# Bài 4: HUẤN LUYỆN MÔ HÌNH VÀ LỖI

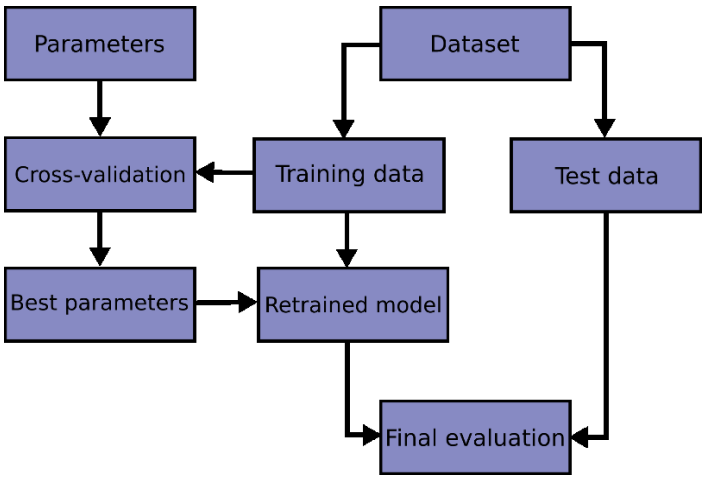
*Hướng dẫn 4.2: k-fold cross-validation với mô hình Hồi quy Logistic*

## 1. Quy trình xây dựng k-fold cross-validation

Sau khi chia tập dữ liệu *D* ban đầu thành 2 tập train – test với tỉ lệ thông thường là 70% - 30%, quá trình huấn luyện diễn ra với tập $D_{train}$ được tiến hành bằng cách lấy ra từ $D_{train}$ một phần nhỏ dữ liệu làm *tập dữ liệu kiểm thử cho bước huấn luyện thứ i – gọi là* $D_{validation}$. Phần dữ liệu còn lại đóng vai trò là tập dữ liệu huấn luyện. Quá trình này lặp lại nhiều lần cho đến khi mô hình huấn luyện đạt được yêu cầu. Tóm lại, quy trình thực hiện *k-fold cross-validation* (với k thông thường được chọn k = 10 – nên gọi là *10-fold cross-validation*) được tóm tắt như sau:

- Bước 1: Đọc dữ liệu gốc *D*
- Bước 2: Điều chỉnh dữ liệu
- Bước 3: Chia train – test theo tỉ lệ (thông thường là 70% - 30%)
- Bước 4: Xác định k
- Bước 5: Thực hiện huấn luyên mô hình với k-fold cross validation
- Bước 6: Kiểm định mô hình với tập dữ liệu test

Minh họa quy trình này với 5-fold cross-validation theo ví dụ của sklearn




*Hình 1: Minh họa quy trình 5-fold cross-validation (nguồn: sklearn)*

**2. Huấn luyện mô hình hồi quy Logistic với 10-fold cv sử dụng sklearn**

Áp dụng *10-fold CV* trong quá trình huấn luyện mô hình phân lớp *LogisticRegression* sử dụng thư viện sklearn. Những kỹ thuật cơ bản sau cần được hoàn thiện:

| **Kỹ thuật** | **Mã lệnh Python** |
|---|---|
| Bài 1:<br>- Đọc dữ liệu (***ex2data2.txt***);<br>- Chuẩn hóa dữ liệu (lưu ý: chỉ chuẩn hóa ***X***);<br>- Phân chia train – test theo tỉ lệ ***70% - 30%***;<br>- Xây dựng mô hình ***hồi quy Logistic*** sử dụng sklearn với thuật toán tối ưu là ***liblinear***; số bước lặp là ***1500***; thiết lập chế độ nhãn lớp là ***auto***;<br>- Huấn luyện mô hình sử dụng ***cross_val_score()*** với lựa chọn k-fold cv là ***10***, đánh giá mô hình bằng chỉ số ***accuracy*** và áp dụng với tập dữ liệu (***X_train, y_train***);<br>- In kết quả huấn luyện ra màn hình. | *Tham khảo lời giải gợi ý* ***hd4_2_kfold_lg.py*** *và viết chương trình dưới dạng hàm hoàn chỉnh.* |
| Bài 2:<br>- Đọc dữ liệu (***ex2data2.txt***);<br>- Chuẩn hóa dữ liệu (lưu ý: chỉ chuẩn hóa ***X***);<br>- Phân chia train – test theo tỉ lệ ***90% - 10%***;<br>- Xây dựng mô hình ***hồi quy Logistic*** sử dụng sklearn với thuật toán tối ưu là ***liblinear***; số bước lặp là ***1500***; thiết lập chế độ nhãn lớp là ***auto***;<br>- Thiết lập ***10-fold cv*** bằng ***ShuffleSplit*** với tùy chọn ***n_splits = 10, test_size = 20%***;<br>- Huấn luyện mô hình sử dụng ***cross_val_score()*** với cv được dùng là ShuffleSplit nêu trên, đánh giá mô hình bằng chỉ số ***accuracy*** và áp dụng với tập dữ liệu (***X_train, y_train***);<br>- In kết quả huấn luyện ra màn hình. | *Tham khảo lời giải gợi ý* ***hd4_3_kfold_lg_sklearn.py*** *và viết chương trình dưới dạng hàm hoàn chỉnh.* |
| Bài 3:<br>- Đọc dữ liệu (***ex2data2.txt***);<br>- Chuẩn hóa dữ liệu (lưu ý: chỉ chuẩn hóa ***X***); | *Tham khảo lời giải gợi ý* ***hd4_4_kfold_lg_sklearn.py*** *và viết chương trình dưới dạng hàm hoàn chỉnh.* |

| | |
|---|---|
| - Phân chia train – test theo tỉ lệ ***70% - 30%***;<br>- Xây dựng mô hình ***hồi quy Logistic*** sử dụng sklearn với thuật toán tối ưu là ***liblinear***; số bước lặp là ***1500***; thiết lập chế độ nhãn lớp là ***auto***;<br>- Huấn luyện mô hình ***10-fold cv*** với ***cross_val_predict()*** áp dụng với tập dữ liệu (***X_train, y_train***);<br>- In kết quả dự đoán (***y_hat***) ra màn hình. | |
| Bài 4: Sử dụng ***GridSearchCV*** để tìm bộ tham số tối ưu cho mô hình. Với ***hồi quy Logistic***, chúng ta tìm giá trị tối ưu cho tham số ***C*** áp dụng trong quá trình điều tiết (***regularization***) tránh hiện tượng ***overfitting***.<br>- Đọc dữ liệu (***ex2data2.txt***);<br>- Chuẩn hóa dữ liệu (lưu ý: chỉ chuẩn hóa ***X***);<br>- Phân chia train – test theo tỉ lệ ***70% - 30%***;<br>- Xây dựng mô hình ***hồi quy Logistic*** sử dụng sklearn với thuật toán tối ưu là ***liblinear***; số bước lặp là ***1500***; thiết lập chế độ nhãn lớp là ***auto***;<br>- Tạo bộ tham số C có giá trị 1, 10, 20, 50;<br>- Tạo GridSearchCV với mô hình hồi quy Logistic, cv = 10 và bộ tham số ở trên;<br>- Áp dụng với (X_train, y_train)<br>- In ra màn hình giá trị C tối ưu (***best_params_***) của mô hình GridSearchCV ở trên;<br>- Sử dụng mô hình GridSearchCV để dự đoán y_hat cho tập X_test<br>- Đánh giá hiệu năng của mô hình bằng ***accuracy_score*** đối với tập (***y_hat, y_test***). | *Tham khảo lời giải gợi ý* ***hd4_5_grid_lg_sklearn.py*** *và viết chương trình dưới dạng hàm hoàn chỉnh.* |
| Bài 5: Sử dụng ***LogisticRegressionCV*** để đơn giản hóa quá trình huấn luyện mô hình với k-fold cross-validation.<br>- Đọc dữ liệu (***ex2data2.txt***);<br>- Chuẩn hóa dữ liệu (lưu ý: chỉ chuẩn hóa ***X***); | *Tham khảo lời giải gợi ý* ***hd4_6_lgcv_sklearn.py*** *và viết chương trình dưới dạng hàm hoàn chỉnh.* |

| | |
|---|---|
| - Phân chia train – test theo tỉ lệ ***70% - 30%***;<br>- Khởi tạo mô hình LogisticRegressionCV với tùy chọn cv=10, random_state = <số nguyên tùy ý bạn> và áp dụng với tập (X_train, y_train);<br>- Sử dụng mô hình ở trên để dự đoán ***y_hat*** đối với tập ***X_test***;<br>- Đánh giá hiệu năng của mô hình bằng ***accuracy_score*** đối với tập (***y_hat, y_test***). | |

**3. Câu hỏi mở rộng**

1. Mở rộng Bài 1, hãy thay thế thuật toán tối ưu khác cho mô hình LogisticRegression. Tham khảo tại ***phần tham số solver***.
2. Mở rộng Bài 1, hãy thay thế chỉ số đánh giá hiệu năng mô hình. Tham khảo ***tại đây***.