PYTHON機器學習入門

UNIT 6：ENSEMBLE ALGORITHM

授課教師：江尚瑀

MODEL EVALUATION

CLASSIFICATION METRICS

CONFUSION MATRIX

- 以**CIFAR-10** 為例

Predicted

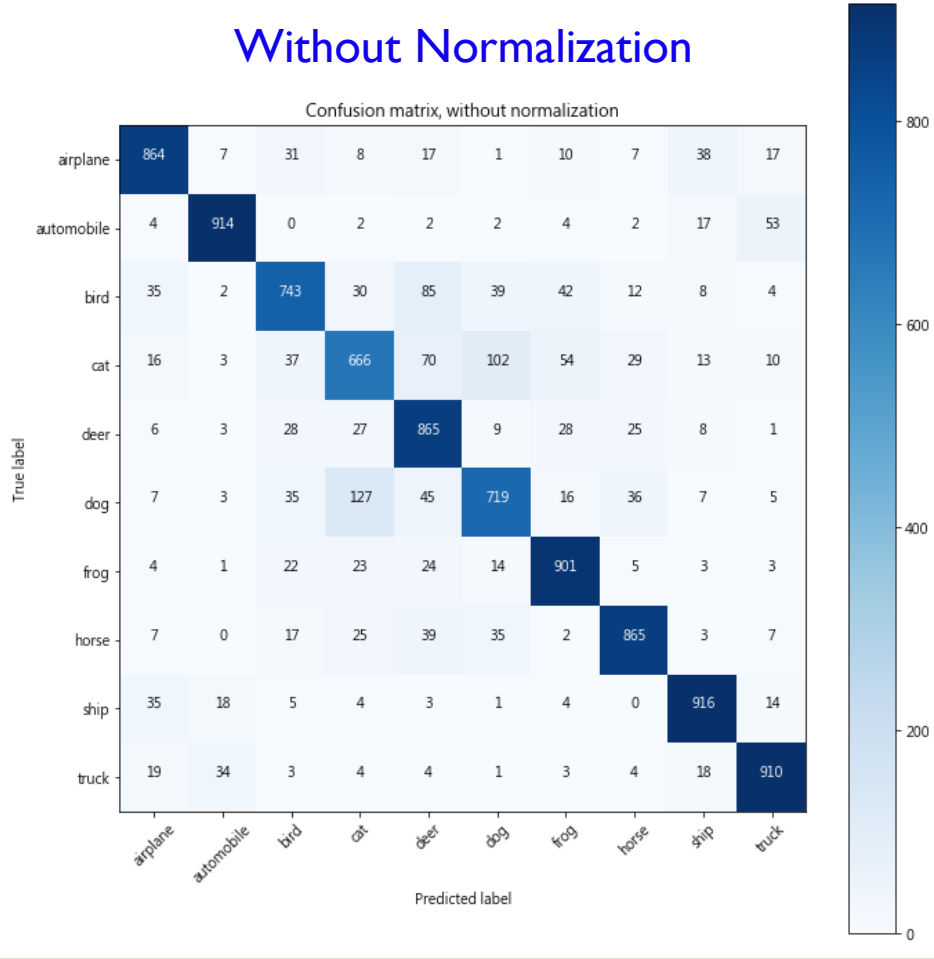
Actual

```
[[864   7  31   8  17   1  10   7  38  17]
 [  4 914   0   2   2   2   4   2  17  53]
 [ 35   2 743  30  85  39  42  12   8   4]
 [ 16   3  37 666  70 102  54  29  13  10]
 [  6   3  28  27 865   9  28  25   8   1]
 [  7   3  35 127  45 719  16  36   7   5]
 [  4   1  22  23  24  14 901   5   3   3]
 [  7   0  17  25  39  35   2 865   3   7]
 [ 35  18   5   4   3   1   4   0 916  14]
 [ 19  34   3   4   4   1   3   4  18 910]]
```

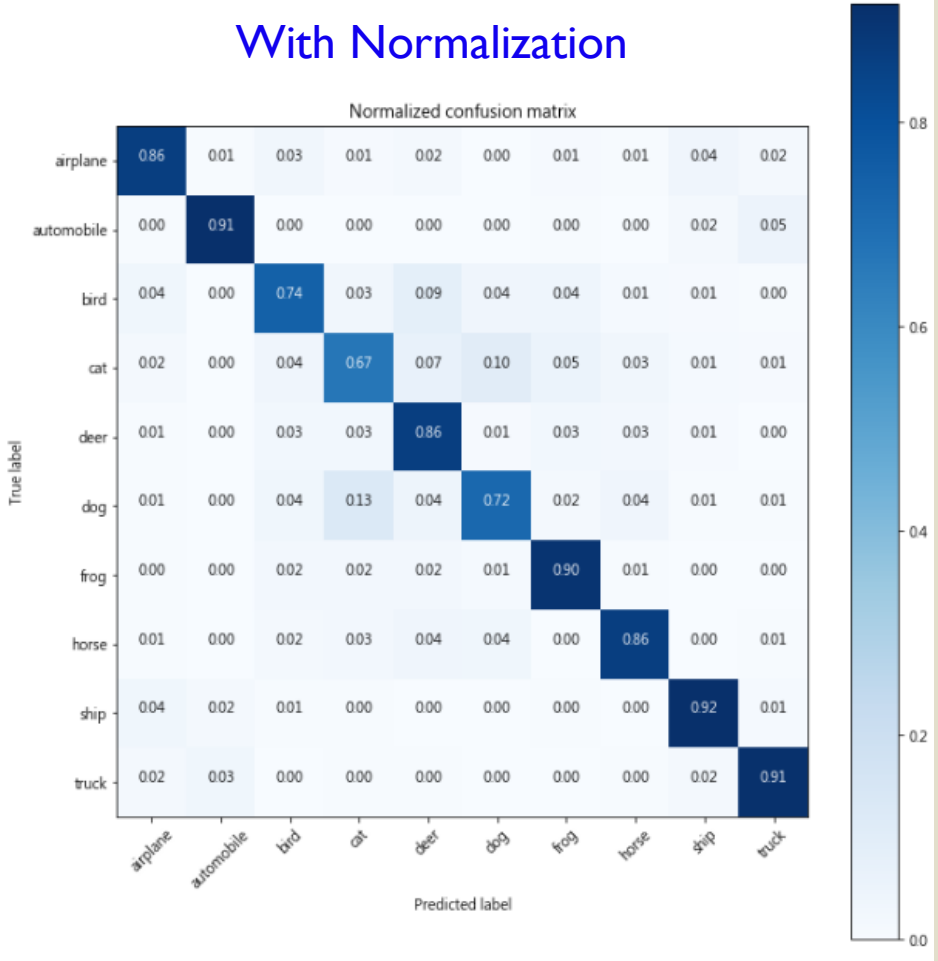
```python
from sklearn.metrics import confusion_matrix, precision_score, recall_score, f1_score, cohen_kappa_score
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
answer = np.argmax(y_test,axis=1)
predicted = np.argmax(y_pred,axis=1)

C=confusion_matrix(answer, predicted)
print(C)
```

CONFUSION MATRIX

Without Normalization



With Normalization



判斷模型預測能力的方法

- **Confusion Matrix** 混淆矩陣
- 通常為一個二分問題，將樣本分為正類與負類 **Positive/Negative實際狀況**
- 會根據預測結果 **True/False預測正確?**，以二分問題來說會出現四種情形

	實際Yes	實際No
預測Yes	**TP** (True Positive)	**FP** (False Positive)
預測No	**FN** (False Negative)	**TN** (True Negative)

判斷模型預測能力的方法

- **Precision(準確率) : TP/(TP+FP)**
  = positive predictive value
  → 用來判定當預測為正時的可信度

- **Recall(召回率) : TP/(TP+FN)**
  = sensitivity = true positive rate
  → 用來判定有多少的正例被檢測出

- **F1-score : 2 * Precision * Recall / (Precision + Recall)**
  = the harmonic mean of precision and recall
  → 算是一個比較概略的指標來看這個模型的表現

	實際Yes	實際No
預測Yes	TP (True Positive)	FP (False Positive)
預測No	FN (False Negative)	TN (True Negative)

EXAMPLE

池塘有1400條鯉魚，300隻蝦，300隻鱉。現在以捕鯉魚為目的。撒網，逮著了700條鯉魚，200隻蝦，100隻鱉。那麼，這些指標分別如下：

Precision = 700 / (700 + 200 + 100) = 70%

Recall = 700 / 1400 = 50%

F1值 = 70% * 50% * 2 / (70% + 50%) = 58.3%

- 若把池子裡的所有的鯉魚、蝦和鱉都一網打盡，這些指標又有何變化：

Precision = 1400 / (1400 + 300 + 300) = 70%

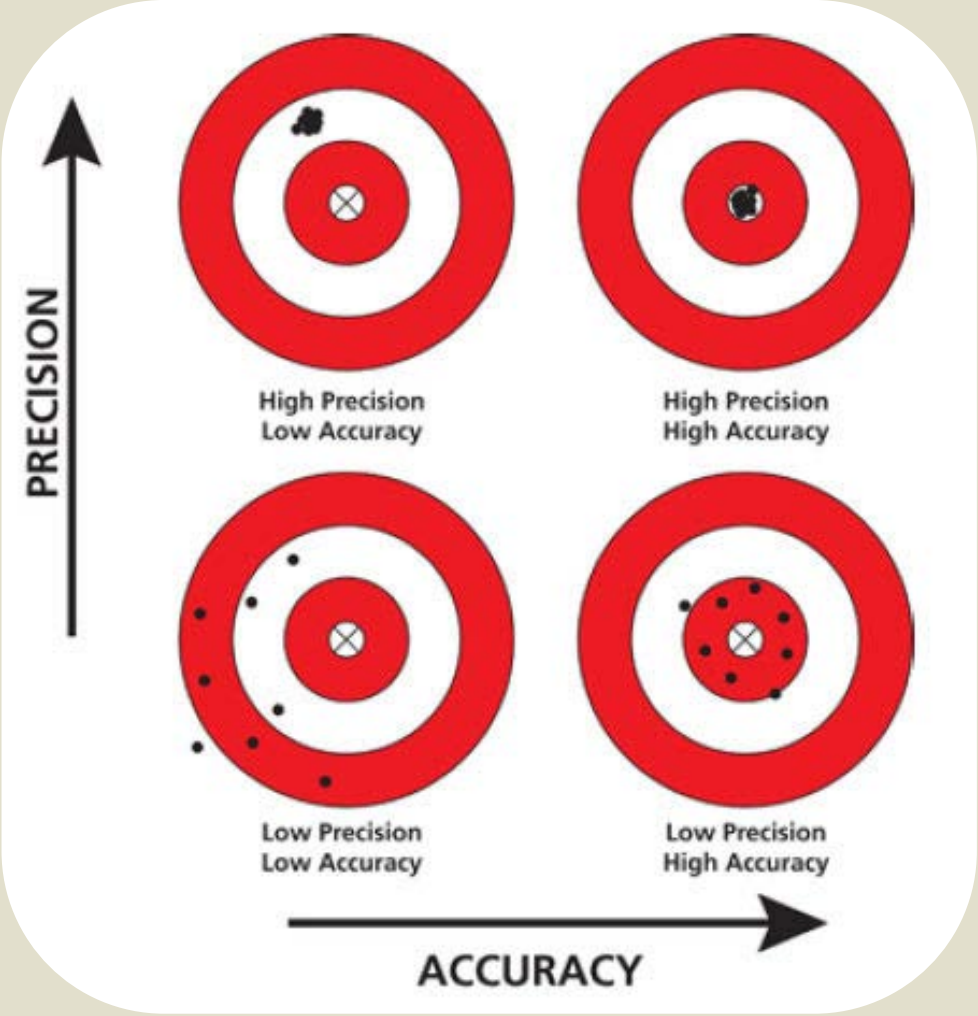
Recall = 1400 / 1400 = 100%

F1值 = 70% * 100% * 2 / (70% + 100%) = 82.35%

Precision : 是評估捕獲的成果中目標成果所占得比例
Recall : 是從關注領域中，召回目標類別的比例
F值 : 則是綜合這二者指標的評估指標，用於綜合反映整體的指標。

判斷模型預測能力的方法

- **Accuracy V.S. Precision**



判斷模型預測能力的方法

- **ROC 曲線(receiver operating characteristic curve)**
  - 以 **FPR** 為 **X** 軸，**TPR**為 **Y** 軸

FPR (False Positive Rate)偽陽性率 **= [1- specificity]**

specificity = TN / (TN + FP) 意指正確判斷出負樣本

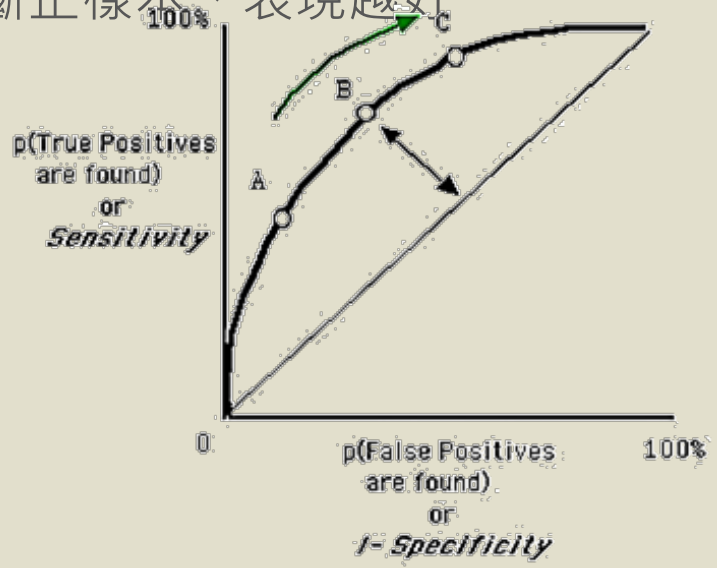
故特異度越高、**FPR**越低，模型越能夠正確判斷負樣本、表現越好

TPR (True Positive Rate)真陽性率 **= [Sensitivity]**

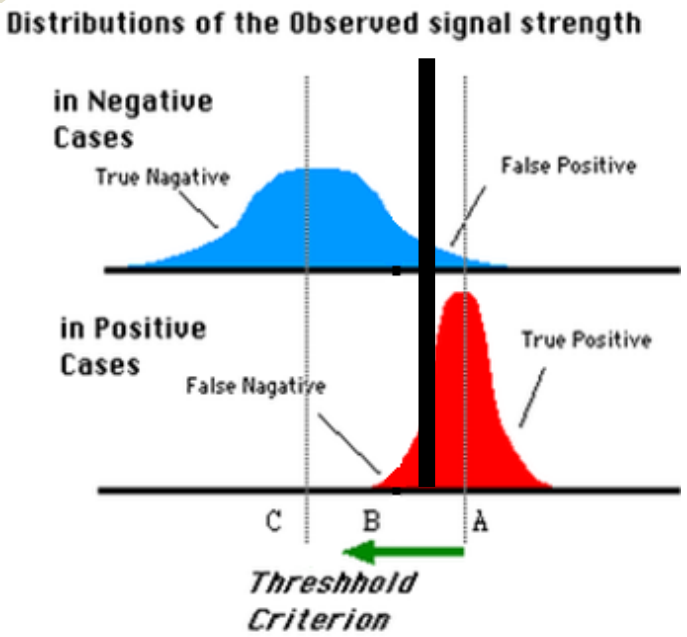
Sensitivity = TP/ (TP + FN)是正確判斷出正樣本

故**TPR**越高則模型越能夠正確判斷正樣本、表現越好

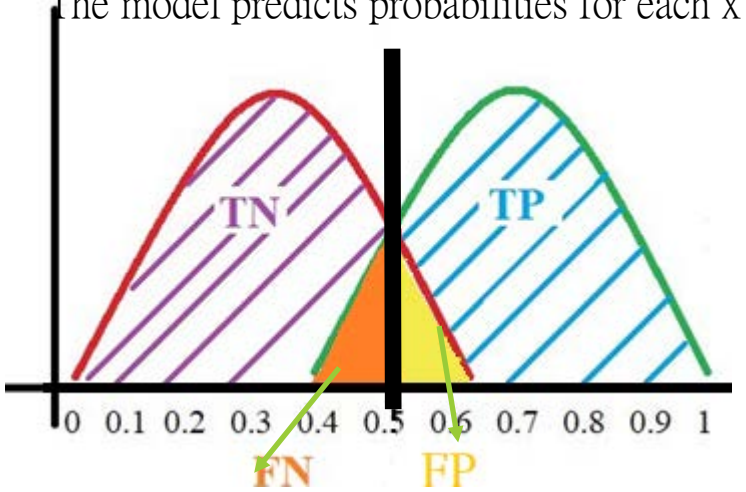
	實際Yes	實際No
預測Yes	TP (True Positive)	FP (False Positive)
預測No	FN (False Negative)	TN (True Negative)



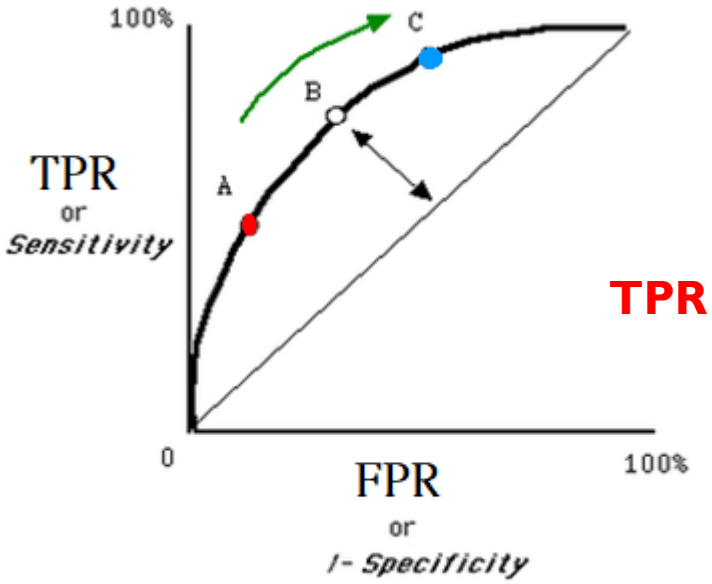
ROC



The model predicts probabilities for each x



ROC 曲線
(依據不同threshold來描繪)

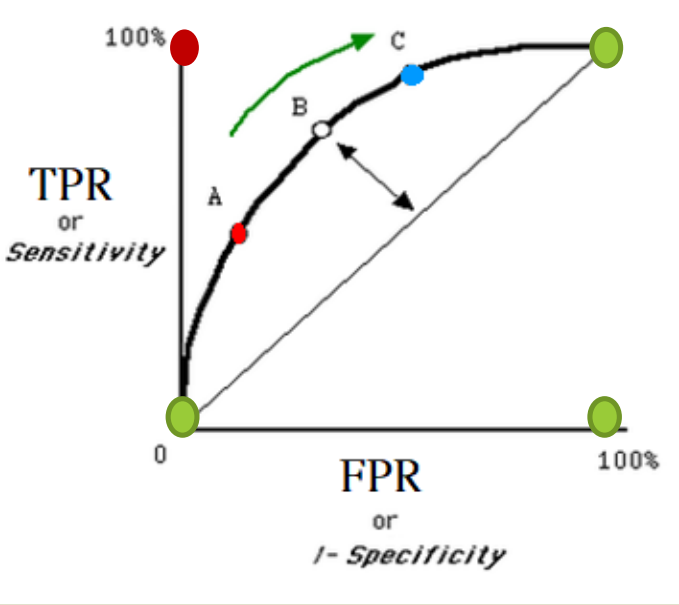


TPR 和 FPR 成正比

* 改變threshold 會改變FN,FP, 即 Sensitivity (TPR)及1-Specificity (FPR)也會跟著改變
* ROC曲線則是根據在不同threshold下所畫出來的曲線。 Threshold 愈接近0 則TPR和FPR愈接近1; 反之,Threshold 愈接近1,TPR和FPR 愈接近0

ROC EXAMPLE - 1

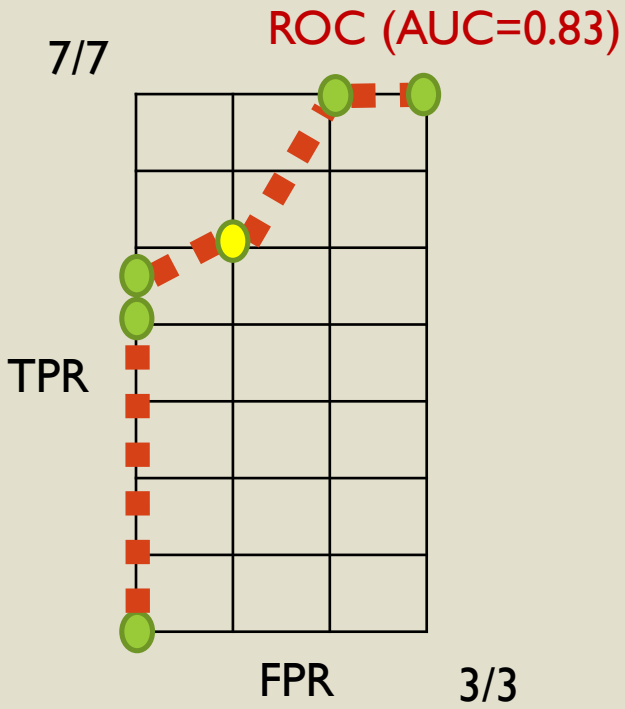
Label	Thresh=0	Thresh=1		
0	0	X	X	0
X	0	X	0	X
0	0	X	X	0
0	0	X	X	0
0	0	X	X	0
X	0	X	0	X
0	0	X	X	0
0	0	X	X	0
0	0	X	X	0
X	0	X	0	X
FPR	1-0/3=3/3	1-3/3=0/3	3/3	0/3
TPR	7/7	0/7	0/7	7/7



ROC EXAMPLE - 2

Label	Pr.	1	0.8	0.7	0.5	0.3	0
0	0.6	X	X	X	0	0	0
X	0.2	X	X	X	X	X	0
0	0.5	X	X	X	X	0	0
0	0.5	X	X	X	X	0	0
0	1.0	X	0	0	0	0	0
X	0.4	X	X	X	X	0	0
0	0.8	X	X	0	0	0	0
0	0.9	X	0	0	0	0	0
0	1.0	X	0	0	0	0	0
X	0.6	X	X	X	0	0	0
FPR		0/3	0/3	0/3	1/3	2/3	3/3
TPR		0/7	3/7	4/7	5/7	7/7	7/7

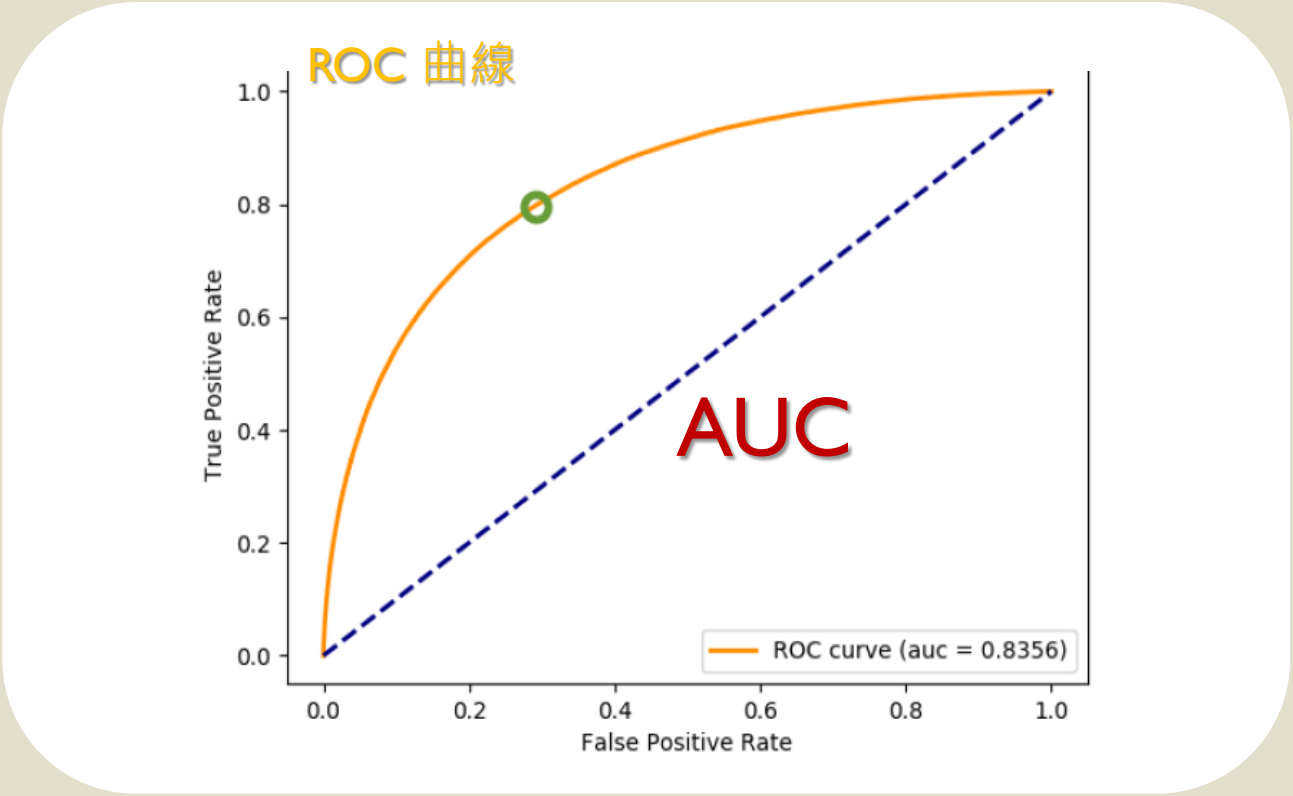
Threshold



TPR=5/7=71.4%
FPR=1/3=33.3%

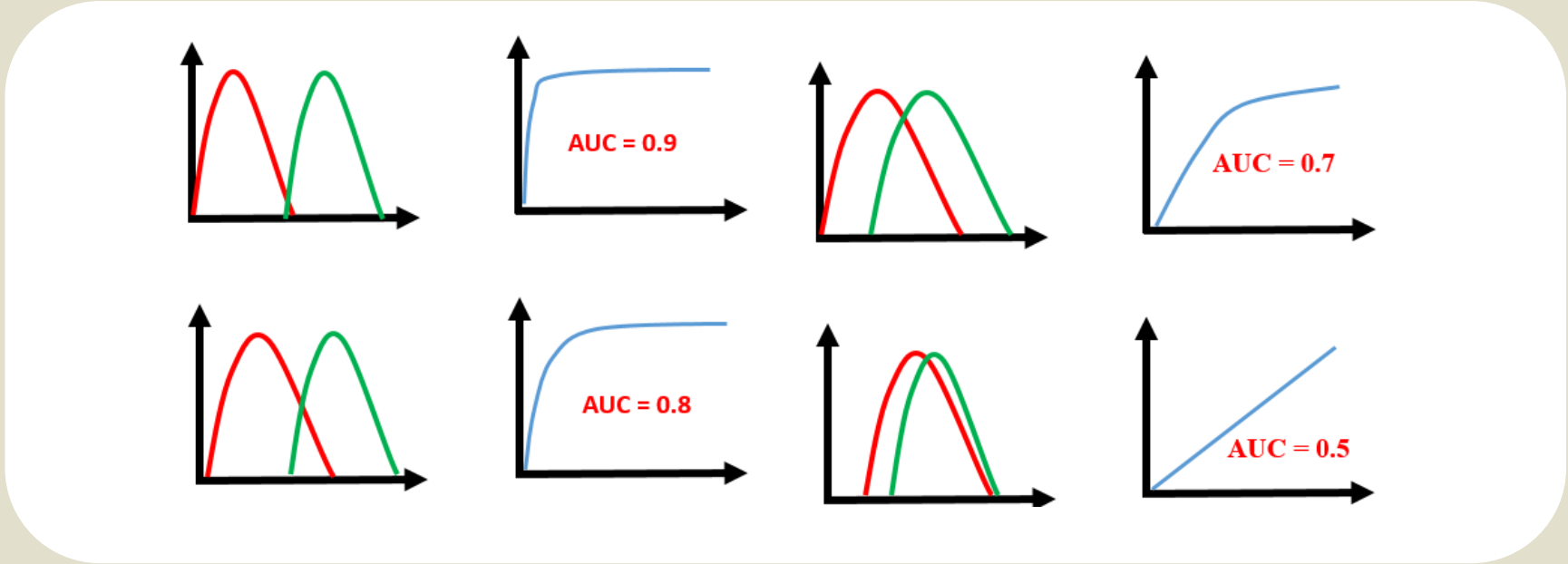
判斷模型預測能力的方法

- **AUC (Area Under the Curve)**
  - **ROC (*Receiver Operating Characteristic*)** 曲線下方的面積
  - 在**1x1**的方格裡求面積，**AUC**必在**0~1**之間
  - **AUC**值越大的分類器，正確率越高



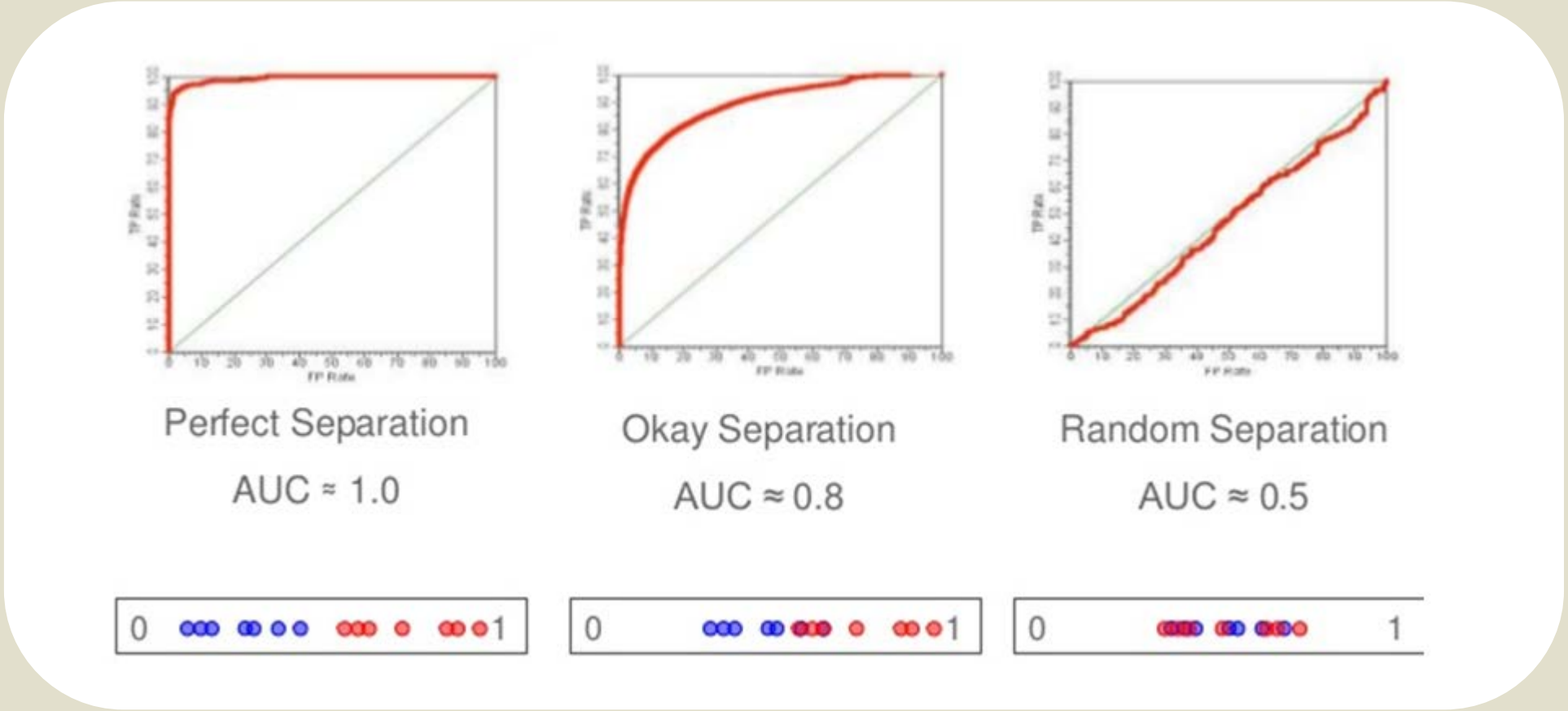
AUC

- **AUC值越大的分類器，正確率越高**
  - **AUC = 1**，是完美分類器，採用這個預測模型時，存在一個閥值能得出完美預測
  - **0.5 < AUC < 1**，優於隨機猜測。分類器（模型）妥善設定閥值的話，能有預測能力
  - **AUC = 0.5**，跟隨機猜測一樣，模型沒有預測能力
  - **AUC < 0.5**，比隨機猜測還差；但只要總是反預測而行，就優於隨機猜測



https://medium.com/kkstream/auc-explained-85537509e3b3

ROC & AUC



https://www.slideshare.net/asurkis/machine-learning-actsmeeting2015

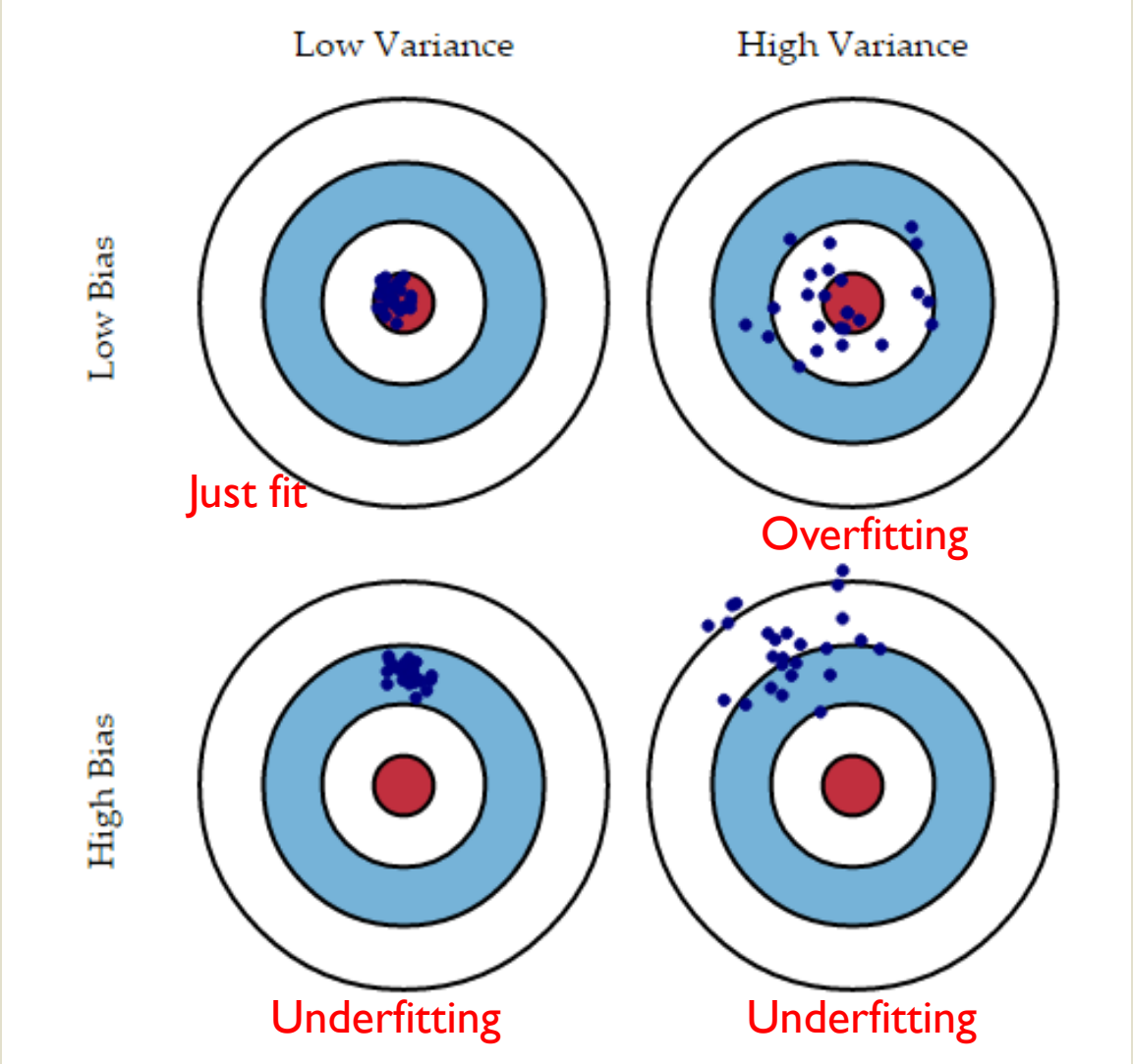
EXERCISE

- 請寫一個**Python**程式完成**AUC**繪圖

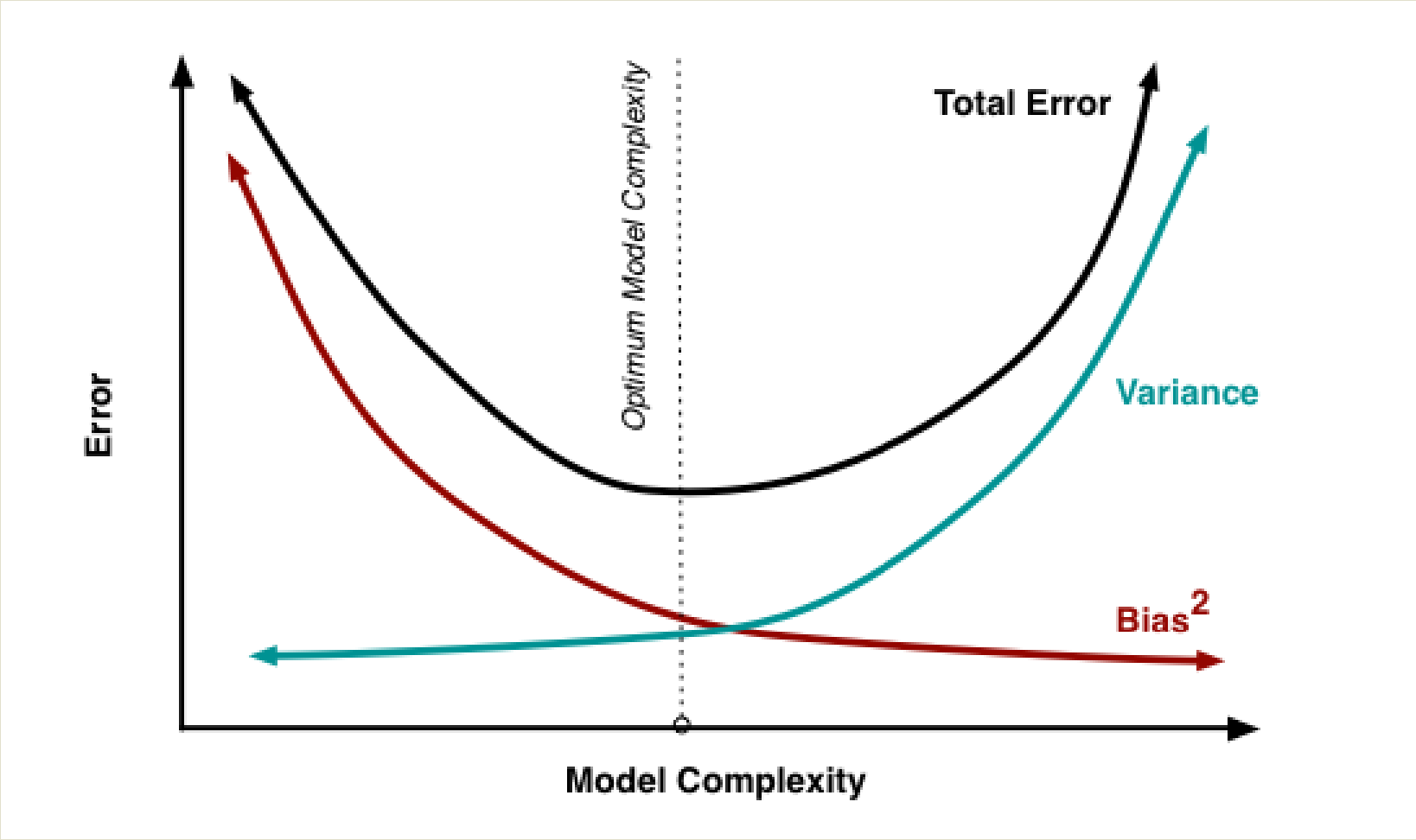
ENSEMBLE LEARNING

WHAT CAUSES ERROR IN THE MODEL ?

Noise, Bias, Variance

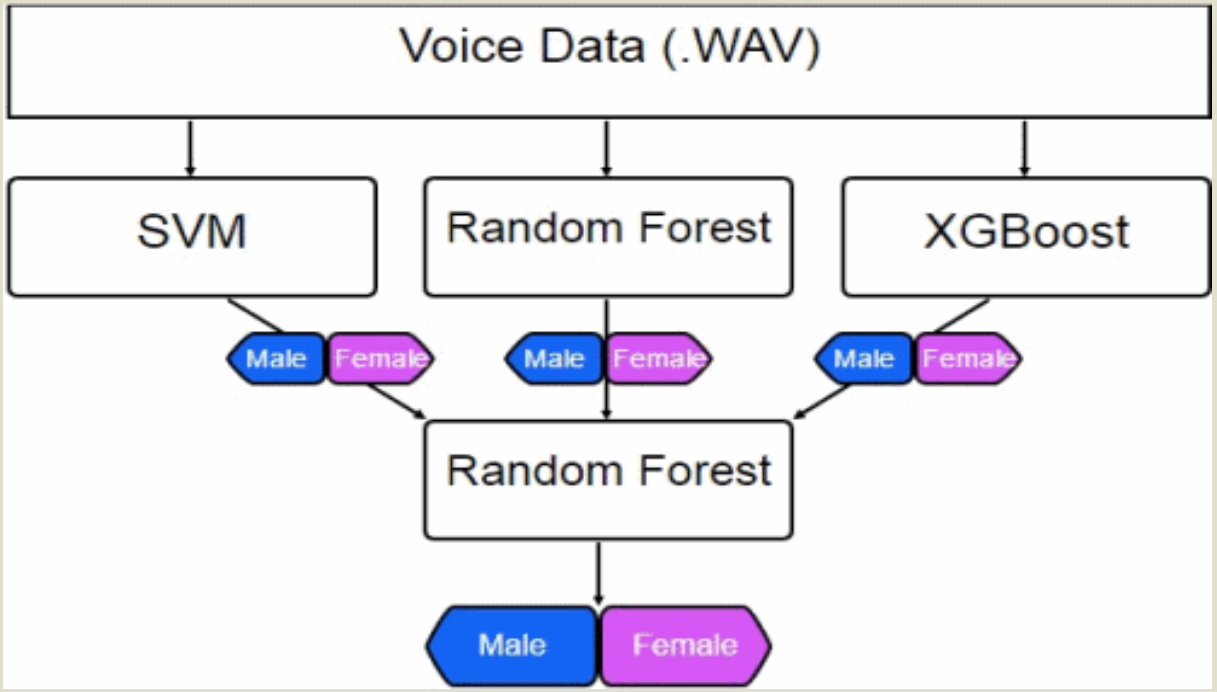


- A champion model should maintain a balance between these two types of errors. This is known as the trade-off management of bias-variance errors



WHAT IS ENSEMBLE LEARNING?

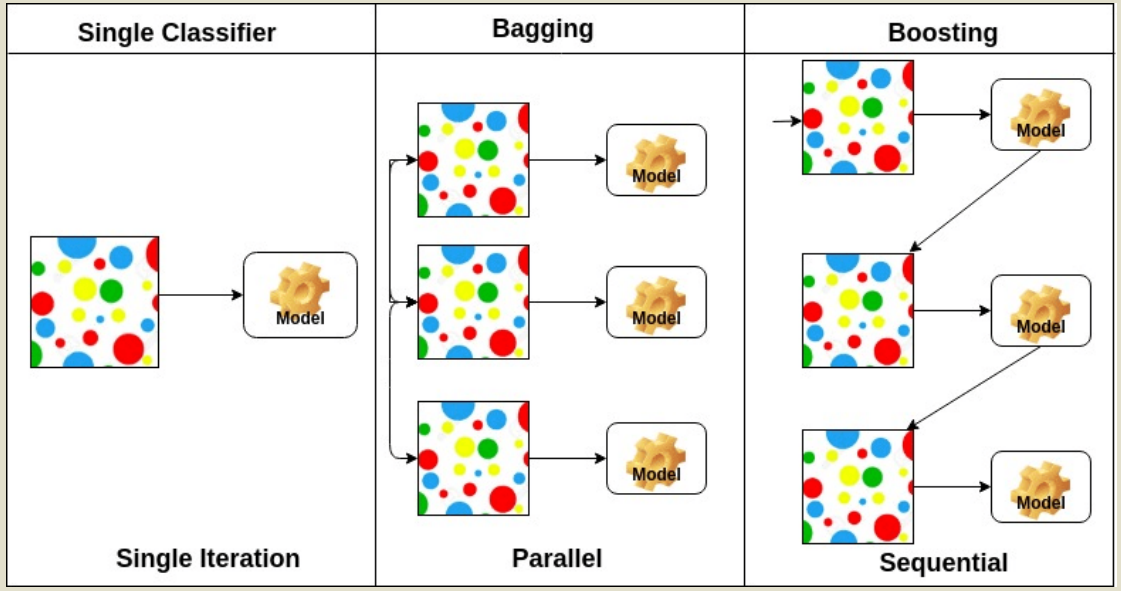
- **Ensemble helps to minimize these error factors.**
- **Ensemble learning** 組合多個弱學習器來建構一個強穩的模型，如此模型比較不會有**bias**或是發生**Overfitting**的情況
- **Random Forest** 即採用**Ensemble**方式



三個臭皮匠勝過一個諸葛亮

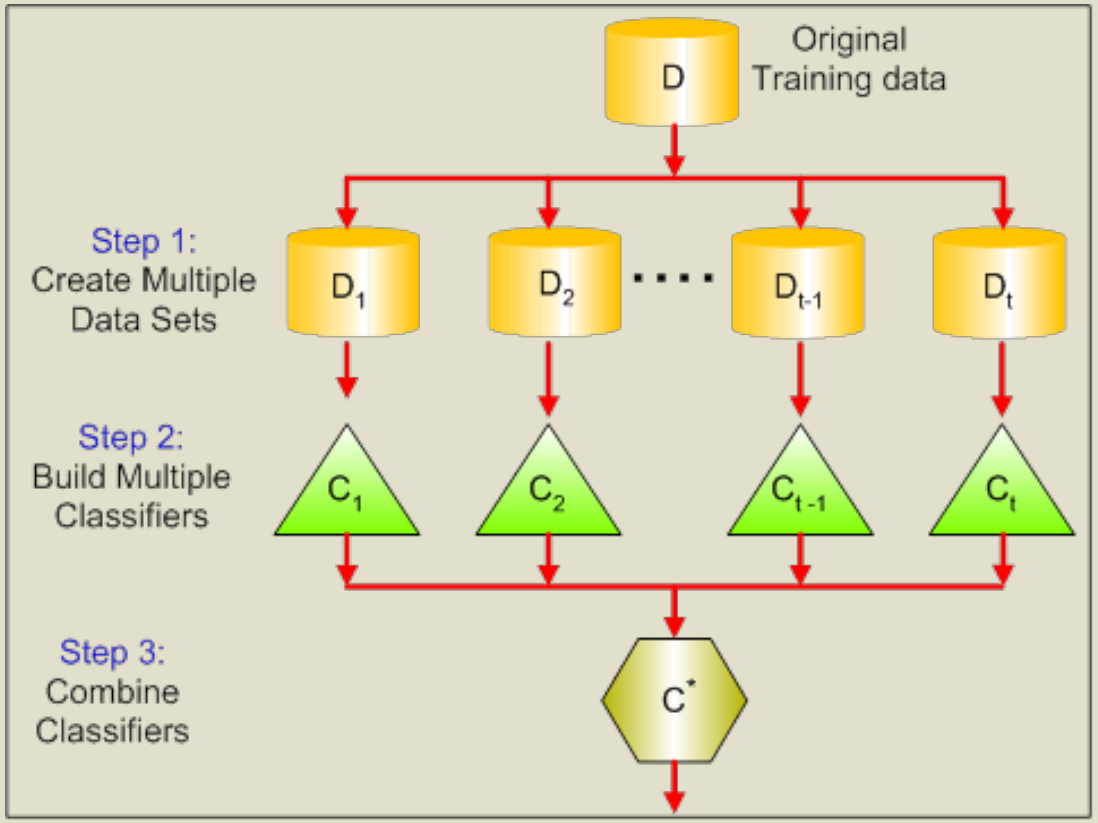
ENSEMBLE LEANING

1. Bagging(Bootstrap Aggregating)
   - mainly used to decrease the model's variance.
   - Common Bagging algorithms: **Random forest,**
2. Boosting
   - mainly used to decrease the model's bias.
   - Common Boosting algorithms: AdaBoost, XGBoost, …



BAGGING(BOOTSTRAP AGGREGATING)

- Implement similar learners on small sample populations and then takes a mean of all the predictions (in case of a regression model)



BAGGING(BOOTSTRAP AGGREGATING)

- Main Steps involved in bagging are :
  - 1. Creating multiple datasets: Sampling is done with a replacement on the original data set and new datasets are formed from the original dataset.
  - 2. Building multiple classifiers: On each of these smaller datasets, a classifier is built, usually, the same classifier is built on all the datasets.
  - 3. Combining Classifiers: The predictions of all the individual classifiers are now combined to give a better classifier, usually with very less variance compared to before.

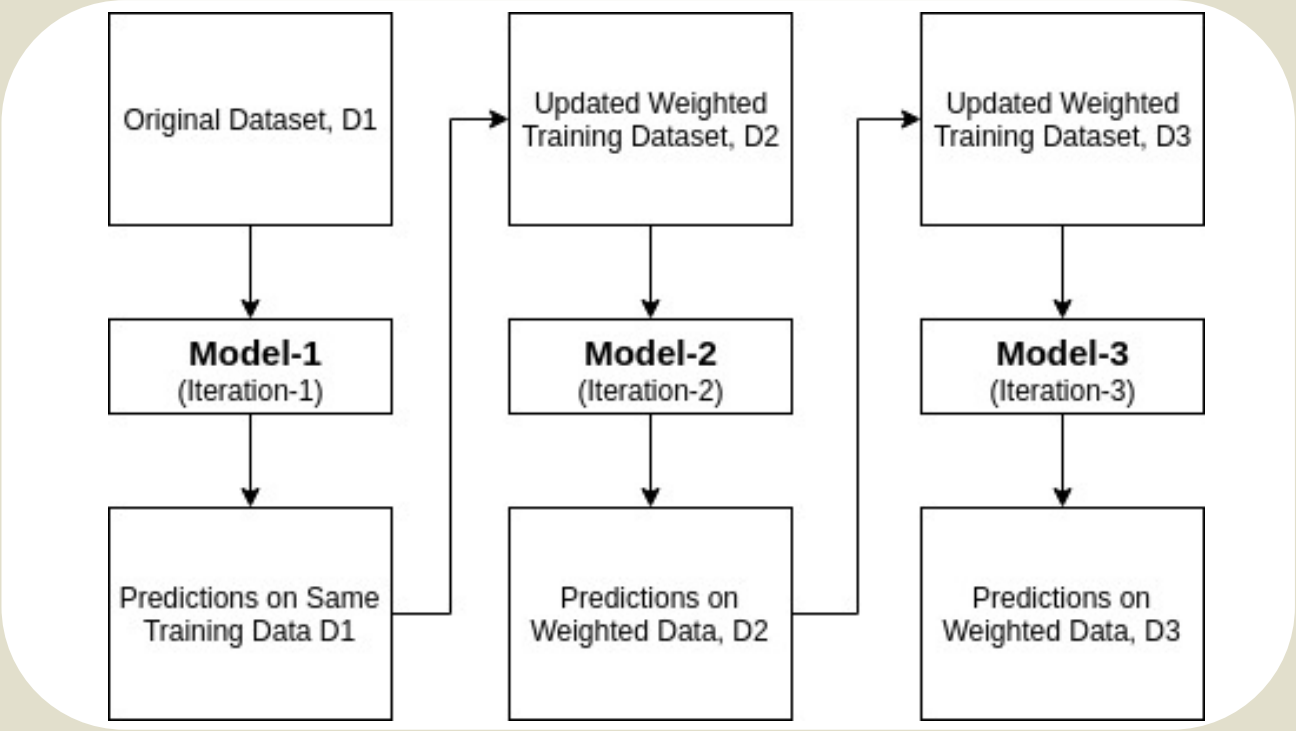
Ref: https://www.youtube.com/watch?v=2Mg8QD0F1dQ

BAGGING 優點

- **Bagging**的精神在於從樣本中抽樣，如果模型不是分類問題而是預測的問題，分類器部份則改成**regression**，最後投票方式改成計算平均數即可。使用**Bagging** 時會希望單一分類器能夠是一個效能比較好的分類器。
- **Bagging**的優點在於原始訓練樣本中有噪聲資料**(**不好的資料**)**，透過**Bagging**抽樣就有機會不讓有噪聲資料被訓練到，所以可以降低模型的不穩定性。
- 由於**Bagging**的每個**model**獨立，可平行計算

BOOSTING(ADABOOST)

- 基本分類器有所關聯，分錯的樣本會得到加強，加權後的全體樣本再次被用來訓練下一個基本分類器。
- 每一輪中加入一個新的弱分類器，直到達到某個預定的足夠小的錯誤率或達到預先指定的最大反覆運算次數，最後模型準確度高的給予較高權重。



Ref : https://www.youtube.com/watch?v=GM3CDQfQ4sw

ADABOOST ALGORITHM

AdaBoost全名為**Adaptive Boosting**是一种迭代算法，其核心思想是針對同一個訓練集**(training set)**訓。 **AdaBoost**練不同的分類器**(**弱分類器**)**，然後把這些弱分類器集合起來，構成一個更強的最终分類器**(**強分類器**)**。

It is often easy to come up with a **weak classifier**, one that is only slightly better than random guessing.

$$\Pr(h(X) \neq Y) = \frac{1}{2} - \epsilon$$

A learning algorithm that can consistently generate such classifiers is called a **weak learner**.

Is it possible to systematically boost the quality of a weak learner?

Blueprint:

- Think of the data set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$ as a distribution D on $\mathcal{X} \times \mathcal{Y}$
- Repeat for $t = 1, 2, \ldots$:
  - Feed D to the weak learner, get back a weak classifier h_t
  - Reweight D to put more emphasis on points that h_t gets wrong
- Combine all these h_t's somehow

ADABOOST

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

T: number of weak learners

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$.

The classifiers that performed well are given higher importance or weight.
正確率高, α 愈大; 反之, α 愈小

- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

Weights of misclassified training samples are increased ;
Weights of correctly classified samples are decreased

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

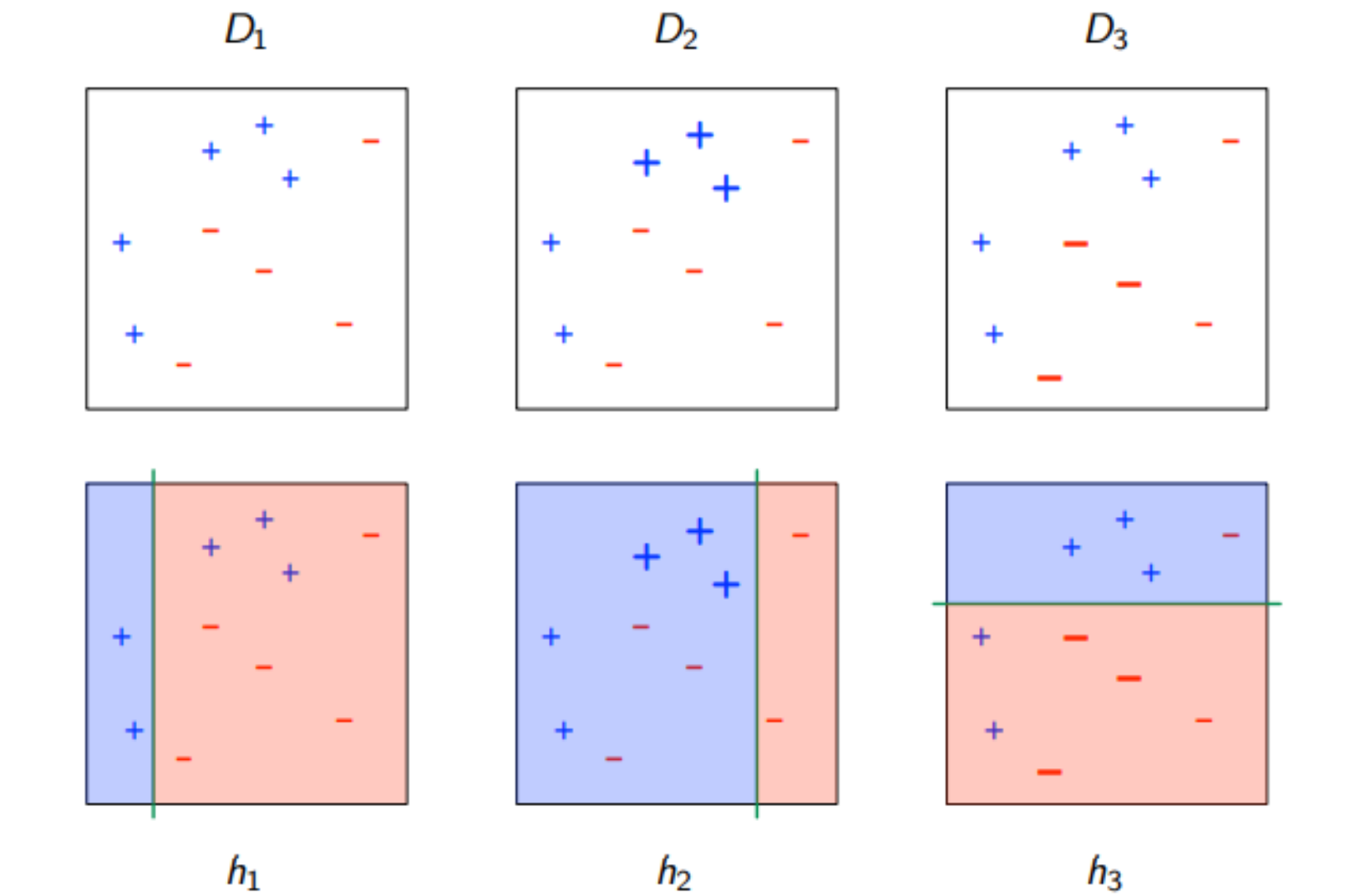
Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Final strong classifier with class Result {+1, -1}

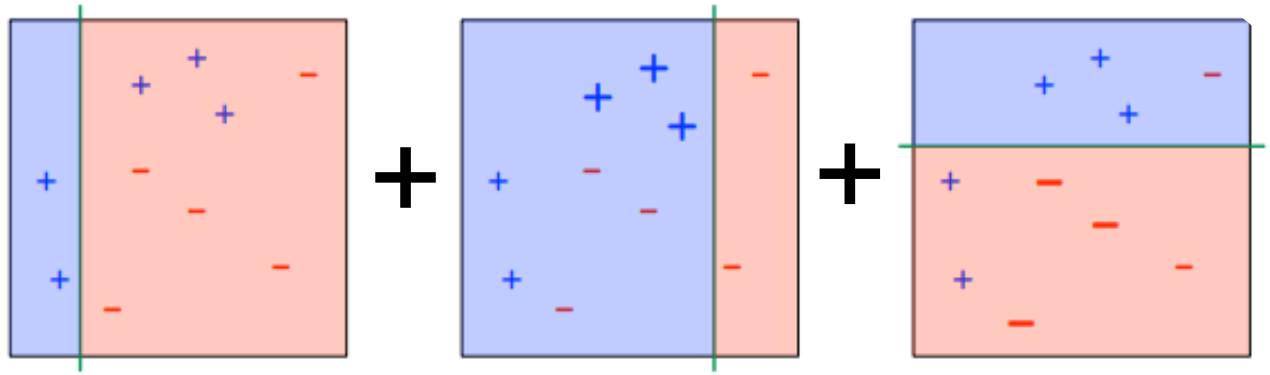
Weighted majority voting scheme

EXAMPLE : ADABOOSTING



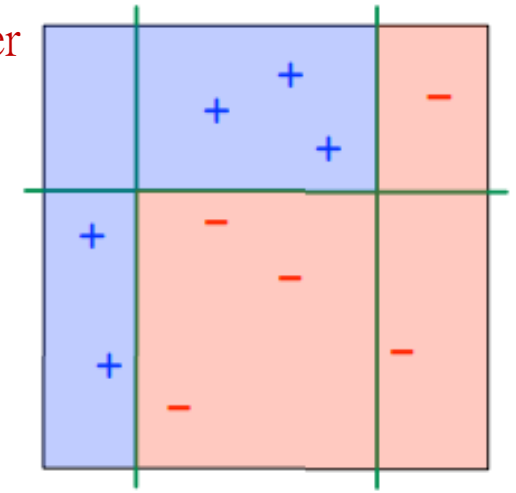
EXAMPLE : ADABOOSTING (CONT.)

由3人共同投票決定，但每個人的權重不同



h_1 $\alpha_1 = 0.42$ h_2 $\alpha_2 = 0.65$ h_3 $\alpha_3 = 0.92$

strong classifier (h1+h2+h3)



Final classifier:

$\text{sign}(0.42h_1(x)+0.65h_2(x)+0.92h_3(x))$

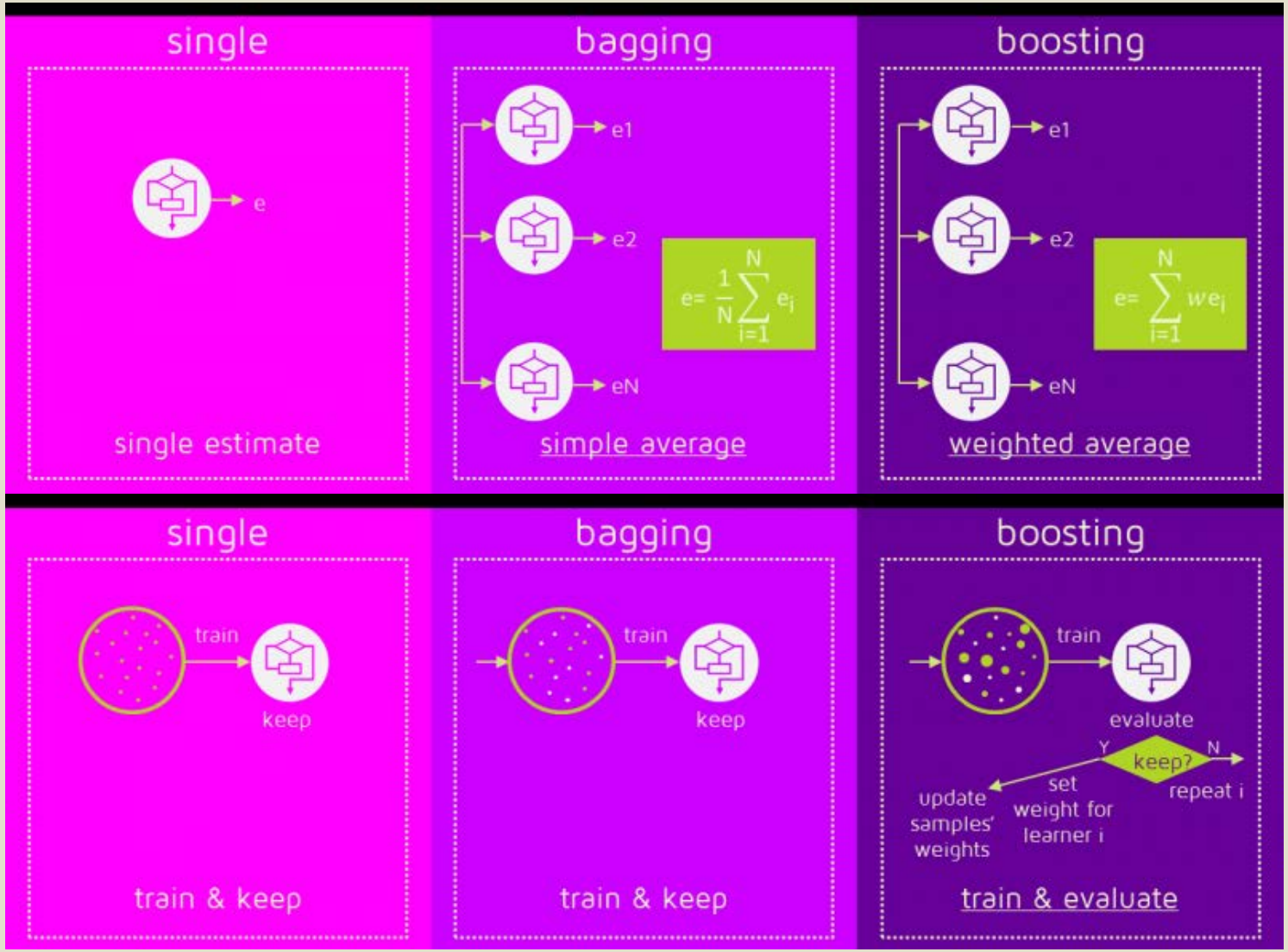
sign(x) <=0 ➔ +

sign(x) > 0 ➔ -

PROS AND CONS

- Pros
  - AdaBoost is easy to implement.
  - It iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners.
  - You can use many base classifiers with AdaBoost.
  - AdaBoost is not prone to overfitting. This can be found out via experiment results, but there is no concrete reason available.
- Cons
  - AdaBoost is sensitive to noise data. It is highly affected by outliers because it tries to fit each point perfectly.

BAGGING AND BOOSTING



SUMMARY ON BAGGING AND BOOSTING

Similarities	Differences
Both are ensemble methods to get N learners from 1 learner…	… but, while they are built independently for Bagging, Boosting tries to add new models that do well where previous models fail.
Both generate several training data sets by random sampling…	… but only Boosting determines weights for the data to tip the scales in favor of the most difficult cases.
Both make the final decision by averaging the N learners (or taking the majority of them)…	… but it is an equally weighted average for Bagging and a weighted average for Boosting, more weight to those with better performance on training data.
Both are good at reducing variance and provide higher stability…	… but only Boosting tries to reduce bias. On the other hand, Bagging may solve the over-fitting problem, while Boosting can increase it.

APPENDIX -結合不同類型的弱學習器

- Voting

  The VotingClassifier takes in a list of different estimators as arguments and a voting method.

  The hard voting method uses the predicted labels and a majority rules system, while the soft voting method predicts a label based on the argmax