDiabetes2") # 載入資料
```

建模範例資料 - 初探

- 用`glimpse()`函數看一下各資料的型態跟資料筆數

```
glimpse(PimaIndiansDiabetes2)
```

```
## Rows: 768
## Columns: 9
## $ pregnant <dbl> 6, 1, 8, 1, 0, 5, 3, 10, 2, 8, 4, 10, 10, 1, 5, 7, 0, 7,
## $ glucose <dbl> 148, 85, 183, 89, 137, 116, 78, 115, 197, 125, 110, 168,
## $ pressure <dbl> 72, 66, 64, 66, 40, 74, 50, NA, 70, 96, 92, 74, 80, 60, 7
## $ triceps <dbl> 35, 29, NA, 23, 35, NA, 32, NA, 45, NA, NA, NA, NA, 23, 1
## $ insulin <dbl> NA, NA, NA, 94, 168, NA, 88, NA, 543, NA, NA, NA, NA, 846
## $ mass <dbl> 33.6, 26.6, 23.3, 28.1, 43.1, 25.6, 31.0, 35.3, 30.5, NA,
## $ pedigree <dbl> 0.627, 0.351, 0.672, 0.167, 2.288, 0.201, 0.248, 0.134, 0
## $ age <dbl> 50, 31, 32, 21, 33, 30, 26, 29, 53, 54, 30, 34, 57, 59, 5
## $ diabetes <fct> pos, neg, pos, neg, pos, neg, pos, neg, pos, pos, neg, po
```

- 發現有不少空值，要決定如何處理

建模範例資料 - Outcome比例

- 在處理空值前，先看**diabetes**欄位中有病跟沒病的人數與比例

```
PimaIndiansDiabetes2 %>%  
  count(diabetes) %>%  
  mutate(prop = n/sum(n))
```

```
## # A tibble: 2 x 3  
##   diabetes      n  prop  
##   <fct>      <int> <dbl>  
## 1 neg         500 0.651  
## 2 pos         268 0.349
```

- 大概是65:35，有些不平均，但還行

刪除空（遺漏）值

- 臨床研究通常會刪除有缺值的資料
- 為了後續敘述性統計方便，若狀況許可，會在第一步刪除資料
- 某些演算法不能用有空值的資料建模

```
PimaIndiansDiabetes2<-  
PimaIndiansDiabetes2 %>%  
filter(complete.cases(PimaIndiansDiabetes2))
```


刪除遺漏值

- 臨床研究通常會刪除有缺值的資料
- 為了後續敘述性統計方便，若狀況許可，會在第一步刪除資料
- 某些演算法不能用有空值的資料建模

```
PimaIndiansDiabetes2<-  
  PimaIndiansDiabetes2 %>%  
  filter(complete.cases(PimaIndiansDiabetes2))
```

- `filter()` 為 `dplyr` 套件的功能，可針對資料列(row)做子集

ID Name

1 a
2 b
3 c

ID Name

1 a
3 c

刪除遺漏值

- 臨床研究通常會刪除有缺值的資料
- 為了後續敘述性統計方便，若狀況許可，會在第一步刪除資料
- 某些演算法不能用有空值的資料建模

```
PimaIndiansDiabetes2<-  
  PimaIndiansDiabetes2 %>%  
  filter(complete.cases(PimaIndiansDiabetes2))
```

- `filter()`為dplyr套件的功能，可針對資料列(row)做子集
- 用`complete.cases()`辨識是否為完整資料，若沒有空值，回傳TRUE

刪除遺漏值後 - Outcome比例

刪除空後資料筆數有些微改變，但大概還是6x:3x的比例

```
PimaIndiansDiabetes2 %>%  
  count(diabetes) %>%  
  mutate(prop = n/sum(n))
```

```
## # A tibble: 2 x 3  
##   diabetes      n  prop  
##   <fct>      <int> <dbl>  
## 1 neg         262 0.668  
## 2 pos         130 0.332
```

敘述性統計

醫學資料分析中的敘述性統計

- 通常是論文中的必要表格
- 描述有病沒病的兩群人的基本資料，並做統計檢定

Patient characteristics in case and control groups.

	Metastasis group	Non-metastasis group	P-value
Patient, n	69	136	
Age, mean (SD)^a	45. 94 (12. 24)	49. 90 (11. 00)	0. 020 [*]
T, n (%)			0. 024 [*]
0	5 (7. 4)	13 (9. 7)	
is	2 (2. 9)	3 (2. 2)	
1mi	1 (1. 5)	7 (5. 2)	
1a	4 (5. 9)	10 (7. 5)	
1b	7 (10. 3)	22 (16. 4)	
1c	10 (14. 7)	34 (25. 4)	
2	29 (42. 6)	41 (30. 6)	
3	8 (11. 8)	2 (1. 5)	
4 (4a,4b,4c,4d)	2 (2. 9)	2 (1. 5)	
N, n (%)			< 0. 001
0	18 (26. 1)	71 (52. 2)	
1	17 (24. 6)	37 (27. 2)	
2	13 (18. 8)	17 (12. 5)	
3	21 (30. 4)	11 (8. 1)	
M = 0, n (%)	69 (100. 0)	136 (100. 0)	

Source

單變量分析

- 使用tableone套件完成表格，第一次使用前要先安裝
- 將去除空值的資料PimaIndiansDiabetes2放入CreateTableOne()函數

```
library(tableone)
tableone<-
  CreateTableOne(data=PimaIndiansDiabetes2)
print(tableone)
```

```
##
##                                Overall
##      n                                392
##      pregnant (mean (SD))    3.30 (3.21)
##      glucose (mean (SD))    122.63 (30.86)
##      pressure (mean (SD))   70.66 (12.50)
##      triceps (mean (SD))    29.15 (10.52)
##      insulin (mean (SD))   156.06 (118.84)
##      mass (mean (SD))       33.09 (7.03)
##      pedigree (mean (SD))   0.52 (0.35)
##      age (mean (SD))        30.86 (10.20)
##      diabetes = pos (%)      130 (33.2)
```

單變量分析 - 依outcome分組

```
library(tableone)
tableone<-
  CreateTableOne(data=PimaIndiansDiabetes2,
    strata = "diabetes")
print(tableone)
```

```
##                               Stratified by diabetes
##                               neg                pos                p          test
##    n                262                130
##    pregnant (mean (SD))    2.72 (2.62)    4.47 (3.92)    <0.001
##    glucose (mean (SD))    111.43 (24.64)    145.19 (29.84)    <0.001
##    pressure (mean (SD))    68.97 (11.89)    74.08 (13.02)    <0.001
##    triceps (mean (SD))    27.25 (10.43)    32.96 (9.64)    <0.001
##    insulin (mean (SD))    130.85 (102.63)    206.85 (132.70)    <0.001
##    mass (mean (SD))        31.75 (6.79)    35.78 (6.73)    <0.001
##    pedigree (mean (SD))    0.47 (0.30)    0.63 (0.41)    <0.001
##    age (mean (SD))        28.35 (8.99)    35.94 (10.63)    <0.001
##    diabetes = pos (%)      0 (0.0)        130 (100.0)    <0.001
```


單變量分析 - 貼到Excel的表格

```
library(tableone)
tableone<-
  CreateTableOne(data=PimaIndiansDiabetes2,
                 strata = "diabetes")
print(tableone,quote = T)
```

```
##           "Stratified by diabetes"
##      ""           "neg"           "pos"           "p"           "tes
##      "n"           "  262"           "  130"           ""           ""
##      "pregnant (mean (SD))" "  2.72 (2.62)" "  4.47 (3.92)" "<0.001" ""
##      "glucose (mean (SD))" "111.43 (24.64)" "145.19 (29.84)" "<0.001" ""
##      "pressure (mean (SD))" " 68.97 (11.89)" " 74.08 (13.02)" "<0.001" ""
##      "triceps (mean (SD))" " 27.25 (10.43)" " 32.96 (9.64)" "<0.001" ""
##      "insulin (mean (SD))" "130.85 (102.63)" "206.85 (132.70)" "<0.001" ""
##      "mass (mean (SD))" " 31.75 (6.79)" " 35.78 (6.73)" "<0.001" ""
##      "pedigree (mean (SD))" "  0.47 (0.30)" "  0.63 (0.41)" "<0.001" ""
##      "age (mean (SD))" " 28.35 (8.99)" " 35.94 (10.63)" "<0.001" ""
##      "diabetes = pos (%)" "      0 (0.0)" "    130 (100.0)" "<0.001" ""
```

單變量分析 - Excel處理步驟 (1/7)

調整Console視窗大小，讓結果不被強迫換行

```
""          "Stratified by diabetes"
""          "neg"          "pos"          "p"          "test"
"n"         " 262"         " 130"         ""          ""
"pregnant (mean (SD))" " 2.72 (2.62)" " 4.47 (3.92)" "<0.001" ""
"glucose (mean (SD))" "111.43 (24.64)" "145.19 (29.84)" "<0.001" ""
"pressure (mean (SD))" " 68.97 (11.89)" " 74.08 (13.02)" "<0.001" ""
"triceps (mean (SD))" " 27.25 (10.43)" " 32.96 (9.64)" "<0.001" ""
"insulin (mean (SD))" "130.85 (102.63)" "206.85 (132.70)" "<0.001" ""
"mass (mean (SD))"    " 31.75 (6.79)" " 35.78 (6.73)" "<0.001" ""
"pedigree (mean (SD))" " 0.47 (0.30)" " 0.63 (0.41)" "<0.001" ""
"age (mean (SD))"     " 28.35 (8.99)" " 35.94 (10.63)" "<0.001" ""
"diabetes = pos (%)"  "      0 (0.0)" "    130 (100.0)" "<0.001" ""
```

單變量分析 - Excel處理步驟 (2/7)

全選表格

"Stratified by diabetes"				
"n"	"neg"	"pos"	"p"	"test"
"n"	" 262"	" 130"	" "	" "
"pregnant (mean (SD))"	" 2.72 (2.62)"	" 4.47 (3.92)"	"<0.001"	" "
"glucose (mean (SD))"	"111.43 (24.64)"	"145.19 (29.84)"	"<0.001"	" "
"pressure (mean (SD))"	" 68.97 (11.89)"	" 74.08 (13.02)"	"<0.001"	" "
"triceps (mean (SD))"	" 27.25 (10.43)"	" 32.96 (9.64)"	"<0.001"	" "
"insulin (mean (SD))"	"130.85 (102.63)"	"206.85 (132.70)"	"<0.001"	" "
"mass (mean (SD))"	" 31.75 (6.79)"	" 35.78 (6.73)"	"<0.001"	" "
"pedigree (mean (SD))"	" 0.47 (0.30)"	" 0.63 (0.41)"	"<0.001"	" "
"age (mean (SD))"	" 28.35 (8.99)"	" 35.94 (10.63)"	"<0.001"	" "
"diabetes = pos (%)"	" 0 (0.0)"	" 130 (100.0)"	"<0.001"	" "

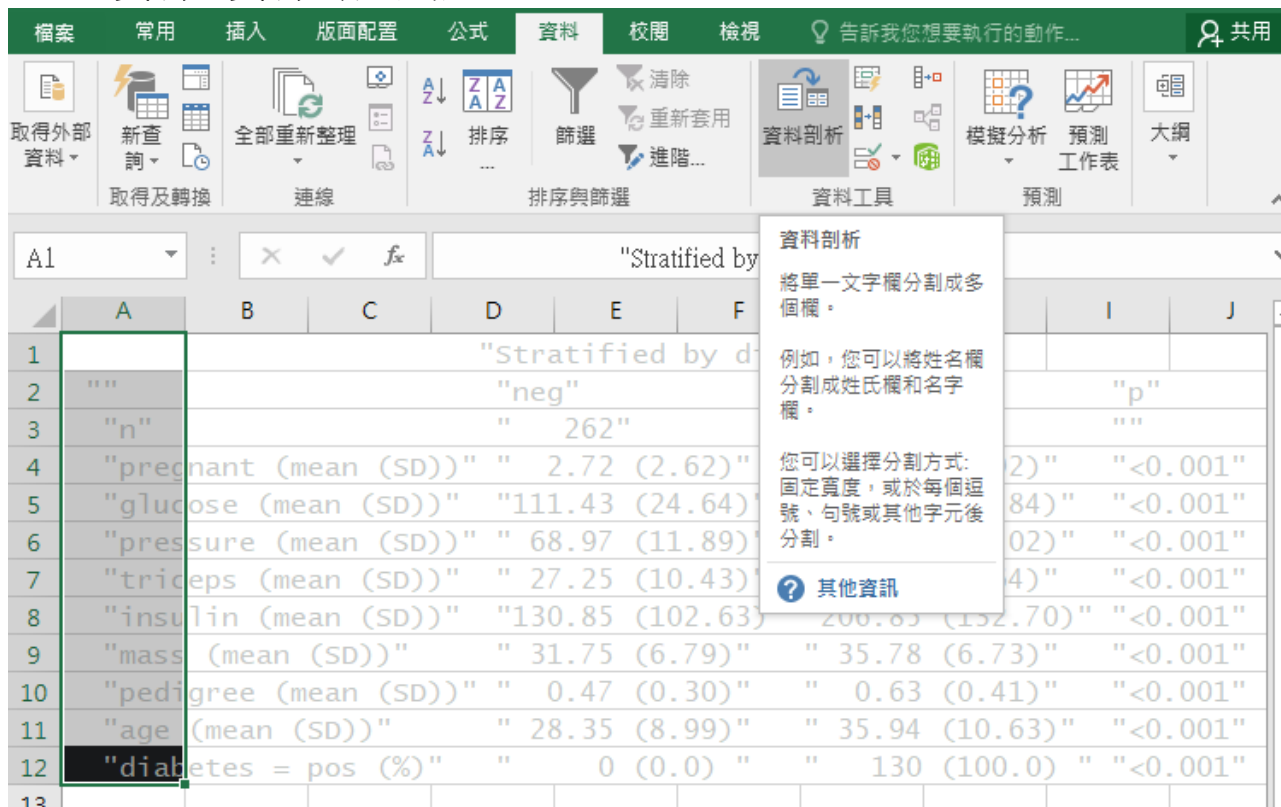
單變量分析 - Excel處理步驟 (3/7)

打開Excel貼上

	A	B	C	D	E	F	G	H	I	J	K	L
1	"Stratified by diabetes"											
2	""			"neg"			"pos"		"p"	"test"		
3	"n"			" 262"			" 130"		""	""		
4	"pregnant (mean (SD))"			" 2.72 (2.62)"			" 4.47 (3.92)"		"<0.001"	""		
5	"glucose (mean (SD))"			"111.43 (24.64)"			"145.19 (29.84)"		"<0.001"	""		
6	"pressure (mean (SD))"			" 68.97 (11.89)"			" 74.08 (13.02)"		"<0.001"	""		
7	"triceps (mean (SD))"			" 27.25 (10.43)"			" 32.96 (9.64)"		"<0.001"	""		
8	"insulin (mean (SD))"			"130.85 (102.63)"			"206.85 (132.70)"		"<0.001"	""		
9	"mass (mean (SD))"			" 31.75 (6.79)"			" 35.78 (6.73)"		"<0.001"	""		
10	"pedigree (mean (SD))"			" 0.47 (0.30)"			" 0.63 (0.41)"		"<0.001"	""		
11	"age (mean (SD))"			" 28.35 (8.99)"			" 35.94 (10.63)"		"<0.001"	""		
12	"diabetes = pos (%)"			" 0 (0.0)"			" 130 (100.0)"		"<0.001"	""		

單變量分析 - Excel處理步驟 (4/7)

打開Excel->資料->資料剖析功能



資料剖析

將單一文字欄分割成多個欄。

例如，您可以將姓名欄分割成姓氏欄和名字欄。

您可以選擇分割方式：固定寬度，或於每個逗號、句號或其他字元後分割。

[其他資訊](#)

	A	B	C	D	E	F	I	J
1				"Stratified by d				
2	"n"			"neg"			"p"	
3	"n"			" 262"			" "	
4	"pregnant (mean (SD))"			" 2.72 (2.62)"			2)"	"<0.001"
5	"glucose (mean (SD))"			"111.43 (24.64)"			84)"	"<0.001"
6	"pressure (mean (SD))"			" 68.97 (11.89)"			02)"	"<0.001"
7	"triceps (mean (SD))"			" 27.25 (10.43)"			4)"	"<0.001"
8	"insulin (mean (SD))"			"130.85 (102.63)"			200.85 (152.70)"	"<0.001"
9	"mass (mean (SD))"			" 31.75 (6.79)"			" 35.78 (6.73)"	"<0.001"
10	"pedigree (mean (SD))"			" 0.47 (0.30)"			" 0.63 (0.41)"	"<0.001"
11	"age (mean (SD))"			" 28.35 (8.99)"			" 35.94 (10.63)"	"<0.001"
12	"diabetes = pos (%)"			" 0 (0.0) "			" 130 (100.0) "	"<0.001"
13								

單變量分析 - Excel處理步驟 (5/7)

設定使用分隔符號切割資料

資料剖析精靈 - 步驟 3 之 1

資料剖析精靈判定資料類型為固定寬度。
若一切設定無誤，請選取 [下一步]，或選取適當的資料類別。

原始資料類型

請選擇最適合剖析您的資料的檔案類型：

☒ 分隔符號(D) - 用分欄字元，如逗號或 TAB 鍵，區分每一個欄位。

☐ 固定寬度(W) - 每個欄位固定，欄位間以空格區分。

預覽選取的資料：

1	"Stratified by diabetes"				
2	" "	"neg"	"pos"	"p"	"tes"
3	"n"	" 262"	" 130"	" "	" "
4	"pregnant (mean (SD))"	" 2.72 (2.62)"	" 4.47 (3.92)"	"<0.001"	" "
5	"glucose (mean (SD))"	"111.43 (24.64)"	"145.19 (29.84)"	"<0.001"	" "

< 取消 < 上一步(B) 下一步(N) > 完成(F)

單變量分析 - Excel處理步驟 (6/7)

用空格當作分隔符號，並將文字辨識符號設為雙引號

資料剖析精靈 - 步驟 3 之 2

您可在此畫面中選擇輸入資料中所包含的分隔符號，您可在預覽視窗內看到分欄的結果。

分隔符號

☐ Tab 鍵(T)

☐ 分號(M)

☐ 逗點(C)

☒ 空格(S)

☐ 其他(O):

☒ 連續分隔符號視為單一處理(R)

文字辨識符號(Q):

預覽分欄結果(P)

Stratified by diabetes	neg	pos	p	test
n	262	130		
pregnant (mean (SD))	2.72 (2.62)	4.47 (3.92)	<0.001	
glucose (mean (SD))	111.43 (24.64)	145.19 (29.84)	<0.001	

< >

取消 < 上一步(B) 下一步(N) > 完成(F)

單變量分析 - Excel處理步驟 (7/7)

完成！！

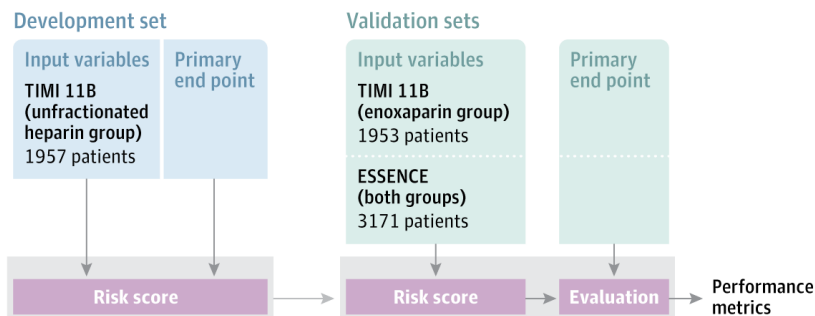
	neg	pos	p
n	262	130	
pregnant (mean (SD))	2.72 (2.62)	4.47 (3.92)	<0.001
glucose (mean (SD))	111.43 (24.64)	145.19 (29.84)	<0.001
pressure (mean (SD))	68.97 (11.89)	74.08 (13.02)	<0.001
triceps (mean (SD))	27.25 (10.43)	32.96 (9.64)	<0.001
insulin (mean (SD))	130.85 (102.63)	206.85 (132.70)	<0.001
mass (mean (SD))	31.75 (6.79)	35.78 (6.73)	<0.001
pedigree (mean (SD))	0.47 (0.30)	0.63 (0.41)	<0.001
age (mean (SD))	28.35 (8.99)	35.94 (10.63)	<0.001
diabetes = pos (%)	0 (0.0)	130 (100.0)	<0.001

開始建模 - 策略說明與資料分割

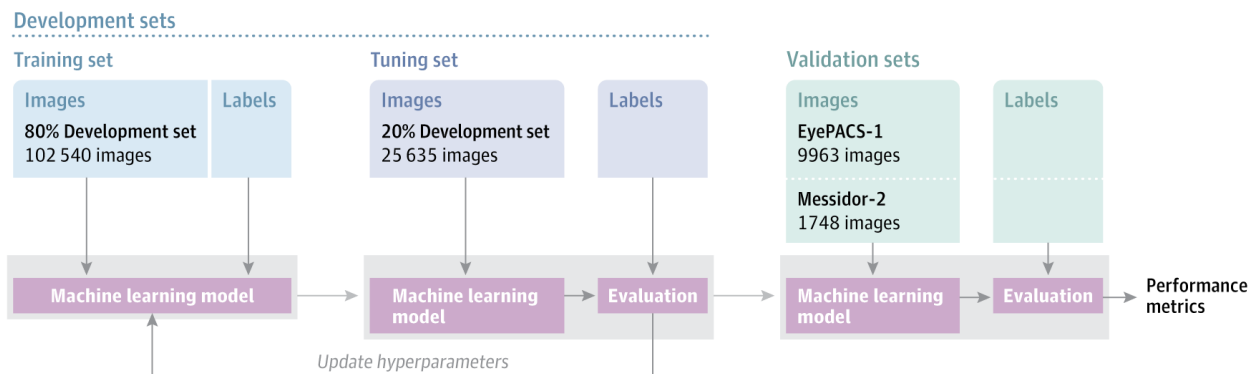
訓練/調整與驗證模型效能策略

How to Read Articles That Use Machine Learning - Users' Guides to the Medical Literature **JAMA**, 說明在機器學習時代如何建立預測模型，跟傳統的不同

A Decision-rule model



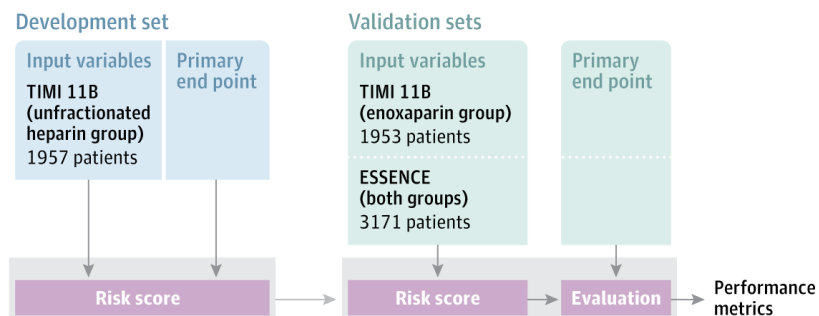
B Machine learning model



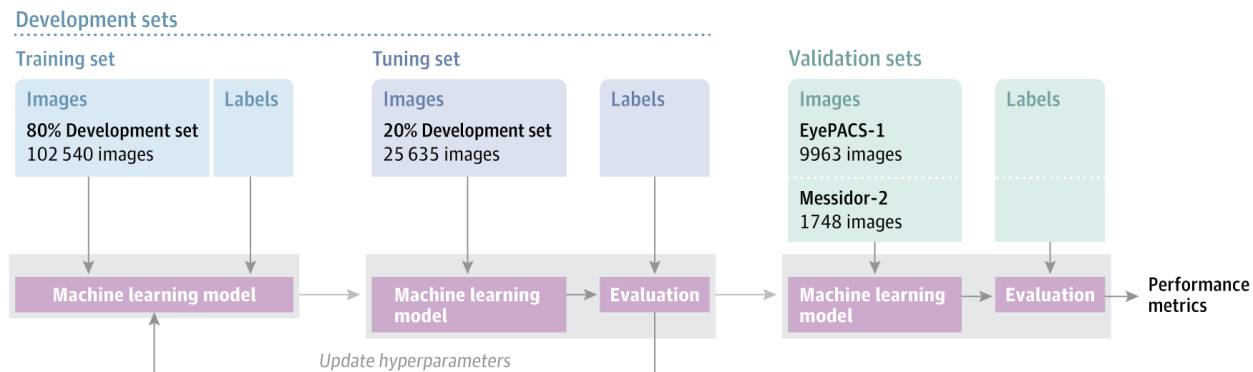
調整參數的必要性

- 在傳統"非"機器學習模型(圖上半部)，分訓練組與測試組
- 訓練模型時，記得不可以用測試組資料來訓練模型，才能得到真實的預測結果(完全沒偷看答案的意思)

A Decision-rule model



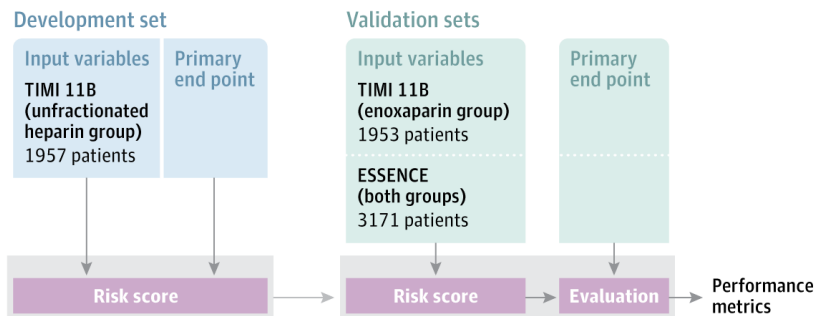
B Machine learning model



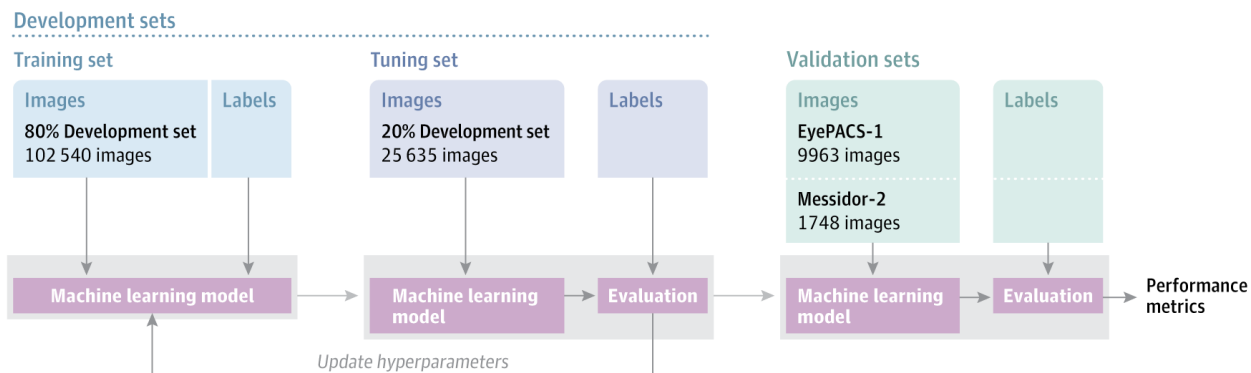
調整參數的必要性

- 在圖下半，主要差異是在訓練模型時多了一組參數調整資料集
- 因為在多種機器學習模型中，有可調整的參數
- 為了讓參數調整效果更好，會將訓練組再切分成小組，用來決定最佳參數
- 決定最佳參數以後，用訓練組搭配最佳參數訓練模型，最後用測試組測試

A Decision-rule model



B Machine learning model

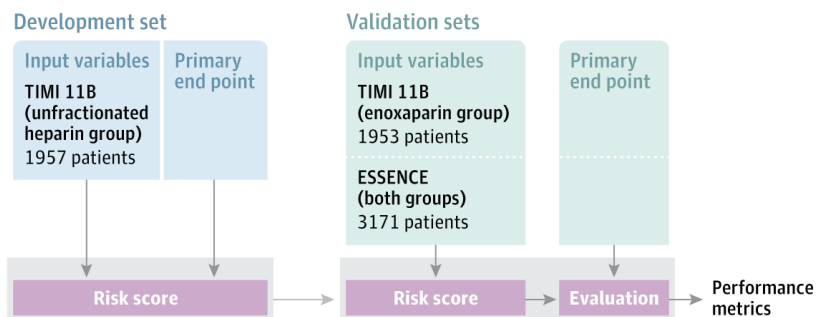


調整參數的必要性

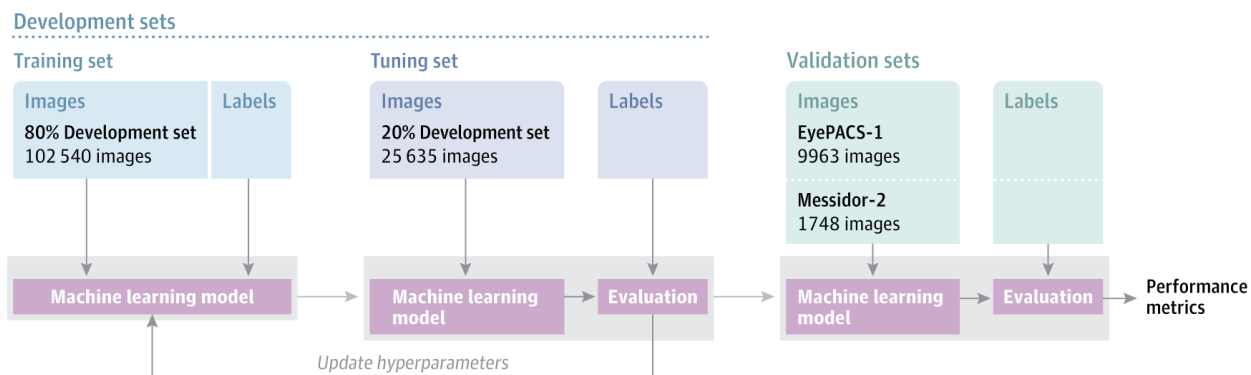
本課程範例：

- 邏輯迴歸**Logistic Regression**示範此圖上半部不調參數的作法
- 隨機森林**Random Forest**示範此圖下半部需要調整參數的作法

A Decision-rule model



B Machine learning model



訓練組、測試組資料分割

- 不管要不要調參數，都需要分割訓練組與測試組
- 用`initial_split()`函數將資料分成訓練組與測試組
 - 第一個參數放資料`PimaIndiansDiabetes2`
 - 第二個參數`prop`放訓練組測試組比例(3/4)
 - 第三個參數`strata`放分組抽樣依據`diabetes`，針對此分組個別抽某個比例的人當訓練組，剩下的就當測試組。

```
set.seed(123)
splits<- initial_split(PimaIndiansDiabetes2,
                        prop=(3/4),
                        strata = diabetes)
DM_train<- training(splits)
DM_test<- testing(splits)
```


訓練組、測試組資料分割

- 不管要不要調參數，都需要分割訓練組與測試組
- 用`initial_split()`函數將資料分成訓練組與測試組
 - 第一個參數放資料`PimaIndiansDiabetes2`
 - 第二個參數`prop`放訓練組測試組比例(3/4)
 - 第三個參數`strata`放分組抽樣依據`diabetes`，針對此分組個別抽某個比例的人當訓練組，剩下的就當測試組。
- 切割完後，用`training()`和`testing()`函數將訓練組測試組正式分開。

```
set.seed(123)
splits<- initial_split(PimaIndiansDiabetes2,
                        prop=(3/4),
                        strata = diabetes)
DM_train<- training(splits)
DM_test<- testing(splits)
```

訓練組、測試組資料分割

- `initial_split()` 函數有隨機的概念
- 為了讓每次的實驗結果相同，要在有隨機事件前設定 **seed** `set.seed()`
- **seed** 為隨機的依據，讓隨機每次都一樣，確保實驗結果可重複

```
set.seed(123)
splits<- initial_split(PimaIndiansDiabetes2,
                        prop=(3/4),
                        strata = diabetes)
DM_train<- training(splits)
DM_test<- testing(splits)
```

查看分組結果

分組完後，查看訓練組的生病比例

```
DM_train %>%  
  count(diabetes) %>%  
  mutate(prop = n/sum(n))
```

```
## # A tibble: 2 x 3  
##   diabetes      n prop  
##   <fct>    <int> <dbl>  
## 1 neg       197 0.668  
## 2 pos        98 0.332
```

查看測試組的生病比例

```
DM_test %>%  
  count(diabetes) %>%  
  mutate(prop = n/sum(n))
```

```
## # A tibble: 2 x 3  
##   diabetes      n prop  
##   <fct>    <int> <dbl>  
## 1 neg        65 0.670  
## 2 pos        32 0.330
```

開始建模 - 資料前處理

建立資料前處理“食譜”

- 資料前處理也是建立兩種模型都必須經歷的方法
- 前處理包括將類別變項轉為虛擬變項(dummy variables)，數值變項取log，數值變項正規化，以及日期資料處理等
- 首先使用recipe()函數，設定模型訓練公式diabetes ~ .以及訓練用資料DM_train
 - 注意這邊只能用訓練組資料，不能用全部的資料
 - 公式：用~前方的diabetes當outcome (想要預測的值)，~後方的.代表其他所有剩下的欄位都當成predictors (又稱variables 或是 features)

```
gen_recipe <-  
  recipe(diabetes ~ ., data = DM_train) %>%  
    step_dummy(all_nominal(), -all_outcomes()) %>%  
    step_zv(all_predictors()) %>%  
    step_normalize(all_predictors())  
summary(gen_recipe)
```

```
## # A tibble: 9 x 4  
##   variable type      role      source  
##   <chr>      <chr>    <chr>    <chr>  
## 1 pregnant numeric predictor original
```

建立資料前處理“食譜”

完成模型公式與資料設定後，就開始逐一加上想要做的資料前處理方法

- `step_dummy(all_nominal(), -all_outcomes())` 將所有的類別變項轉成虛擬變項，除了Outcome以外
- `step_zv(all_predictors())` 若有變項都是一樣的值，刪掉。舉例來說，若是整個資料都是女性，那性別欄位就不用拿來當作features了
- `step_normalize(all_predictors())` 將所有數值變項正規化

```
gen_recipe <-  
  recipe(diabetes ~ ., data = DM_train) %>%  
  step_dummy(all_nominal(), -all_outcomes()) %>%  
  step_zv(all_predictors()) %>%  
  step_normalize(all_predictors())  
summary(gen_recipe)
```

```
## # A tibble: 9 x 4  
##   variable type      role      source  
##   <chr>      <chr>   <chr>   <chr>  
## 1 pregnant numeric predictor original  
## 2 glucose  numeric predictor original  
## 3 pressure numeric predictor original
```

建立資料前處理“食譜”

還有很多其他的前處理方法：

- `step_naomit(everything(), skip = TRUE)` 如果沒有在一開始將NA資料刪掉，通常要去除NA值
- `step_rose(diabetes)` 設定oversampleing或是undersampling，範例用ROSE作為oversampling的演算法
- 參考`recipe()`函數的說明文件

```
gen_recipe <-  
  recipe(diabetes ~ ., data = DM_train) %>%  
  step_dummy(all_nominal(), -all_outcomes()) %>%  
  step_zv(all_predictors()) %>%  
  step_normalize(all_predictors())  
summary(gen_recipe)
```

```
## # A tibble: 9 x 4  
##   variable type      role      source  
##   <chr>      <chr>   <chr>    <chr>  
## 1 pregnant numeric predictor original  
## 2 glucose  numeric predictor original  
## 3 pressure numeric predictor original
```

邏輯迴歸Logistic Regression模型建立與效能評估範例

Step 1 設定用邏輯迴建立模型

- 以邏輯迴歸為例
- 用`logistic_reg()`函數指定使用邏輯迴歸
- 用`set_engine("glm")`設定模型建立演算法

```
lr_mod <-  
  logistic_reg() %>%  
  set_engine("glm")
```

Step 2 設定建模流程workflow

將建模 (model, 從step 1來的)與資料前處理方法 (recipe, 從前處理來的)串成單一
工作流程workflow

```
lr_wflow <-  
  workflow() %>%  
  add_model(lr_mod) %>%  
  add_recipe(gen_recipe)  
lr_wflow
```

```
## == Workflow =====  
## Preprocessor: Recipe  
## Model: logistic_reg()  
##  
## — Preprocessor —————  
## 3 Recipe Steps  
##  
## ● step_dummy()  
## ● step_zv()  
## ● step_normalize()  
##  
## — Model —————
```

Step 3 訓練模型

- 使用剛剛串起來的工作流程，加上`fit()`函數，完成建模

```
lr_fit <-  
  lr_wflow %>%  
  fit(data=DM_train)
```

Step 3 訓練模型

- 使用剛剛串起來的工作流程，加上`fit()`函數，完成建模

```
lr_fit <-  
  lr_wflow %>%  
  fit(data=DM_train)
```

- 用`pull_workflow_fit()`與`tidy()`呈現建模結果，注意這裡也是只能用訓練組資料DM_train

```
lr_fit %>%  
  pull_workflow_fit() %>%  
  tidy()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-0.9640371	0.1597350	-6.0352290	0.0000000
pregnant	0.3490945	0.1920579	1.8176525	0.0691173
glucose	1.0494165	0.1948386	5.3860817	0.0000001
pressure	0.0038956	0.1589881	0.0245027	0.9804516

Step 4 使用測試組資料驗證效能

- 剛剛訓練出來的模型lr_fit丟入predict()函數，指定以一開始分出的測試組DM_test產生預測結果
- 注意這邊只能用測試組資料DM_test

```
lr_pred <- lr_fit %>%  
  predict(DM_test)
```

```
lr_pred
```

.pred_class
pos
neg
pos
neg
neg
pos

Step 4 使用測試組資料驗證效能

- 通常會直接將predict()函數預測結果與真實答案結合
- 答案為測試組資料DM_test中的diabetes欄位
- 將預測結果丟入bind_cols(), 並指定與答案直接結合

```
lr_pred <- lr_fit %>%  
  predict(DM_test) %>%  
  bind_cols(DM_test %>% select(diabetes))
```

lr_pred

.pred_class	diabetes
pos	pos
neg	neg
pos	pos
neg	neg
neg	neg

Step 4 使用測試組資料驗證效能

使用`accuracy()`函數，輸入剛剛整合的預測結果與真實答案計算正確率

```
lr_pred %>%  
  accuracy(truth = diabetes,  
           .pred_class)
```

```
## # A tibble: 1 x 3  
##   .metric .estimator .estimate  
##   <chr>    <chr>         <dbl>  
## 1 accuracy binary         0.825
```

Step 4 使用測試組資料驗證效能

- 很多時候我們需要回報**Area under the ROC curve**
- 此時不能直接採用模型預測pos或neg的結果
- 需要的連續性的預測數值來畫**ROC curve**
- 將predict()函數的type參數設定為prob，即回傳各組預測值

```
lr_pred <- lr_fit %>%  
  predict(DM_test,  
    type = "prob") %>%  
  bind_cols(DM_test %>% select(diabetes))
```

```
lr_pred
```

.pred_neg	.pred_pos	diabetes
0.1785513	0.8214487	pos
0.9668423	0.0331577	neg
0.0591382	0.9408618	pos
0.9421452	0.0578548	neg

Step 4 使用測試組資料驗證效能

將預測結果lr_pred丟入roc_curve(), 加上autoplot()畫ROC curve

```
lr_pred %>%  
  roc_curve(truth = diabetes,  
            .pred_pos) %>%  
  autoplot()
```

Step 4 使用測試組資料驗證效能

當然也能用`roc_auc()`算Area under the ROC curve

```
lr_pred %>%  
  roc_auc(truth = diabetes,  
          .pred_pos)
```

.metric	.estimator	.estimate
roc_auc	binary	0.9072115

Step 5 解釋模型

- 以迴歸為例，參數即可解釋各變數與模型的關係
- 用pull_workflow_fit()與tidy()呈現建模結果

```
lr_fit %>%  
  pull_workflow_fit() %>%  
  tidy()
```

term	estimate	std.error	statistic	p.value
(Intercept)	-0.9640371	0.1597350	-6.0352290	0.0000000
pregnant	0.3490945	0.1920579	1.8176525	0.0691173
glucose	1.0494165	0.1948386	5.3860817	0.0000001
pressure	0.0038956	0.1589881	0.0245027	0.9804516
triceps	0.1655423	0.2039447	0.8117017	0.4169628
insulin	-0.1612772	0.1746670	-0.9233411	0.3558295
mass	0.5546995	0.2187121	2.5362087	0.0112060
pedigree	0.2894106	0.1640091	1.7646014	0.0776308

完成

以上就是使用邏輯迴歸建立模型與效能測試流程，可以發現完全沒有調整任何參數，因基本邏輯迴歸不用調參數。

隨機森林Random Forest模型建立、 參數調整與效能評估範例

Step 1 設定平行處理

因為模型參數調整需要一直不斷建立模型與測試，所以設定平行處理會快一些，`tidymodels`套組支援平行處理，細節可參考[官方文件](#)

```
library(parallel)
library(doParallel)
all_cores <- detectCores(logical = FALSE)
cl <- makePSOCKcluster(all_cores)
registerDoParallel(cl)
```

Step 2 設定用隨機森林建立模型以及要調整的參數

- 用`rand_forest()`函數指定要建立隨機森林模型
- 用`set_engine("ranger")`設定模型建立演算法為基於ranger套件的隨機森林演算法，沒裝過ranger套件的話需要先裝
 - `permutation`: effect on the prediction performance (Mean decrease accuracy)
 - `impurity`: Gini index for classification
- 因為隨機森林有迴歸版與分類版，因此使用`set_mode("classification")`設定我們要用分類演算法

```
install.packages("ranger")
```

```
rf_mod <-  
  rand_forest(mtry = tune(), min_n = tune(),  
              trees = 1000) %>%  
  set_engine("ranger",  
             importance= "permutation") %>%  
  set_mode("classification")
```

Step 2 設定用隨機森林建立模型以及要調整的參數

在隨機森林`rand_forest()`函數中，可設定幾個參數，說明如下：

- `mtry`: 在切割節點時，隨機抽取`n`個特徵，並從中選最適合的特徵當作節點
- `min_n`: 每個節點的最小資料數，如果設為10，當該節點的資料剩十筆或更少時，就不會再切割
- `trees`: 建模要用幾棵樹 在這個範例中，我將要建幾棵樹設定為1000，其他兩個參數則是想用資料調整，因此將想調的參數值設為`tune()`，表示這些參數要調，不想在現階段指定。

```
rf_mod <-  
  rand_forest(mtry = tune(), min_n = tune(),  
              trees = 1000) %>%  
  set_engine("ranger",  
             importance= "permutation") %>%  
  set_mode("classification")
```


Step 2 設定用隨機森林建立模型以及要調整的參數

模型設定完成後，輸出如下

```
rf_mod
```

```
## Random Forest Model Specification (classification)
##
## Main Arguments:
##   mtry = tune()
##   trees = 1000
##   min_n = tune()
##
## Engine-Specific Arguments:
##   importance = permutation
##
## Computational engine: ranger
```

Step 3 設定建模流程workflow

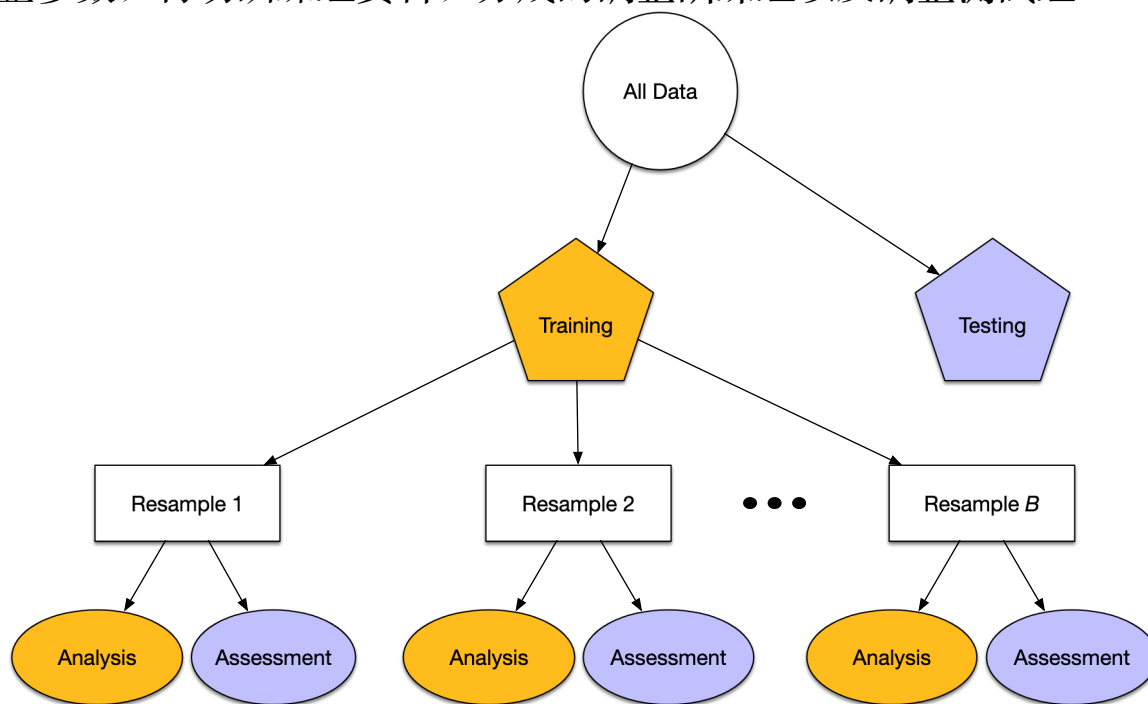
建模流程同邏輯迴歸，將模型 (model，從step 2來的)與資料前處理方法(recipe，從前處理來的)串接成一個工作流程

```
rf_wflow <- workflow() %>%  
  add_model(rf_mod) %>%  
  add_recipe(gen_recipe)  
rf_wflow
```

```
## == Workflow ==  
## Preprocessor: Recipe  
## Model: rand_forest()  
##  
## — Preprocessor —  
## 3 Recipe Steps  
##  
## ● step_dummy()  
## ● step_zv()  
## ● step_normalize()  
##  
## — Model —  
## Random Forest Model Specification (classification)
```

Step 4 參數調整組資料分割

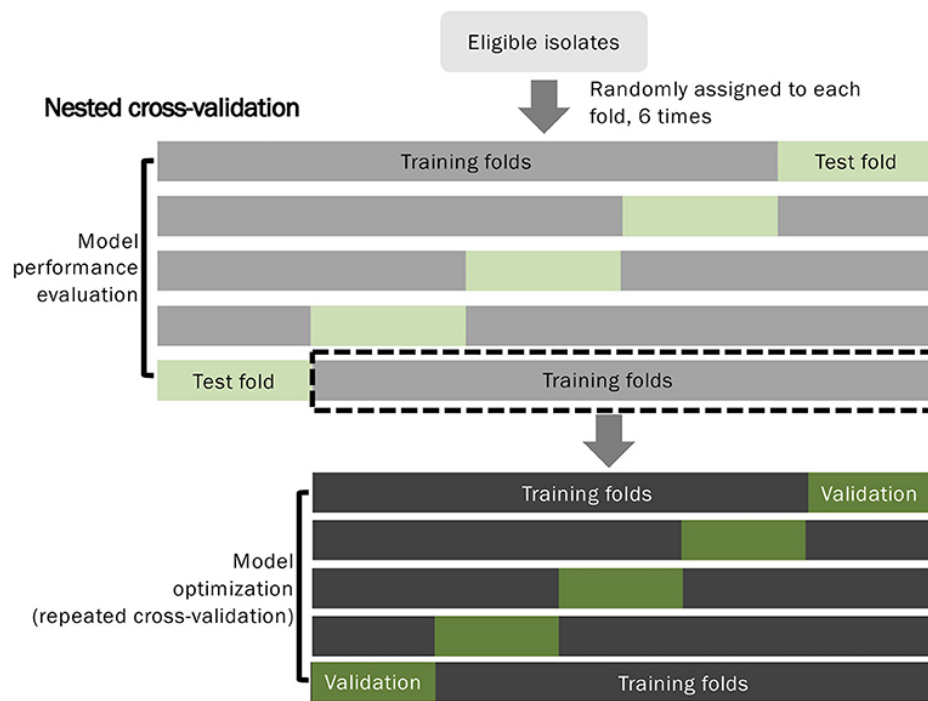
- 剛剛有提到要以資料調整的參數`mtry`以及`min_n`
- 首先將所有資料分成測試組訓練組，也就是本篇文章一開始做的切割
- 為了調整參數，再切訓練組資料，分成的調整訓練組以及調整測試組。



圖片來源

Step 4 參數調整組資料分割

- 切割參數調整組有很多種方法，可以用**bootstrap**法隨機抽，也可使用這邊的交叉驗證(Cross Validation)範例
- 圖片下半部為5-fold CV 的示意圖
- 每份資料都會被拿來當作調整訓練以及調整測試組
- 經過幾次測試後，用調整測試組的效能來決定一組最佳參數



Step 4 參數調整組資料分割

- 使用10-fold CV 為例，先用vfold_cv()函數
 - 設定分割基準為訓練組DM_train
 - 要分10份v=10
 - 分割時要注意糖尿病的比例不能差太多，因此設為分組依據 strata = diabetes

```
set.seed(345)
folds <- vfold_cv(DM_train,
                  v = 10,
                  strata = diabetes)
folds
```

```
## # 10-fold cross-validation using stratification
## # A tibble: 10 x 2
##   splits          id
##   <named list>    <chr>
## 1 <split [265/30]> Fold01
## 2 <split [265/30]> Fold02
## 3 <split [265/30]> Fold03
## 4 <split [265/30]> Fold04
## 5 <split [265/30]> Fold05
```

Step 5 調整參數

- 將Step 3所建立的建模流程rf_wflow串接至tune_grid()函數，設定參數調整的方法
 - 設定切割好的參數調整組 resamples = folds
 - 要測試幾組參數 grid = 50
 - 測試時要用什麼效能評估方式，設定為Area under the ROC curve
metrics = metric_set(roc_auc)
- 因為要重複訓練測試多次，因此這程式碼執行會需要一些時間。

```
rf_res <-  
  rf_wflow %>%  
  tune_grid(  
    resamples = folds,  
    grid = 50,  
    metrics = metric_set(roc_auc),  
    control=control_resamples(save_pred = TRUE)  
  )
```

Step 5 調整參數

執行完後，可用`collect_metrics()`查看各參數效能

```
rf_res %>%  
  collect_metrics()
```

mtry	min_n	.metric	.estimator	mean	n	std_err
1	9	roc_auc	binary	0.8219883	10	0.0271489
1	25	roc_auc	binary	0.8248187	10	0.0280118
1	30	roc_auc	binary	0.8283772	10	0.0283969
2	5	roc_auc	binary	0.8191579	10	0.0275482
2	13	roc_auc	binary	0.8194942	10	0.0286831
2	25	roc_auc	binary	0.8233567	10	0.0288690

Step 5 調整參數

也可畫個圖呈現參數調整對效能的影響，由圖可知在這個範例中min_n越大效能越好

```
rf_res %>%  
  collect_metrics() %>%  
  mutate(mtry=factor(mtry)) %>%  
  ggplot(aes(min_n, mean, color = mtry)) +  
  geom_line(size=1)
```


Step 5 調整參數

搭配show_best()函數可呈現Area under the ROC curve最優的幾組參數

```
rf_res %>%  
  show_best("roc_auc")
```

mtry	min_n	.metric	.estimator	mean	n	std_err
2	33	roc_auc	binary	0.8298567	10	0.0289917
1	30	roc_auc	binary	0.8283772	10	0.0283969
2	31	roc_auc	binary	0.8281550	10	0.0280554
2	29	roc_auc	binary	0.8276871	10	0.0290739
2	39	roc_auc	binary	0.8271404	10	0.0298687

Step 6 使用最佳參數與訓練組資料建立最終模型

- 使用最佳參數(意即Area under the ROC curve最高的一組參數)來建立最終模型
- 將剛剛調參數的結果rf_res輸入select_best()函數，選出最好的一組參數best_rf

```
best_rf <- rf_res %>%  
  select_best("roc_auc")
```

```
best_rf
```

mtry	min_n
2	33

Step 6 使用最佳參數與訓練組資料建立最終模型

將Step 3 建立的工作流程輸入`finalize_workflow()`函數，並指定要用剛剛選出的最佳參數`best_rf`，建立一個最終建模流程

```
final_wflow <-  
  rf_wflow %>%  
  finalize_workflow(best_rf)
```

```
final_wflow
```

```
## == Workflow ==  
## Preprocessor: Recipe  
## Model: rand_forest()  
##  
## — Preprocessor —  
## 3 Recipe Steps  
##  
## ● step_dummy()  
## ● step_zv()  
## ● step_normalize()
```

Step 6 使用最佳參數與訓練組資料建立最終模型

最終建模流程 `final_wflow` 建立後，即可用 `fit()` 建模，注意這邊用的是完整的訓練資料 `DM_train`

```
final_rf_model <-  
  final_wflow %>%  
  fit(data = DM_train)
```

```
final_rf_model
```

```
## == Workflow [trained] =====
```

```
## Preprocessor: Recipe
```

```
## Model: rand_forest()
```

```
##
```

```
## — Preprocessor —————
```

```
## 3 Recipe Steps
```

```
##
```

```
## ● step_dummy()
```

```
## ● step_zv()
```

```
## ● step_normalize()
```

Step 7 用測試組資料驗證最終效能

- 將剛剛訓練出來的模型`final_rf_model`輸入`predict()`函數
- 並指定使用一開始分出的測試組`DM_test`產生預測結果
- 注意要用測試組資料

```
rf_pred <- final_rf_model %>%  
  predict(DM_test)
```

```
rf_pred
```

.pred_class

pos

neg

pos

neg

neg

pos

Step 7 用測試組資料驗證最終效能

- 通常會直接將predict()函數預測結果與真實答案結合
- 答案為測試組資料DM_test中的diabetes欄位
- 將預測結果丟入bind_cols(), 並指定與答案直接結合

```
rf_pred <- final_rf_model %>%  
  predict(DM_test) %>%  
  bind_cols(DM_test %>% select(diabetes))
```

rf_pred

.pred_class	diabetes
pos	pos
neg	neg
pos	pos
neg	neg
neg	neg

Step 7 用測試組資料驗證最終效能

使用預測結果與真實答案計算正確率

```
rf_pred %>%  
  accuracy(truth = diabetes,  
           .pred_class)
```

.metric	.estimator	.estimate
accuracy	binary	0.8247423

Step 7 用測試組資料驗證最終效能

- 很多時候我們需要回報**Area under the ROC curve**
- 此時不能直接採用模型預測pos或neg的結果
- 需要的連續性的預測數值來畫**ROC curve**
- 將predict()函數的type參數設定為prob，即回傳各組預測值

```
rf_pred <- final_rf_model %>%  
  predict(DM_test,  
    type = "prob") %>%  
  bind_cols(DM_test %>% select(diabetes))
```

```
rf_pred
```

.pred_neg	.pred_pos	diabetes
0.3973083	0.6026917	pos
0.9651058	0.0348942	neg
0.1754237	0.8245763	pos
0.9276385	0.0723615	neg

Step 7 用測試組資料驗證最終效能

將預測結果lr_pred丟入roc_curve(), 加上autoplot()畫ROC curve

```
rf_pred %>%  
  roc_curve(truth = diabetes,  
            .pred_pos) %>%  
  autoplot()
```

Step 7 用測試組資料驗證最終效能

當然也能用`roc_auc()`算Area under the ROC curve

```
rf_pred %>%  
  roc_auc(truth = diabetes,  
          .pred_pos)
```

.metric	.estimator	.estimate
roc_auc	binary	0.8947115

Step 8 解釋模型

- 查看最終模型final_rf_model中，各變數的重要性
- 用pull_workflow_fit()將模型資訊取出
- 丟入vip()查看各變數重要性

```
library(vip)
final_rf_model %>%
  pull_workflow_fit() %>%
  vi()
```

```
## # A tibble: 8 x 2
##   Variable Importance
##   <chr>           <dbl>
## 1 glucose    0.0424
## 2 age        0.0227
## 3 mass       0.0113
## 4 insulin    0.0110
## 5 pregnant   0.00713
## 6 triceps    0.00545
## 7 pedigree   0.00376
## 8 pressure   0.00000833
```

Step 8 解釋模型

- 查看最終模型`final_rf_model`中，各變數的重要性
- 用`pull_workflow_fit()`將模型資訊取出
- 丟入`vip()`查看各變數重要性

```
library(vip)
final_rf_model %>%
  pull_workflow_fit() %>%
  vip()
```

完成

以上就是使用隨機森林建立模型、調整參數以及與效能測試流程，多了使用交叉驗證法調整參數的步驟。

正紅的XGBoost模型建立、參數調整與效能評估範例

XGBoost不分步驟，一次搞定 (1/3)

```
#install.packages("xgboost")
xgb_mod <-
  boost_tree(mtry = tune(),
             min_n = tune(),
             trees = 1000) %>%
  set_engine("xgboost",
             importance= "permutation") %>%
  set_mode("classification")
xgb_wflow <- workflow() %>%
  add_model(xgb_mod) %>%
  add_recipe(gen_recipe)
xgb_res <-
  xgb_wflow %>%
  tune_grid(
    resamples = folds,
    grid = 50,
    metrics = metric_set(roc_auc),
    control=control_resamples(save_pred = TRUE)
  )
```

i Creating pre-processing data to finalize unknown parameter: mtry

XGBoost不分步驟，一次搞定 (2/3)

```
best_xgb <- xgb_res %>%  
  select_best("roc_auc")  
final_wflow <-  
  xgb_wflow %>%  
  finalize_workflow(best_xgb)  
final_xgb_model <-  
  final_wflow %>%  
  fit(data = DM_train)  
xgb_pred <- final_xgb_model %>%  
  predict(DM_test,  
    type = "prob")%>%  
  bind_cols(DM_test %>% select(diabetes))
```


XGBoost不分步驟，一次搞定 (3/3)

```
xgb_pred %>%  
  roc_curve(truth = diabetes,  
            .pred_pos) %>%  
  autoplot()
```

上課教的SVM模型建立、參數調整與效能評估範例

SVM不分步驟，一次搞定 (1/3)

```
#install.packages("kernlab")
svm_mod <-
  svm_rbf(cost=tune(),
          rbf_sigma=tune(),
          margin=tune()) %>%
  set_engine("kernlab",
             importance= "permutation") %>%
  set_mode("classification")
svm_wflow <- workflow() %>%
  add_model(svm_mod) %>%
  add_recipe(gen_recipe)
svm_res <-
  svm_wflow %>%
  tune_grid(
    resamples = folds,
    grid = 50,
    metrics = metric_set(roc_auc),
    control=control_resamples(save_pred = TRUE)
  )
```

SVM不分步驟，一次搞定 (2/3)

```
best_svm <- svm_res %>%  
  select_best("roc_auc")  
final_wflow <-  
  svm_wflow %>%  
  finalize_workflow(best_svm)  
final_svm_model <-  
  final_wflow %>%  
  fit(data = DM_train)
```

```
## maximum number of iterations reached 9.479447e-05 9.47946e-05
```

```
svm_pred <- final_svm_model %>%  
  predict(DM_test,  
    type = "prob") %>%  
  bind_cols(DM_test %>% select(diabetes))
```

SVM不分步驟，一次搞定 (3/3)

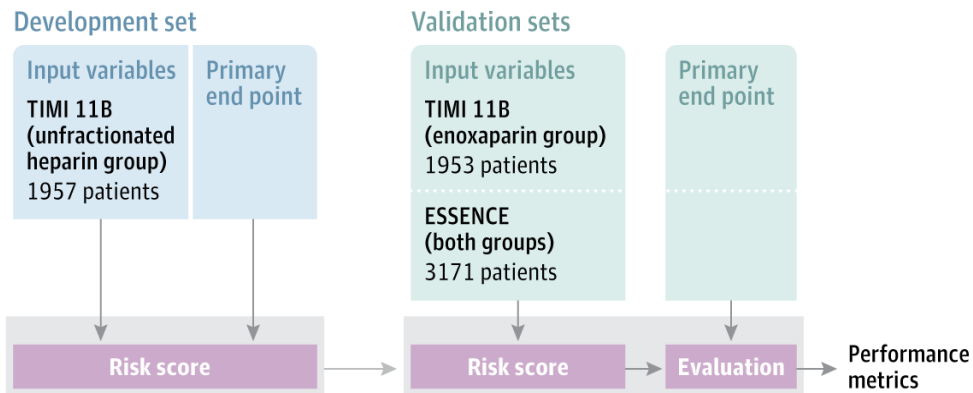
```
svm_pred %>%  
  roc_curve(truth = diabetes,  
            .pred_pos) %>%  
  autoplot()
```

tidymodels總結

- 官方說明文件完整
- 建模、調參數、預測流程架構清楚
- 目前支援21大類模型，可透過模型清單查找，自行替換模型
- 不知道每個模型有何參數可調？一樣可透過參數清單查找
- 不知道參數意義？了解各模型的原理還是很重要的！

訓練/調整與驗證模型效能策略

A Decision-rule model



B Machine learning model

Development sets

