# Bài 5: XÁC NHẬN CHÉO – k-fold CV

*(k-fold cross validation with sklearn)*

## 1. Chuẩn bị dữ liệu

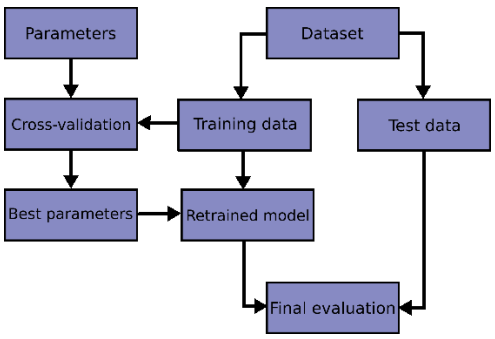
Cho tập dữ liệu gốc $D = \{(X, y) \mid X \in R^{m \times n}, y \in R^m, m \text{ và } n \in N\}$, tập dữ liệu D có thể được chuẩn hóa trước khi phân chia thành tập dữ liệu huấn luyện (training set) và tập dữ liệu kiểm thử (test set). Lưu ý:

- Đối với bài toán hồi quy (tuyến tính/phi tuyến): có thể tiến hành chuẩn hóa toàn bộ tập *D* (bao gồm cả *X* và *y*);
- Đối với bài toán phân lớp (lưu ý: mô hình hồi quy logistic được xếp vào nhóm mô hình phân lớp), chỉ tiến hành chuẩn hóa tập dữ liệu *X* của *D*.

Tập *D* sẽ được phân chia thành tập $D_{train}$ và $D_{test}$ theo một tỉ lệ định trước (thông thường là *70%-30%* hoặc *80%-20%*, trường hợp đặc biệt là *90%-10%*) tùy theo trường hợp cụ thể. Tập dữ liệu $D_{train}$ tiếp theo sẽ được sử dụng để huấn luyện mô hình. Phần còn lại của tài liệu này trình bày phương pháp huấn luyện mô hình bằng *k-fold cross validation theo tài liệu hướng dẫn của sklearn*.

## 2. Thực hiện k-fold CV

Mục đích của quá trình huấn luyện k-fold CV là nhằm tránh hiện tượng **overfitting** bằng cách tìm ra bộ tham số tối ưu của mô hình thông qua quá trình huấn luyện k-fold CV. Quá trình này có thể minh họa bằng Hình 1.



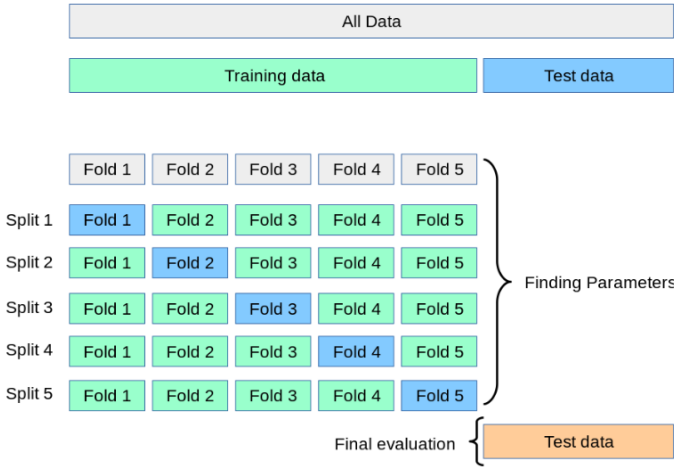

*Hình 1: Quy trình huấn luyện mô hình theo k-fold CV (nguồn: sklearn)*

Đối với k-fold CV, tập dữ liệu $D_{train}$ sẽ được sử dụng toàn bộ mà không trích lại một phần nhỏ để làm tập dữ liệu xác nhận ($D_{validation}$). Thay vào đó $D_{train}$ sẽ được chia thành $k$ – tập dữ liệu nhỏ hơn – gọi là k-fold. Một nhóm $k$-fold sẽ được sử dụng như sau:

- Mô hình sử dụng nhóm ($k$-$1$) – fold để huấn luyện;

- Phần còn lại (fold còn lại) được sử dụng để đánh giá mô hình.

Quá trình này được lặp lại nhiều lần và hiệu năng của mô hình được tính là trung bình cộng hiệu năng trên mỗi lần lặp huấn luyện. Hình 2 minh họa quá trình k-fold CV với k = 5.




*Hình 2: 5-fold CV (nguồn: sklearn)*

## 2.1. Đánh giá hiệu năng mô hình huấn luyện theo k-fold CV

Ở mức đơn giản nhất, để huấn luyện mô hình với k-fold CV, thư viện sklearn cung cấp hàm ***cross_val_score*** để huấn luyện mô hình và cung cấp các chỉ số đánh giá hiệu năng mô hình được huấn luyện.

*Bảng 1: 5-fold CV với mô hình SVM dùng cross_val_score*

```
>>> from sklearn.model_selection import cross_val_score
>>> clf = svm.SVC(kernel='linear', C=1, random_state=42)
>>> scores = cross_val_score(clf, X, y, cv=5)
>>> scores
array([0.96..., 1. , 0.96..., 0.96..., 1. ])
```

## 2.2. Sử dụng kết quả dự đoán của mô hình huấn luyện theo quy trình k-fold CV

Công cụ cross_val_score không cung cấp kết quả dự đoán của mô hình, để lấy kết quả dự đoán của mô hình huấn luyện bởi k-fold CV, sklearn cung cấp công cụ ***cross_val_predict***.

*Bảng 2: Huấn luyện mô hình hồi quy Lasso với 3-fold CV và lấy kết quả dự đoán*

```
>>> from sklearn import datasets, linear_model
>>> from sklearn.model_selection import cross_val_predict
>>> diabetes = datasets.load_diabetes()
>>> X = diabetes.data[:150]
>>> y = diabetes.target[:150]
>>> lasso = linear_model.Lasso()
>>> y_pred = cross_val_predict(lasso, X, y, cv=3)
```

## 3. Các phương pháp chia dữ liệu trong k-fold CV

### 3.1. Phương pháp KFold

Phương pháp KFold chia $D_{train}$ thành $k$ nhóm có kích thước bằng nhau. Mô hình sẽ được huấn luyện với ($k$-$1$) nhóm và được kiểm thử với nhóm còn lại.

*Bảng 3: 2-fold CV trên tập dữ liệu có 4 mẫu*

```
>>> import numpy as np
>>> from sklearn.model_selection import KFold

>>> X = ["a", "b", "c", "d"]
>>> kf = KFold(n_splits=2)
>>> for train, test in kf.split(X):
...     print("%s %s" % (train, test))
[2 3] [0 1]
[0 1] [2 3]
```

### 3.2. Công cụ KFold lặp (Repeated KFold)

Phương pháp RepeatedKFold lặp lại KFold $n$ lần.

*Bảng 4: 2-fold CV được lặp lại 2 lần*

```
>>> import numpy as np
>>> from sklearn.model_selection import RepeatedKFold
>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
>>> random_state = 12883823
>>> rkf = RepeatedKFold(n_splits=2, n_repeats=2, random_state=random_state)
>>> for train, test in rkf.split(X):
...     print("%s %s" % (train, test))
...
[2 3] [0 1]
```

```
[0 1] [2 3]
[0 2] [1 3]
[1 3] [0 2]
```

### 3.3. Phương pháp LOO – Leave One Out

Phương pháp này thực hiện một cách đơn giản bằng cách lấy ra 1 mẫu dữ liệu trong tập dữ liệu huấn luyện làm tập dữ liệu kiểm thử. Với 1 tập dữ liệu $n$-mẫu, phương pháp LOO sẽ cho ra $n$ tập dữ liệu huấn luyện và $n$-tập dữ liệu kiểm thử khác nhau.

*Bảng 5: Ví dụ LOO với tập dữ liệu có 4 mẫu*

```
>>> from sklearn.model_selection import LeaveOneOut

>>> X = [1, 2, 3, 4]
>>> loo = LeaveOneOut()
>>> for train, test in loo.split(X):
...     print("%s %s" % (train, test))
[1 2 3] [0]
[0 2 3] [1]
[0 1 3] [2]
[0 1 2] [3]
```

### 3.4. Phương pháp LPO – Leave P Out

Phương pháp LPO là trường hợp tổng quát hơn so với LOO. Trong đó với 1 tập dữ liệu $n$-mẫu, LPO sẽ để ra $p$-mẫu ($p < n$) để làm tập dữ liệu kiểm thử. Do đó, LPO sẽ tạo ra $\binom{n}{p} = \frac{n!}{p!(n-p)!}$ các cặp train – test khác nhau.

*Bảng 6: Leave-2-Out trên tập dữ liệu 4 - mẫu*

```
>>> from sklearn.model_selection import LeavePOut

>>> X = np.ones(4)
>>> lpo = LeavePOut(p=2)
>>> for train, test in lpo.split(X):
...     print("%s %s" % (train, test))
[2 3] [0 1]
[1 3] [0 2]
[1 2] [0 3]
[0 3] [1 2]
[0 2] [1 3]
[0 1] [2 3]
```

### 3.5. Phương pháp hoán vị ngẫu nhiên - ShuffleSplit

ShuffleSplit thực hiện xáo tập $D_{train}$ sau đó chọn ngẫu nhiên 1 lượng $p\%$ từ $D_{train}$ làm tạp dữ liệu kiểm thử. Phần còn lại làm tập dữ liệu huấn luyện.

*Bảng 7: ShuffleSplit với test size 25%*

```
>>> from sklearn.model_selection import ShuffleSplit
>>> X = np.arange(10)
>>> ss = ShuffleSplit(n_splits=5, test_size=0.25, random_state=0)
>>> for train_index, test_index in ss.split(X):
...     print("%s %s" % (train_index, test_index))
[9 1 6 7 3 0 5] [2 8 4]
[2 9 8 0 6 7 4] [3 5 1]
[4 5 1 0 6 9 7] [2 3 8]
[2 7 5 8 0 3 4] [6 1 9]
[4 1 0 6 8 9 3] [5 2 7]
```

## 4. Thực hiện k-fold CV trong trường hợp số lượng các nhãn lớp không cân bằng

Trong thực tế vector nhãn lớp $y$ tập dữ liệu $D$ hay cụ thể hơn là $D_{train}$ thường chứa số lượng nhãn lớp không cân bằng (đối với bài toán phân lớp). Vậy làm thế nào để thực hiện phân chia train – test trong quá trình huấn luyện k-fold CV mà vẫn giữ được tỉ lệ các nhãn lớp như tập dữ liệu gốc là điều cần quan tâm.

### 4.1. Phương pháp StratifiedKFold

Phương pháp StratifiedKFold là một biến thể của KFold áp dụng cho trường hợp tập dữ liệu có số lượng nhãn lớp không cân bằng.

*Bảng 8: Minh họa StratifiedKFold với dữ liệu giả lập*

```
>>> from sklearn.model_selection import StratifiedKFold, KFold
>>> import numpy as np
>>> X, y = np.ones((50, 1)), np.hstack(([0] * 45, [1] * 5))
>>> skf = StratifiedKFold(n_splits=3)
>>> for train, test in skf.split(X, y):
...     print('train -  {}   |   test -  {}'.format(
...         np.bincount(y[train]), np.bincount(y[test])))
train -  [30  3]   |   test -  [15  2]
train -  [30  3]   |   test -  [15  2]
train -  [30  4]   |   test -  [15  1]
>>> kf = KFold(n_splits=3)
>>> for train, test in kf.split(X, y):
...     print('train -  {}   |   test -  {}'.format(
...         np.bincount(y[train]), np.bincount(y[test])))
```

```
train -  [28  5]   |   test -  [17]
train -  [28  5]   |   test -  [17]
train -  [34]   |   test -  [11  5]
```

### 4.2. Phương pháp StratifiedShuffleSplit

Phương pháp StratifiedShuffleSplit là biến thể của phương pháp ShuffleSplit nhằm đảm bảo tỉ lệ nhãn lớp được duy trì.

*Bảng 9: Ví dụ về StratifiedShuffleSplit*

```
>>> import numpy as np
>>> from sklearn.model_selection import StratifiedShuffleSplit
>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4], [1, 2], [3, 4]])
>>> y = np.array([0, 0, 0, 1, 1, 1])
>>> sss = StratifiedShuffleSplit(n_splits=5, test_size=0.5, random_state=0)
>>> sss.get_n_splits(X, y)
5
>>> print(sss)
StratifiedShuffleSplit(n_splits=5, random_state=0, ...)
>>> for train_index, test_index in sss.split(X, y):
...     print("TRAIN:", train_index, "TEST:", test_index)
...     X_train, X_test = X[train_index], X[test_index]
...     y_train, y_test = y[train_index], y[test_index]
TRAIN: [5 2 3] TEST: [4 1 0]
TRAIN: [5 1 4] TEST: [0 2 3]
TRAIN: [5 0 2] TEST: [4 3 1]
TRAIN: [4 1 0] TEST: [2 3 5]
TRAIN: [0 5 1] TEST: [3 4 2]
```

## 5. Thực hiện k-fold CV với dữ liệu chia nhóm

Trong thực tế, dữ liệu đôi khi được thu thập theo từng nhóm. Ví dụ, dữ liệu y khoa lấy từ một nhóm bệnh nhân. Trong đó, mỗi bệnh nhân được thu thập nhiều mẫu bệnh phẩm. Hay nói cách khác dữ liệu của mỗi bệnh nhân hình thành nên 1 nhóm mẫu cho bệnh nhân đó. Một số phương pháp phổ biến áp dung trong trường hợp này như sau

| | |
|---|---|
| GroupKFold | `>>> from sklearn.model_selection import GroupKFold`<br><br>`>>> X = [0.1, 0.2, 2.2, 2.4, 2.3, 4.55, 5.8, 8.8, 9, 10]`<br>`>>> y = ["a", "b", "b", "b", "c", "c", "c", "d", "d", "d"]`<br>`>>> groups = [1, 1, 1, 2, 2, 2, 3, 3, 3, 3]`<br><br>`>>> gkf = GroupKFold(n_splits=3)`<br>`>>> for train, test in gkf.split(X, y, groups=groups):`<br>`...  print("%s %s" % (train, test))`<br>`[0 1 2 3 4 5] [6 7 8 9]`<br>`[0 1 2 6 7 8 9] [3 4 5]`<br>`[3 4 5 6 7 8 9] [0 1 2]` |
| StratifiedGroupKFold | `>>> from sklearn.model_selection import StratifiedGroupKFold`<br>`>>> X = list(range(18))`<br>`>>> y = [1] * 6 + [0] * 12`<br>`>>> groups = [1, 2, 3, 3, 4, 4, 1, 1, 2, 2, 3, 4, 5, 5, 5, 6, 6, 6]`<br>`>>> sgkf = StratifiedGroupKFold(n_splits=3)`<br>`>>> for train, test in sgkf.split(X, y, groups=groups):`<br>`...  print("%s %s" % (train, test))`<br>`[ 0  2  3  4  5  6  7 10 11 15 16 17] [ 1  8  9 12 13 14]`<br>`[ 0  1  4  5  6  7  8  9 11 12 13 14] [ 2  3 10 15 16 17]`<br>`[ 1  2  3  8  9 10 12 13 14 15 16 17] [ 0  4  5  6  7 11]` |
| GroupShuffleSplit | `>>> from sklearn.model_selection import GroupShuffleSplit`<br><br>`>>> X = [0.1, 0.2, 2.2, 2.4, 2.3, 4.55, 5.8, 0.001]`<br>`>>> y = ["a", "b", "b", "b", "c", "c", "c", "a"]`<br>`>>> groups = [1, 1, 2, 2, 3, 3, 4, 4]`<br>`>>> gss = GroupShuffleSplit(n_splits=4, test_size=0.5, random_state=0)`<br>`>>> for train, test in gss.split(X, y, groups=groups):`<br>`...  print("%s %s" % (train, test))`<br>`...`<br>`[0 1 2 3] [4 5 6 7]`<br>`[2 3 6 7] [0 1 4 5]`<br>`[2 3 4 5] [0 1 6 7]`<br>`[4 5 6 7] [0 1 2 3]` |

## 6. Thực hiện k-fold CV với dữ liệu chuỗi thời gian

Trong thực tế, dữ liệu gắn với chiều thời gian (hay dữ liệu xuất hiện theo thứ tự), xuất hiện rất nhiều trong cuộc sống. Việc phân chia dữ liệu dạng này trong quá trình thực hiện k-fold CV đòi hỏi phải sử dụng phương pháp riêng. Thư viện sklearn cung cấp công cụ TimeSeriesSplit để thực hiện nhiệm vụ này.

*Bảng 10: 3-fold CV trên tập dữ liệu chuỗi thời gian 6 mẫu*

```
>>> from sklearn.model_selection import TimeSeriesSplit

>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4], [1, 2], [3, 4]])
```

```
>>> y = np.array([1, 2, 3, 4, 5, 6])
>>> tscv = TimeSeriesSplit(n_splits=3)
>>> print(tscv)
TimeSeriesSplit(gap=0, max_train_size=None, n_splits=3, test_size=None)
>>> for train, test in tscv.split(X):
...     print("%s %s" % (train, test))
[0 1 2] [3]
[0 1 2 3] [4]
[0 1 2 3 4] [5]
```