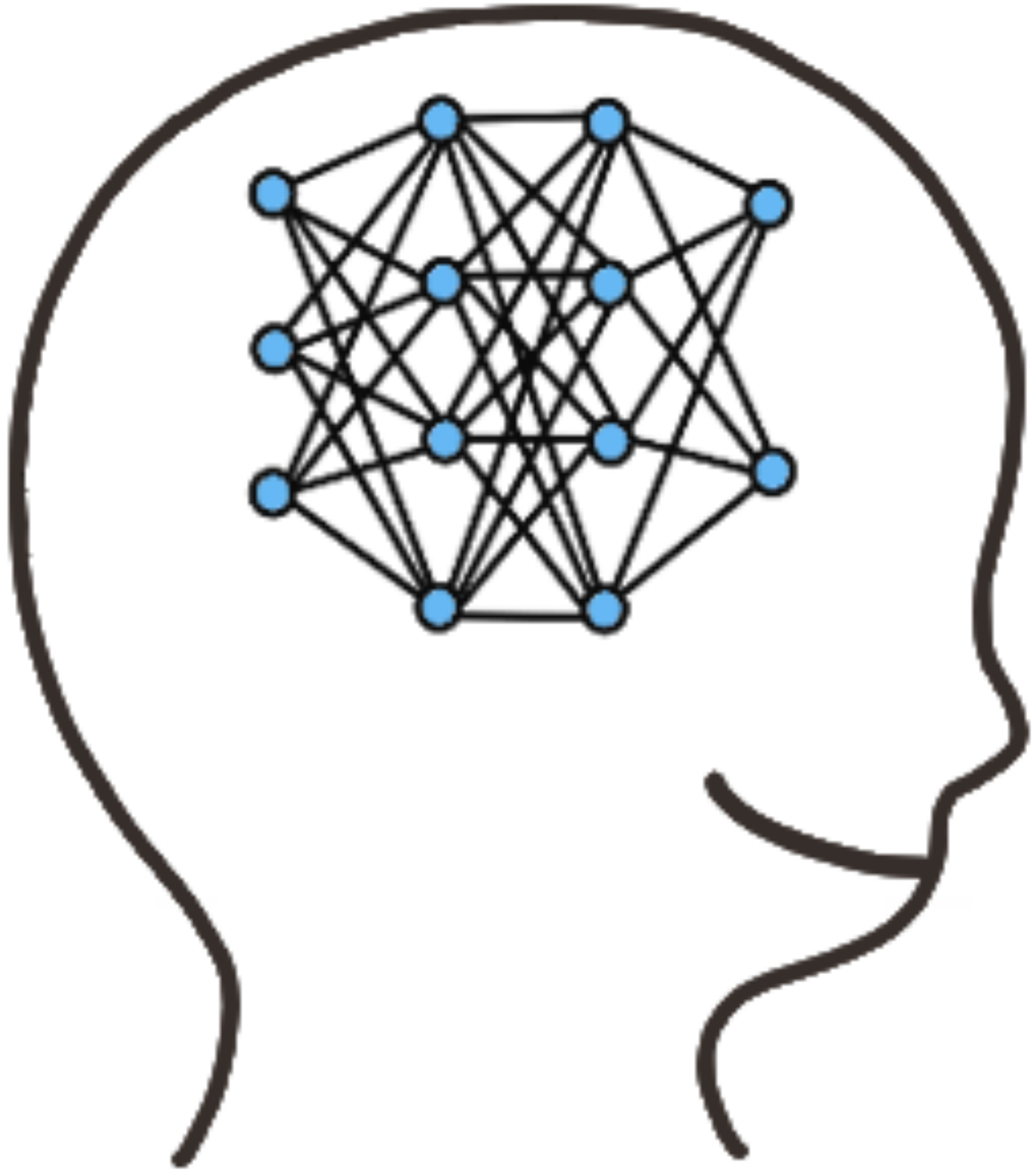


整體學習
Ensemble Learning

原理

- **「三個臭皮匠(裨將)，勝過一個諸葛亮」**
  - 裨將：副將
  - 三個才能平庸的人一起集思廣益，勝過一個優秀的人。

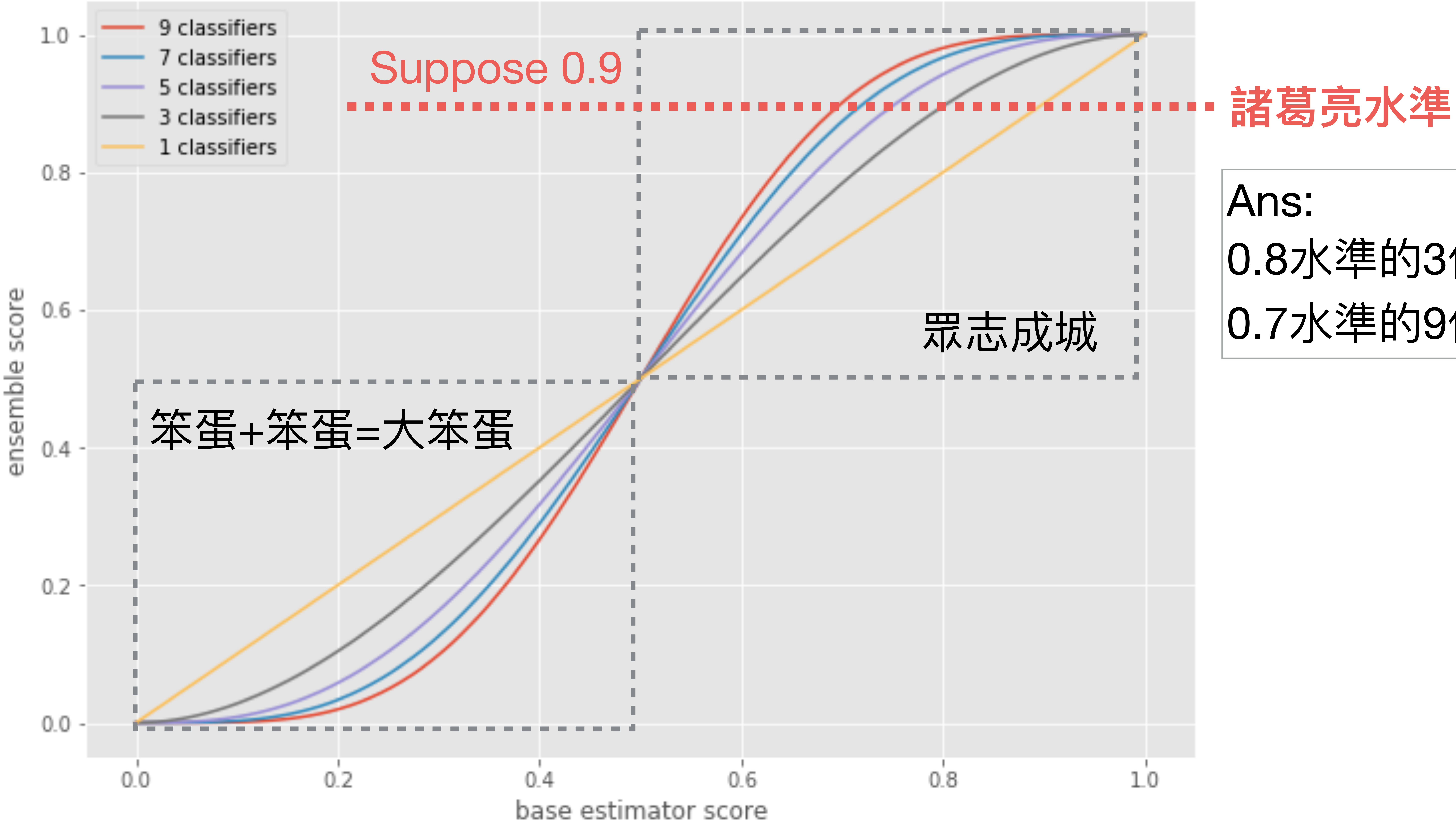
三個臭裨將真的可以勝過諸葛亮？

- 問題：燈泡故障的機率是10%，請問取3個燈泡中至少有2個是正常的機率？
- 二項分佈(Binominal Distribution) (獨立事件) $C_k^n p^k (1-p)^{n-k}$
- 成功(p)/失敗(1-p)：0.9(正常)/0.1(故障)
- P = P(3個燈泡都正常) + P(其中2個燈泡正常)

$$
\begin{aligned}
&= C_2^3 (0.9)^2 \cdot (0.1)^1 + C_3^3 (0.9)^3 \\
&= 3 \cdot 0.81 \cdot 0.1 + 1 \cdot 0.729 \\
&= 0.243 + 0.729 \\
&= 0.972
\end{aligned}
$$

三個臭裨將真的可以勝過諸葛亮？

- 若諸葛亮和臭裨將一起來做分類(0,1)問題，諸葛亮自己預測y值，臭裨將們先各自預測y後，再採多數決方式決定最後要預測的y值。
- 諸葛亮(強學習器，strong learner)
  - 假設準確度0.9
- 臭裨將(弱學習器，weak learner)
  - 準確度要多高才能超越諸葛亮？
  - 要幾位裨將才夠超越諸葛亮？



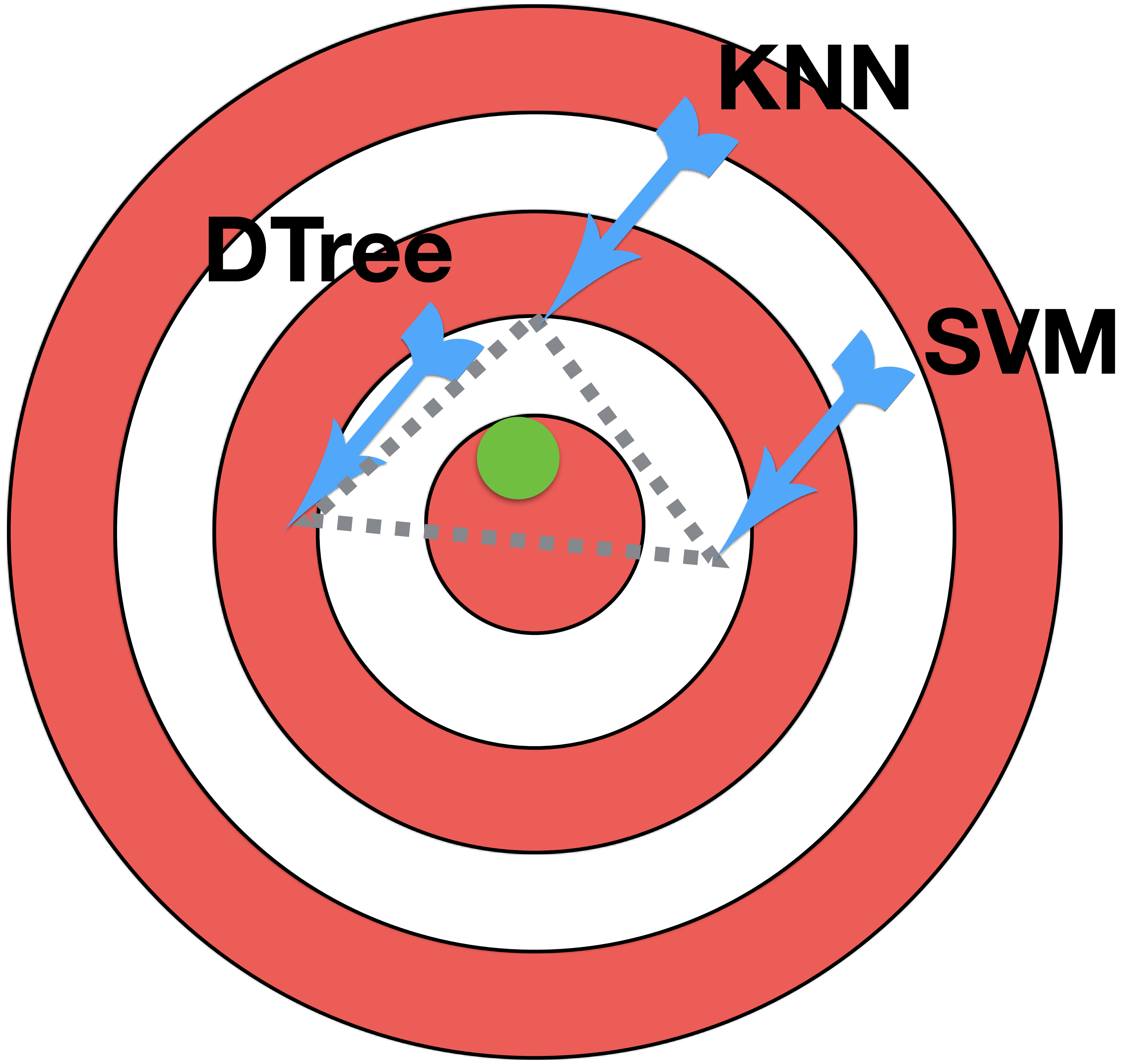
Ans:
0.8水準的3位裨將
0.7水準的9位裨將

使用原則

- 每個分類器準確度至少>0.5
- 每個分類器種類不同 (要各有所長，盡量符合獨立事件)

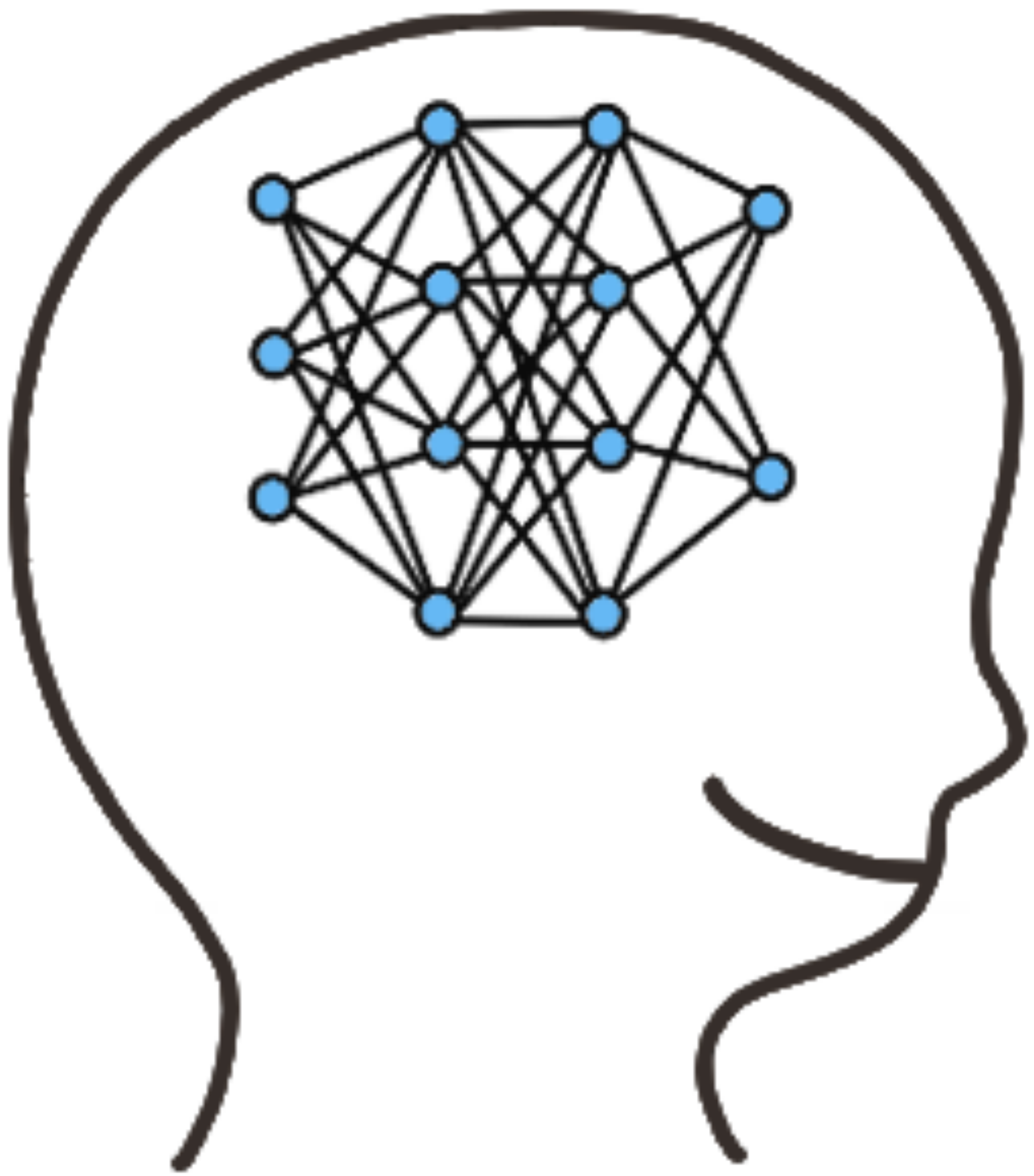
整合多種演算法效果

- 降低因不同學習演算法特性所產生的bias



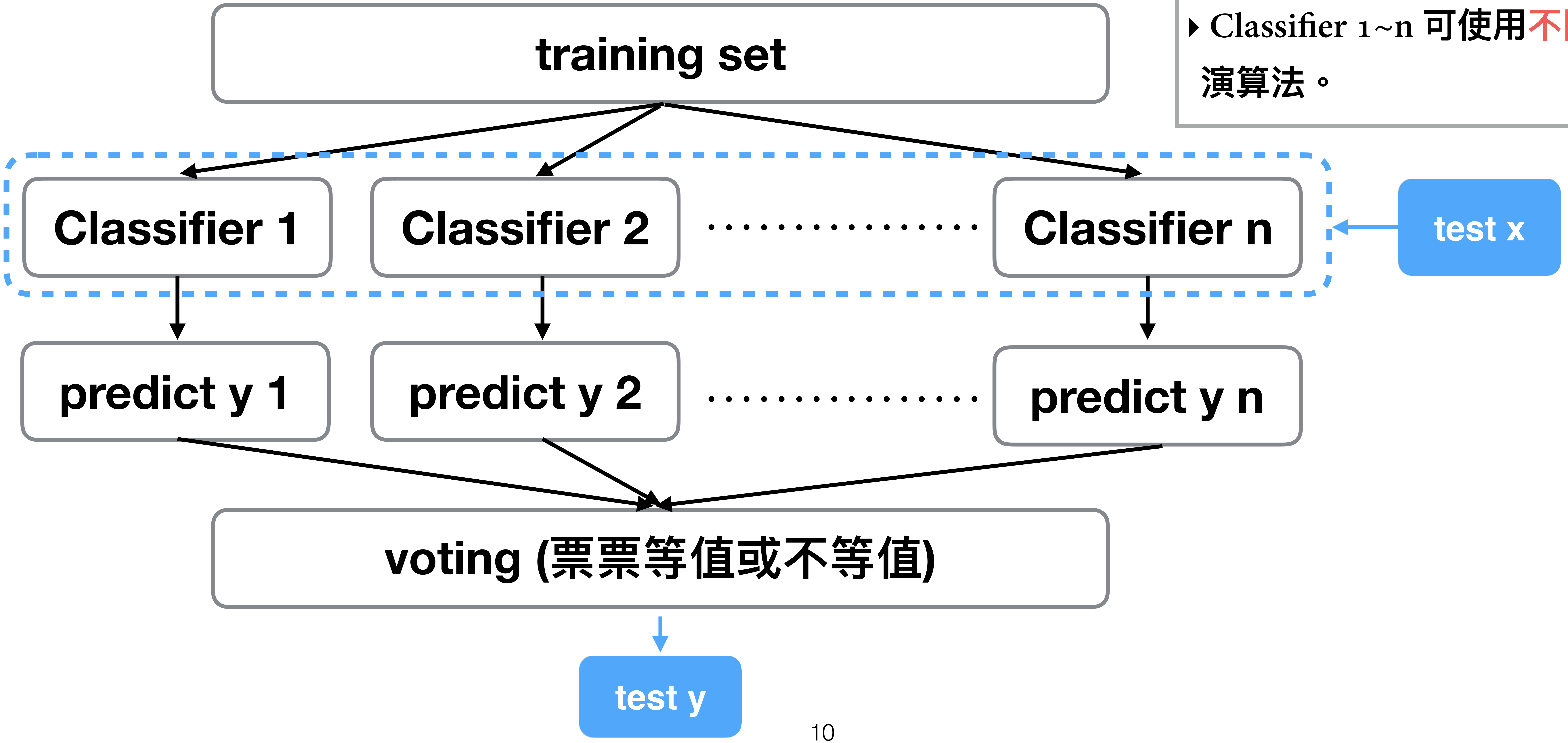
Ensemble Learning with sklearn

- `ensemble.AdaBoostClassifier`([…])
- An AdaBoost classifier.
- `ensemble.AdaBoostRegressor`([base_estimator, …])
- An AdaBoost regressor.
- `ensemble.BaggingClassifier`([base_estimator, …])
- A Bagging classifier.
- `ensemble.BaggingRegressor`([base_estimator, …])
- A Bagging regressor.
- `ensemble.ExtraTreesClassifier`([…])
- An extra-trees classifier.
- `ensemble.ExtraTreesRegressor`([n_estimators, …])
- An extra-trees regressor.
- `ensemble.GradientBoostingClassifier`([loss, …])
- Gradient Boosting for classification.
- `ensemble.GradientBoostingRegressor`([loss, …])
- Gradient Boosting for regression.
- `ensemble.IsolationForest`([n_estimators, …])
- Isolation Forest Algorithm
- `ensemble.RandomForestClassifier`([…])
- A random forest classifier.
- `ensemble.RandomForestRegressor`([…])
- A random forest regressor.
- `ensemble.RandomTreesEmbedding`([…])
- An ensemble of totally random trees.
- `ensemble.VotingClassifier`(estimators[, …])
- Soft Voting/Majority Rule classifier for unfitted estimators.



投票分類器
Voting Classifier

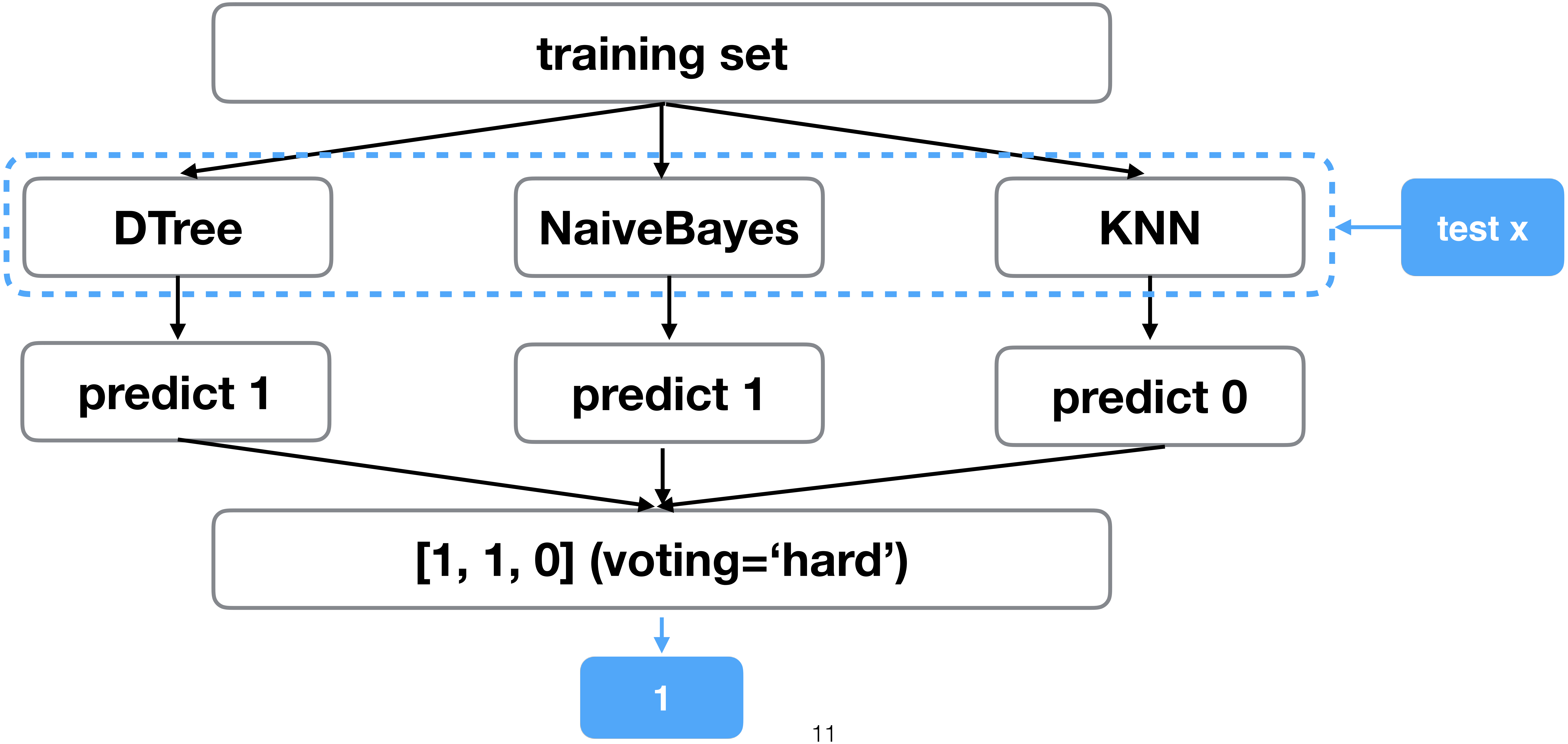
Voting



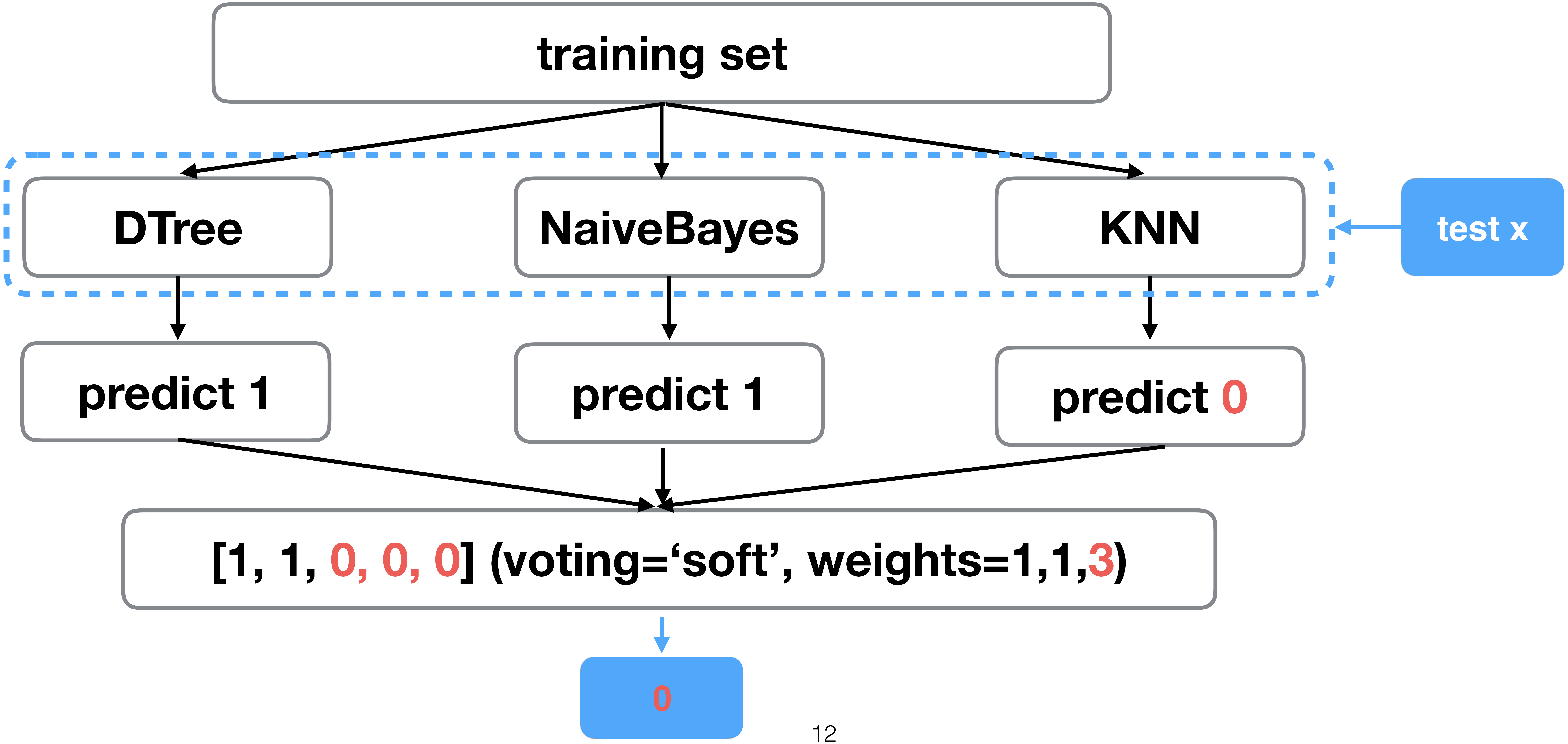
Notes

- Classifier 1~n 可使用不同的分類演算法。

Voting Example 1

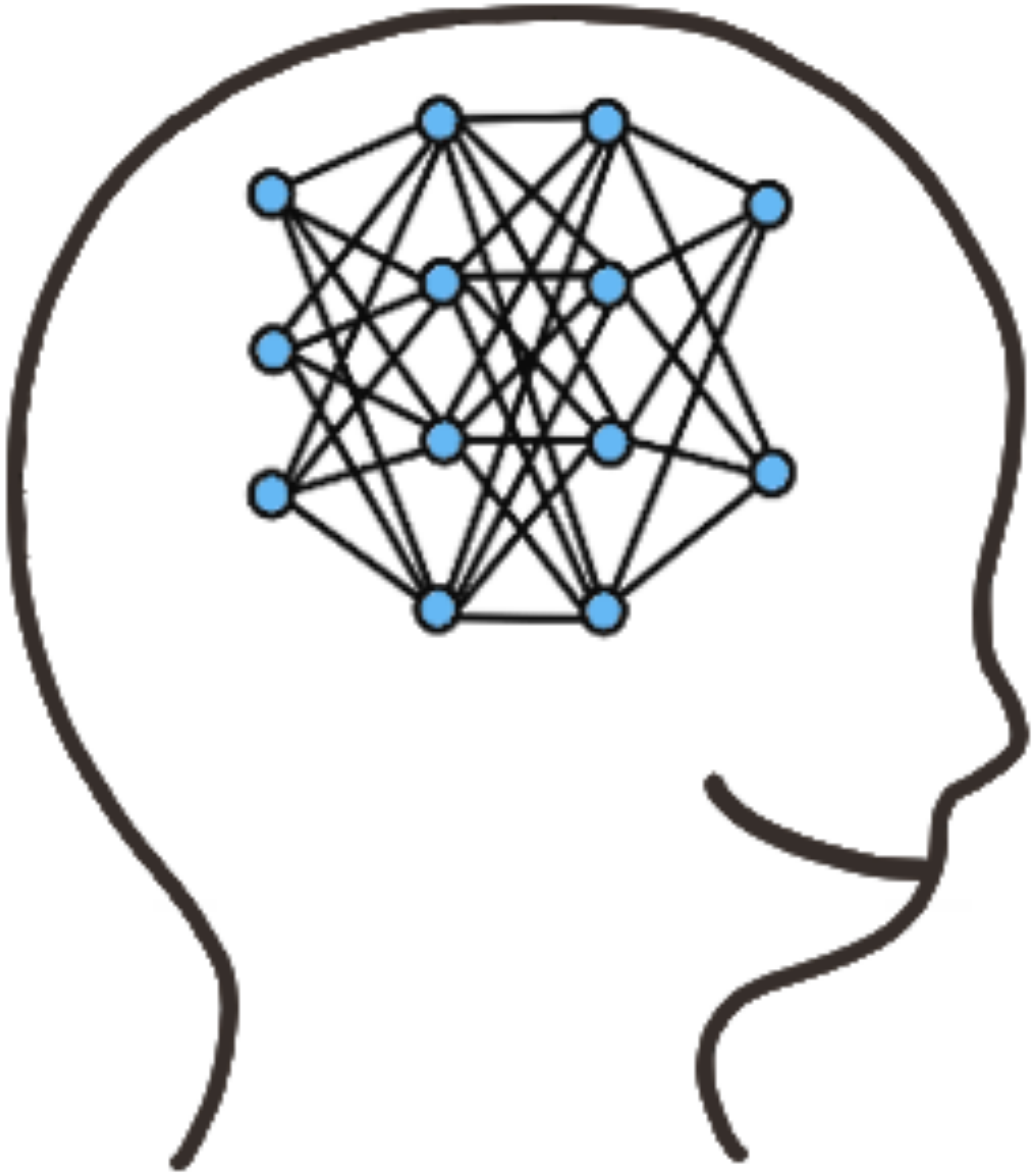


Voting Example 2



VotingClassifier

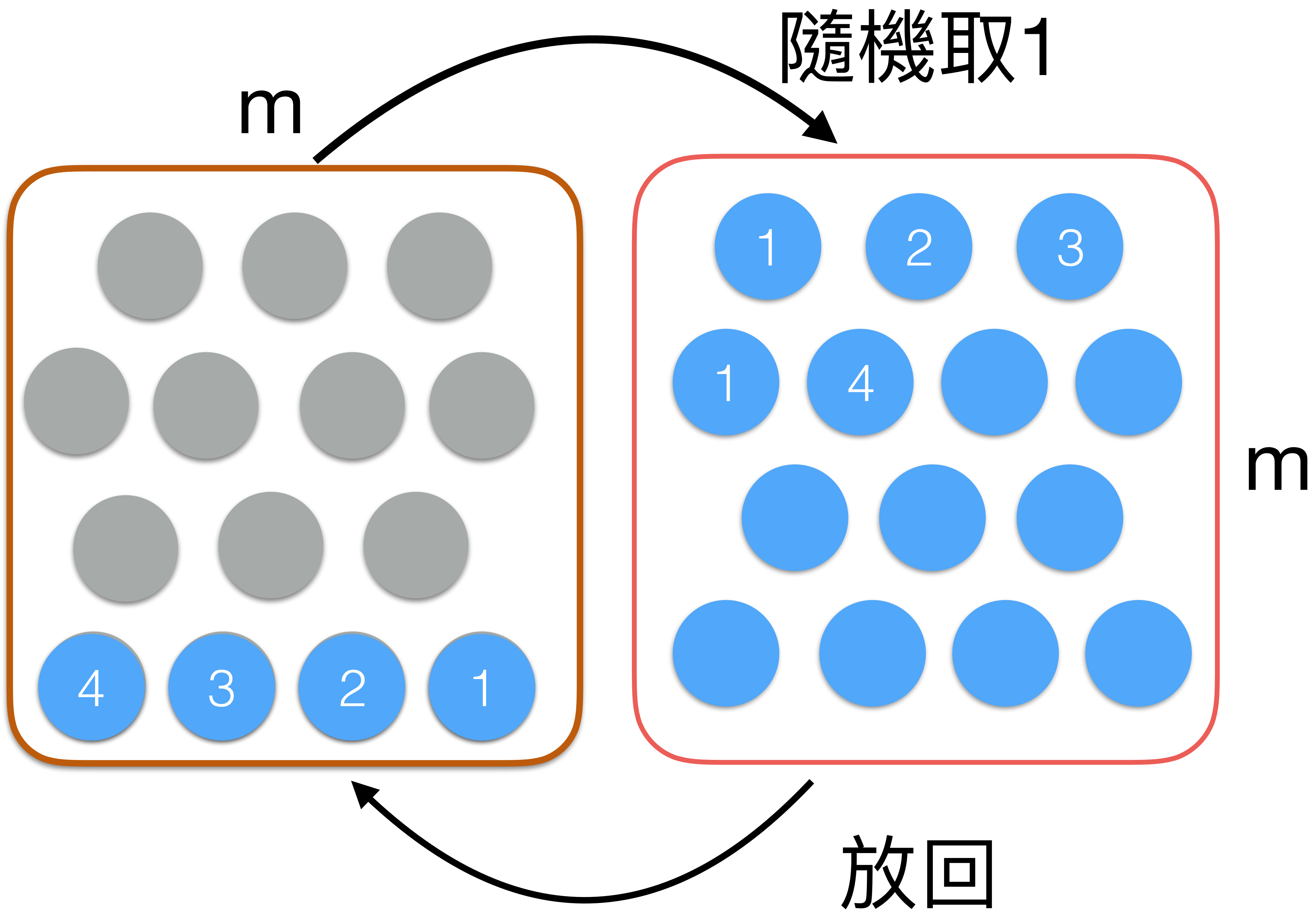
- *class* `sklearn.ensemble.`**`VotingClassifier`**(*estimators*, *voting='hard'*, *weights=None*, *n_jobs=1*, *flatten_transform=None*)
  - estimators: classifiers (命名, clf) tuple list
    - e.g. [('KNN', clf1), ('NB', clf2), ('LogisticRegression', clf3)]
  - voting: 'hard' (票票等值)、'soft' (票票不等值) with weights
  - weights: weight list
    - e.g. [1, 2, 5]



装袋法
Bagging

Bootstrap

- Bagging (Bootstrap Aggregating)
- Bootstrap (sample): “自助”隨機產生訓練資料



Bootstrap

- 訓練資料集中可能包含相同資料。
- 產生各筆資料隨機重要性，抗噪(noise)能力較好、較不易overfitting
    - 被抽中的機率是1/m，沒被抽中的機率是(1-1/m)
    - A完全沒被抽中的機率: $(1-\frac{1}{m})^m$
    - 樣本數很大時，任一樣本被抽中的機率是

$$1-\lim_{m\to\infty}(1-\frac{1}{m})^m = 1-\frac{1}{e} \approx 1-\frac{1}{2.71828} \approx 0.632$$

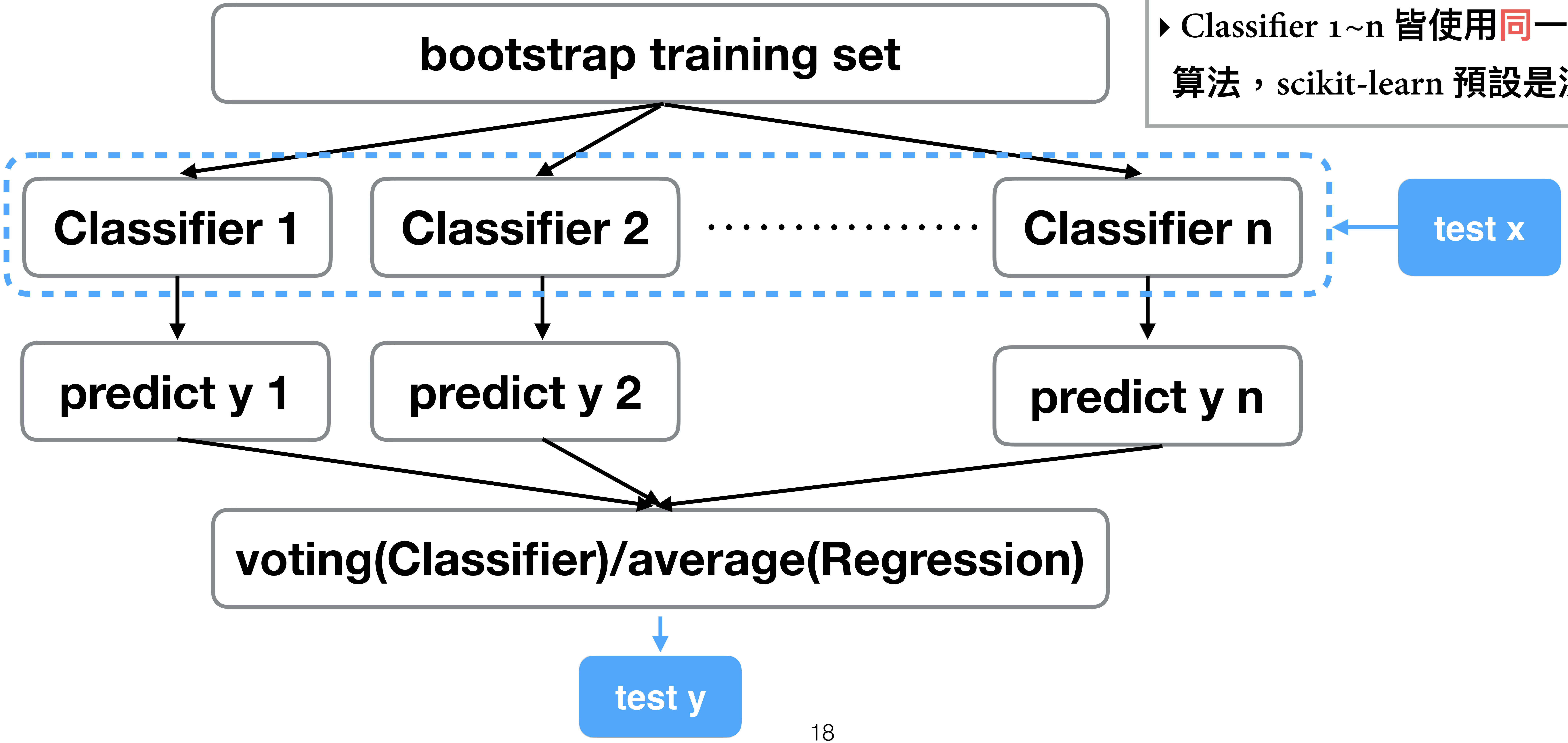
OOB (Out of Bag)

- OOB (Out of Bag): 沒被取到的資料

    - 當資料量大時，約有1/3資料沒被取到，可作為test set使用

$$\frac{1}{e} \approx \frac{1}{2.71828} \approx 0.368 \approx \frac{1}{3}$$

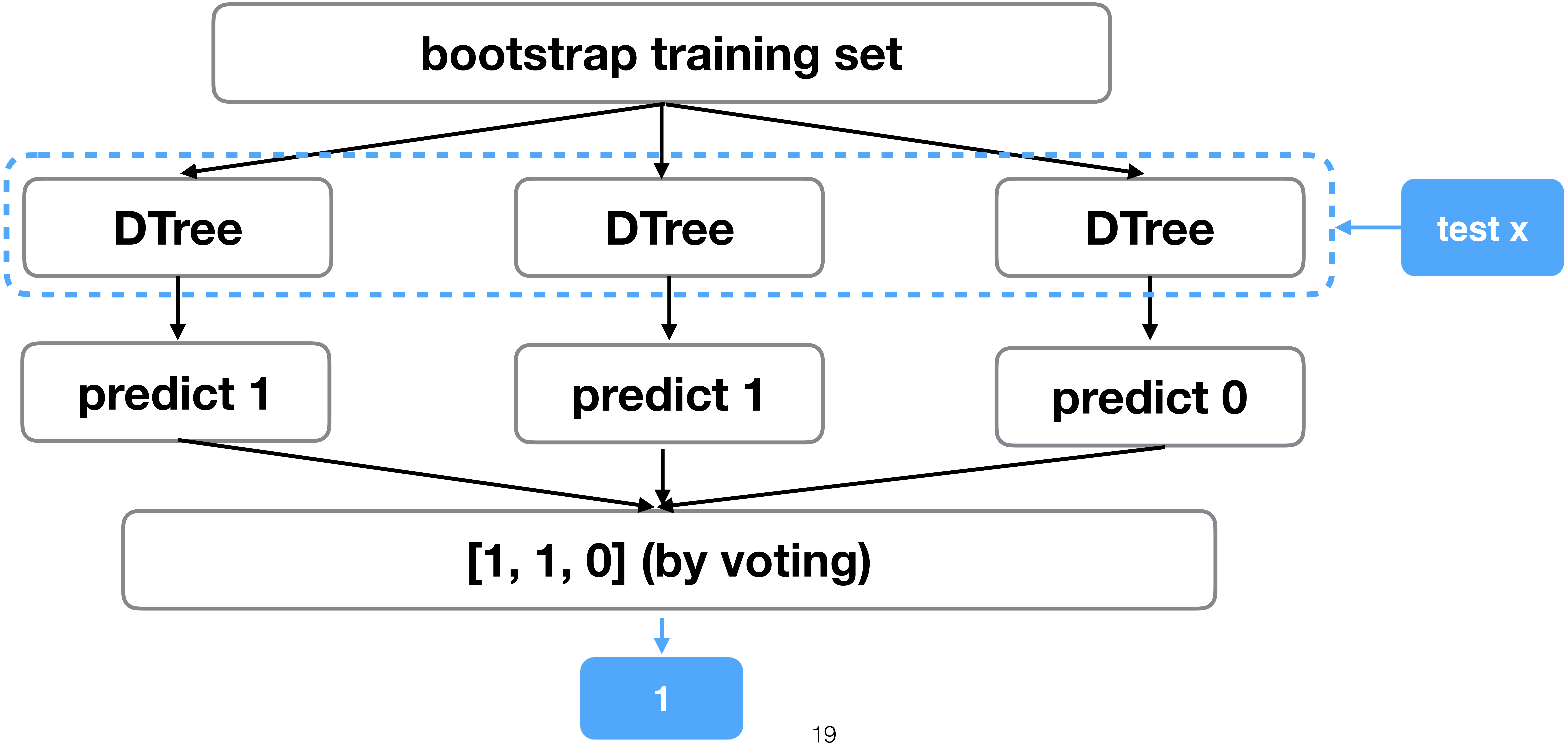
Bagging



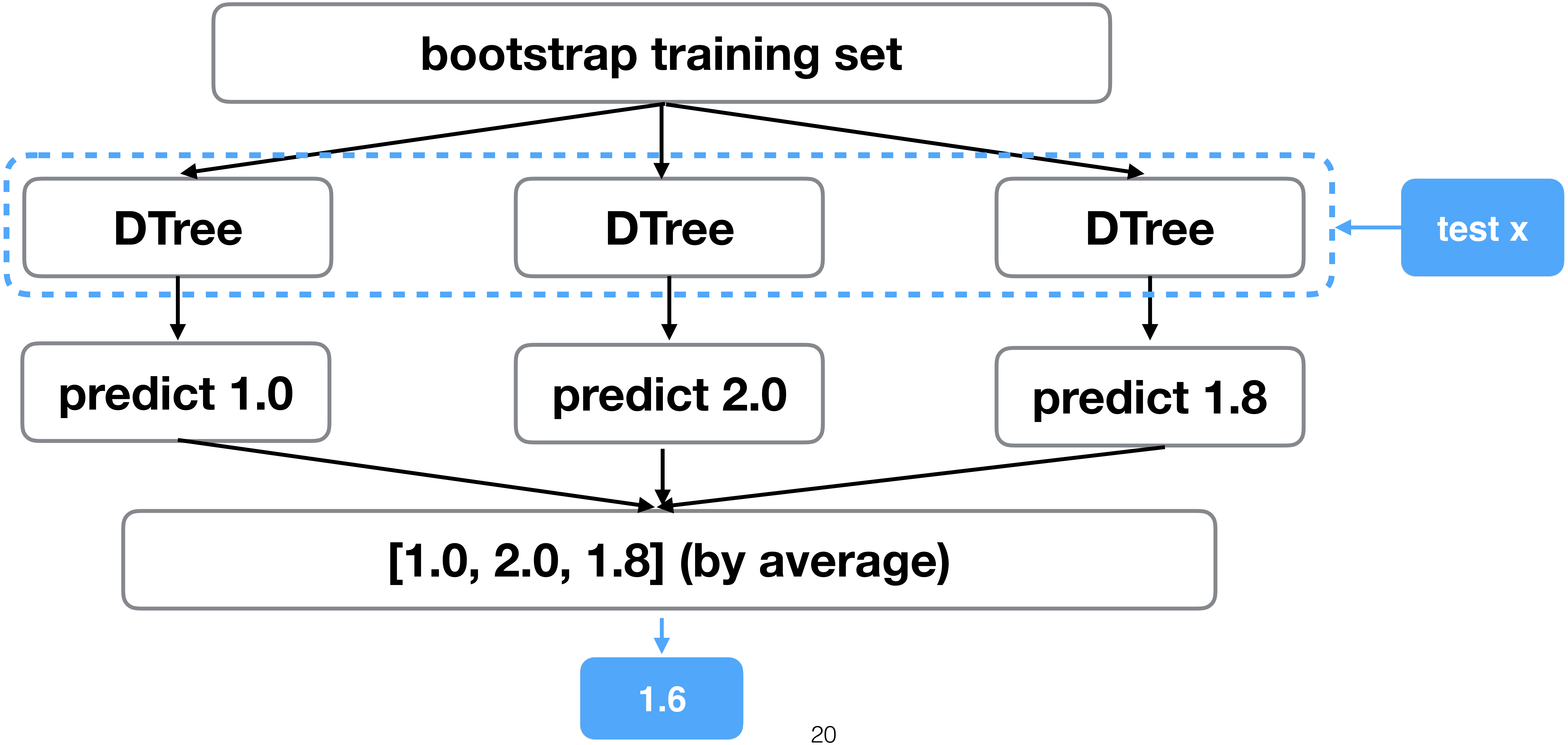
Notes

- Classifier 1~n 皆使用同一種學習演算法，scikit-learn 預設是決策樹。

Bagging Classifier



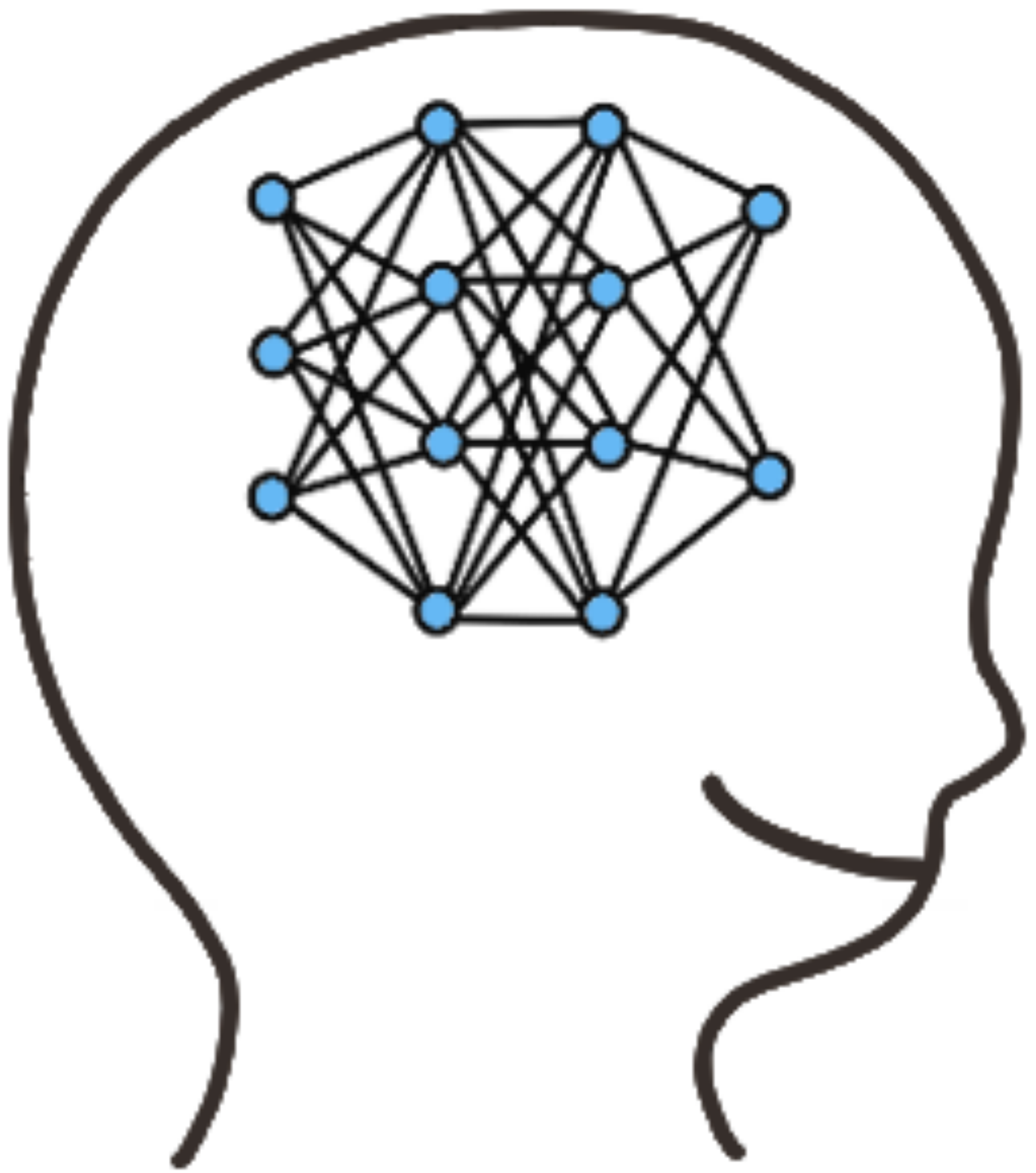
Bagging Regressor



BaggingClassifier/Regressor with sklearn

- *class* `sklearn.ensemble.`**`BaggingClassifier`**(*base_estimator=None*, *n_estimators=10*, *max_samples=1.0*, *max_features=1.0*, *bootstrap=True*, *bootstrap_features=False*, *oob_score=False*, *warm_start=False*, *n_jobs=1*, *random_state=None*, *verbose=0*)
- *class* `sklearn.ensemble.`**`BaggingRegressor`**(*base_estimator=None*, *n_estimators=10*, *max_samples=1.0*, *max_features=1.0*, *bootstrap=True*, *bootstrap_features=False*, *oob_score=False*, *warm_start=False*, *n_jobs=1*, *random_state=None*, *verbose=0*)
  - base_estimator: estimator object
  - n_estimators: estimator數量
  - oob_score: 是否要使用oob計算score(True/False)
  - max_features: 最多使用的訓練特徵量，int 定值、float 比例

- attributes:
  - *oob_score_ : (float)*



隨機森林
Random Forest

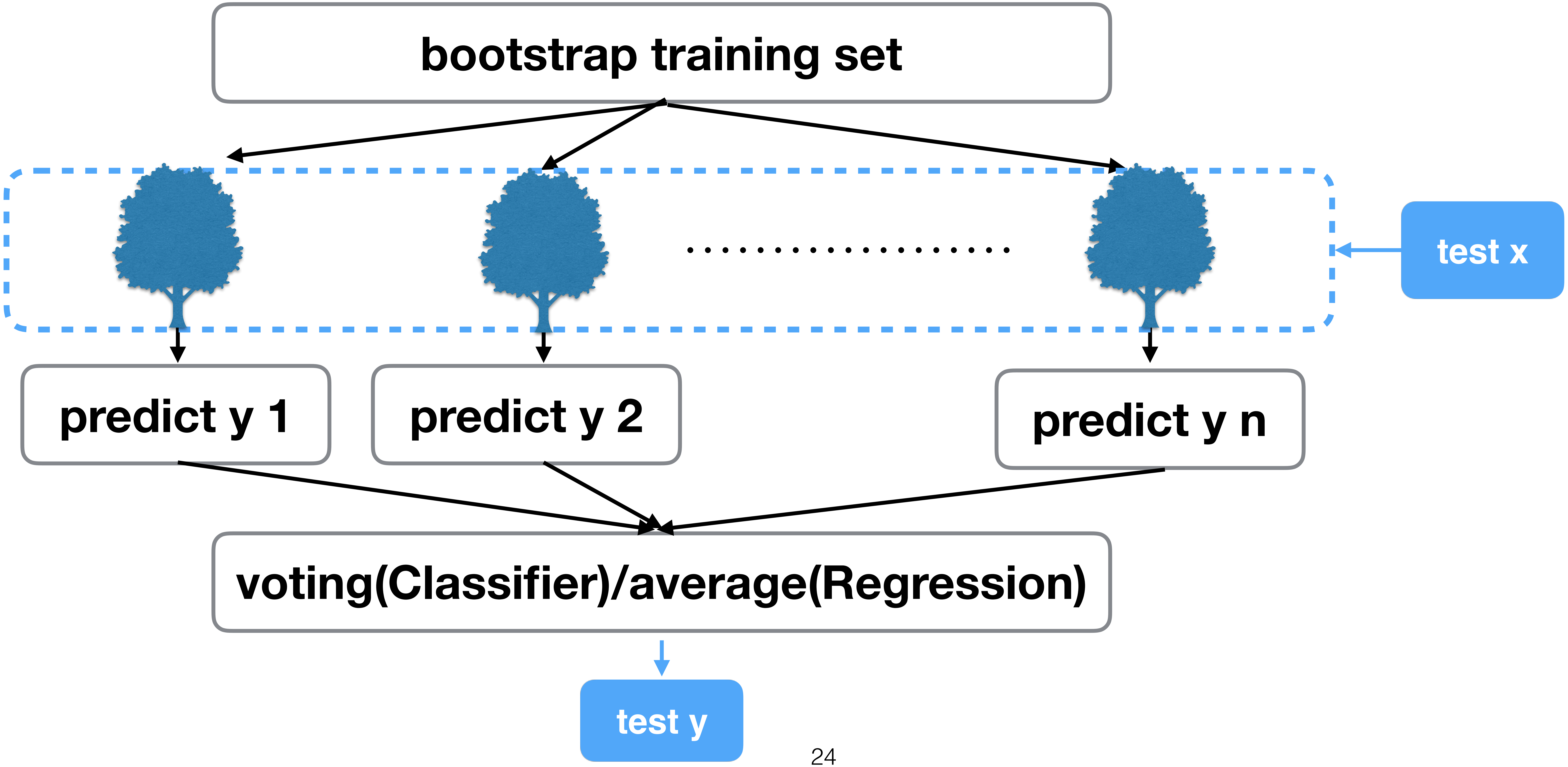
Random Forest

- Random: 隨機挑選特徵做訓練(常用值：特徵數量的平方根)
  - 可以處理高維特徵，不需要降維
- Forest: 訓練多棵決策樹作為weak learners
  - 每棵樹皆完全生長

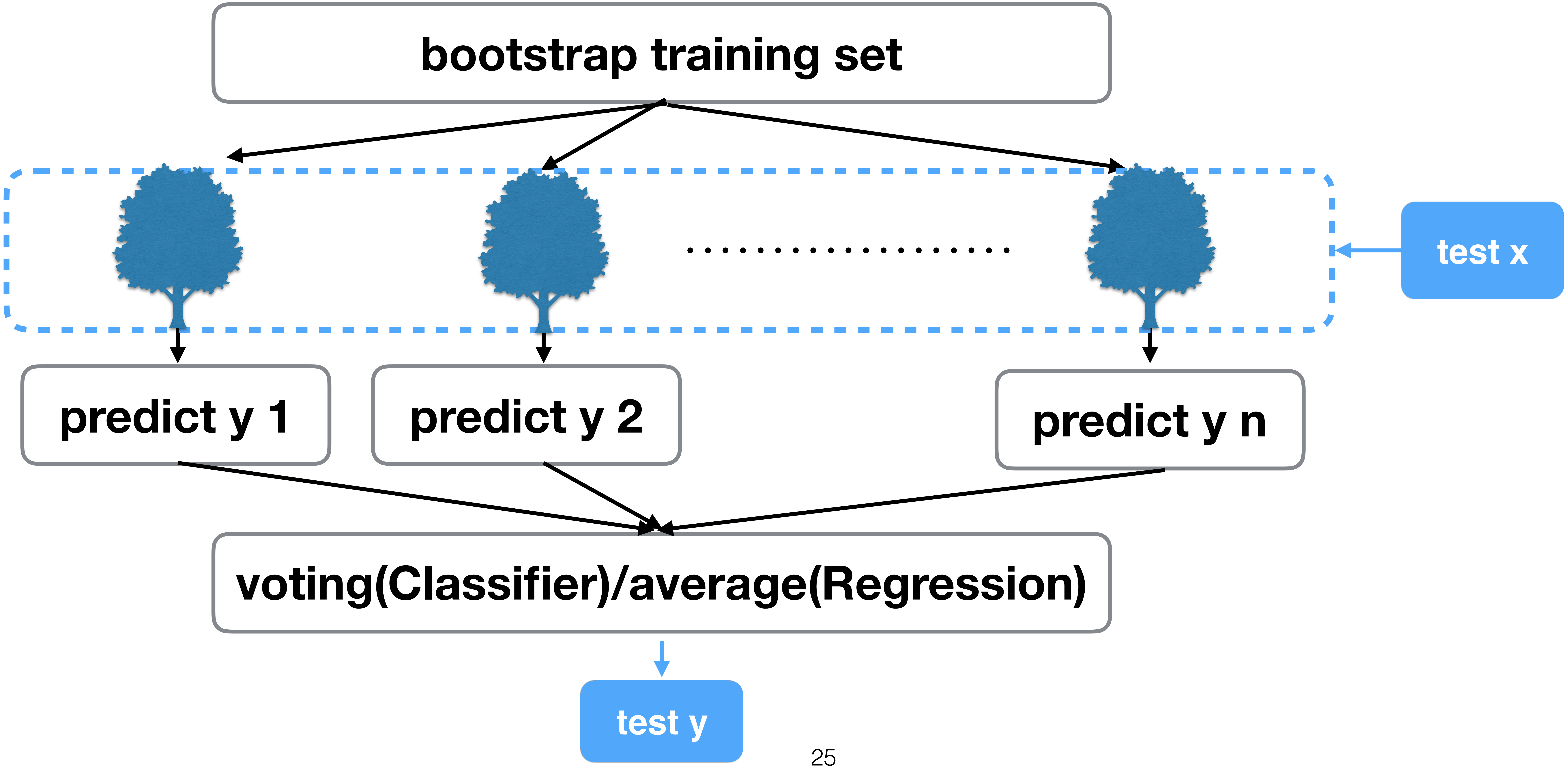
Notes

- Random Forest 決策原理同 Bagging。

Random Forest



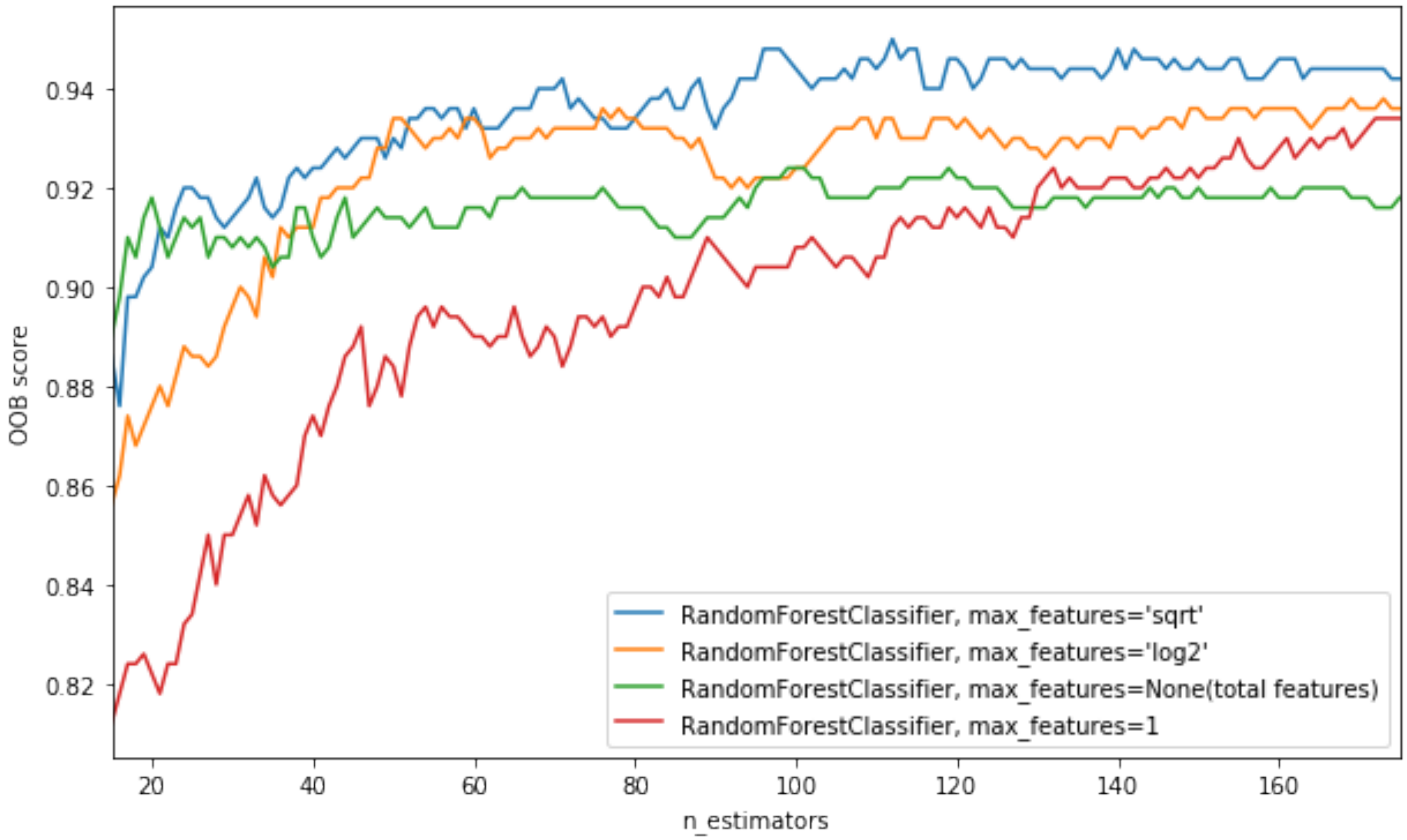
Random Forest



決策樹效果

- 鳶尾花資料集(Iris Dataset) 使用單一特徵完全生長的決策樹測試準確度
  - f1-score
    - 花萼長度: 0.66
    - 花萼寬度: 0.51
    - 花瓣長度: 0.91
    - 花瓣寬度: 0.98

OOB score



OOB score
0.94
0.92
0.90
0.88
0.86
0.84
0.82
20
40
60
80
100
120
140
160
n_estimators
RandomForestClassifier, max_features='sqrt'
RandomForestClassifier, max_features='log2'
RandomForestClassifier, max_features=None(total features)
RandomForestClassifier, max_features=1

優缺點

- 優點：
  - 能處理高維特徵，不用篩選特徵或降維
  - 兩個隨機性(bootstrap+select features)降低雜訊(noise)影響，避免overfitting
  - 分類預測力強，因多數決的關係，不好樹的預測彼此抵銷，由好的樹影響預測結果
- 缺點：
  - 黑盒子，難以控制模型內部運作
  - 迴歸效果較差(因為average會受不好的樹影響)

Random Forest with sklearn

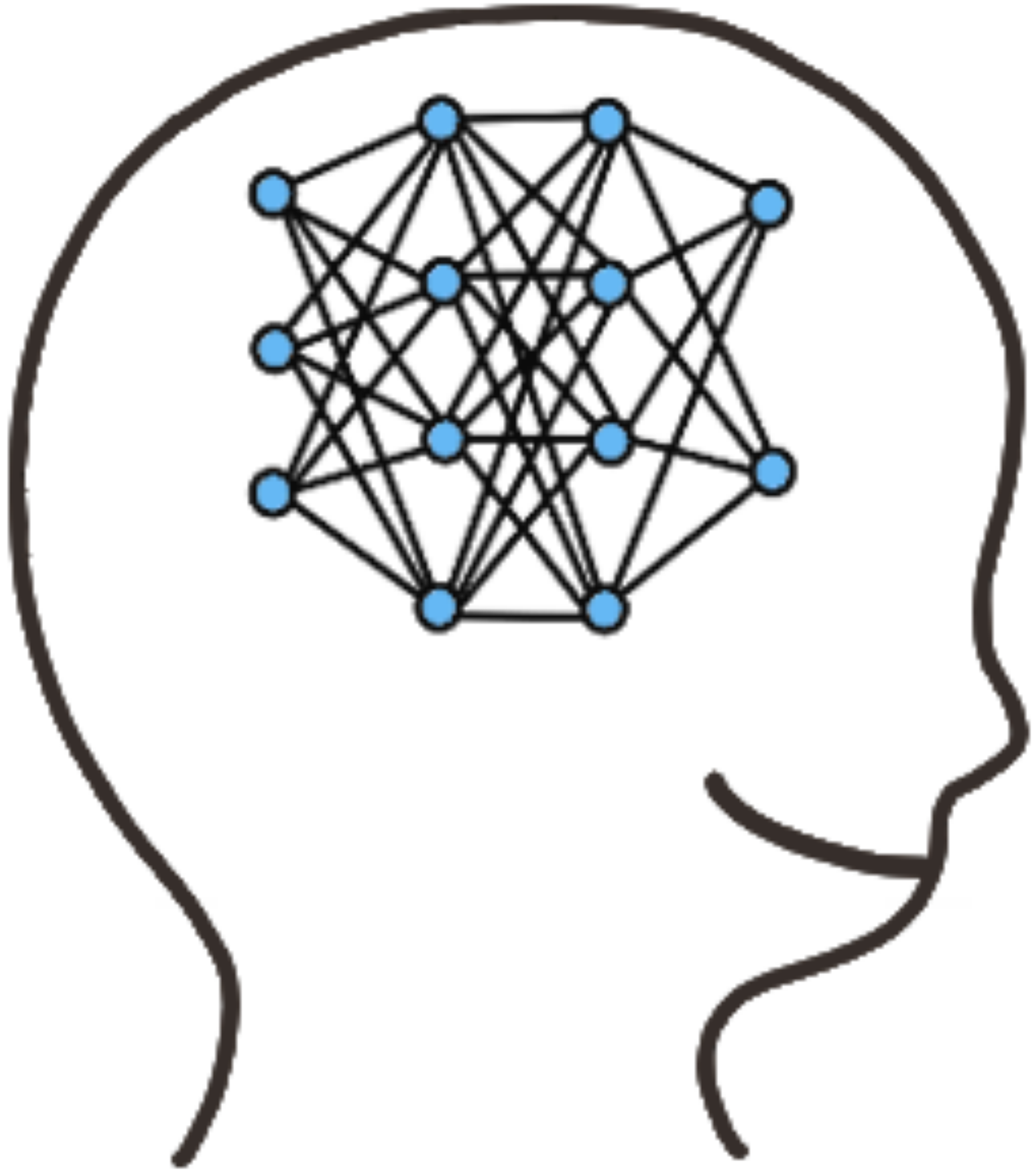
- *class* `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=10*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*)
  - n_estimators: estimator數量
  - criterion: 'gini' or 'entropy'
  - max_feautres: 'auto' (=sqrt(n_features))
  - oob_score: 是否要使用oob計算score(True/False)

- attributes:
  - *oob_score_ : (float)*
  - feature_importances_: 特徵重要性

Random Forest with sklearn

- *class* `sklearn.ensemble.`**`RandomForestRegressor`**(*n_estimators=10*, *criterion='mse'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*)
  - n_estimators: estimator數量
  - max_feautres: 'auto' (=sqrt(n_features))
  - oob_score: 是否要使用oob計算score(True/False)

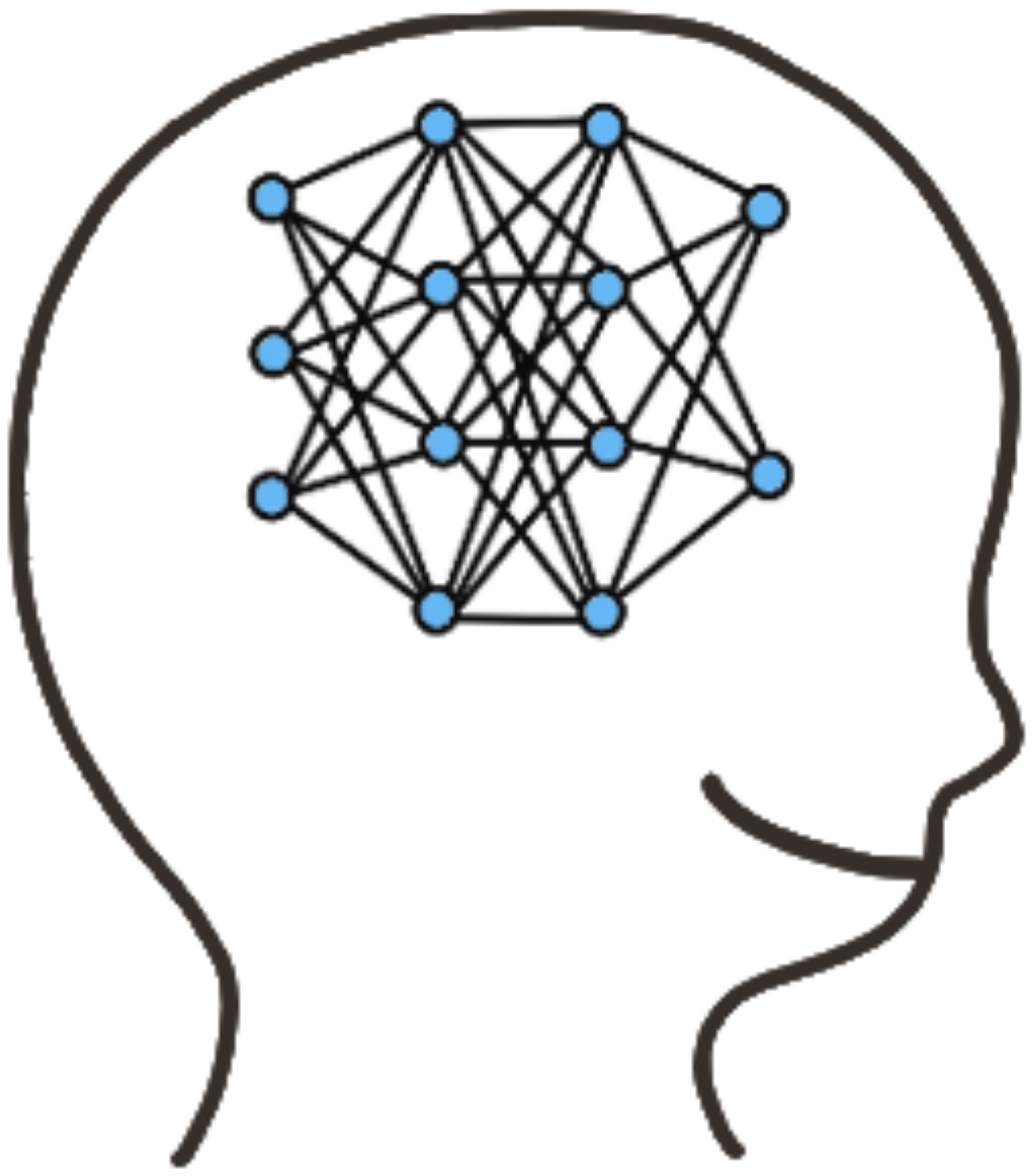
- attributes:
  - *oob_score_ : (float)*
  - feature_importances_: 特徵重要性



強化法
Boosting

原理

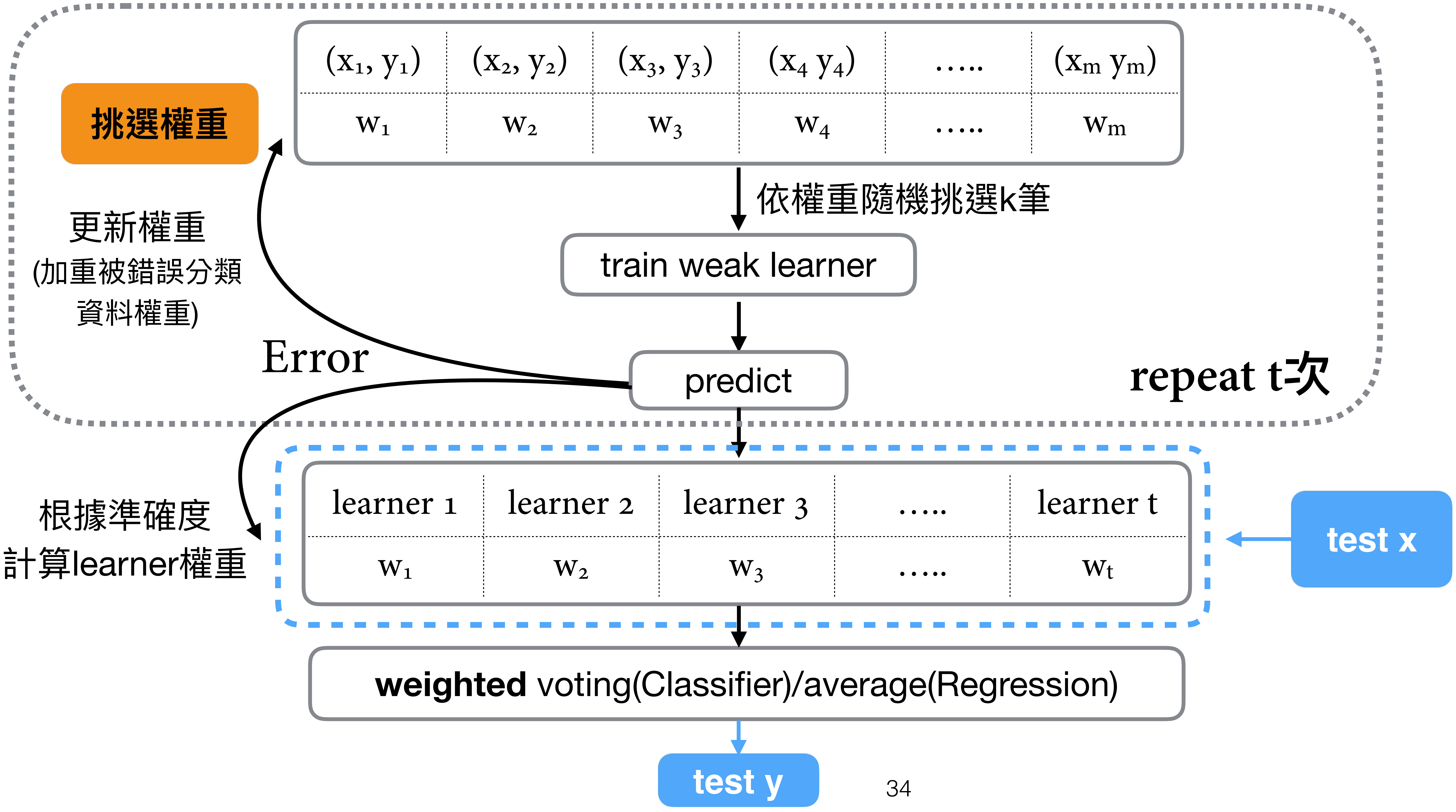
- 全班同學成績有高有低，如果我想要讓全班同學的整體成績提升，應該怎麼做？
  - 學生：加強關照成績較低的學生
  - 老師：找比較好的老師來教



AdaBoost

Adaptive Boosting

AdaBoost



(x_1, y_1) | (x_2, y_2) | (x_3, y_3) | $(x_4\ y_4)$ | ….. | $(x_m\ y_m)$
w_1 | w_2 | w_3 | w_4 | ….. | w_m
挑選權重
依權重隨機挑選k筆
train weak learner
predict
更新權重
(加重被錯誤分類
資料權重)
Error
repeat t次
learner 1 | learner 2 | learner 3 | ….. | learner t
w_1 | w_2 | w_3 | ….. | w_t
根據準確度
計算learner權重
test x
weighted voting(Classifier)/average(Regression)
test y

AdaBoost Algorithm

- m筆訓練資料：權重起始值為 1/m
- 加權**錯誤**率 $\varepsilon_t = \sum_{i:\hat{y}_i \neq y_i} w \cdot i$
  - e.g. 5筆資料，其中2筆預測**錯誤**(1)
  - $\varepsilon_t = 0.2 \times 1 + 0.2 \times 1 = 0.4$
- $\alpha_t = \frac{1}{2} \ln \cdot \frac{1-\varepsilon_t}{\varepsilon_t}$

$$\alpha_t = \frac{1}{2} \cdot \ln \frac{0.6}{0.4} \approx 0.203$$

AdaBoost Algorithm(cont.)

- 更新資料權重
  - $w = w \cdot e^{(-\alpha_t \cdot \hat{y} \cdot y)}$
  - $y_m \in \{1, -1\}$
    - **預測正確：(權重降低)**

$$w_m = 0.2 \cdot \mathrm{e}^{(-0.203 \cdot 1 \cdot 1)} \approx 0.2 \cdot 0.816 \approx 0.163$$

    - **預測錯誤：(權重提高)**

$$w_m = 0.2 \cdot \mathrm{e}^{(-0.203 \cdot 1 \cdot -1)} \approx 0.2 \cdot 1.225 \approx 0.245$$

  - 調整權重加總為1: $w = \dfrac{w}{\sum_m w_m}$

AdaBoost Algorithm(cont.)

- 依照更新的資料權重再抽樣(重複t次，產生t個learner)

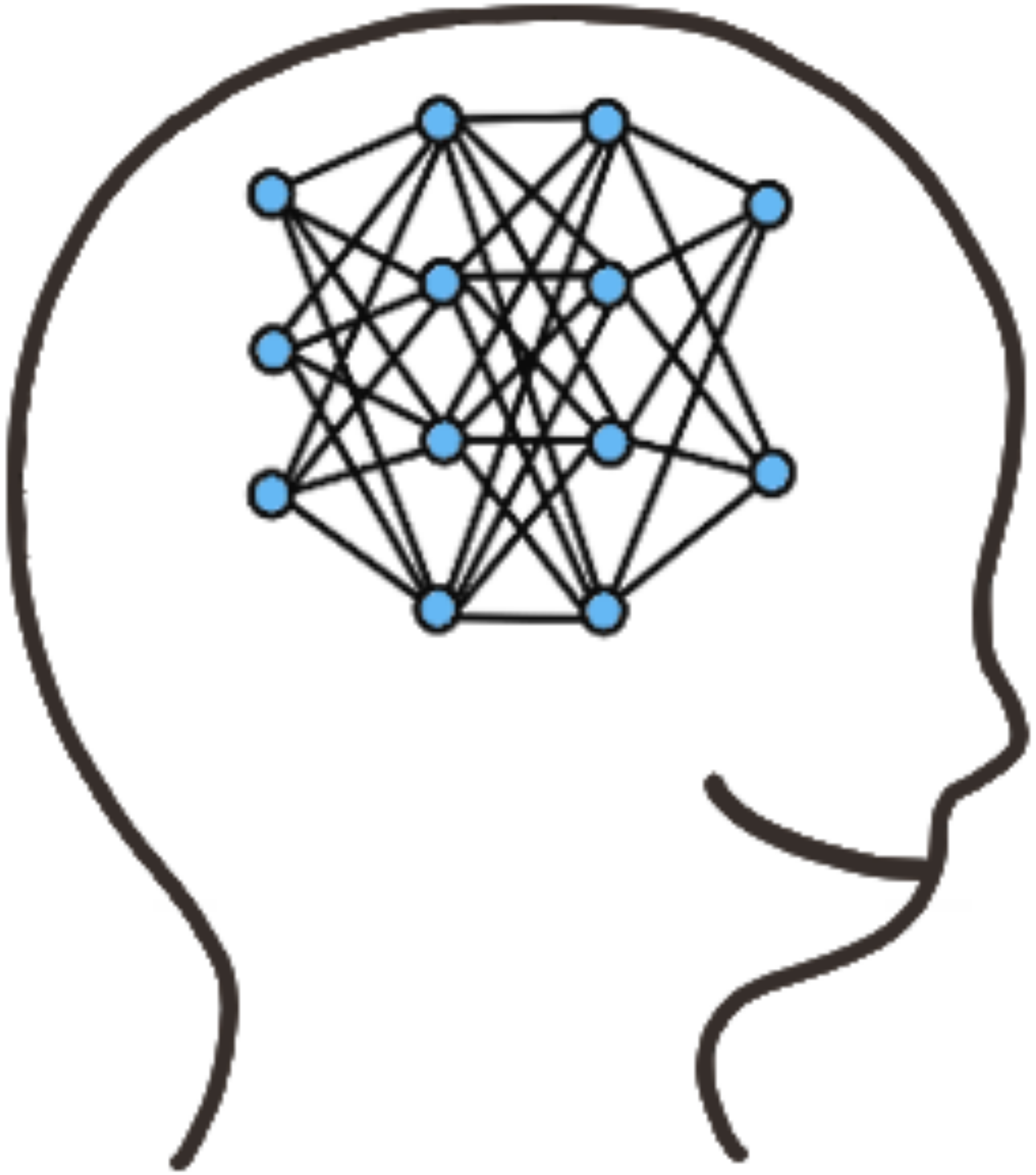
- 最後預測：
  - learner weights: $\alpha_t = \frac{1}{2} \ln \cdot \frac{1-\varepsilon_t}{\varepsilon_t}$
  - Classifier: **weighted** voting
  - Regressor: **weighted** average

AdaBoost with sklearn

- *class* `sklearn.ensemble.`**`AdaBoostClassifier`**(*base_estimator=None*, *n_estimators=50*, *learning_rate=1.0*, *algorithm='SAMME.R'*, *random_state=None)*
- *class* `sklearn.ensemble.`**`AdaBoostRegressor`**(*base_estimator=None*, *n_estimators=50*, *learning_rate=1.0*, *loss='linear'*, *random_state=None*)
  - base_estimator: estimator object (預設'None': 決策樹)
  - *n_estimators:* estimator 數量

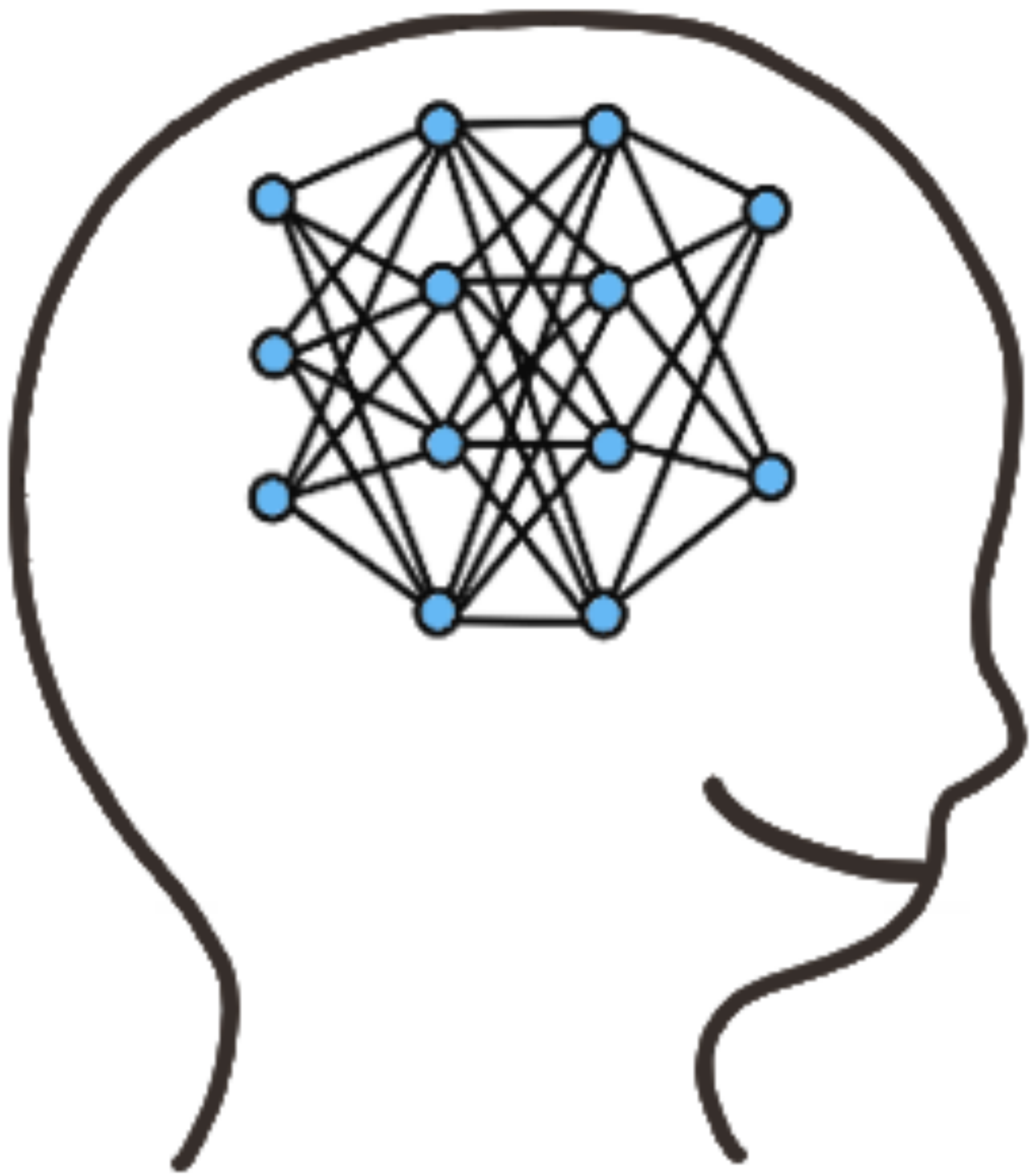
- attributes:
  - estimator_errors_: 各estimators錯誤
  - estimator_*weights*_ : 各estimators weights
  - feature_importances_: 特徵重要性(by 決策樹)

- score function:
  - **`score`**(*X, y[, sample_weight]*)



整體學習(二)

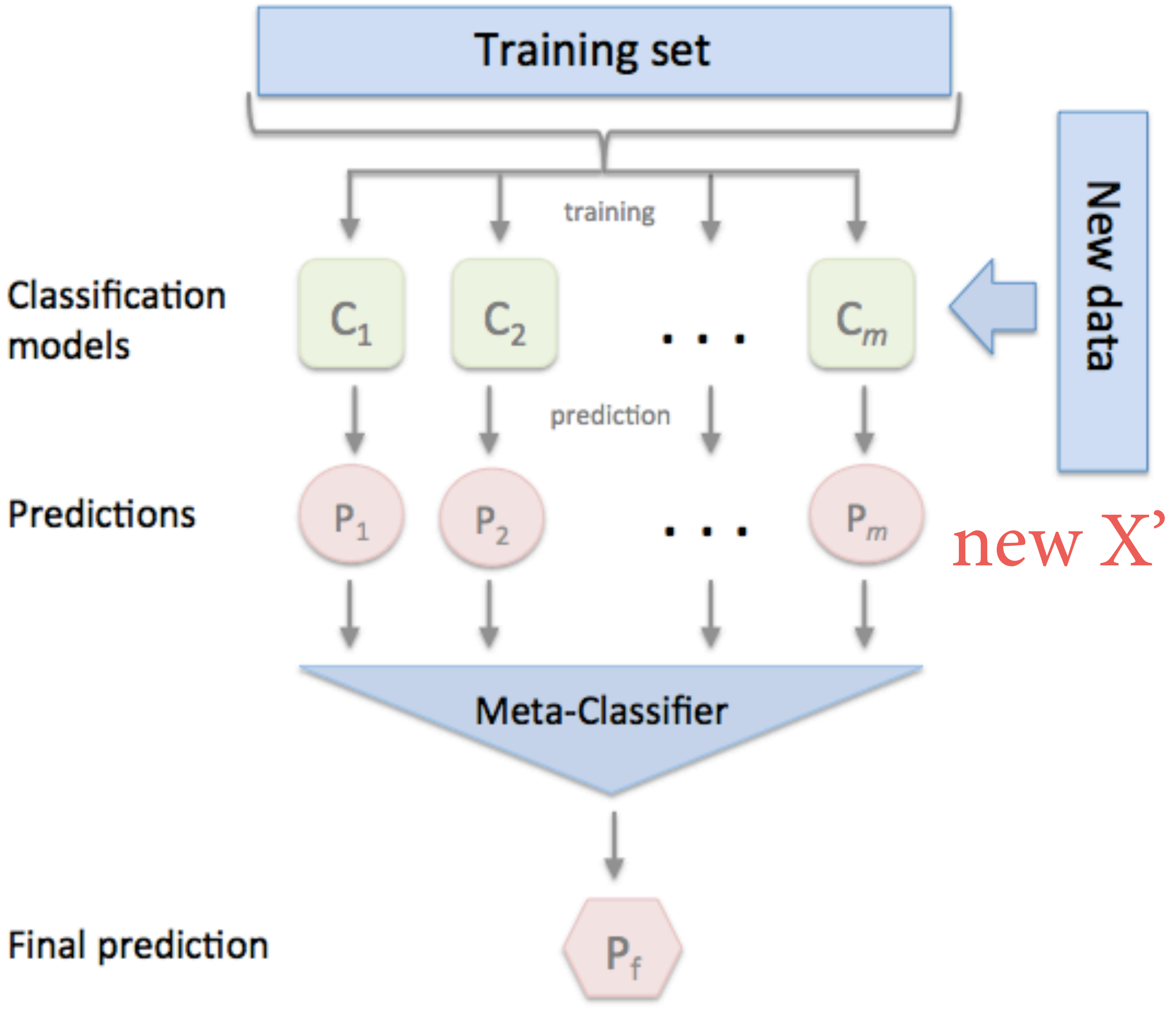
Ensemble Learning (2)



MLXTEND

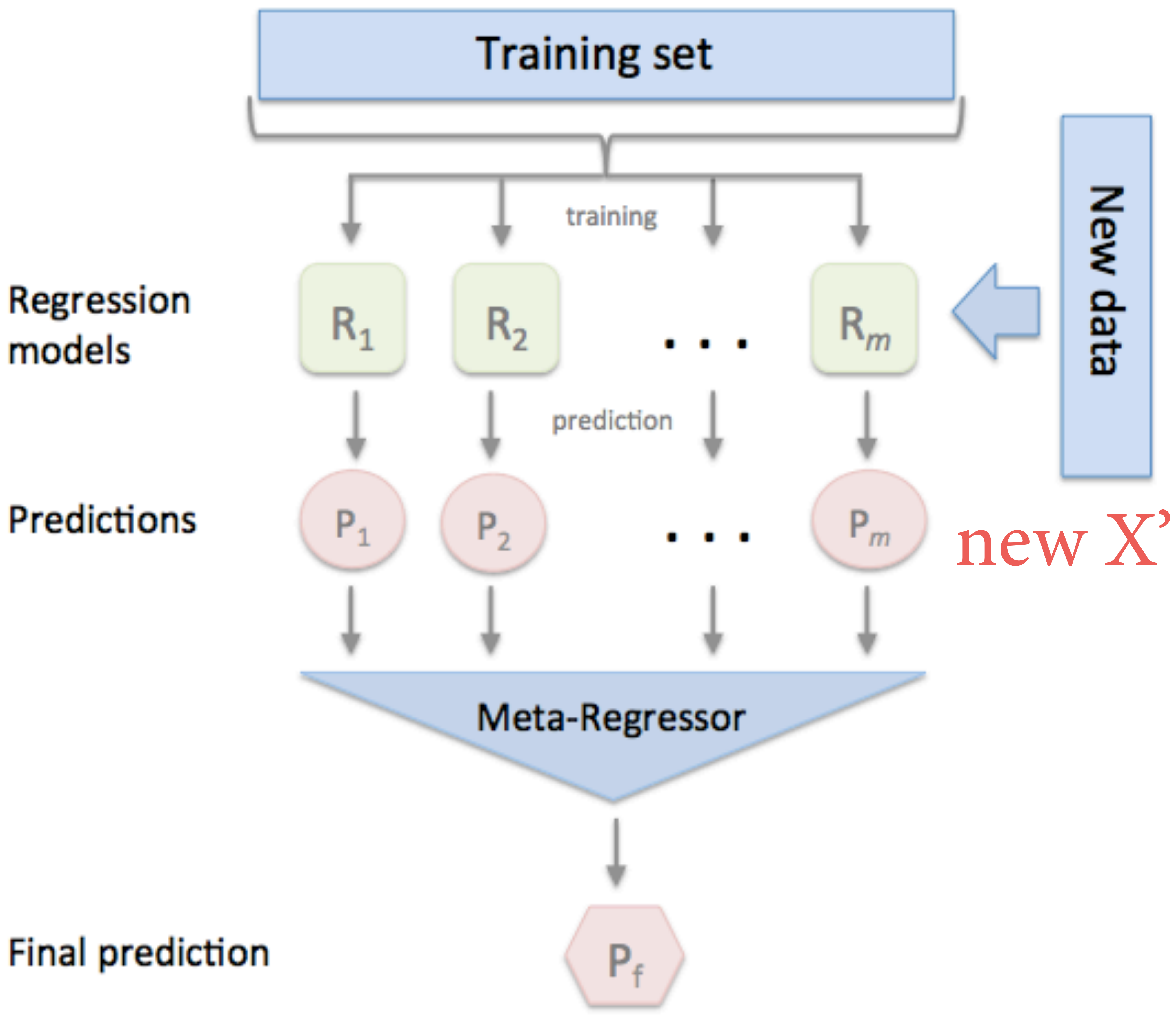
Stacking

Stacking Classifier

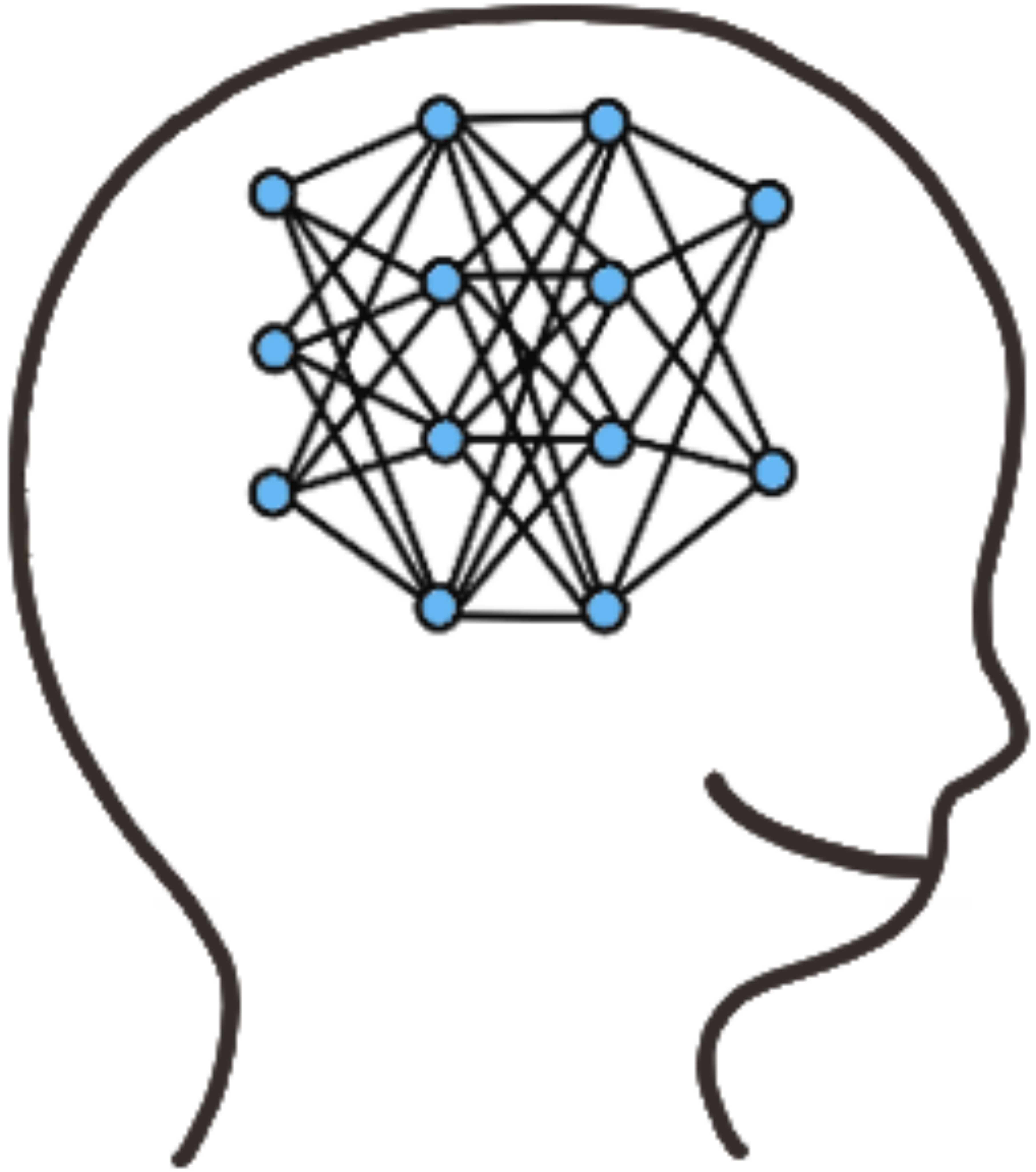


(from: mlxtend)

Stacking Regressor



(from: mlxtend)



XGBoost

AdaBoost

- update (after each iteration):
  - weights of learners
  - weights of samples

梯度提升決策樹

- the same as:
  - GBDT (Gradient Boost Decision Tree)
  - MART (Multiple Additive Regression Tree)
  - GBRT (Gradient Boost Regression Tree)

Boosting

- CART樹的加法(additive)模型
    - 不論是用於迴歸或分類，皆採用迴歸樹
    - 具備自動判斷重要特徵並組合之效果

Boosting

- 累加式訓練 (Additive Training)

$$
\begin{aligned}
\hat{y}_i^{(0)} &= 0 \\
\hat{y}_i^{(1)} &= f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i) \\
\hat{y}_i^{(2)} &= f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i) \\
&\dots \\
\hat{y}_i^{(t)} &= \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

這一回合的new model

上一回合的predict y

Note:

- 以下公式皆源自XGBoost網頁，請見References。

Cost Function

- Training Loss + Regularization

MSE/Cross Entropy L1/L2

$$\text{obj}^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i)$$

$$= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t)$$

這一回合的predict y

= 前一回合的predict y+這一回合的tree model

Cost Function

- MSE

$$\text{obj}^{(t)} = \sum_{i=1}^{n} (y_i - (\hat{y}_i^{(t-1)} + f_t(x_i)))^2 + \sum_{i=1}^{t} \Omega(f_i)$$

$$= \sum_{i=1}^{n} [2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2] + \Omega(f_t)$$

殘差(Residual)

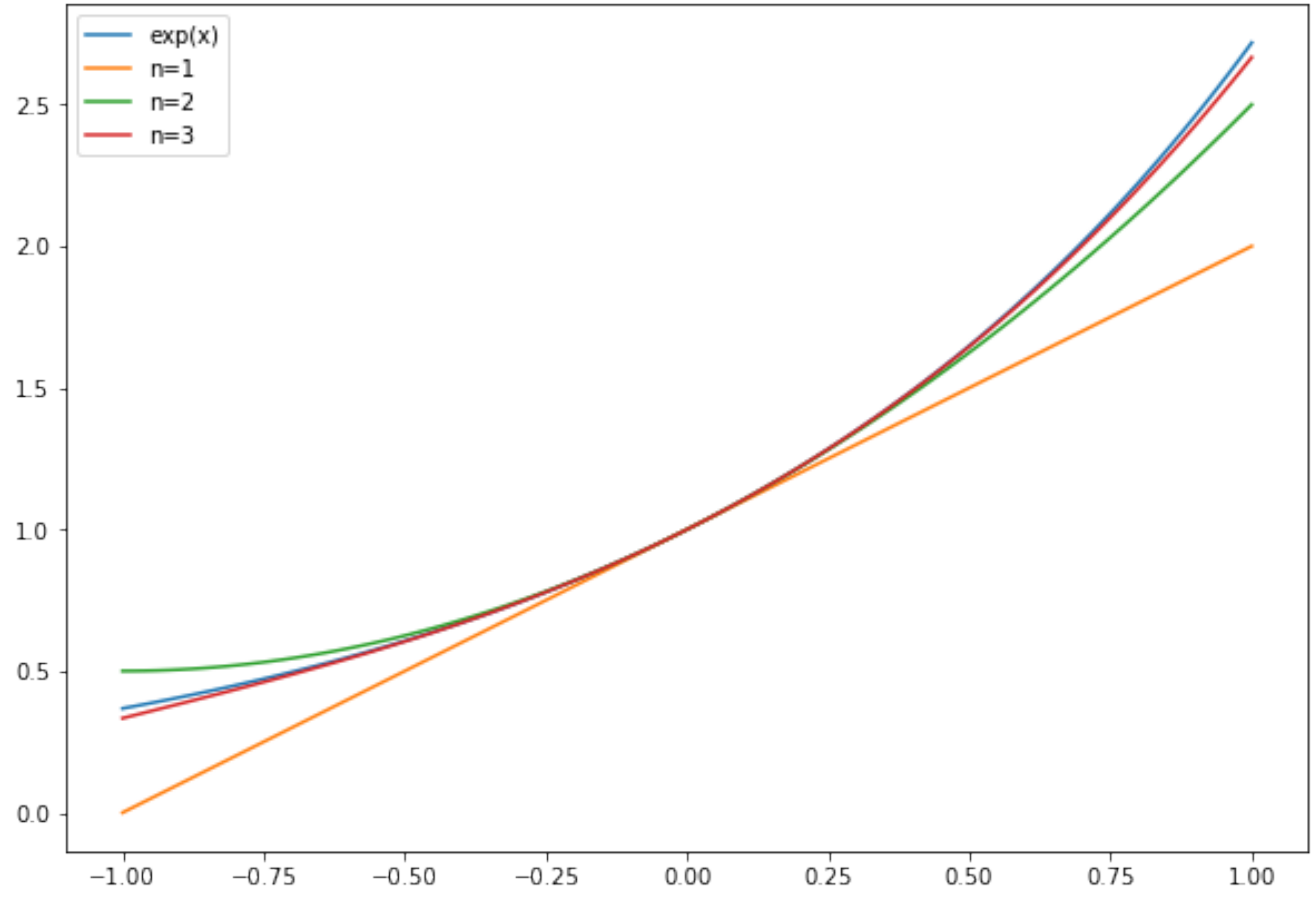
Taylor Expansion

- 泰勒展開式/級數 (Taylor Expansion/Series)
  - 使用目的：推估函數值
  - 使用前提：連續函數且可微分

$$f(x) = f(a) + f'(a)(x-a) + f''(a)\frac{(x-a)^2}{2!} + \cdots + f^{(n)}(a)\frac{(x-a)^n}{n!}$$

Taylor Expansion

for a=0, $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$



Cost Function

$$Obj^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$

$\downarrow$ Taylor Expansion

$$Obj^{(t)} \simeq \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t)$$

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}), \quad h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$$

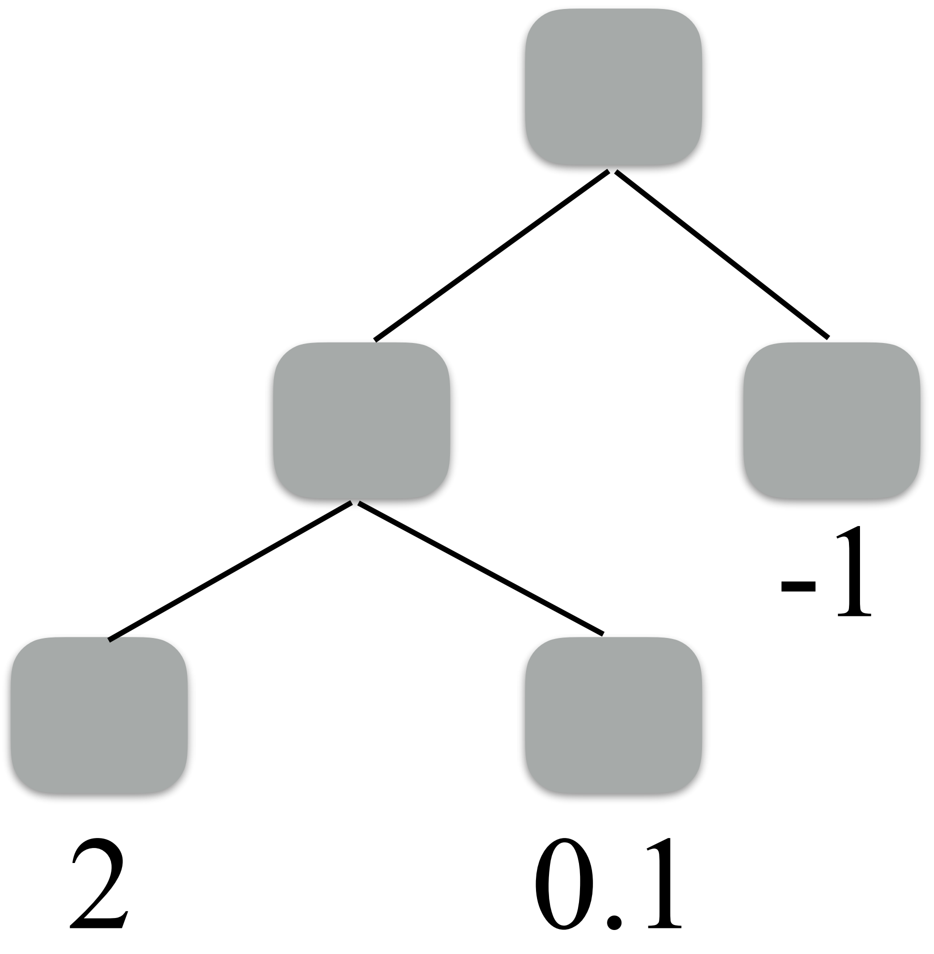
- MSE

$$g_i = \partial_{\hat{y}^{(t-1)}} (\hat{y}^{(t-1)} - y_i)^2 = 2(\hat{y}^{(t-1)} - y_i) \quad h_i = \partial^2_{\hat{y}^{(t-1)}} (y_i - \hat{y}^{(t-1)})^2 = 2$$

模型複雜度

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

leaf 數量　　　leaf 權重



$$\Omega(f) = \gamma(3) + \frac{1}{2}\lambda(2^2 + 0.1^2 + (-1)^2)$$

New Cost Function

$$Obj^{(t)} \approx \sum_{i=1}^{n}[g_i w_{q(x_i)} + \frac{1}{2}h_i w_{q(x_i)}^2] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T$$

- Optimization: 最大化IG (not Gradient Descent) for Splitting

GradientBoosting with sklearn

- class **`sklearn.ensemble.GradientBoostingClassifier`**(loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False, presort='auto')

- class **`sklearn.ensemble.GradientBoostingRegressor`**(loss='ls', learning_rate=0.1, n_estimators=100, subsample=1.0, criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0, min_impurity_split=None, init=None, random_state=None, max_features=None, alpha=0.9, verbose=0, max_leaf_nodes=None, warm_start=False, presort='auto')

What's XGBoost

- **Scalable**, **Portable** and **Distributed** Gradient Boosting (GBDT, GBRT or GBM) Library
    - 優化版GBDT：運算速度更快、準確率較高
    - website: http://xgboost.readthedocs.io/en/latest/
    - GitHub: https://github.com/dmlc/xgboost

References

- Tianqi Chen (2014)
  - slide: https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf
  - paper: https://homes.cs.washington.edu/~tqchen/pdf/xgboost-supp.pdf