

# 集成

# 堆疊泛化(Stacking)



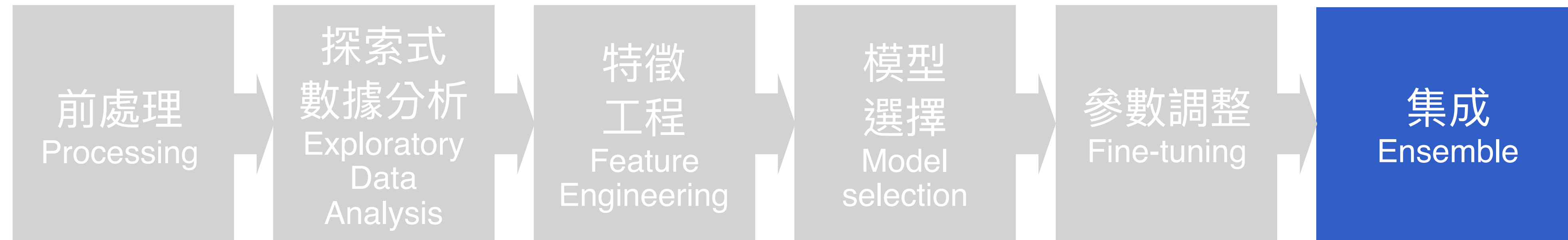
## 出題教練

**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

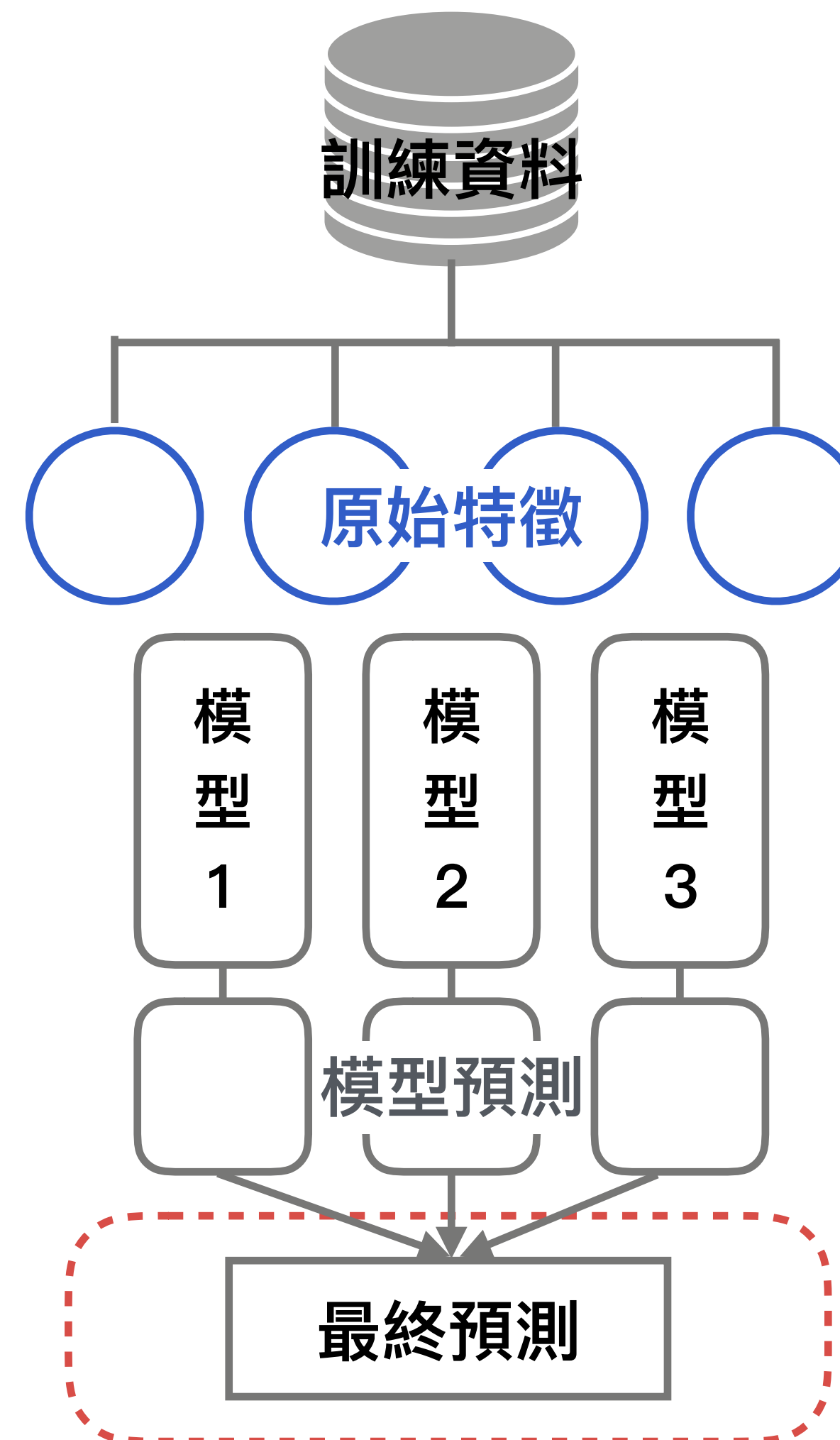
## 相對於 Blending 的改良

- 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

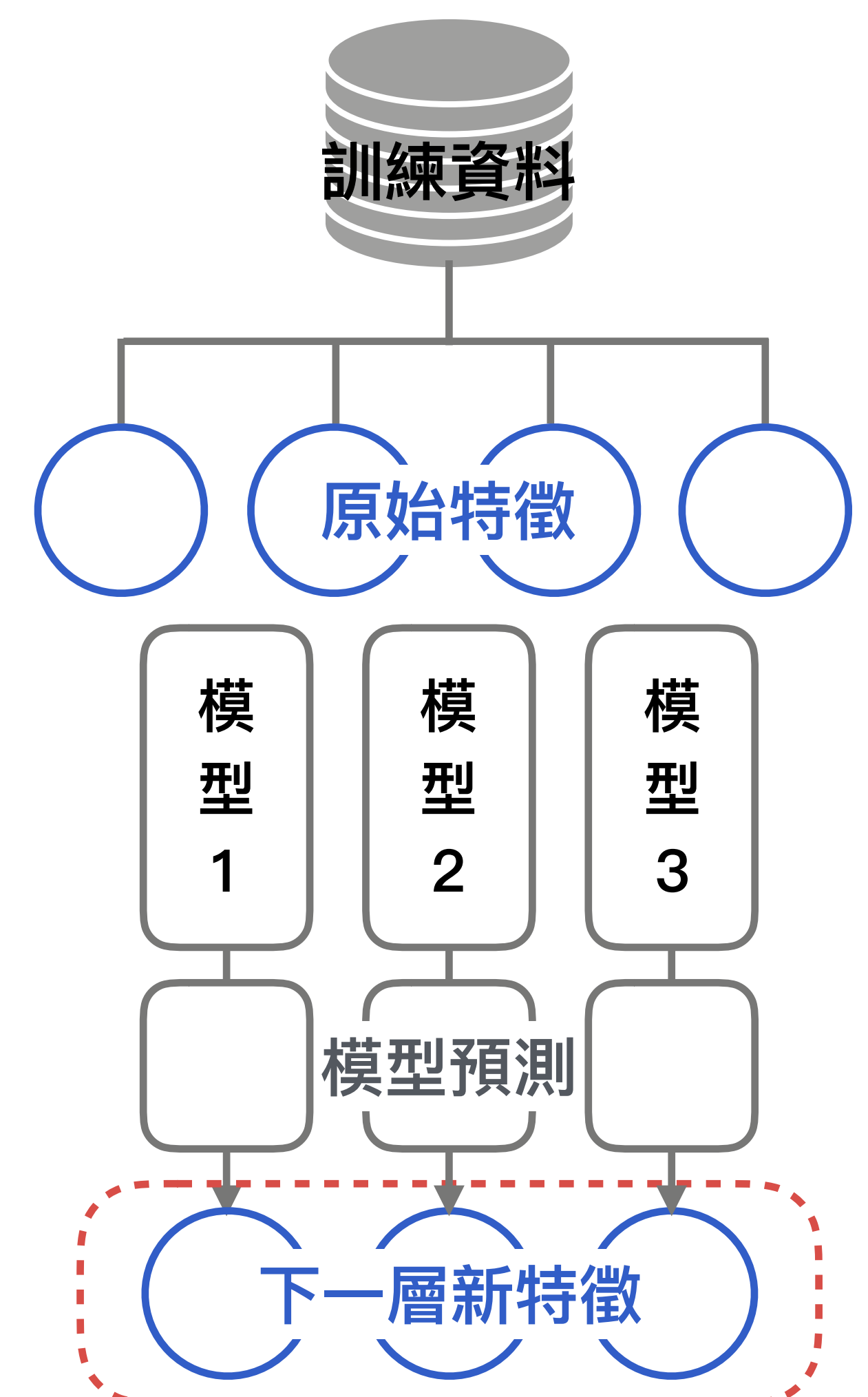
# Stacking 的設計挑戰：訓練測試的不可重複性

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

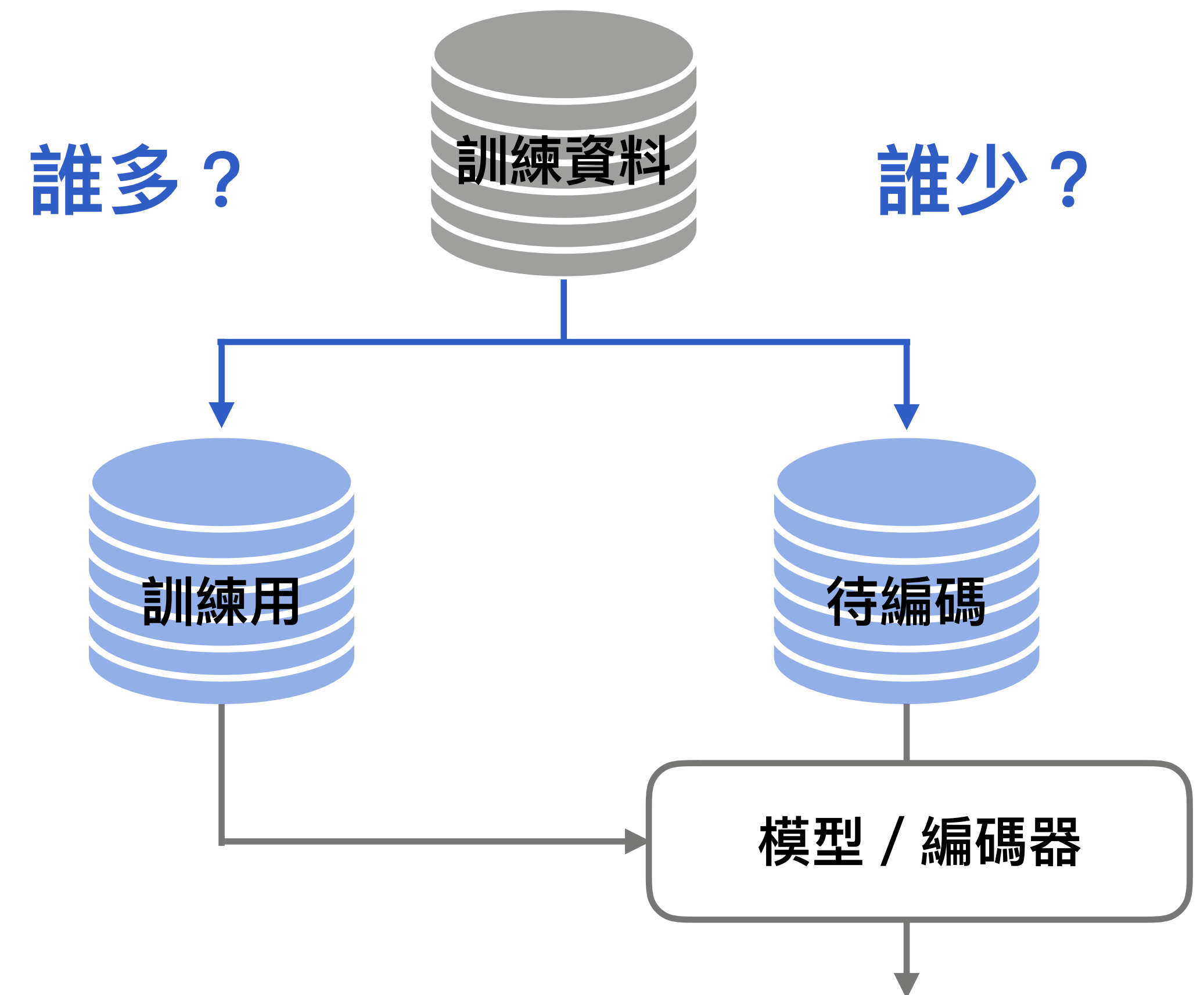


## Stacking



# Blending 與 Stacking 的原理差異

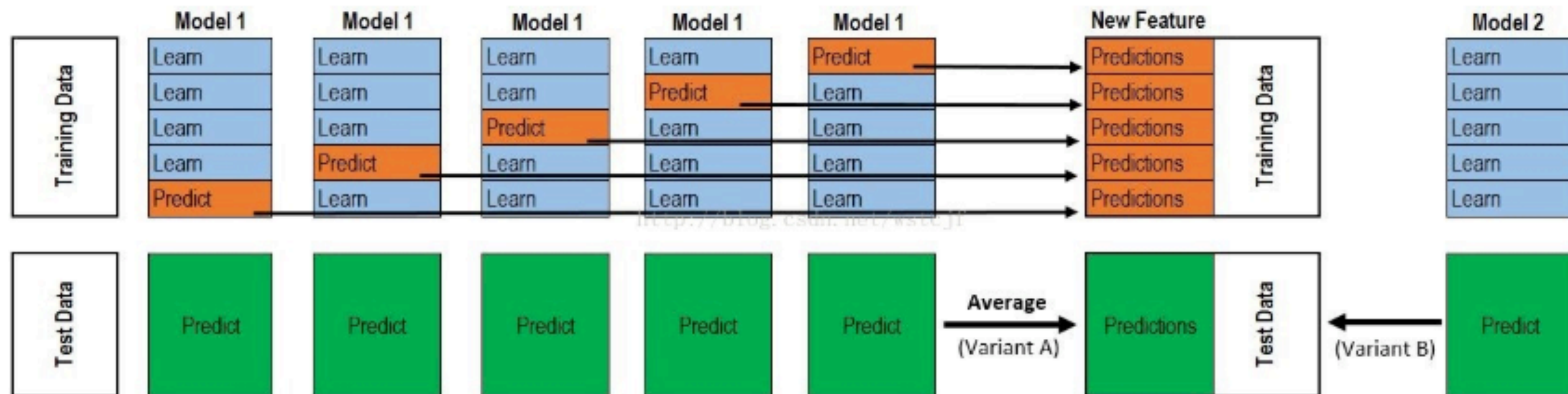
- Stacking 主要是把模型當作下一階的**特徵編碼器**來使用，但是待編碼資料與用來訓練編碼器的資料不可重複 (訓練測試的不可重複性)
- 若將訓練資料切成兩組：**待編碼資料**太少，下一層的資料筆數就會太少，訓練編碼器的資料太少，則編碼器的強度就會不夠，這樣的困境該如何解決呢？





# 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

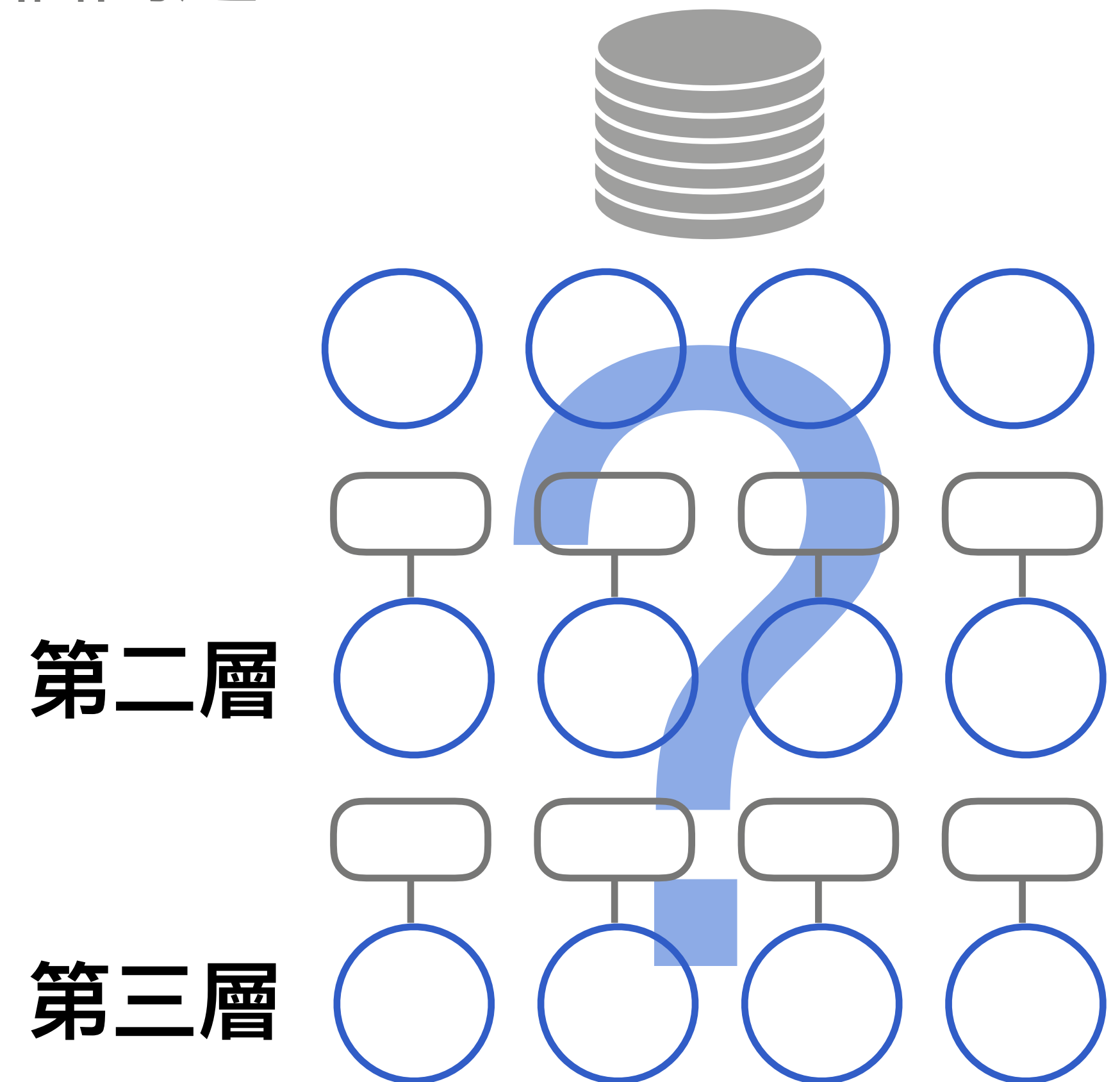


# 自我遞迴的 Stacking ? ( 1 / 3 )

- 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

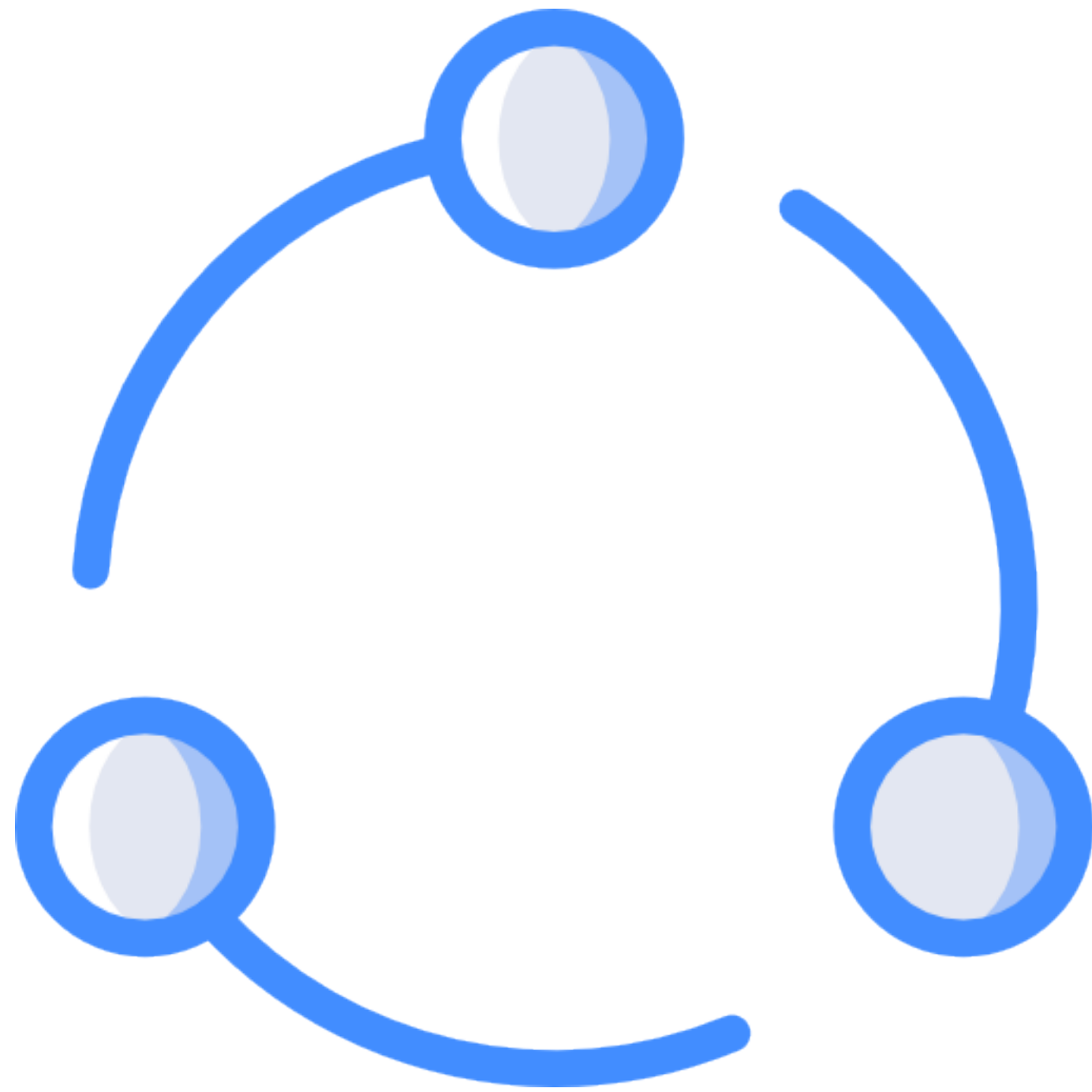
## Stacking 可以做參數調整嗎？

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\_prob=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

