

補  
第四單元

集成

# 混合泛化(Blending)



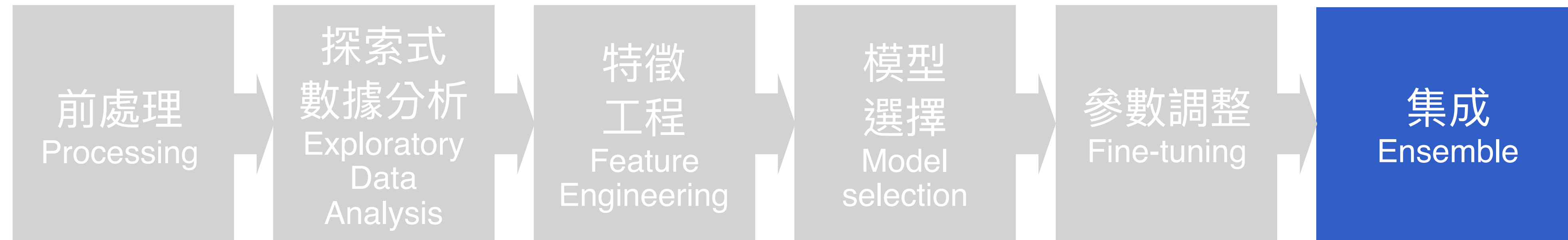
出題教練

陳明佑

# 知識地圖 機器學習- 參數調整 - 超參數調整與優化

## 機器學習概論 Introduction of Machine Learning

### 監督式學習 Supervised Learning



### 非監督式學習 Unsupervised Learning



### 參數調整 Fine-tuning

混合泛化  
Blending

堆疊泛化  
Stacking



# 本日知識點目標

- 資料工程中的集成，有哪些常見的內容？
- 混合泛化為什麼能提升預測力，使用上要注意什麼問題？

# 什麼是集成

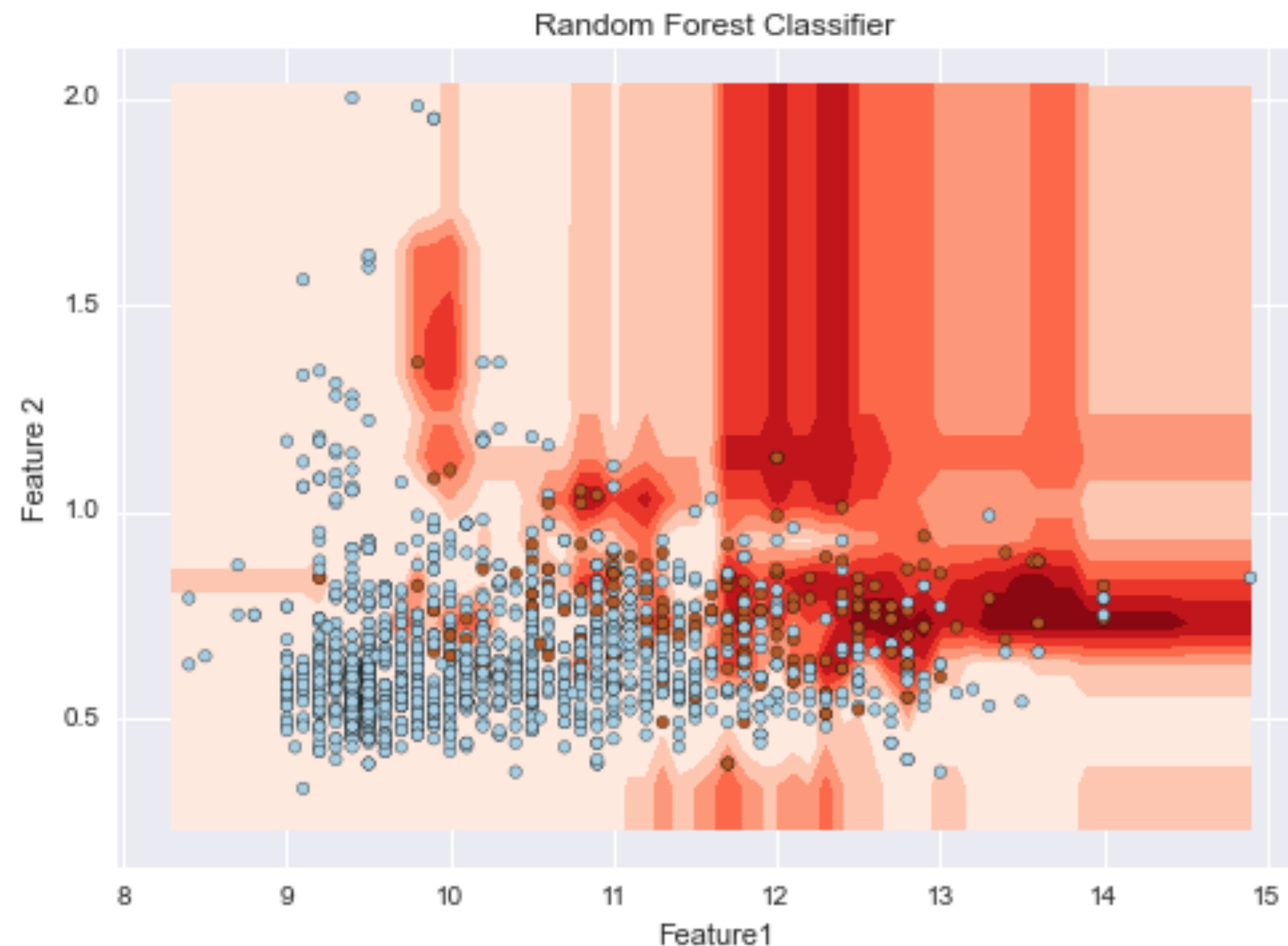
- 集成是使用不同方式，結合多個/多種不同分類器，作為綜合預測的做法統稱
- 將模型截長補短，也可說是機器學習裡的 和議制 / 多數決



- 其中又分為資料面的集成：如裝袋法(Bagging) / 提升法(Boosting)
- 以及模型與特徵的集成：如混合泛化(Blending) / 堆疊泛化(Stacking)

# 資料面集成：裝袋法 ( Bagging )

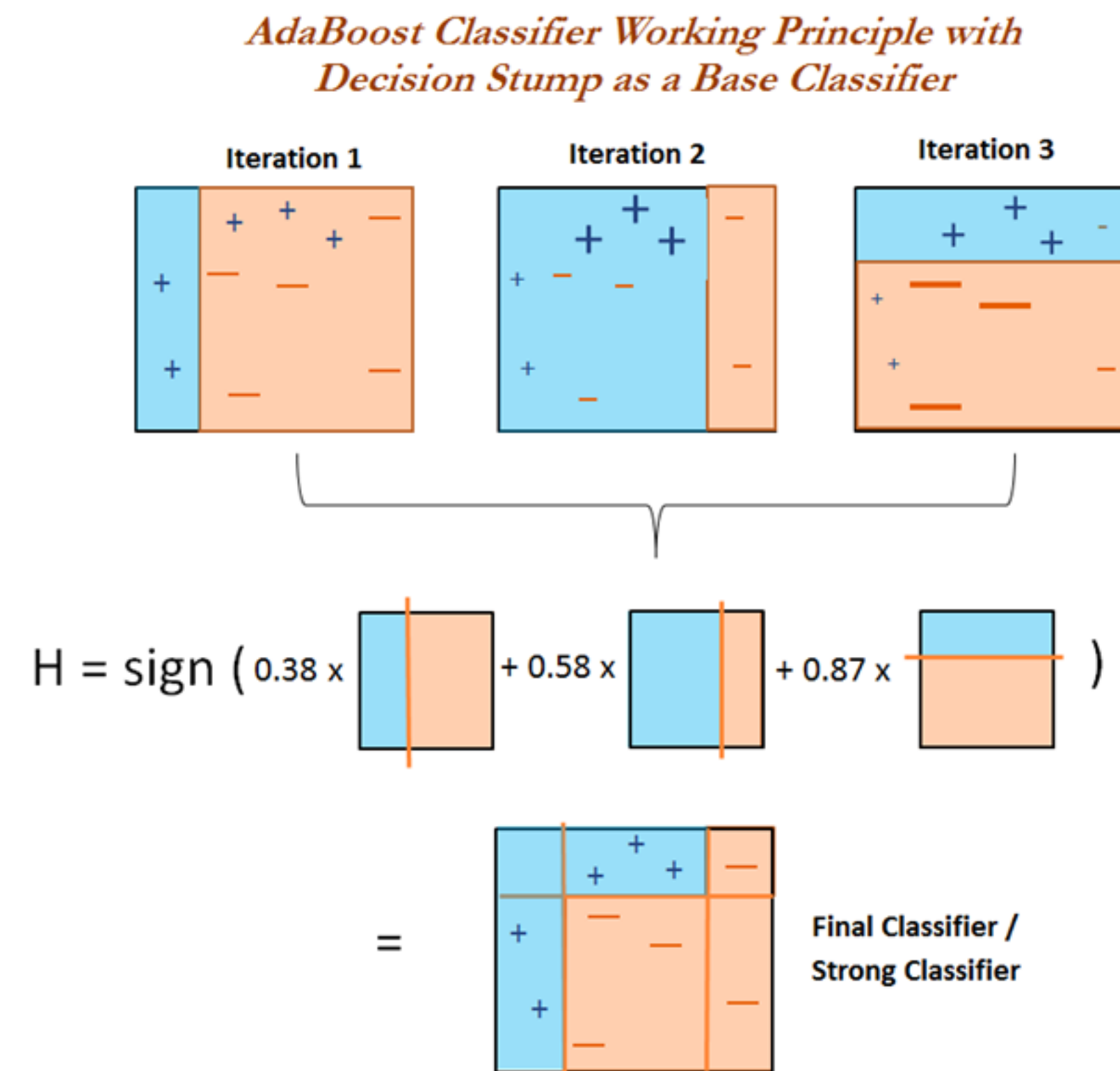
- 裝袋法顧名思義，是將資料放入袋中抽取，每回合結束後全部放回袋中重抽
- 再搭配弱分類器取平均/多數決結果，最有名的就是前面學過的**隨機森林**



圖片來源：stackexchange.

# 資料面集成：提升法 ( Boosting )

- 提升法則是由之前模型的預測結果，去改變資料被抽到的權重或目標值
- 將錯判資料被抽中的機率放大，正確的縮小，就是**自適應提升** (AdaBoost, Adaptive Boosting)
- 如果是依照估計誤差的殘差項調整新目標值，則就是**梯度提升機** (Gradient Boosting Machine) 的作法，只是梯度提升機還加上用梯度來選擇決策樹分支





# 資料集成 v.s. 模型與特徵集成

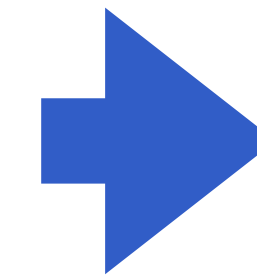
- 兩者雖然都稱為集成，其實適用範圍差異很大，通常不會一起提及
- 這裡為了避免同學混淆，在這邊將兩者做個對比

- 資料集成

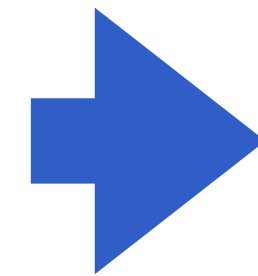
## Bagging / Boosting

- 使用不同訓練資料 + 同一種模型，多次估計的結果合成最終預測

不同資料



相同模型



合成結果

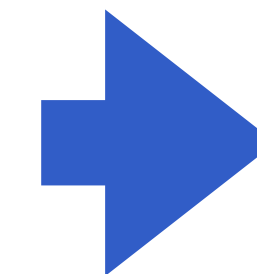


- 模型與特徵集成

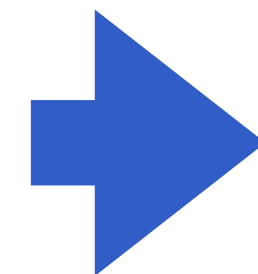
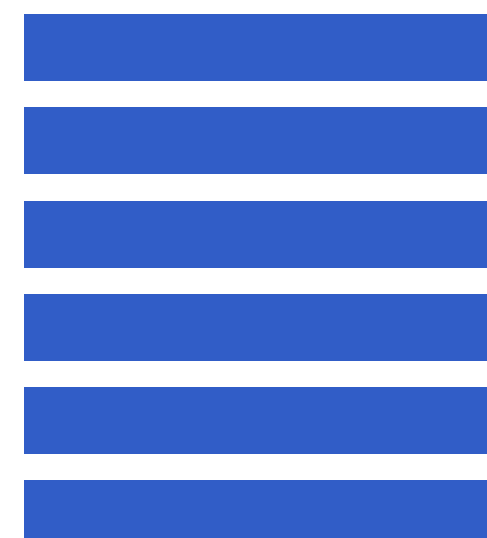
## Voting / Blending / Stacking

- 使用同一資料 + 不同模型，合成出不同預測結果

相同資料



不同模型

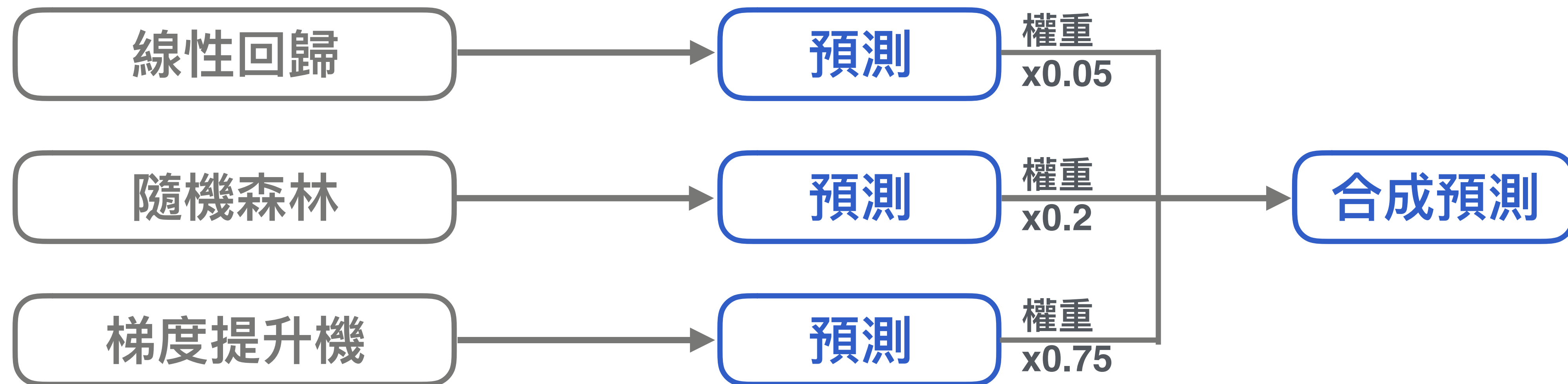


合成結果



# 混合泛化 ( Blending ) ( 1 / 3 )

- 其實混合泛化非常單純，就是將不同模型的預測值加權合成，權重和為 1  
如果取預測的平均 or 一人一票多數決(每個模型權重相同)，則又稱為 **投票泛化(Voting)**



- 雖然單純，但因為最容易使用且有效，至今仍然是競賽中常見的作法



# 混合泛化 ( Blending ) ( 2 / 3 )

---

## 容易使用

- 不只在一般機器學習中 useful，影像處理或自然語言處理等深度學習，也一樣可以使用
- 因為只要有預測值(Submit 檔案)就可以使用，許多跨國隊伍就是靠這個方式合作
- 另一方面也因為只要用預測值就能計算，在競賽中可以快速合成多種比例的答案，妥善消耗掉每一天剩餘的 Submit 次數

# 混合泛化 ( Blending ) ( 3 / 3 )

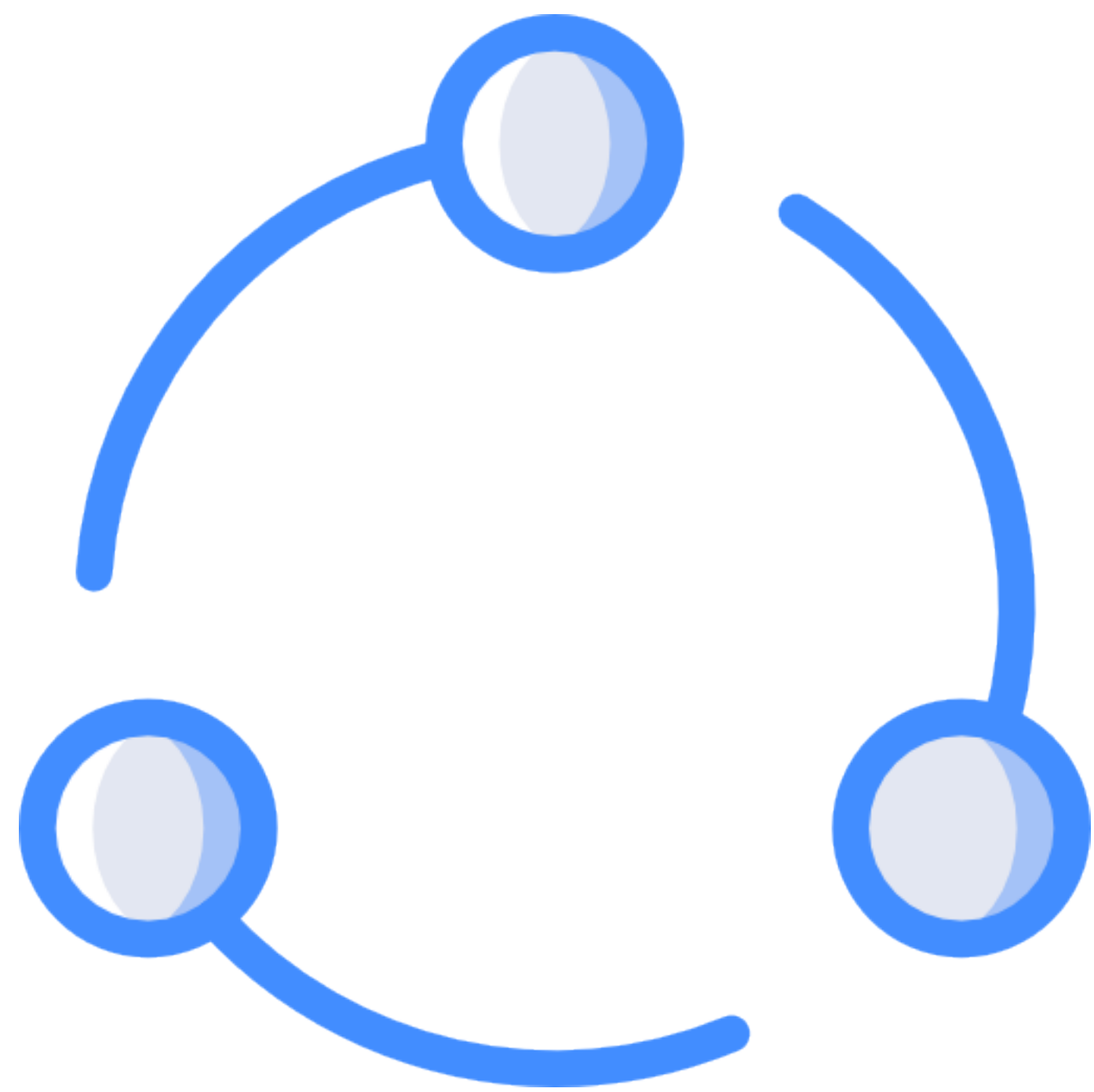
---

## 效果顯著

- Kaggle 競賽截止日前的 Kernel，有許多只是對其他人的輸出結果做 Blending，但是因為分數較高，因此也有許多人樂於推薦與發表
- 在2015年前的大賽中，Blending 仍是主流，例如林軒田老師也曾在課程中提及：有競賽的送出結果，是上百個模型的 Blending

## 注意事項

- Blending 的前提是：個別**單模效果都很好**(有調參)並且**模型差異大**，單模要好尤其重要，如果單模效果差異太大，Blending 的效果提升就相當有限



- 資料工程中的集成，包含了資料面的集成 - **裝袋法(Bagging)** / **提升法(Boosting)**，以及模型與特徵的集成 - **混合泛化(Blending)** / **堆疊泛化(Stacking)**
- 混合泛化提升預測力的原因是基於**模型差異度大**，在預測細節上能互補，因此預測模型只要各自調參優化過且原理不同，通常都能使用混合泛化集成



# 解題時間

## It's Your Turn

請跳出PDF至官網Sample Code & 作業  
開始解題



補  
第四單元

集成

# 堆疊泛化(Stacking)



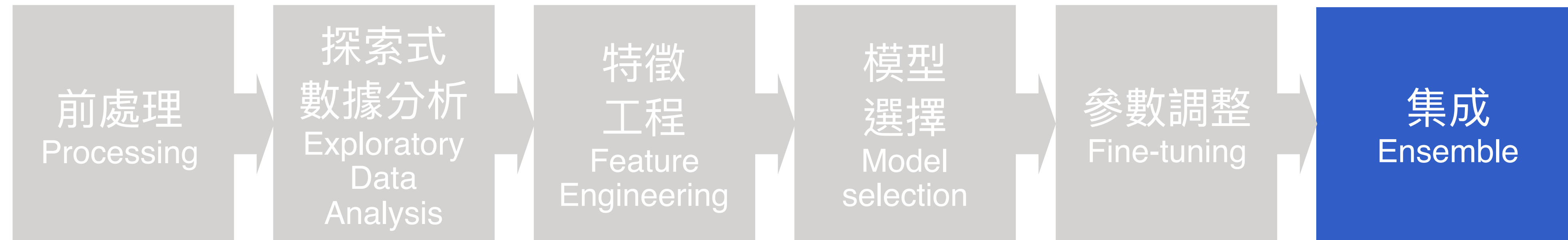
出題教練

陳明佑

# 知識地圖 機器學習- 參數調整 - 超參數調整與優化

## 機器學習概論 Introduction of Machine Learning

### 監督式學習 Supervised Learning



### 非監督式學習 Unsupervised Learning



### 參數調整 Fine-tuning

混合泛化  
Blending

堆疊泛化  
Stacking



# 本日知識點目標

- 為什麼堆疊泛化看起來這麼複雜？
- 堆疊泛化有堆疊層數上的限制嗎？
- 混合泛化相對堆疊泛化來說，有什麼優缺點？

# 堆疊泛化 (Stacking) 的橫空出世

## Stacking 小歷史

- 雖然堆疊泛化 (Stacking) 的論文早在 2012 年，就由 David H. Wolpert 發布 ([原始論文連結](#))
- 但真正被廣泛應用於競賽上，是2014年底的 Kaggle 競賽開始
- 由於 Kaggle 一直有前幾名於賽後發布做法的風氣，所以當有越來越多的前幾名使用 Stacking
- 後，這個技術就漸漸變得普及起來，甚至後來出現了加速混合與計算速度的 StackNet

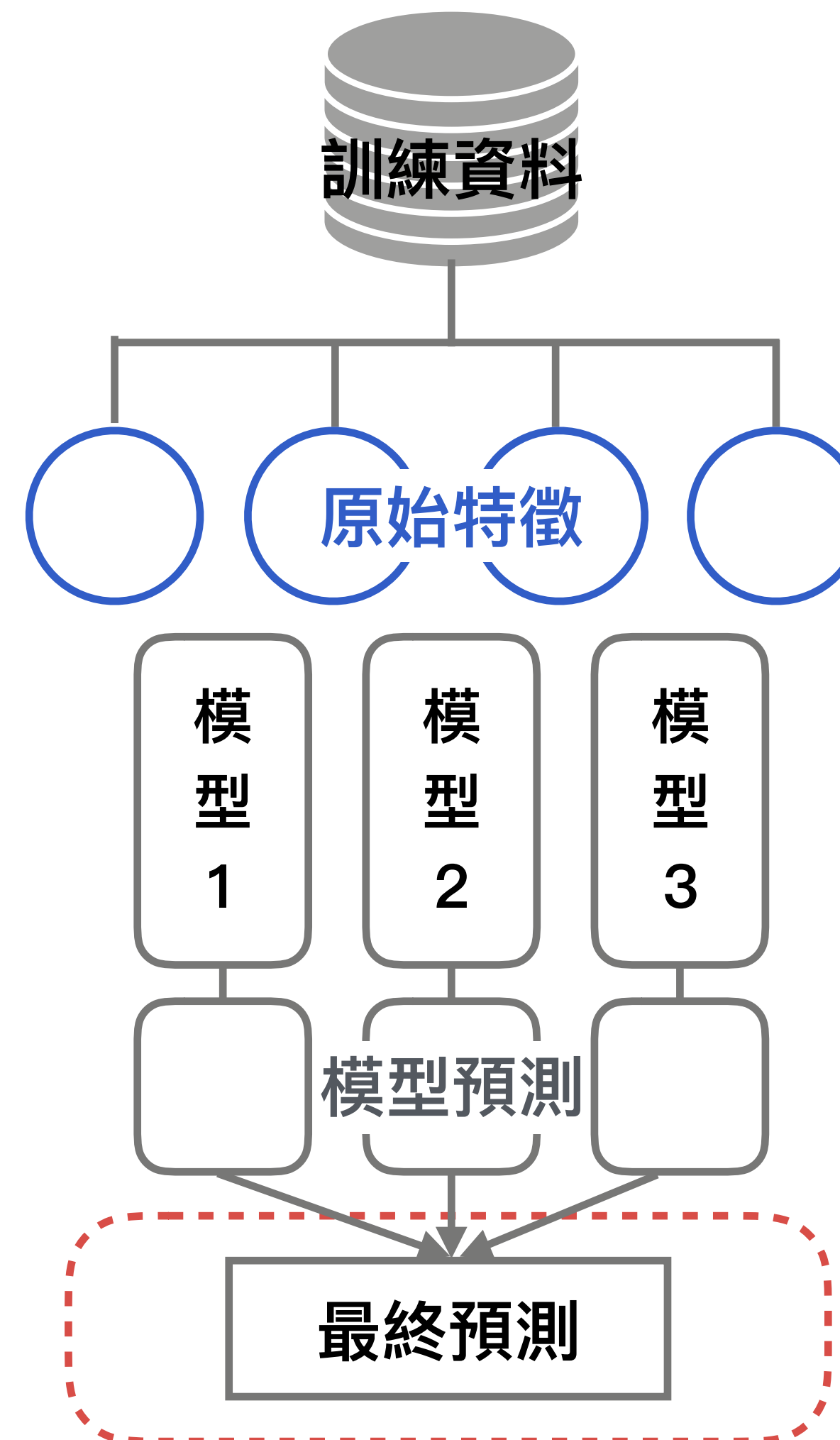
## 相對於 Blending 的改良

- 不只將預測結果混合，而是使用**預測結果當新特徵**
- 更進一步的運用了**資料輔助集成**，但也使得 Stacking 複雜許多

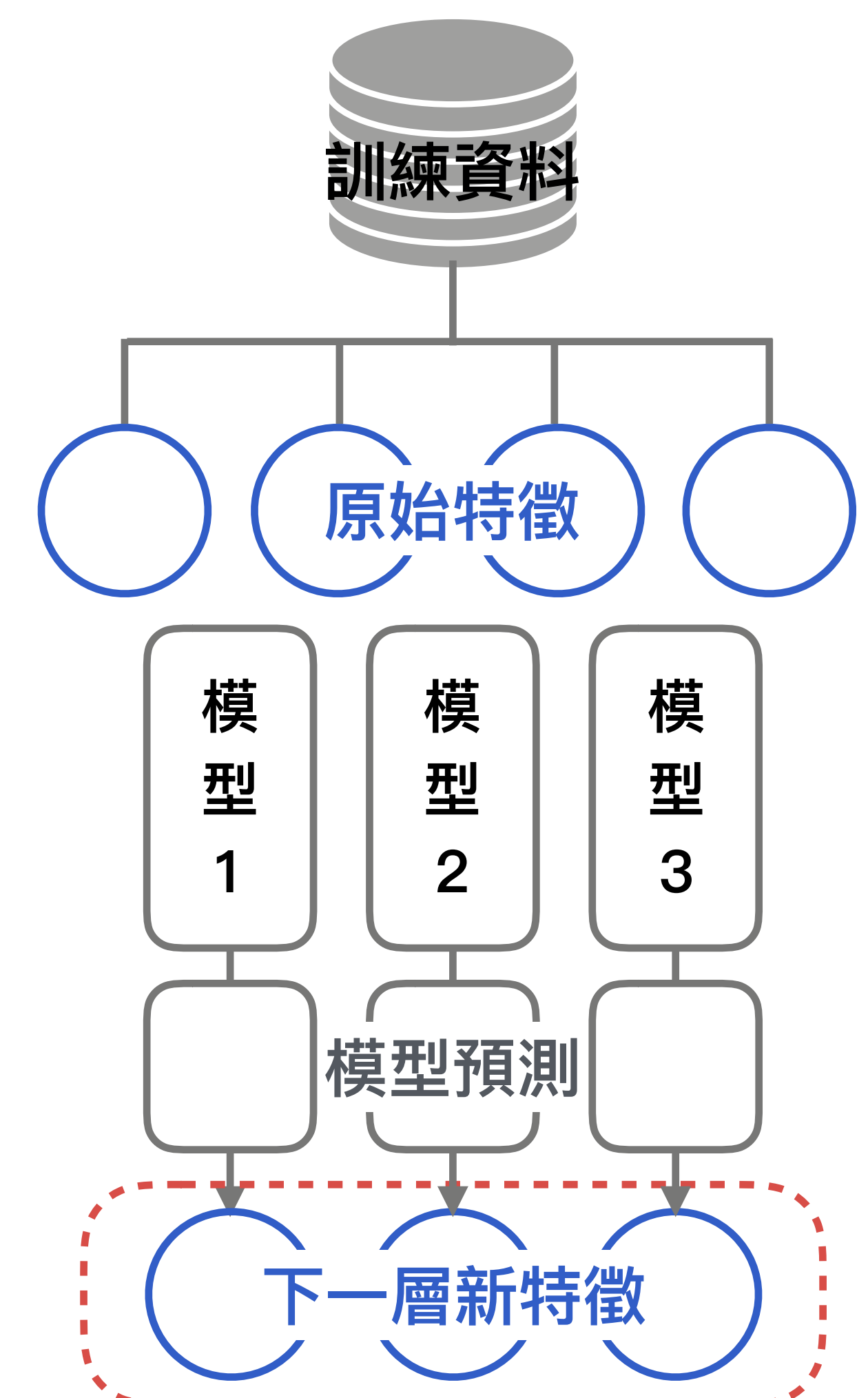
# Stacking 的設計挑戰：訓練測試的不可重複性

Blending 與 Stacking 都是模型集成，但是模型預測結果怎麼使用，是關鍵差異

## Blending



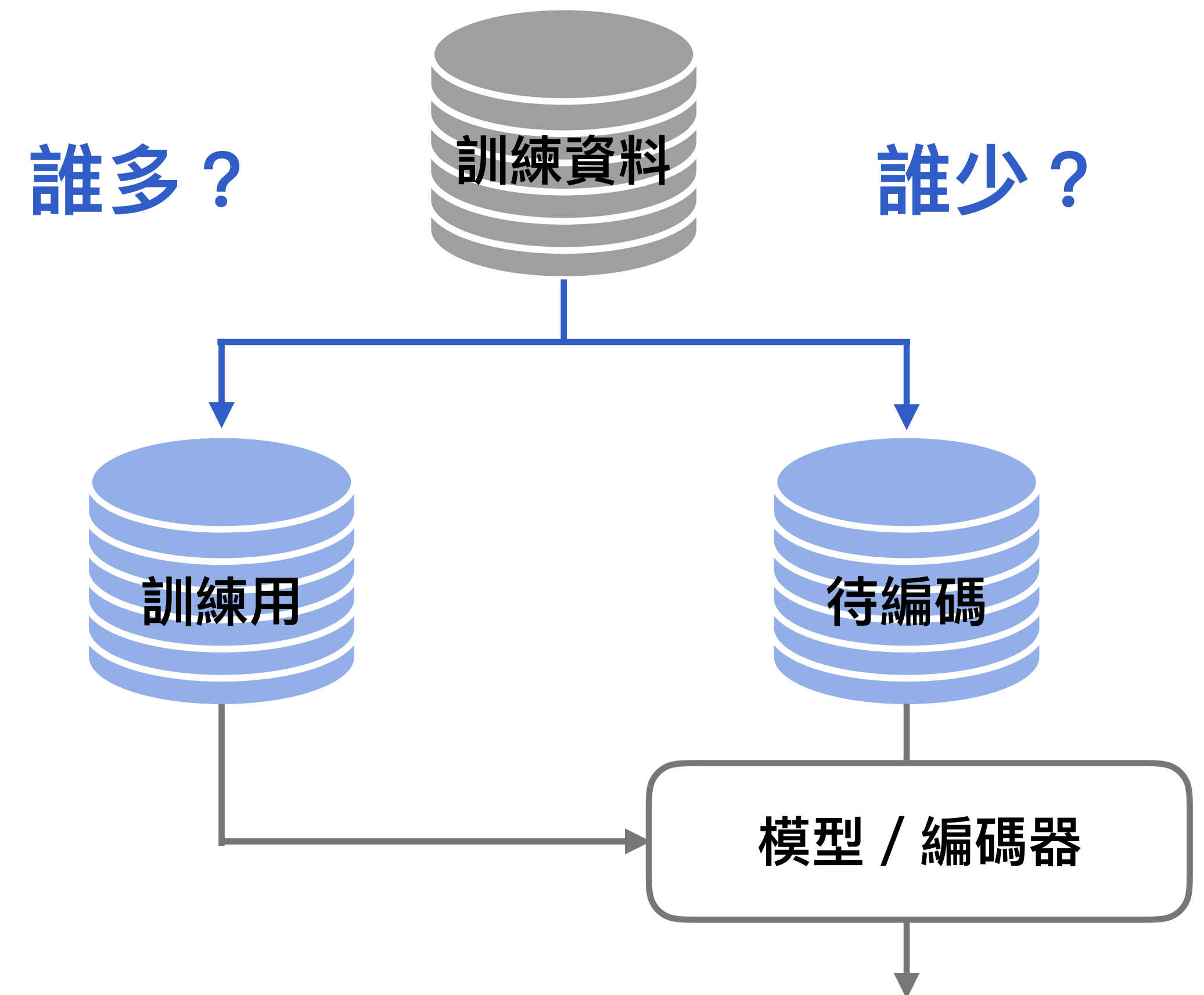
## Stacking





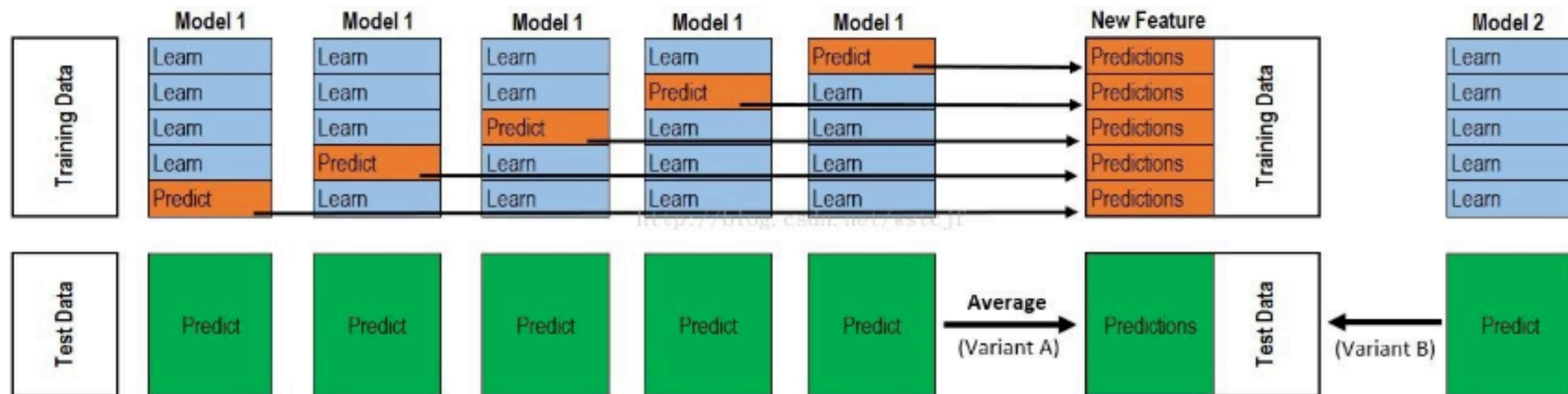
# Blending 與 Stacking 的原理差異

- Stacking 主要是把模型當作下一階的**特徵編碼器**來使用，但是待編碼資料與用來訓練編碼器的資料不可重複 (訓練測試的不可重複性)
- 若將訓練資料切成兩組：**待編碼資料**太少，下一層的資料筆數就會太少，訓練編碼器的資料太少，則編碼器的強度就會不夠，這樣的困境該如何解決呢？



# Stacking 最終設計：巧妙的 K-Fold 拆分

- Stacking 最終採取了下圖設計：將資料拆成  $K$  份 (圖中  $K=5$ )，每  $1/K$  的資料要編碼時，使用其他的  $K-1$  組資料訓練模型/編碼器
- 這樣資料就沒有變少， $K$  夠大時編碼器的強韌性也夠，唯一的問題就是計算時間隨著  $K$  變大而變長，但  $K$  可以調整，且相對深度學習所需的時間來說，這樣的時間長度也還算可接受

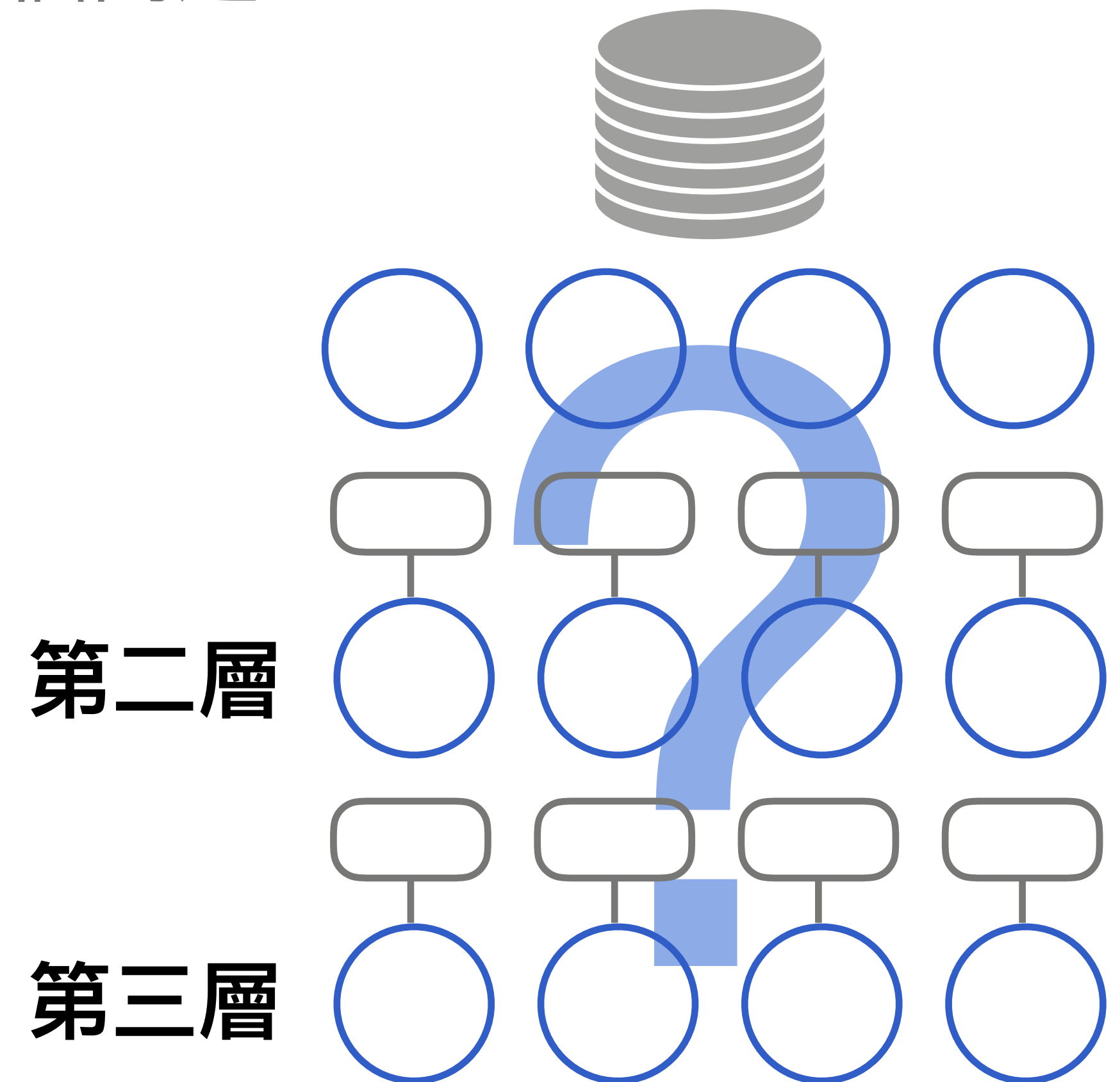


# 自我遞迴的 Stacking ? ( 1 / 3 )

- 大家在看到 Stacking 時可能已經注意到了：既然 Stacking 是在原本特徵上，用**模型**造出**新特徵**，那麼我們自然會想到兩個問題：

- Q1 能不能**新舊特徵一起用**，再用模型去預測呢？

- Q2 新的特徵，能不能**再搭配模型創特徵**，第三層第四層...一直下去呢？





# 自我遞迴的 Stacking ? ( 2 / 3 )

---

Q1：能不能**新舊特徵一起用**，再用模型預測呢？

A1：**可以**，這裡其實有個有趣的思考，也就是：這樣不就可以一直一直無限增加特徵下去？這樣後面的特徵還有意義嗎？ 不會 Overfitting 嗎?...其實加太多次是會 Overfitting 的，必需謹慎切分 Fold 以及新增次數

Q2：新的特徵，能不能**再搭配模型創特徵**，第三層第四層...一直下去呢？

A2：**可以**，但是每多一層，模型會越複雜：因此泛化(又稱為魯棒性)會做得更好，精準度也會下降，所以除非第一層的單模調得很好，否則兩三層就不需要繼續往下了

# 自我遞迴的 Stacking ? ( 3 / 3 )

\*\* 更有趣的其實是下面的問題 (純個人分享，如果感到太抽象的同學可以跳過)

Q3：既然同層新特徵會 Overfitting，層數加深會增加泛化，兩者同時用是不是就能把缺點互相抵銷呢？

A3：可以!! 而且這正是 Stacking 最有趣的地方，但真正實踐時，程式複雜，運算時間又要再往上一個量級，之前曾有大神寫過 StackNet 實現這個想法，用 JVM 加速運算，但實際上使用時調參困難，後繼使用的人就少了

# 真實世界的 Stacking 使用心得

## 實際上寫 Stacking 有這麼困難嗎？

其實不難，就像 sklearn 幫我們寫好了許多機器學習模型，mlxtend 也已經幫我們寫好了 Stacking 的模型，所以用就可以了 (參考今日範例或 [mlxtrend](#) 官網)

## Stacking 結果分數真的比較高嗎？

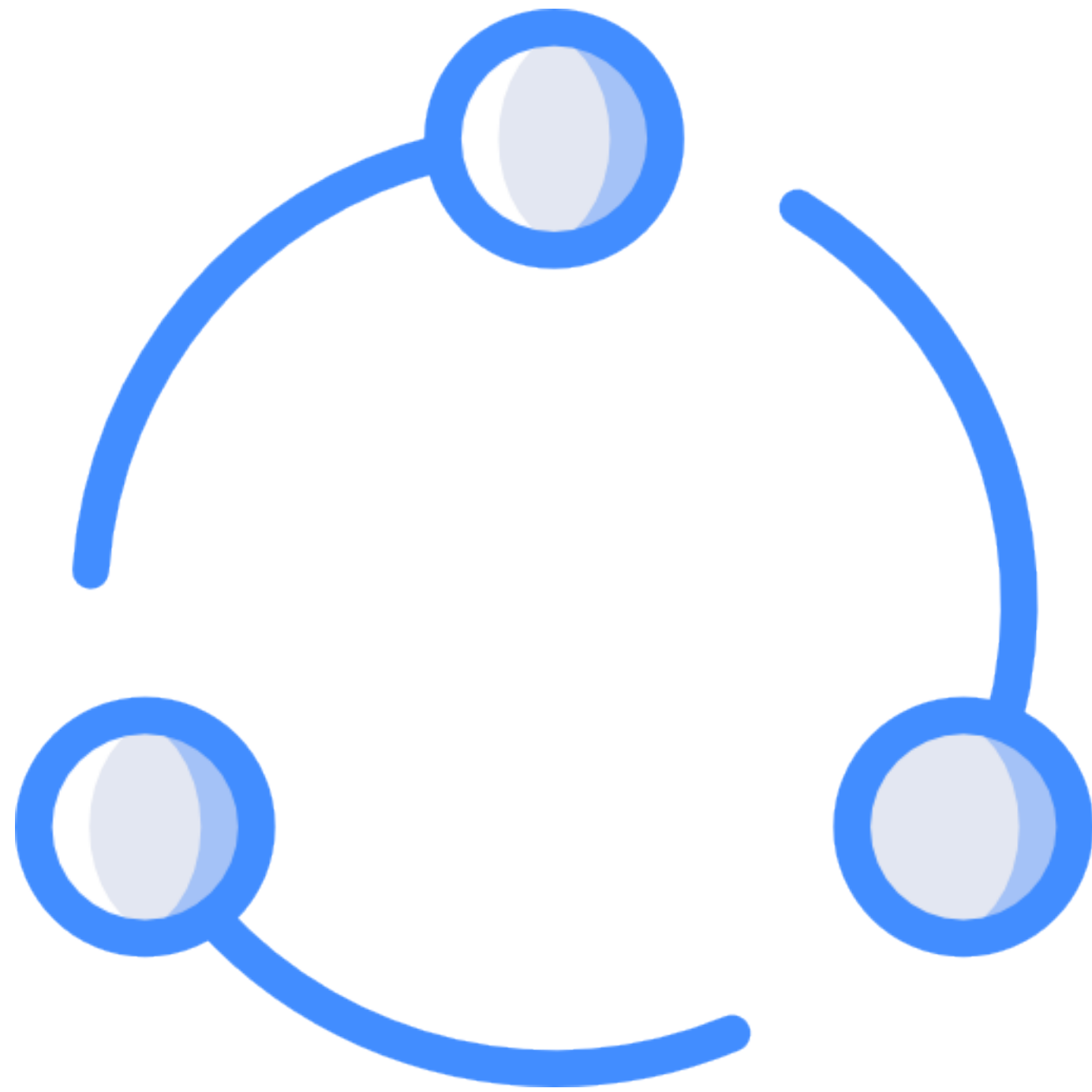
不一定，有時候單模更高，有時候 Blending 效果就不錯，視資料狀況而定

## Stacking 可以做參數調整嗎？

可以，請參考 mlxtrend 的 [調參範例](#)，主要差異是參數名稱寫法稍有不同

## 還有其他做 Stacking 時需要注意的事項嗎？

「分類問題」的 Stacking 要注意兩件事：記得加上 `use_probabilities=True` (輸出特徵才會是機率值)，以及輸出的總特徵數會是：模型數量\*分類數量(回歸問題特徵數=模型數量)



- 堆疊泛化因為將模型預測當作特徵時，要避免**要編碼**的資料與訓練編碼器的資料重疊，因此設計上看起來相當複雜
- 堆疊泛化理論上在**堆疊層數**上**沒有限制**，但如果第一層的單模不夠複雜，堆疊二三層後，改善幅度就有限了
- 混合泛化相對堆疊泛化來說，優點在於**使用容易**，缺點在於**無法更深入的利用資料**更進一步混合模型



# 解題時間 It's Your Turn

請跳出PDF至官網Sample Code & 作業  
開始解題

