



# 5. Ensemble Learning



「三個臭皮匠勝過一個諸葛亮」



# Ensemble Key Words

## Base Learner :

- 被拿來 ensemble 的基礎模型 。
- Train by base learning algorithm · ex. Decision tree



# Ensemble Key Words

## 三大訓練法分支

### (1) Bagging:

- Like Random Forest, sampling from data or features

### (2) Boosting :

- Boost weak learners too strong learners(sequential learners)



# Ensemble Key Words

## (3) Stacking

- Parallel learners.
- Employing different learning algorithms to train individual learners
- Individual learners then combined by second-level learner which is called meta-learner.

“

# Bagging - 隨機森林

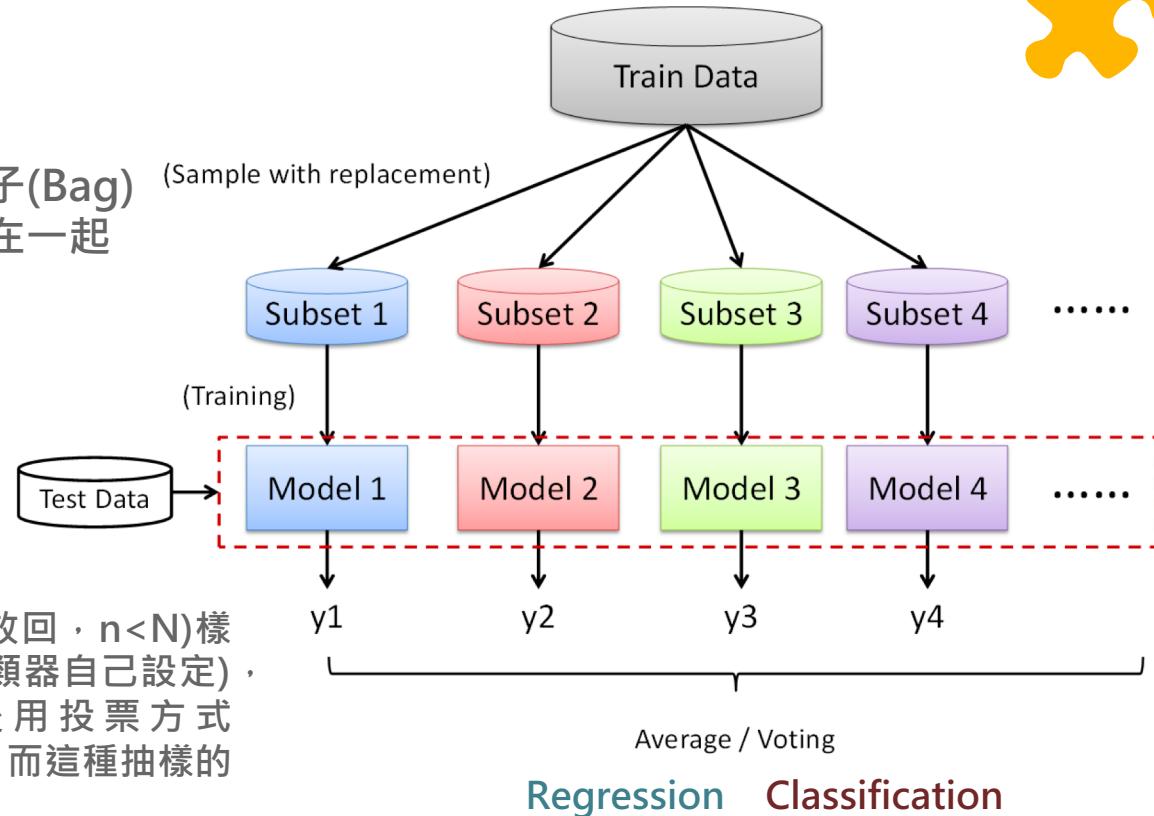
# 民主

Bagging 的核心思想



# Bagging

將資料裝成一個袋子一個袋子(Bag)  
然後將每個袋子的結果結合在一起



從訓練資料中隨機抽取(取出後放回， $n < N$ )樣本訓練多個分類器(要多少個分類器自己設定)，每個分類器的權重一致最後用投票方式(Majority vote)得到最終結果，而這種抽樣的方法在統計上稱為bootstrap。



# Bagging

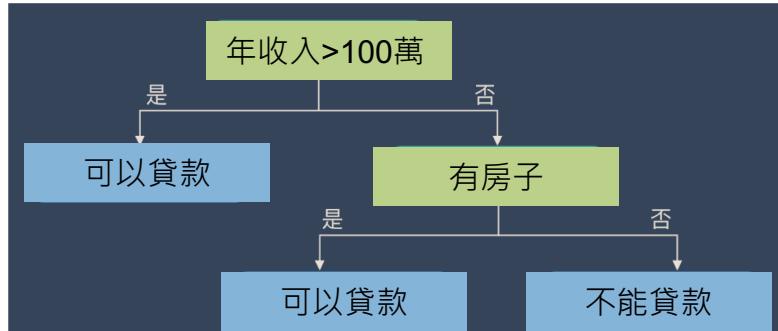
- Bootstrap aggregation (並行訓練一堆分類器)
- 典型算法：隨機森林
- 隨機：數據隨機採樣，特徵隨機選擇
- 森林：很多棵決策數並行運算

Random Forest = Bagging + Decision Tree



# 什麼是隨機森林？

## 決策樹

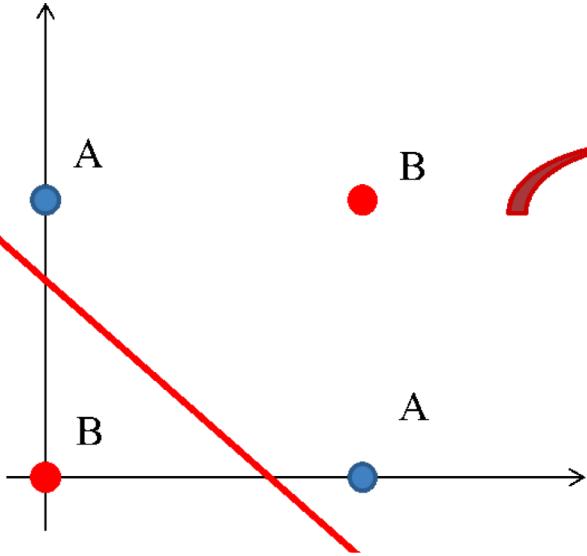


## 隨機森林

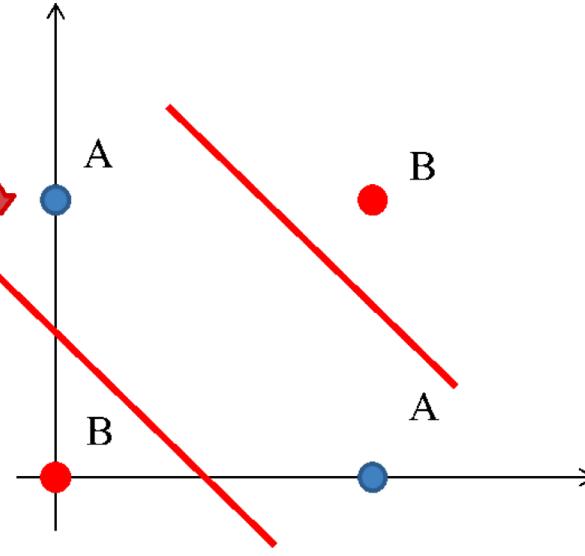


隨機森林是由很多決策樹構成的，不同決策樹之間沒有關聯。

進行分類任務時，新輸入樣本進入，就讓森林中的每一棵決策樹分別進行判斷和分類，每個決策樹會得到一個自己的分類結果，決策樹分類結果中哪一個分類最多，那麼隨機森林就會把這個結果當做最終的結果。



一個分類器/hyperplane分不出來



兩個分類器/hyperplane就分的出來



# 隨機森林 4 步驟



Step 1

隨機抽樣  
訓練決策樹



Step 2

隨機選取屬性  
做節點分裂屬性



Step 3

重複步驟2  
直到不能再分裂



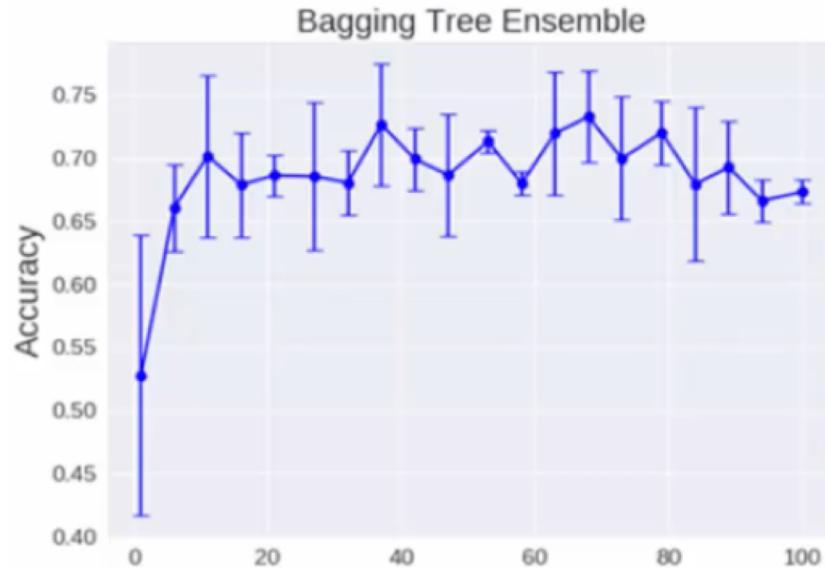
Step 4

建立大量決策樹  
形成森林



# 樹越多越好嗎？

理論上越多效果會越好  
但基本超過一定數量  
Accuracy就平穩了





# 優點

- 可處理高維度的數據，且不用降維，無需做特徵選擇
- 可判斷特徵重要程度
- 不易過擬合
- 訓練速度較快，易做成並行
- 對於不平衡的數據集來說，可平衡誤差
- 如果有很大一部分的特徵遺失，仍可以維持準確度



# 缺點

- 隨機森林已被證明在某些噪音較大的分類或迴歸問題上會過擬合
- 黑盒子，無法控制模型內部的執行。只能不斷在參數和隨機種子間測試
- 可能有很多相似的決策樹，掩蓋真實的結果。
- 對於小資料或低維資料，可能不能產生很好的分類

“

# Boosting – XGBoost

eXtreme Gradient Boosting(極限梯度提升)

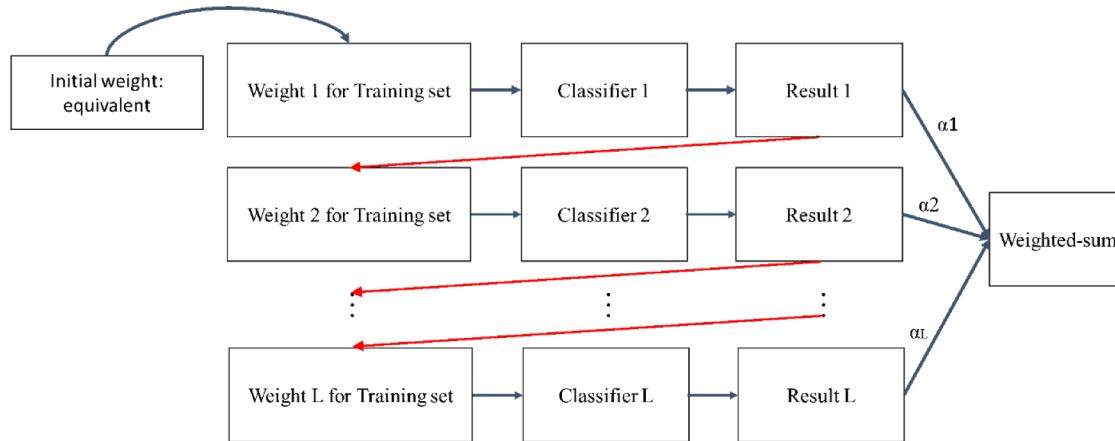


# 挑選精英

Boosting 的核心思想



# Boosting

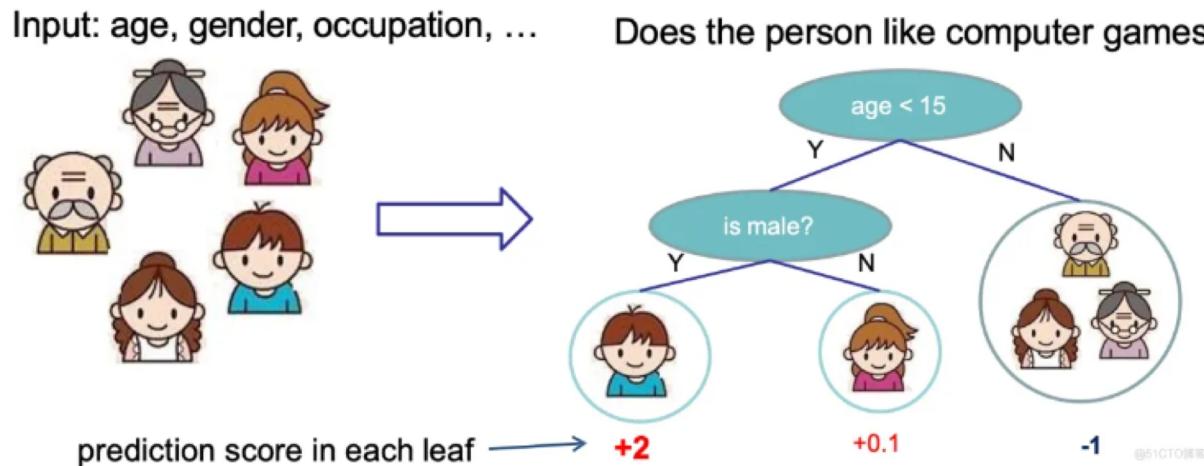


Boosting 算法是將很多個弱的分類器(weak classifier)進行合成變成一個強分類器(Strong classifier)，分類器之間是有關聯性的，透過將舊分類器錯誤歸類權重提升，然後交給下一棵樹，訓練新的分類器，這樣新的分類器就會學習到錯誤分類資料(misclassified data)的特性，進而提升分類結果。



# 預測一家人對電子遊戲的喜好程度

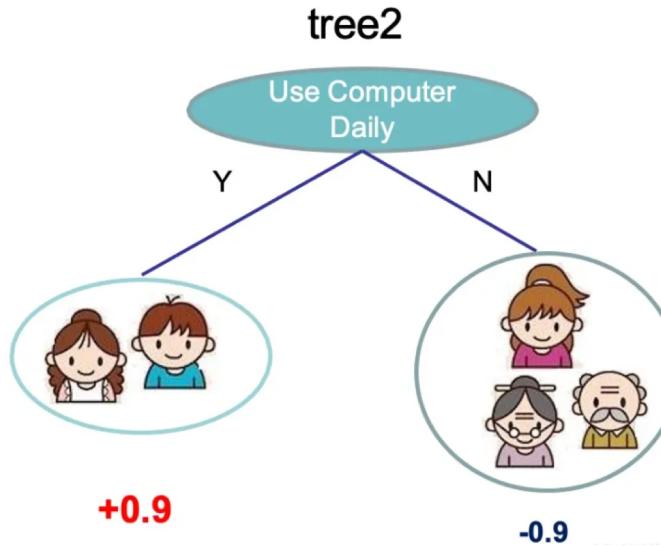
- Tree1: 考慮到年輕和年老相比，年輕更可能喜歡電子遊戲，以及男性和女性相比，男性更喜歡電子遊戲，故先根據年齡大小區分小孩和大人，然後再通過性別區分開是男是女，逐一給各人在電子遊戲喜好程度上打分數





# 預測一家人對電子遊戲的喜好程度

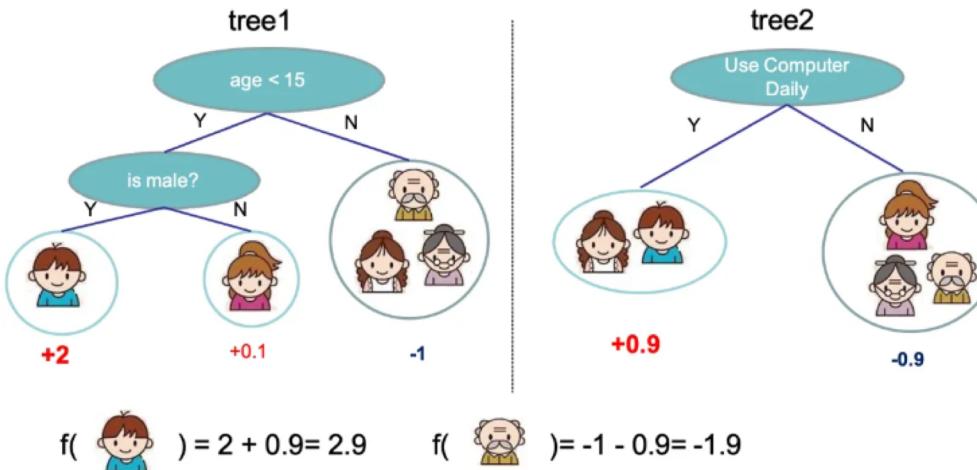
- Tree2: 根據日常是否使用電腦來看





# 預測一家人對電子遊戲的喜好程度

- 訓練出2棵樹tree1和tree2，兩棵樹的結論累加起來便是最終的結論
- 小孩的預測分數就是兩棵樹中小孩所落到的結點的分數相加： $2 + 0.9 = 2.9$ 。  
爺爺的預測分數： $-1 + (-0.9) = -1.9$ 。



Prediction of is sum of scores predicted by each of the tree

©EICTD編輯



- 不斷地添加樹，不斷地進行特徵分裂來生長一棵樹，每次添加一個樹，就是學習一個新函數 $f(x)$ ，去擬合上次預測的殘差。

$$\hat{y} = \phi(x_i) = \sum_{k=1}^K f_k(x_i)$$

where  $F = \{f(x) = w_{q(x)}\} (q : R^m \rightarrow T, w \in R^T)$

$W_{q(x)}$ 為葉子節點 $q$ 的分數， $f$ 對應所有 $K$ 顆樹的集合，而 $f(X)$ 為其中一棵樹

- 訓練完成得到 $k$ 棵樹，要預測一個樣本的分數，其實就是根據這個樣本的特徵，在每棵樹中會落到對應的一個葉子節點，每個葉子節點就對應一個分數。
- 最後只需要將每棵樹對應的分數加起來就是該樣本的預測值。



目標：使得樹群的預測值盡量接近真實值，且有盡量大的泛化能力。

損失函數

$$\text{目標函數 } L(\phi) = \sum_i l(\hat{y}_i - y_i) + \sum_k \Omega(f_k)$$

預測值                    真實值

正則化(regularization)  
L1(Lasso) / L2(Ridge)

- 損失函數：  
揭示訓練誤差，鼓勵模型儘量去擬合訓練數據，使得最後的模型會有較少的bias。
- 正則化：  
定義複雜度的函數，鼓勵更簡單的模型，值越小複雜化越低，泛化能力越強。



# Note

由於Boosting將注意力集中在分類錯誤的資料上  
因此對訓練資料的噪聲非常敏感  
如果一筆訓練資料噪音資料很多  
那後面分類器都會集中在進行噪聲資料上分類  
反而會影響最終的分類性能



# 手動調參

參數	功能說明
n_estimators	樹的數量，太少會欠擬合，太多可能過擬合
max_depth	樹的最大深度，影響模型複雜度
learning_rate (eta)	每棵樹學習貢獻度，太大會過擬合，太小則學習太慢
subsample	每次訓練用多少比例樣本（避免過擬合）
colsample_bytree	每棵樹訓練時隨機取多少比例的特徵
gamma	最小 loss 減少才允許分裂的閾值（越大越保守）
min_child_weight	控制葉節點最小樣本總權重（防止過擬合）



`n_estimators` : 樹的數量 (迭代次數)。

- 定義：模型中要長幾棵決策樹。
- 預設值：100
- 影響：

每棵樹都是在幫前一棵「修正錯誤」。

樹長越多，模型學得越完整；但也可能太複雜、記太多細節（過擬合）。

如果樹太少，模型可能學不夠。

- 太少 → 欠擬合 (underfitting)
- 太多 → 容易過擬合 (overfitting)，除非搭配 `early_stopping`

- 建議調法：
  - 可搭配 `early_stopping_rounds` 自動停止
  - 通常搭配較小的 `learning_rate` 增加 `n_estimators`



## max\_depth：每棵樹的最大深度

- 定義：決定每棵樹能長多深（幾層判斷）
- 預設值：6
- 影響：
  - 值越大 → 每棵樹模型越複雜、學得越細，容易過擬合
  - 值越小 → 模型簡單，泛化力強但可能無法捕捉複雜模式
- 建議範圍：3–10 為常見區間
- 技巧：
  - 資料量大或特徵多時，可稍微放寬
  - 若搭配 min\_child\_weight 使用，可防止深樹過擬合



`learning_rate` ( 或 `eta` ) : 每棵樹的學習速率，每棵樹改多少

- 定義：每棵樹對總預測結果的貢獻比例
- 預設值：0.3 ( 稍偏大 )
- 影響：
  - 小 → 學習慢但精細，需較多 `n_estimators`(樹的數量)
  - 大 → 學習快但不穩，容易震盪或過擬合
- 建議範圍：0.01 ~ 0.2 ( 0.1 最常見 )
- 常見搭配：
  - `learning_rate` ↓ → `n_estimators` ↑
  - 較小的 `learning_rate` 更容易取得穩定效果



**subsample**：訓練每棵樹使用的樣本比例，每棵樹看幾成的資料

- 定義：每棵樹訓練時，隨機抽多少比例的樣本。
- 不是每棵樹都看全部資料，有時只給它 80%、90%，讓每棵樹「看得不一樣」，增加隨機性，也比較不會太執著某些資料。  
→ 有點像「分組學習」，大家各自練習，整體更穩定
- 預設值：1.0（使用全部樣本）
- 作用：增加隨機性、降低過擬合
- 建議範圍：0.5–1.0（偏低時模型變保守）
- 技巧：
  - 可以看作是 Bagging 思想的應用
  - 設為 0.8 是經驗法則起點



## colsample\_bytree：訓練每棵樹使用的特徵比例，每棵樹看幾成的特徵

- 定義：每棵樹訓練時，隨機選用多少比例的特徵
- 預設值：1.0
- 作用：減少特徵間共線性、降低過擬合
- 建議範圍：0.5–1.0
- 技巧：
  - 若特徵數非常多，建議用 0.7 或更低
  - 與 subsample 互補調節模型隨機性



gamma：像「讓模型停下來深挖的門檻」，設定高一點，模型就會更保守、不輕易分支。

- 定義：決定「要不要繼續分支」的門檻。  
    🔍 「如果分一刀，只讓模型變好一點點，那就別分。」
- 預設值：0（無限制）
- 作用：加強分裂條件、使模型更保守，避免學過多雜訊
- 建議範圍：0–5

#### 🧠 怎麼理解？

gamma = 0：只要模型有一點點進步，就會讓樹繼續分裂（比較積極）

gamma = 5：只有當分裂能讓模型「明顯進步」，才會分裂（比較保守）

- 技巧：
  - 增加 gamma → 樹會長得越簡單，避免過擬合
  - 可配合 max\_depth 精細控制模型結構



`min_child_weight`：最小葉節點樣本總權重，一個節點下，至少要有多少人

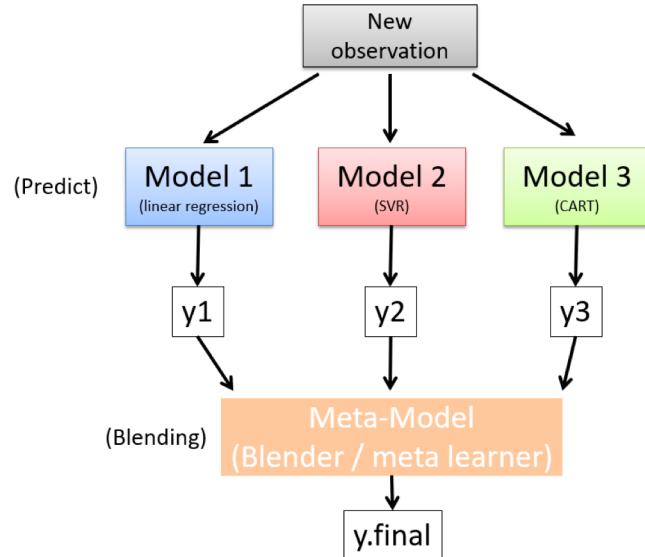
- 定義：每個葉節點所需的最小樣本總和（樣本權重加總）
- 預設值：1
- 影響：
  - 值越大 → 模型越保守、不易分裂 → 降低過擬合風險
  - 值越小 → 容易過擬合，尤其資料雜訊大時
- 建議範圍：1–10（偏高時適合處理雜訊資料）
- 技巧：
  - 與 `max_depth` 搭配調整，效果更佳
  - 可理解為：需有多少人支持這個分裂才值得做

“  
Stacking



# Stacking

- Stacking 是一種集成學習方法，將多個模型的預測結果結合起來，並使用一個新的模型（Meta-Model）來做最終的預測。
- 目的是利用多個模型的優勢，克服單一模型的限制。

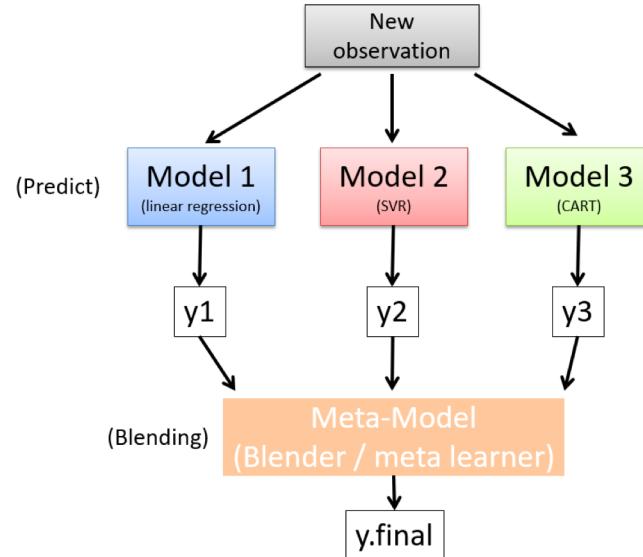




# Stacking

基本流程

1. 第一層：使用多個基礎模型（Base Models），每個模型都對訓練資料進行學習。
2. 第二層：將基礎模型的預測結果作為特徵，輸入到 Meta-Model 中，進行最終預測。
3. 最終預測：Meta-Model 將給出最終的分類或迴歸結果。





# Stacking - 優勢

1. 提高模型準確性：Stacking 通常能夠比單一模型提供更高的預測準確度，因為它結合了多個模型的優勢。
2. 減少過擬合風險：通過將多個不同模型的預測結果結合，Stacking 能減少過擬合的風險，特別是在模型本身表現不穩定時。
3. 增強模型的泛化能力：當使用多樣化的基礎模型時，Stacking 能利用不同模型的優勢，對未見資料的預測更加穩定和準確。
4. 能處理不同類型的資料：Stacking 可以與多種不同類型的基礎模型（如決策樹、SVM、神經網絡等）結合，靈活應對不同的數據特徵和問題。
5. 解釋性和調整：由於 Meta-Model 是相對簡單的線性回歸或邏輯迴歸等模型，Stacking 的最終模型可以較容易進行調整。



# Stacking - 劣勢

1. 計算成本高：由於需要訓練多個基礎模型以及一個 Meta-Model，Stacking 會顯著增加計算開銷和訓練時間。
2. 過於複雜：Stacking 需要管理多個基礎模型及其預測，這使得實現和維護變得相對複雜。
3. 模型選擇的挑戰：選擇合適的基礎模型和 Meta-Model 是一項挑戰，選錯模型會導致性能下降或無法達到預期效果。
4. 過度依賴基礎模型：Stacking 依賴於基礎模型的多樣性和準確性，若基礎模型過於相似或預測效果差，Stacking 的表現將會受限。

“

# Summary



# 樣本選擇

## Bagging

每次的訓練集皆為隨機抽取，每個樣本權重一致，抽出可放回，以獨立同分布選取訓練集的子集，訓練弱分類器

## Boosting

每次選擇的訓練集都是依賴上一次學習得結果，彼此間相依，並根據錯誤率，給予彼此之間相依不同的權重取樣



# 分類器

## Bagging

每個分類器的權重相等

## Boosting

每個弱分類器都有相應權重，對於分類誤差小的分類器會有更大權重



# 每個分類器的形成

**Bagging**

可並行生成

**Boosting**

每個弱分類器只能依賴上一次的分類器順序生成



# 優點

## Bagging

**can significantly reduce the variance**

原始訓練樣本中有噪聲資料，透過Bagging抽樣，不讓有噪資料被訓練到

## Boosting

**can significantly reduce the bias**

逐漸提高訓練難度與變換不同面向，避免建構重蹈覆徹的模型



# 缺點

**black box**，難以解釋  
調參較為不易



# Summary

Reduce	Bias	Variance	Parallelization
Bagging		降低	n
Boosting	降低		1
Stacking		降低	n



# Thanks!