

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**HỌC MÁY THỐNG KÊ**  
**LAB 2**

**Họ và tên : Lưu Quang Tiến Hoàng**

**MSSV : 20521342**

**Lớp : DS102.M21**

**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).**

```
In [142]: import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
import os
import numpy as np
value_counts = pd.DataFrame(y_train).value_counts().sort_index(ascending=True)
```

```
In [143]: value_counts
```

```
Out[143]: 0    36
          1    39
          2    45
          dtype: int64
```

```
In [144]: dataframe_value_counts = pd.DataFrame(value_counts)
dataframe_value_counts.columns = ['counts']
dataframe_value_counts = dataframe_value_counts.reset_index()
```

```
In [145]: dataframe_value_counts.columns = ['label', 'counts']
```

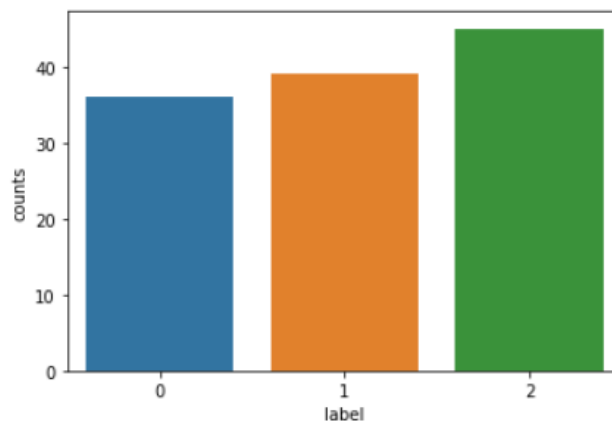
```
In [146]: dataframe_value_counts
```

```
Out[146]:
```

	label	counts
0	0	36
1	1	39
2	2	45

```
In [147]: sns.barplot(x='label', y='counts', data=dataframe_value_counts)
```

```
Out[147]: <AxesSubplot:xlabel='label', ylabel='counts'>
```



Hình 1. Số lượng nhãn của tập train

```
In [148]: value_counts_test = pd.DataFrame(y_test).value_counts().sort_index(ascending=True)
```

```
In [149]: dataframe_value_counts_test = pd.DataFrame(value_counts_test)
dataframe_value_counts_test.columns = ['counts']
dataframe_value_counts_test = dataframe_value_counts_test.reset_index()
```

```
In [150]: dataframe_value_counts_test.columns = ['label', 'counts']
```

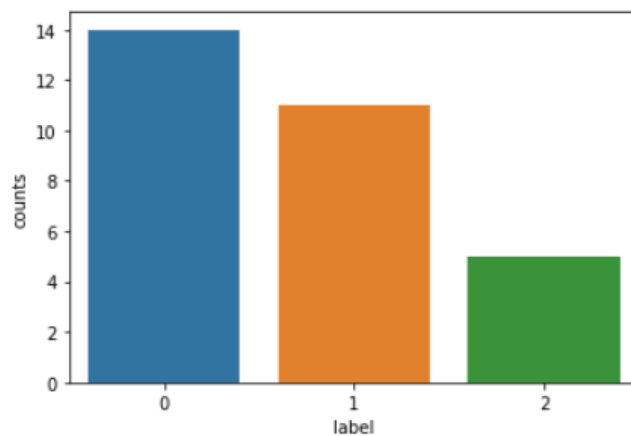
```
In [151]: dataframe_value_counts_test
```

Out[151]:

	label	counts
0	0	14
1	1	11
2	2	5

```
In [152]: sns.barplot(x='label', y='counts', data=dataframe_value_counts_test)
```

Out[152]: <AxesSubplot:xlabel='label', ylabel='counts'>



Hình 2. Số lượng nhãn tập Test

## Bài 2: Thực hiện huấn luyện mô hình Logistic Regression trên bộ dữ liệu

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

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

```
X.shape
```

```
(150, 2)
```

```
y.shape
```

```
(150,)
```

```
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)
```

```
X_train.shape
```

```
(120, 2)
```

```
X_test.shape
```

```
(30, 2)
```

```
y_train.shape
```

```
(120,)
```

```
y_test.shape
```

```
(30,)
```

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

```
LogisticRegression()
```

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

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

```
83.33333333333334
```

Hình 3. Thực hiện mô hình huấn luyện Logistic Regression

### 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.

```
: from sklearn.neighbors import KNeighborsClassifier
: modell1 = KNeighborsClassifier()
: modell1.fit(X_train,y_train)

: KNeighborsClassifier()

: y_pred_2 = modell1.predict(X_test)

: y_pred_2
: array([2, 0, 0, 0, 2, 0, 1, 1, 1, 1, 0, 0, 2, 2, 2, 1, 0, 1, 1, 1, 0, 0,
:        1, 0, 1, 0, 0, 2, 1, 0])

: y_test
: array([2, 0, 0, 0, 2, 0, 1, 2, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 2, 0, 0, 0,
:        1, 0, 1, 0, 0, 2, 1, 0])

: y_pred
: array([2, 0, 0, 0, 1, 0, 1, 2, 1, 2, 0, 0, 1, 2, 1, 1, 0, 1, 1, 1, 0, 0,
:        1, 0, 1, 0, 0, 2, 1, 0])

: accuracy_score(y_test, y_pred_2)*100
: 80.0
```

Hình 4. Thực hiện huấn luyện mô hình K láng giềng gần nhất (KNN)

\*Nhận xét: Sau mỗi lần chạy tất cả với mỗi lần chia bộ train và bộ test khác nhau thì chênh lệch giữa độ chính xác của mô hình KNN và LogisticRegression khác nhau nên độ chính xác của mô hình KNN so với LogisticRegression phụ thuộc vào đặc điểm của bộ dữ liệu.

**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.**

```
from sklearn.metrics import precision_score
precision_score(y_test, y_pred_2, average='macro')*100
```

74.24242424242425

```
from sklearn.metrics import recall_score
recall_score(y_test, y_pred_2, average='macro')*100
```

75.1948051948052

```
from sklearn.metrics import f1_score
f1_score(y_test, y_pred_2, average='macro')*100
```

74.5230078563412

```
precision_score(y_test, y_pred, average='macro')*100
```

78.33333333333333

```
recall_score(y_test, y_pred, average='macro')*100
```

78.22510822510823

```
f1_score(y_test, y_pred, average='macro')*100
```

78.1857219538379

*Hình 5. Đánh giá hai mô hình vừa xây dựng trên 3 độ đo sau: precision\_score, recall\_score và f1\_score sử dụng macro average.*

\*Nhận xét: Tất cả mô hình của LogisticRegression đều cao hơn mô hình của KNN

**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.**

```
grid_values = { 'C': [0.1, 1, 10, 100, 1000],
                 'penalty': ['l1', 'l2'] }
```

*Hình 6. Biến lưu trữ tham số*

```

from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report

grid =GridSearchCV(model,grid_values,refit = True, verbose =3 )
grid.fit(X_train, y_train)

```

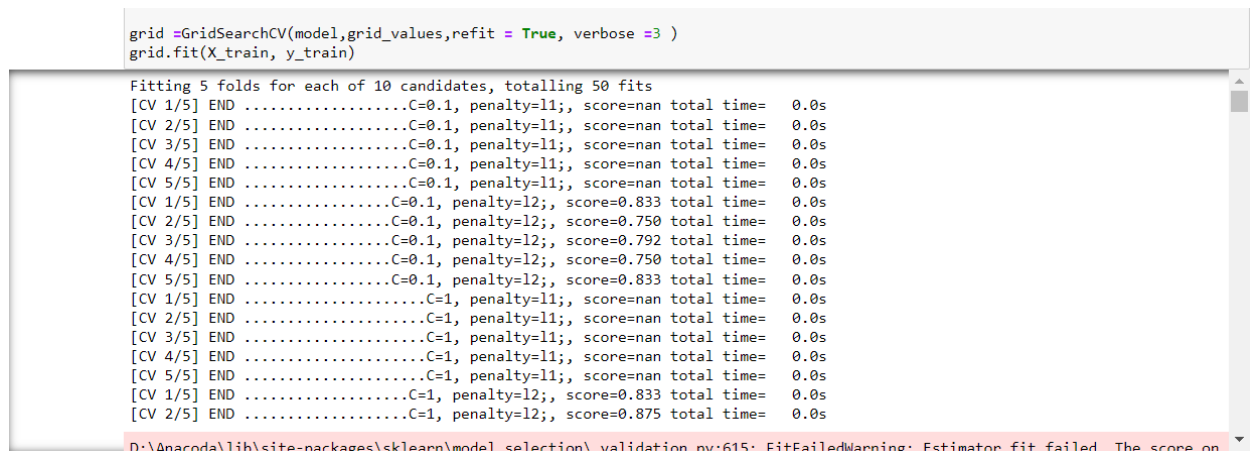
Hình 7. Khai báo thư viện và gọi mô hình GridSreachCV

Khai báo 2 thư viện sklearn để thực hiện mô hình GridSreachCV. Gọi mô hình trong thư viện và gán vào biến grid khi gọi model đã được gán cho mô hình Logistic Regression.

```

grid =GridSearchCV(model,grid_values,refit = True, verbose =3 )
grid.fit(X_train, y_train)

```



```

Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV 1/5] END .....C=0.1, penalty=l1;, score=nan total time= 0.0s
[CV 2/5] END .....C=0.1, penalty=l1;, score=nan total time= 0.0s
[CV 3/5] END .....C=0.1, penalty=l1;, score=nan total time= 0.0s
[CV 4/5] END .....C=0.1, penalty=l1;, score=nan total time= 0.0s
[CV 5/5] END .....C=0.1, penalty=l1;, score=nan total time= 0.0s
[CV 1/5] END .....C=0.1, penalty=l2;, score=0.833 total time= 0.0s
[CV 2/5] END .....C=0.1, penalty=l2;, score=0.750 total time= 0.0s
[CV 3/5] END .....C=0.1, penalty=l2;, score=0.792 total time= 0.0s
[CV 4/5] END .....C=0.1, penalty=l2;, score=0.750 total time= 0.0s
[CV 5/5] END .....C=0.1, penalty=l2;, score=0.833 total time= 0.0s
[CV 1/5] END .....C=1, penalty=l1;, score=nan total time= 0.0s
[CV 2/5] END .....C=1, penalty=l1;, score=nan total time= 0.0s
[CV 3/5] END .....C=1, penalty=l1;, score=nan total time= 0.0s
[CV 4/5] END .....C=1, penalty=l1;, score=nan total time= 0.0s
[CV 5/5] END .....C=1, penalty=l1;, score=nan total time= 0.0s
[CV 1/5] END .....C=1, penalty=l2;, score=0.833 total time= 0.0s
[CV 2/5] END .....C=1, penalty=l2;, score=0.875 total time= 0.0s
D:\Anagoda\lib\site-packages\sklearn\model_selection\_validation.py:615: FitFailedWarning: Estimator fit failed. The score on

```

Hình 8. Điều chỉnh mô hình và chạy kết quả tốt nhất trên tập train.

Đầu tiên, nó chạy cùng một vòng lặp với xác nhận chéo, để tìm ra tích hợp tham số tốt nhất. Sau khi có sự phối hợp tốt nhất, nó sẽ chạy khớp trở lại trên tổng thể tài liệu được truyền cho vừa khớp ( không xác nhận chéo ), để tạo một quy mô mới duy nhất bằng cách sử dụng thiết lập thông số kỹ thuật tốt nhất.

```

In [167]: print(grid.best_params_)
          print(grid.best_estimator_)

          {'C': 1, 'penalty': 'l2'}
          LogisticRegression(C=1)

In [168]: grid_predictions = grid.predict(X_test)

In [169]: print(classification_report(y_test, grid_predictions))

```

	precision	recall	f1-score	support
0	1.00	0.93	0.96	14
1	0.75	0.82	0.78	11
2	0.60	0.60	0.60	5
accuracy			0.83	30
macro avg	0.78	0.78	0.78	30
weighted avg	0.84	0.83	0.84	30

---

```

In [171]: print(grid.best_score_*100)|
          80.0

```

---

*Hình 9.* Thực hiện mô hình với tập test và kiểm tra lại kết quả.

Chạy lại những Dữ kiện và xem báo cáo giải trình phân loại trên đối tượng người dùng lưới này giống như bạn làm với quy mô thông thường

\*Nhận xét: Kết quả sau khi tinh chỉnh tham số Grid Search CV để tìm ra bộ tham số tốt nhất cho mô hình Logistic Regression đã làm tăng độ chính xác của mô hình lên một cách đáng kể và cao hơn của KNN.

---HẾT---