Machine Learning

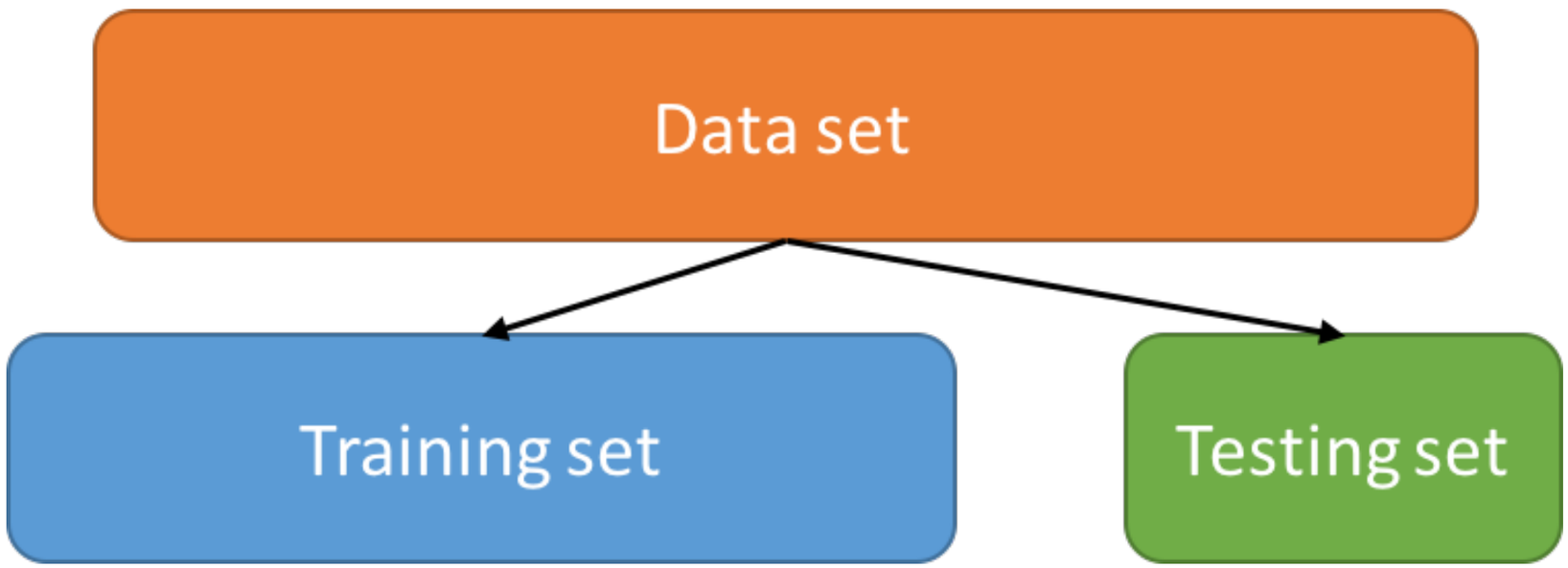
Lecture 10　學習模型評估

比較與評估分類方法

- **使用下列標準比較與評估分類方法**
  - **準確率**：對新的或未知的資料正確判斷或猜測的能力
  - **速度**：產生和使用模型的計算成本之花費
  - **穩定性(Robust)**：給定噪音資料或有空缺值的資料，模型正確預測或判斷的能力
  - **可量度性**：對大量資料，有效的構建模型的能力
  - **可解釋性**：學習模型提供的了解程度。難評量。

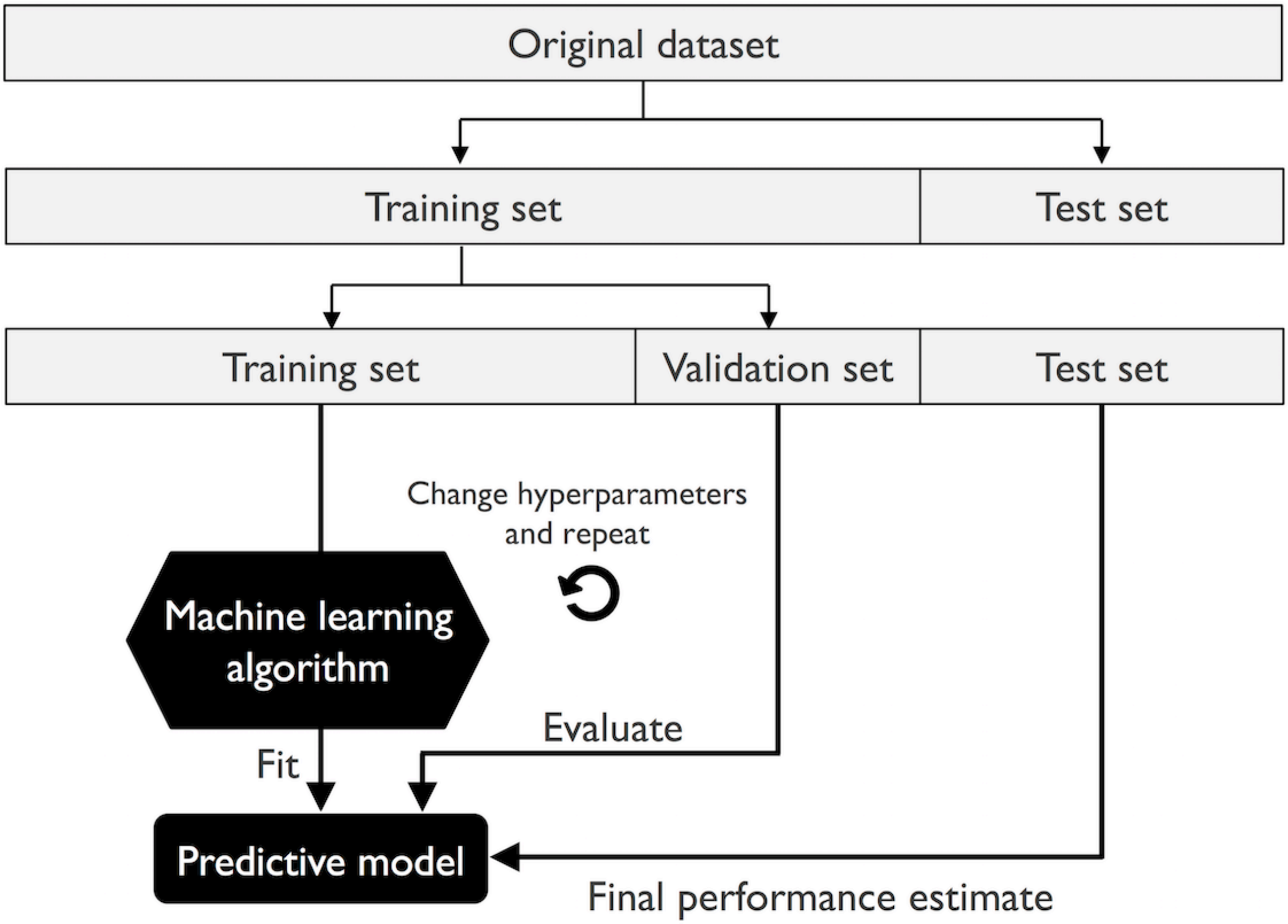
衡量分類模型的正確性

- **測試(Holdout)法**
  - **作法1：將資料隨機切成兩個獨立部分來計算正確率/錯誤率**
    - **訓練資料 (例., 2/3) 用於建立模型**
    - **測試資料 (例., 1/3) 用於評估模型的正確率/錯誤率**
  - **作法2：隨機子取樣**
    - **進行k次測試，正確率為k次的平均**

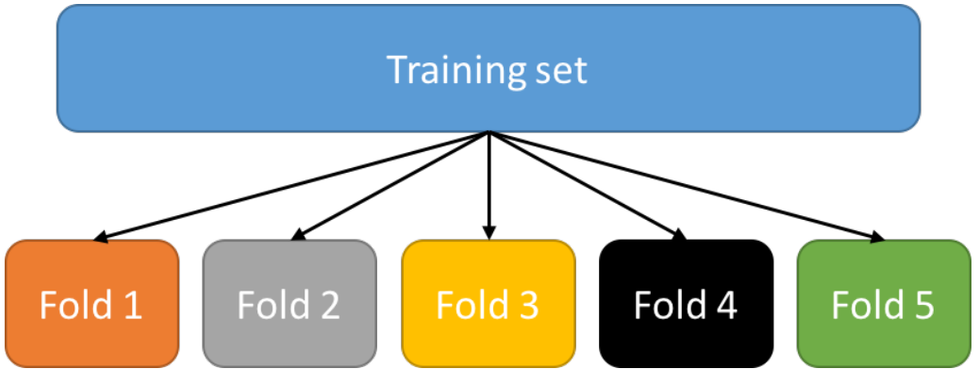


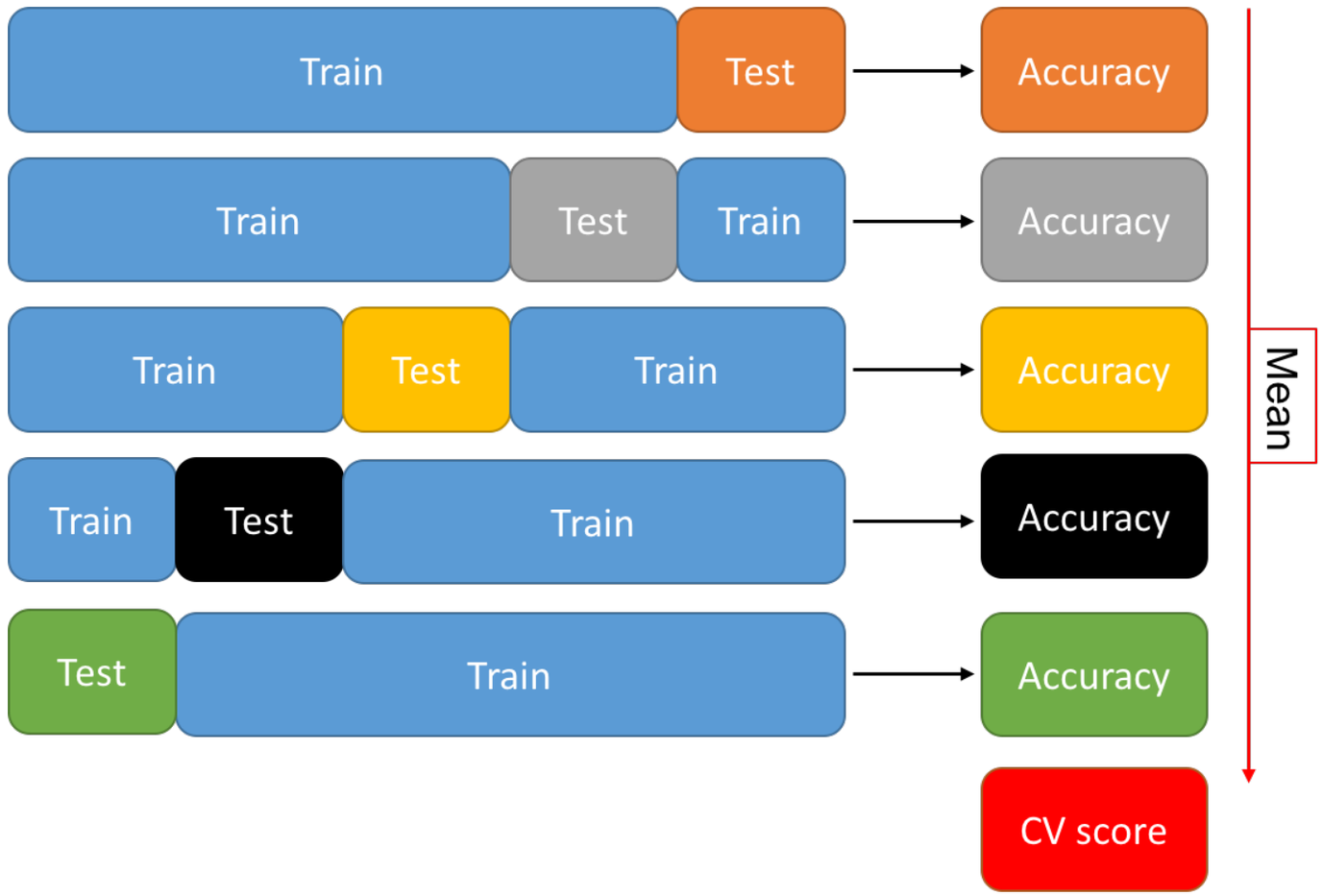
Source: https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/

保留交叉驗證法


Original dataset
Training set
Test set
Training set
Validation set
Test set
Change hyperparameters
and repeat
Machine learning
algorithm
Evaluate
Fit
Predictive model
Final performance estimate

k-fold cross-validation





Source: https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/

衡量分類模型的正確性

- **k 次交叉驗證 (k-fold cross validation, k = 10 為最普遍)**
  - **隨機將原始資料分割成 k個不重疊的資料子集合 (folds) D_i，其中 i = 1, 2, …, k，每個部份的資料量大小相近**
  - **需執行k個回合。當執行到第 i 回合時，以 D_i 設為測試資料集，剩下的資料子集合當作訓練資料來建構模型。因此，每個子集合皆會被當成測試資料。**
  - **總錯誤率即為 k 次錯誤率的總合。**

改良版本： **分層k折交叉驗證 (Stratified k-fold cross-validation)**

“每折數據”中的類別大小比例 = 原始“訓練集”的類別大小比例

其他效能指標

混淆矩陣 (Confusion matrix)

預測類別 / 實際類別	Class 1	Class 2
Class 1	TP（true positive）	FN（false negative）
Class 2	FP（false positive）	TN（true negative）

偽陽性：檢測結果為陽性，但其實沒有生病

- 根據分類結果，可計算出正確率或分類錯誤率

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad Error\,rate = 1 - Accuracy = \frac{FP + FN}{TP + TN + FP + FN}$$

其他效能指標

預測類別 / 實際類別	Class 1	Class 2
Class 1	TP（true positive）	FN（false negative）
Class 2	FP（false positive）	TN（true negative）

處理不平衡類別時

敏感度 (Sensitivity)

該類別確實正確被預測的比率

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

實際上：P

真正有患病的人，被正確檢驗出陽性的機率

準確度 (Specificity)

為另一類別且確實被分為另一類別的比率

$$\text{Specificity} = \frac{TN}{TN+FP}$$

實際上：N

沒患病的人，被正確檢驗判斷為陰性的機率

以腫瘤診斷為例，我們更關心：檢查出“惡性腫瘤”以幫助病人治療。
但同時要“減少”將病人的“良性腫瘤”錯誤分類為“惡性腫瘤”

其他效能指標

實際類別 \ 預測類別	Class 1	Class 2
Class 1	TP（true positive）	FN（false negative）
Class 2	FP（false positive）	TN（true negative）

敏感度 (Sensitivity)

該類別確實正確被預測的比率

$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

準確度 (Specificity)

為另一類別且確實被分為另一類別的比率

$$\text{Specificity} = \frac{TN}{TN+FP}$$

精準率 (precision)

預測類別中，多少比率的資料剛好屬於該類別

$$p = \frac{TP}{TP+FP}$$

預測：P

回想率 (recall)

實際為某類別，同時被判為該類別的比率

$$r = \frac{TP}{TP+FN}$$

兩指標合併

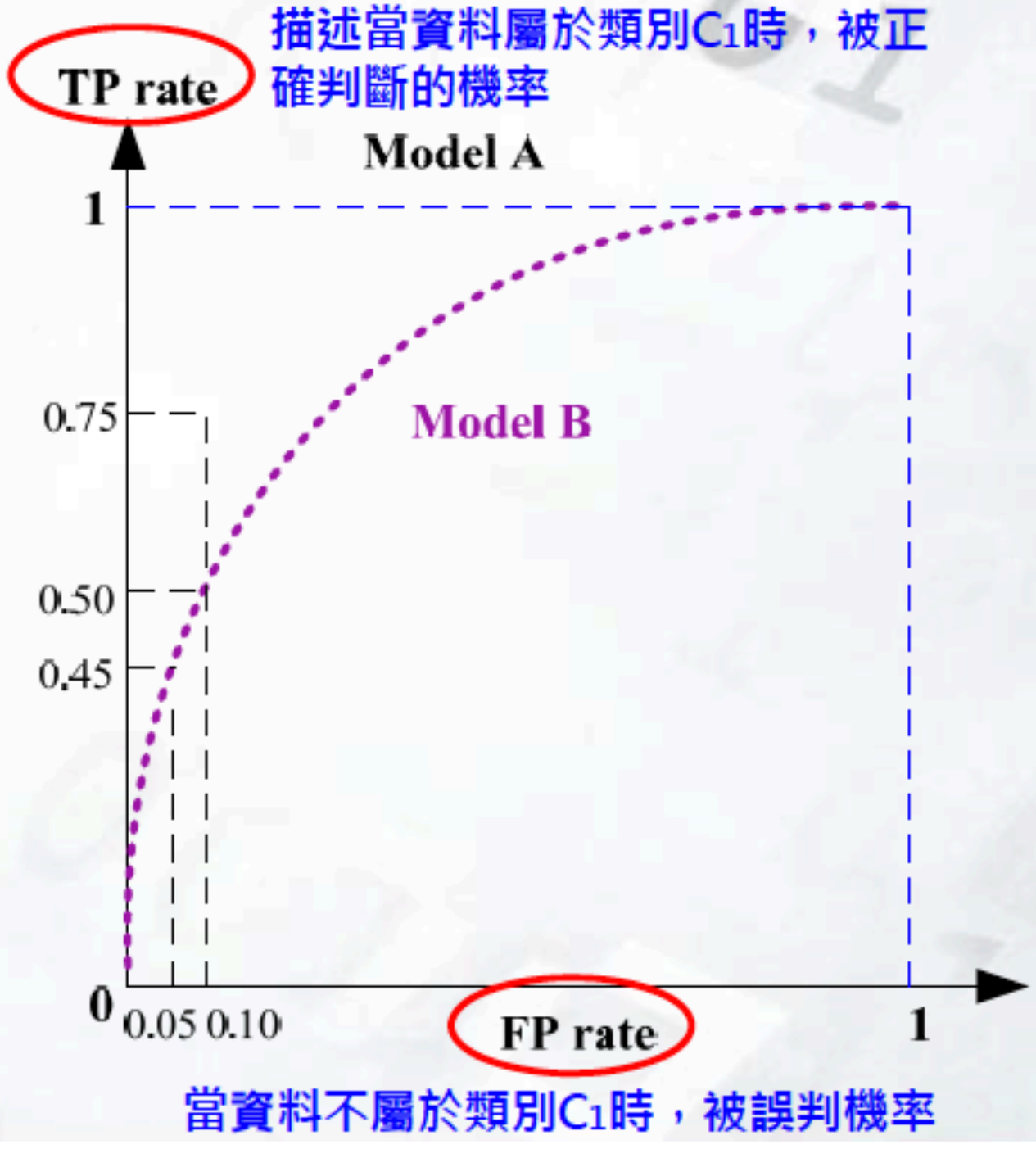
$$F_1 = \frac{2}{\frac{1}{r}+\frac{1}{p}} = \frac{2rp}{r+p}$$

Receiver operating characteristic, ROC

- **ROC曲線**:可作為衡量不同FP rate下TP rate的變化
  - TP rate為越大越好
  - FP rate為越小越好
  - 準確度=1－FP rate，敏感度(TP rate)增加，準確度也會減少，即FP rate會增加

- **AUC (Area under the Curve of ROC)**，也就是ROC曲線下方的面積。

→面積越大，模式分類效果越好



Python

- 畫出3-fold ROC曲線

```
from sklearn.metrics import roc_curve, auc
from scipy import interp

lr = LogisticRegression(penalty='l2', random_state=1, C=100.0)

cv = list(StratifiedKFold(n_splits=3,
                          random_state=1).split(X_train, y_train))

fig = plt.figure(figsize=(7, 5))

mean_tpr = 0.0
mean_fpr = np.linspace(0, 1, 100)
all_tpr = []

for i, (train, test) in enumerate(cv):
    probas = lr.fit(X_train[train],
                    y_train[train]).predict_proba(X_train[test])

    fpr, tpr, thresholds = roc_curve(y_train[test],
                                     probas[:, 1], pos_label=1)
    mean_tpr += interp(mean_fpr, fpr, tpr)
    mean_tpr[0] = 0.0
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr,
             tpr,
             label='ROC fold %d (area = %0.2f)'
                   % (i+1, roc_auc))
```

Python

- 畫出隨機猜測曲線，及平均ROC曲線

```
plt.plot([0, 1],
         [0, 1],
         linestyle='--',
         color=(0.6, 0.6, 0.6),
         label='random guessing')

mean_tpr /= len(cv)
mean_tpr[-1] = 1.0
mean_auc = auc(mean_fpr, mean_tpr)
plt.plot(mean_fpr, mean_tpr, 'k--',
         label='mean ROC (area = %0.2f)' % mean_auc, lw=2)
plt.plot([0, 0, 1],
         [0, 1, 1],
         linestyle=':',
         color='black',
         label='perfect performance')

plt.xlim([-0.05, 1.05])
plt.ylim([-0.05, 1.05])
plt.xlabel('false positive rate')
plt.ylabel('true positive rate')
plt.legend(loc="lower right")

plt.tight_layout()
# plt.savefig('images.png', dpi=300)
plt.show()
```

結論

- 使用k折交叉驗證法來做模型評估
- 利用k折交叉驗證法，可以繪製學習曲線和驗證曲線，用來判斷是否有過度適合或低度適合的問題
- 使用混淆矩陣和其他效能指標來進一步做模型效能的評估

Exercise

- 改用高斯貝氏模型(GaussianNB)
- 請利用學習曲線、Precision、Recall、F1 score、及ROC曲線評估此模型

威斯康辛乳癌數據集

- **Breast Cancer Wisconsin (Diagnostic)**

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Python

- 載入 Breast Cancer Wisconsin data set (1/14)

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

df = pd.read_csv('wdbc.data', header=None)

df.head()

df.shape
```

威斯康辛乳癌數據集

- **Breast Cancer Wisconsin (Diagnostic)**

https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

- 569 個 惡性腫瘤細胞 （**m**alignant tumor cell）樣本和良性腫瘤細胞 （**b**enign tumor cell）。
- 32 個特徵
  - Column 1：唯一識別編號 (unique ID number)
  - Column 2：診斷 (M / B)
  - Column 3-32：實數值特徵

Python

- 將類別標籤轉換成整數
- 標準化並拆分成訓練集和測試集


```
X = df.loc[:, [4, 14]].values
y = df.loc[:, 1].values

le = LabelEncoder()
y = le.fit_transform(y)
le.classes_

le.transform(['M', 'B'])

X_train, X_test, y_train, y_test =  train_test_split(X, y,
                     test_size=0.20, stratify=y, random_state=1)

sc = StandardScaler()
sc.fit(X_train)
X_train = sc.transform(X_train)
X_test = sc.transform(X_test)
```

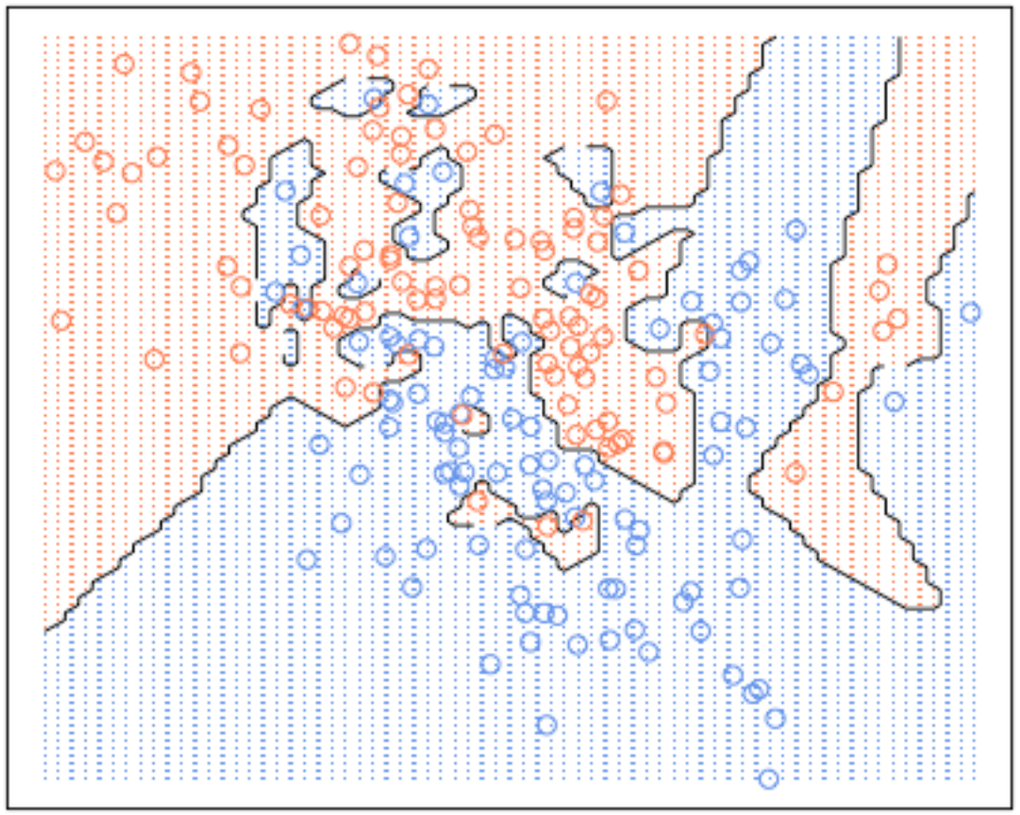
Python

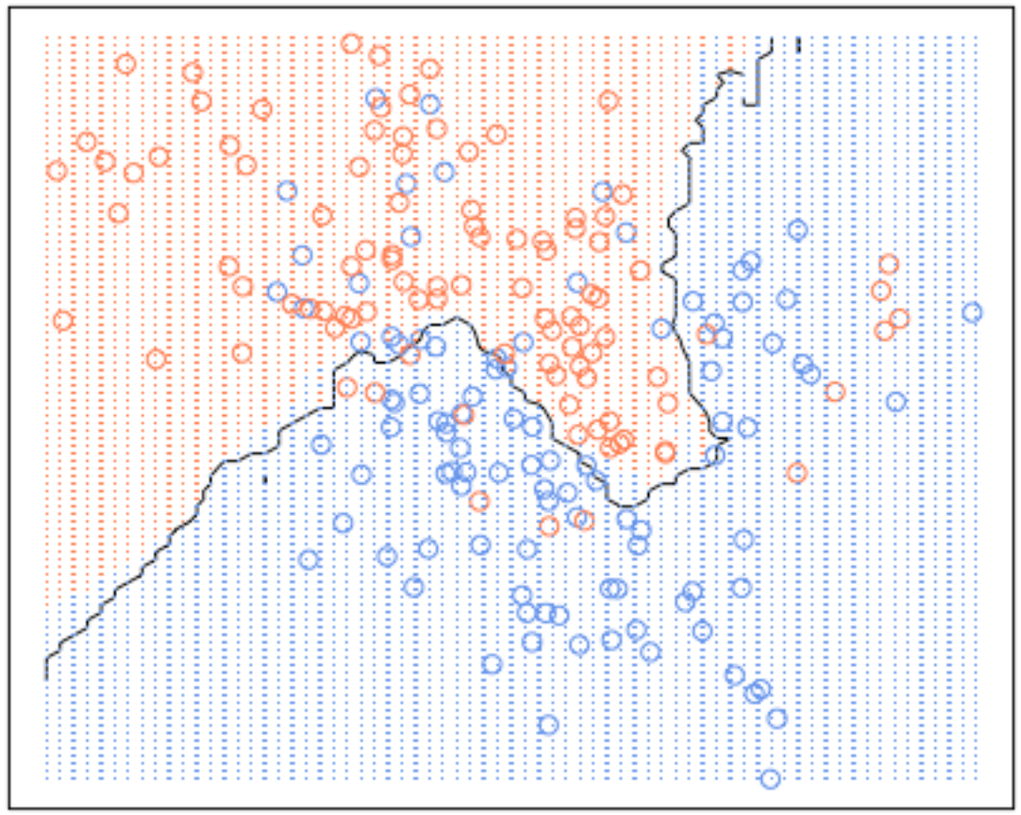
- 訓練 logistic Regression

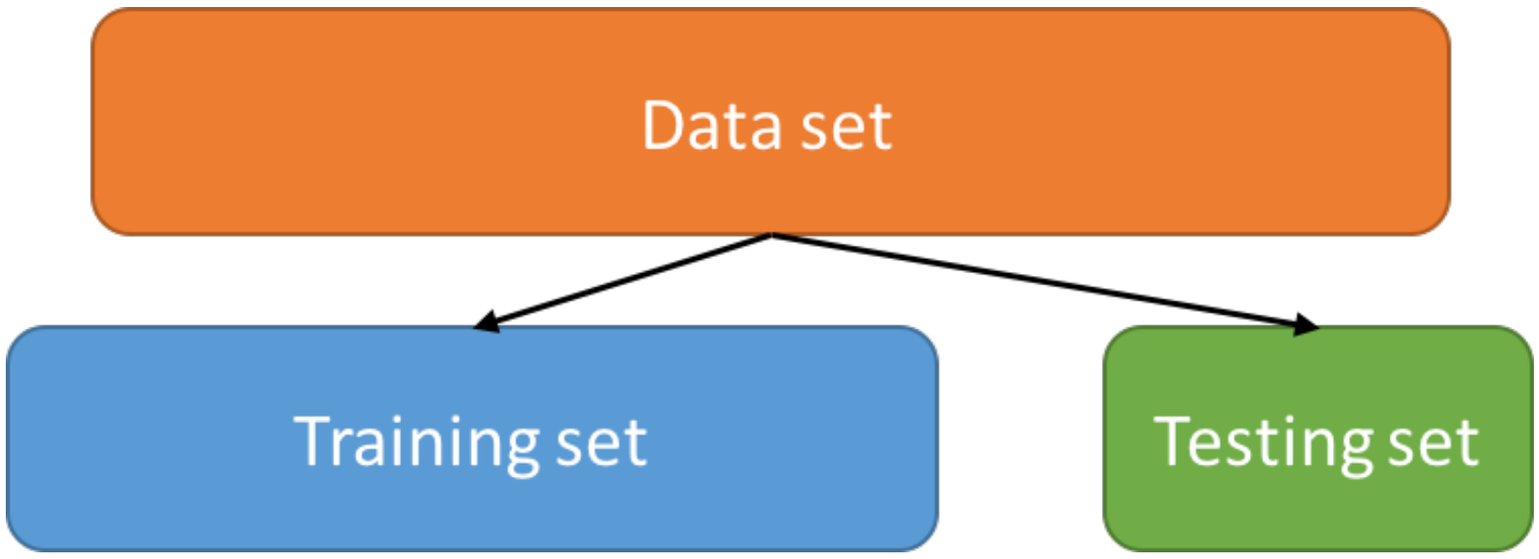

```
from sklearn.linear_model import LogisticRegression



lr = LogisticRegression(random_state=1)
lr.fit(X_train, y_train)
y_pred = lr.predict(X_test)
print('Test Accuracy: %.3f' % lr.score(X_test, y_test))
```

Underfitting v.s. Overfitting

https://kevinzakka.github.io/2016/07/13/k-nearest-neighbor/#exploring-knn-in-code







Source: https://aldro61.github.io/microbiome-summer-school-2017/sections/basics/

Python

- 分層k折交叉驗證

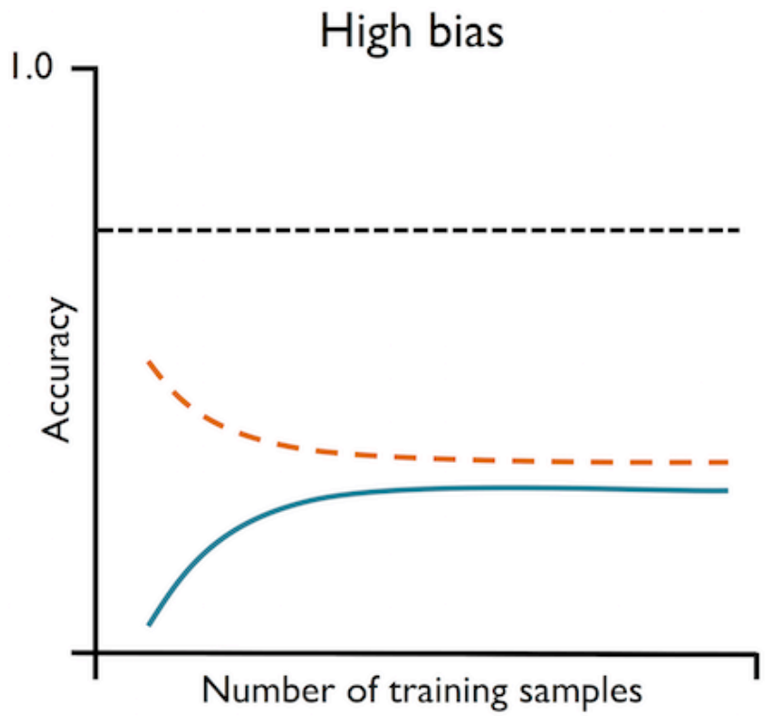

```
import numpy as np
from sklearn.model_selection import StratifiedKFold

kfold = StratifiedKFold(n_splits=10,
                        random_state=1).split(X_train, y_train)


scores = []
for k, (train, test) in enumerate(kfold):
    lr.fit(X_train[train], y_train[train])
    score = lr.score(X_train[test], y_train[test])
    scores.append(score)
    print('Fold: %2d, Class dist.: %s, Acc: %.3f' % (k+1,
          np.bincount(y_train[train]), score))

print('\nCV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

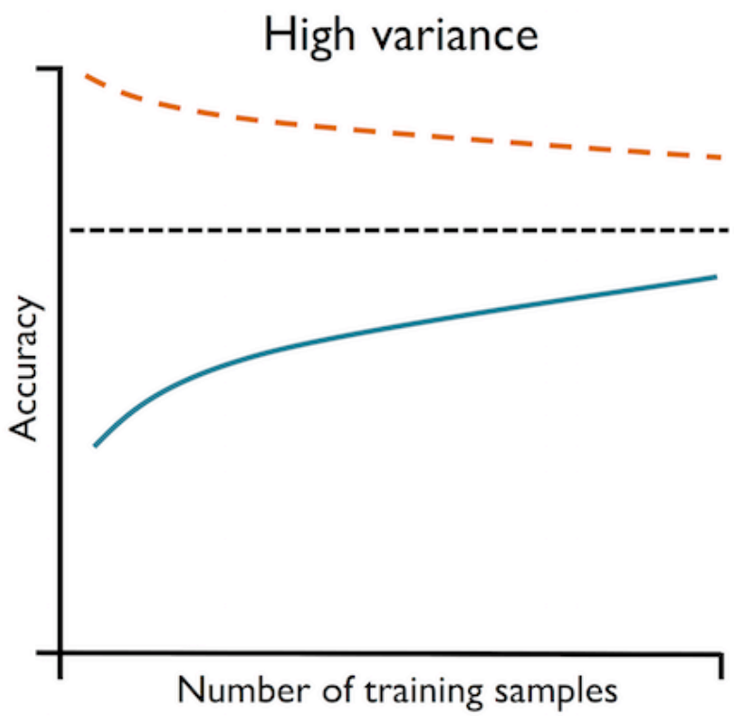
Python

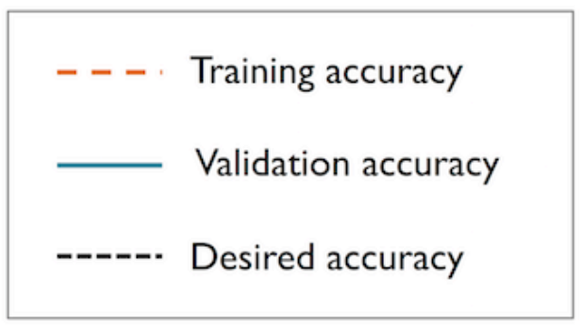
- 用scikit-learn 實作 “分層k折交叉驗證” 的 “計分器”

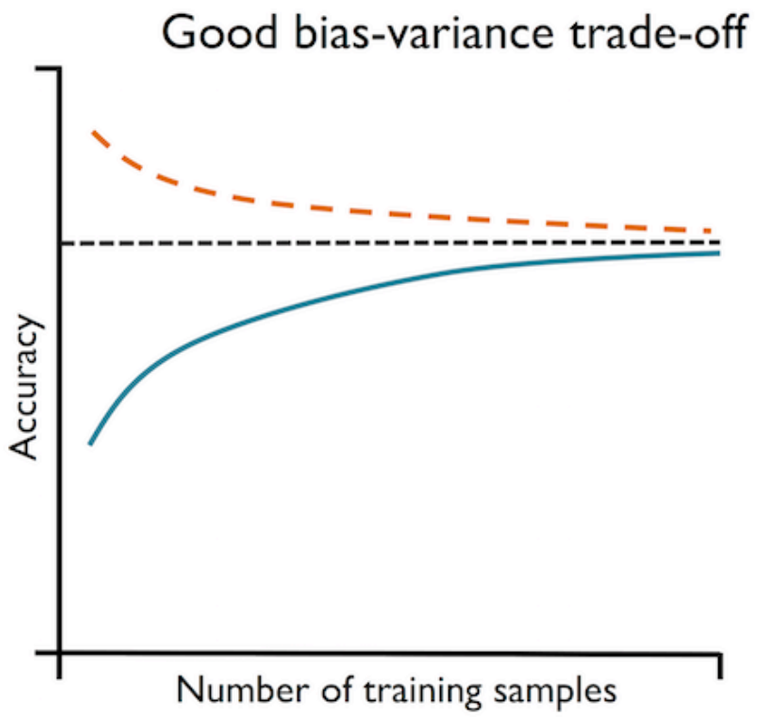
```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(estimator=lr, X=X_train, y=y_train,
                         cv=10, n_jobs=1)
print('CV accuracy scores: %s' % scores)
print('CV accuracy: %.3f +/- %.3f' % (np.mean(scores), np.std(scores)))
```

學習曲線


High bias
1.0
Accuracy
Number of training samples


High variance
Accuracy
Number of training samples


Training accuracy
Validation accuracy
Desired accuracy


Good bias-variance trade-off
Accuracy
Number of training samples

Python

- 繪製學習曲線 **(6/14)**

```
import matplotlib.pyplot as plt
from sklearn.model_selection import learning_curve

lr = LogisticRegression(penalty='l2', random_state=1)

train_sizes, train_scores, test_scores = learning_curve(estimator=lr,
                                        X=X_train, y=y_train,
                                        train_sizes=np.linspace(0.1, 1.0, 10),
                                        cv=10, n_jobs=1)

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

plt.plot(train_sizes, train_mean,
         color='blue', marker='o',
         markersize=5, label='training accuracy')

plt.fill_between(train_sizes,
                 train_mean + train_std,
                 train_mean - train_std,
                 alpha=0.15, color='blue')
```

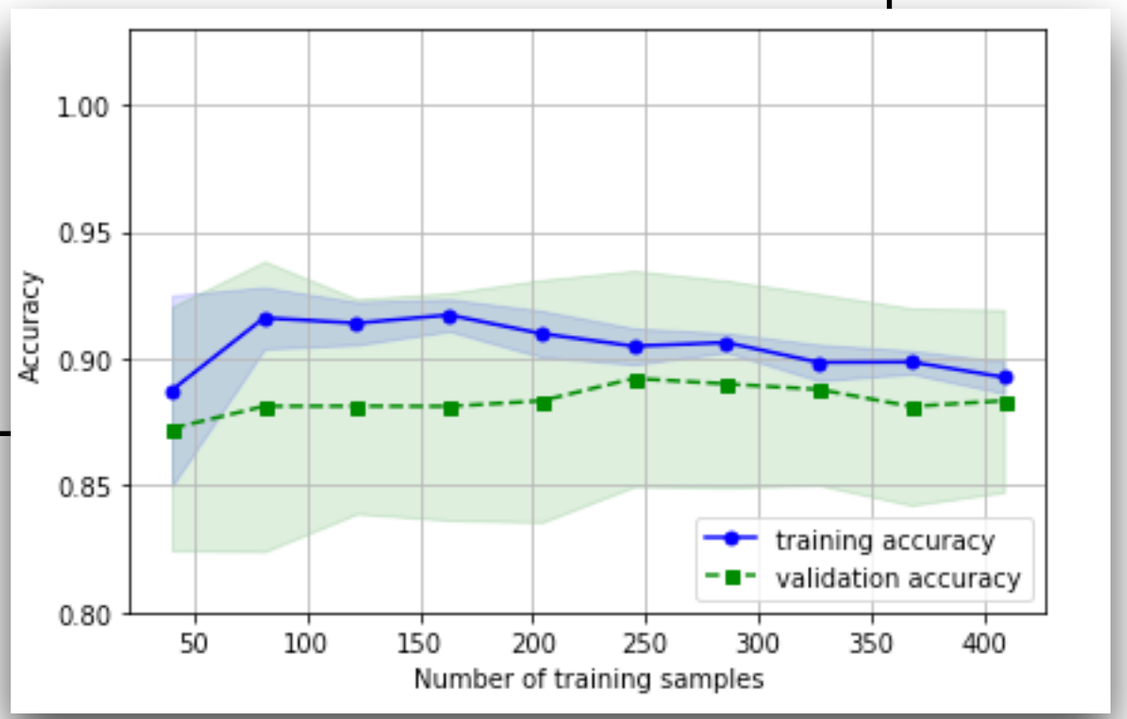
設定10個相對均勻的訓練樣本區間

在圖形中畫出平均正確率的標準差

Python

- 繪製學習曲線 **(7/14)**

```
plt.plot(train_sizes, test_mean,
         color='green', linestyle='--',
         marker='s', markersize=5,
         label='validation accuracy')

plt.fill_between(train_sizes,
                 test_mean + test_std,
                 test_mean - test_std,
                 alpha=0.15, color='green')

plt.grid()
plt.xlabel('Number of training samples')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.ylim([0.8, 1.03])
plt.tight_layout()
#plt.savefig('images', dpi=300)
plt.show()
```



Python

- 繪製驗證曲線

```
from sklearn.model_selection import validation_curve

# ## Addressing over- and underfitting with validation curves

param_range = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0]
train_scores, test_scores = validation_curve(estimator=lr, X=X_train,
                y=y_train, param_name='C',
                param_range=param_range, cv=10)

train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)

plt.plot(param_range, train_mean,
         color='blue', marker='o',
         markersize=5, label='training accuracy')

plt.fill_between(param_range, train_mean + train_std,
                 train_mean - train_std, alpha=0.15,
                 color='blue')
```

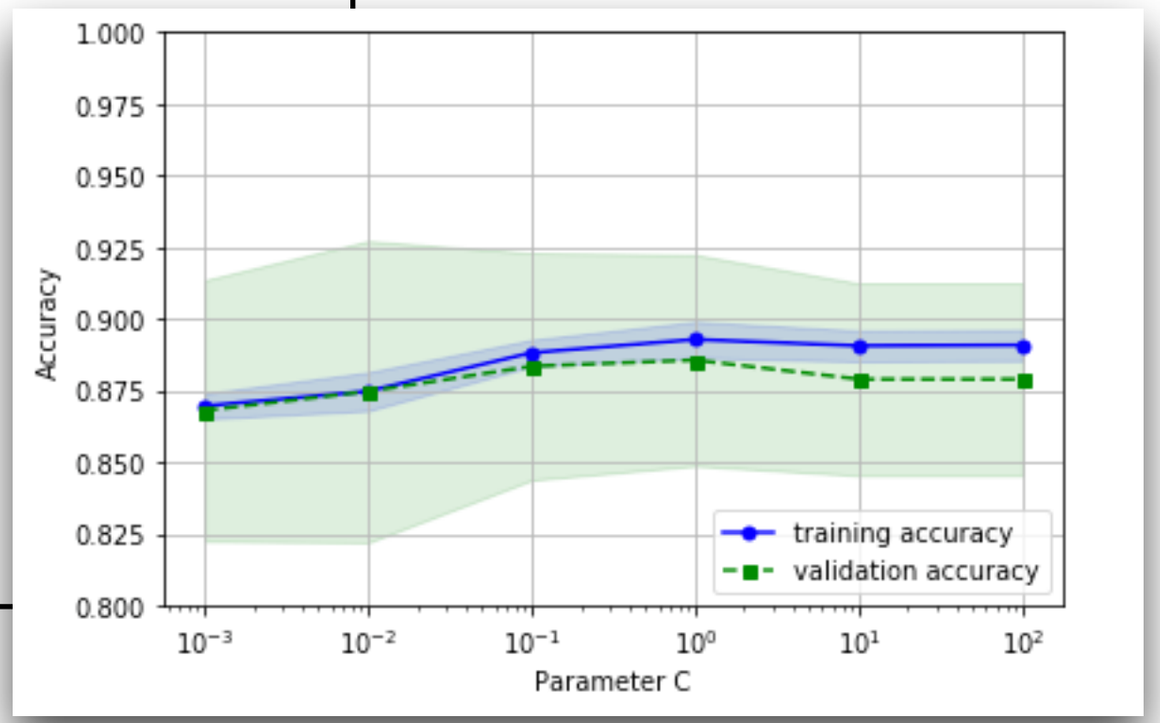
設定參數的範圍

指定想要評估的參數

Python

- 繪製驗證曲線 (9/14)

```
plt.plot(param_range, test_mean,
         color='green', linestyle='--',
         marker='s', markersize=5,
         label='validation accuracy')

plt.fill_between(param_range,
                 test_mean + test_std,
                 test_mean - test_std,
                 alpha=0.15, color='green')

plt.grid()
plt.xscale('log')
plt.legend(loc='lower right')
plt.xlabel('Parameter C')
plt.ylabel('Accuracy')
plt.ylim([0.6, 1.0])
plt.tight_layout()
# plt.savefig('images.png', dpi=300)
plt.show()
```



Python

- 用 scikit-learn 計算計分指標

```
from sklearn.metrics import precision_score, recall_score, f1_score



print('Precision: %.3f' % precision_score(y_true=y_test, y_pred=y_pred))
print('Recall: %.3f' % recall_score(y_true=y_test, y_pred=y_pred))
print('F1: %.3f' % f1_score(y_true=y_test, y_pred=y_pred))
```


Note: scikit-learn 的 P 是使用類別標籤 1的類別