THỰC HÀNH 1: BÀI TOÁN PHÂN LỚP

Mục tiêu: Hiểu được các bước cơ bản để huấn luyện mộ dự án máy học cho bài toán phân lớp ảnh.

1. Đọc và tìm hiểu cấu trúc của dữ liệu.

Bộ dữ liệu: Iris. Bộ dữ liệu này chứa khoảng 150 hình về các loài hoa. Có tổng cộng 3 loài hoa gồm: Setosa, Versicolour, and Virginica.

Mỗi điểm dữ liệu gồm 4 thuộc tính là

- Sepal Length: chiều dài của đài hoa.
- Sepal Width: chiều rộng của đài hoa.
- Petal Length: chiều dài của cánh hoa.
- Petal Width: chiều rộng của cánh hoa.

Bộ dữ liệu này được hỗ trợ sẵn bởi sklearn. Cách đọc dữ liệu như sau:

```
from sklearn.datasets import load_iris
iris = load_iris()
```

Dữ liệu huấn luyện cho bài toán phân lớp sẽ gồm 2 phần:

- X: các thuộc tính của dữ liệu.
- y: thuộc tính nhãn.

Trong bài tập thực hành này, ta chỉ sử dụng 2 thuộc tính là Sepal Length và Sepal Width để làm thuộc tính của dữ liệu. Do đó, ta thực hiện như sau:

```
X = iris.data[:, :2] # đối với X, ta chỉ sử dụng 2 thuộc
tính sepal length và sepal width để dự đoán cho dữ liệu.
y = iris.target      # y: nhãn, gồm 3 nhãn
```

Để xem chiều của dữ liệu X, ta dùng lệnh sau: `X.shape`

Kết quả sẽ là `(150,2)`. Như vậy, X gồm 150 điểm dữ liệu, mỗi điểm dữ liệu có 2 thuộc tính. X sẽ được biểu diễn dưới dạng 1 ma trận 150 dòng và 2 cột.

Tương tự, để xem chiều của dữ liệu y, ta dùng lệnh: `y.shape`.

Kết quả sẽ là: `(150, )`. Như vậy, y là 1 danh sách gồm có 150 phần tử, mỗi phần tử thuộc 1 trong 3 nhãn của bộ dữ liệu. y sẽ được biểu diễn dưới dạng vector cột.

Ghi chú: trong bộ dữ liệu, các nhãn đã được mã hoá như sau:

- 0: Setosa
- 1: Versicolour
- 2: Virginica

2. Phân chia dữ liệu

Từ tập dữ liệu ban đầu, ta sẽ phân chia ra làm 2 phần:

- Tập huấn luyện (train): dùng để huấn luyện cho mô hình.
- Tập kiểm thử (test): dùng để kiểm tra độ chính xác của mô hình.

Tỉ lệ giữa tập huấn luyện (train) và kiểm thử (test) thường được dùng là 8 - 2. Tức là 80% dữ liệu sẽ dùng cho huấn luyện (train) và 20% dữ liệu sẽ dùng cho kiểm thử (test).

Để phân chia dữ liệu, ta dùng hàm train_test_split() trong thư viện sklearn như sau:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2)
```

Hàm ***train_test_split()*** sẽ nhận vào 3 tham số: X là biến thuộc tính, y là nhãn của dữ liệu, và test_size là tỉ lệ của tập test.

Hàm train_test_split sẽ trả về một bộ (tuple) gồm 4 giá trị:

- X_train: thuộc tính của tập huấn luyện.
- X_test: thuộc tính của tập kiểm thử.
- y_train: nhãn của tập huấn luyện.
- y_test: nhãn của tập kiểm thử.

Các bạn hãy cho biết chiều (shape) của từng tập dữ liệu sau khi đã phân chia ra huấn luyện và kiểm thử.

3. Huấn luyện mô hình và dự đoán

Sử dụng mô hình LogisticRegression trong thư viện sklearn để huấn luyện trên dữ liệu huấn luyện (X_train, y_train). Mô hình được lưu vào biến model.

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train, y_train)
```

Dự đoán ra kết quả cho dữ liệu kiểm thử X_test và lưu vào biến y_pred. Sử dụng hàm ***predict().***

```
y_pred = model.predict(X_test)
```

4. Đánh giá khả năng dự đoán của mô hình

Để đánh giá khả năng dự đoán của mô hình, ta tiến hành so khớp giữa kết quả dự đoán (nhãn) của mô hình trên dữ liệu kiểm thử (y_pred) và nhãn thực sự của dữ liệu (y_test), từ đó kết luận khả năng dự đoán của mô hình.

Để định lượng được khả năng dự đoán của mô hình, ta cần dùng các độ đo để đánh giá (xem lại bài **Phân lớp** để ôn lại các độ đo).

Trong ví dụ này, ta sử dụng độ chính xác (Accuracy) để đánh giá cho mô hình. Sử dụng độ đo accuracy trong thư viện sklearn như sau:

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)*100
```

5. Bài tập

Bài 1: Hãy thống kê số lượng nhãn (label) trên tập training và tập test vừa chia. Vẽ biểu đồ phân bố nhãn (Gợi ý: sử dụng ***barplot*** trong thư viện ***seaborn***).

Bài 2: Thực hiện huấn luyện mô hình Logistic Regression trên bộ dữ liệu (tham khảo theo các bước đã hướng dẫn).

Bài 3: Thực hiện huấn luyện mô hình K láng giềng gần nhất (KNN) trên bộ dữ liệu, sau đó so sánh độ chính xác (Accuracy) với mô hình LogisticRegression.

Bài 4: Đánh giá 2 mô hình vừa xây dựng trên 3 độ đo sau: precision_score, recall_score và f1_score sử dụng macro average.

Bài 5*: Hãy sử dụng chiến lược tinh chỉnh tham số GridSearchCV để tìm ra bộ tham số tốt nhất cho mô hình Logistic Regression. So sánh kết quả với mô hình gốc.