BÀI GIẢNG HỌC MÁY

Bài 3-Tiết 2: Thực hành xây dựng mô hình Support Vector Machine

Trình bày: Ths Phạm Việt Anh

Viện Công nghệ HaUI
Trường Đại học Công nghiệp Hà Nội

Nội dung của bài thực hành

1. Mô tả bài toán dự đoán bênh tim
2. Chuẩn bị dữ liệu
3. Xây dựng mô hình SVM
4. Đánh giá mô hình và cải tiến

Tổng quan về bài toán

- Bài toán đưa ra việc dự đoán bệnh nhân có khả năng mắc bệnh tim mạch thông qua một bộ dữ liệu được thu thập từ các chuyên gia:

STT	Ký hiệu	Mô tả	Kiểu
1	age	Tuổi	Liên tục
2	sex	Giới tính	Phân loại
3	cp	Loại đau ngực	Phân loại
4	threstbps	Huyết áp lúc nghỉ (tính bằng mm Hg)	Liên tục
5	chol	Cholesterol mg/dl	Liên tục
6	fps	Lượng đường trong máu	Phân loại
7	restecg	Kết quả điện tim đồ lúc nghỉ ngơi	Phân loại
8	thalach	Nhịp tim tối đa đạt được	Liên tục
9	exang	Tập thể dục có gây đau thắt lưng không	Phân loại
10	oldpeak	Chênh lệch đoạn STAN khi tập thể dục so với lúc nghỉ	Liên tục
11	slope	Độ dốc tại đỉnh của đoạn STAN khi tập thể dục	Phân loại
12	ca	Số lượng đoạn mạch chính	Phân loại
13	thal	Dấu hiệu của sóng	Phân loại
14	target	Kết quả chẩn đoán	Phân loại

Tổng quan về bài toán

❑ Quan sát dữ liệu:

In [1]:
```
#importing library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
```

Bổ sung thư viện

In [2]:
```
# Đọc dữ liệu sử dụng pandas
df = pd.read_csv("heart.csv", header = 0, sep = ",", skipinitialspace = True)
df.head()
```

Đọc dữ liệu bằng thư viện pandas

Out[2]:

	age	sex	cp	trestpbs	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	dial
0	67	1	4	160	286	0	2	108	1	1.5	2	3	3	True
1	67	1	4	120	229	0	2	129	1	2.6	2	2	7	True
2	37	1	3	130	250	0	0	187	0	3.5	3	0	3	False
3	41	0	2	130	204	0	2	172	0	1.4	1	0	3	False
4	56	1	2	120	236	0	0	178	0	0.8	1	0	3	False

Thuộc tính quyết định

Tổng quan về bài toán

❑ Quan sát dữ liệu:

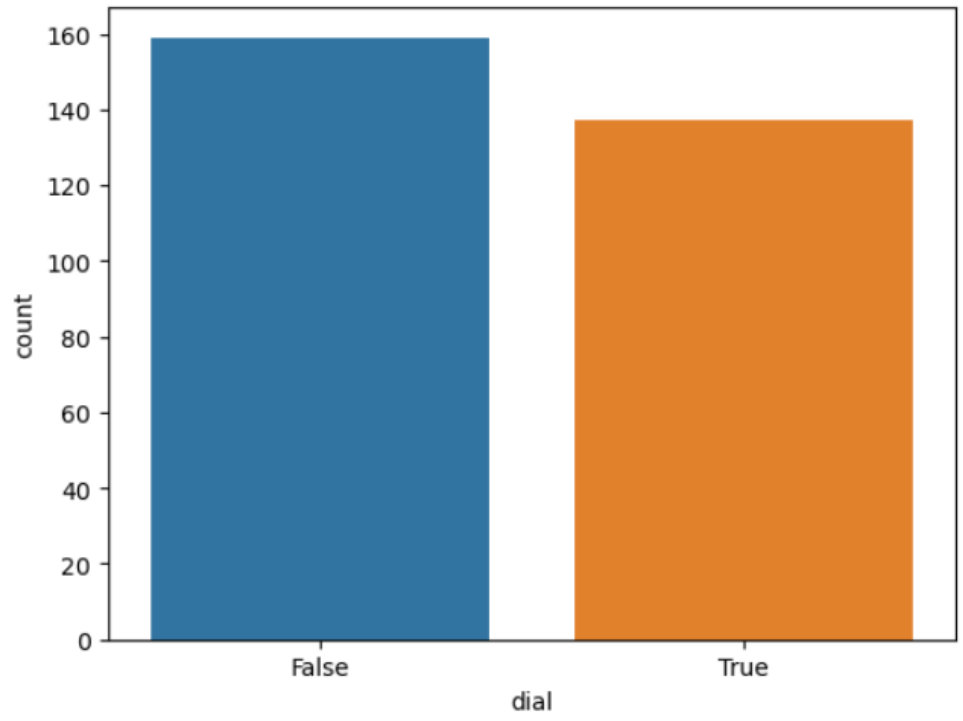
```
In [3]: # Kiểm tra thông tin của data
        # Kiểm tra dữ liệu thiếu sử dụng df.isna().any()
        df.info()

RangeIndex: 296 entries, 0 to 295
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       296 non-null    int64
 1   sex       296 non-null    int64
 2   cp        296 non-null    int64
 3   trestpbs  296 non-null    int64
 4   chol      296 non-null    int64
 5   fbs       296 non-null    int64
 6   restecg   296 non-null    int64
 7   thalach   296 non-null    int64
 8   exang     296 non-null    int64
 9   oldpeak   296 non-null    float64
 10  slope     296 non-null    int64
 11  ca        296 non-null    int64
 12  thal      296 non-null    int64
 13  dial      296 non-null    bool
dtypes: bool(1), float64(1), int64(12)
```

- ❖ Có tổng cộng 296 quan sát, 14 thuộc tính.
- ❖ Các dữ liệu đầy đủ và không bị thiếu.

Tổng quan về bài toán

```
In [4]: print(df['dial'].value_counts())
        sns.countplot(data = df, x = 'dial');
```

```
False    159
True     137
Name: dial, dtype: int64
```



- ❖ Kiểm tra tính cân bằng của dữ liệu
- ❖ Dữ liệu hai lớp được coi là cân bằng nhau.

Tổng quan về bài toán

❑ Tiền xử lý dữ liệu:

In [5]:
```
from sklearn.preprocessing import LabelEncoder
Le = LabelEncoder()
df["dial"] = Le.fit_transform(df["dial"])
df.head()
```

Out[5]:

	age	sex	cp	trestpbs	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	dial
0	67	1	4	160	286	0	2	108	1	1.5	2	3	3	1
1	67	1	4	120	229	0	2	129	1	2.6	2	2	7	1
2	37	1	3	130	250	0	0	187	0	3.5	3	0	3	0
3	41	0	2	130	204	0	2	172	0	1.4	1	0	3	0
4	56	1	2	120	236	0	0	178	0	0.8	1	0	3	0

❖ Sử dụng lệnh fit_transform với các thuộc tính đầu vào được biểu diễn dưới dạng ký tự

Xây dựng mô hình

❑ Chuẩn bị dữ liệu:

```
In [6]: from sklearn.model_selection import train_test_split
        from sklearn.svm import SVC

        # Tạo dữ liệu đầu vào và nhãn đầu ra tương ứng
        df_X = df.iloc[:, :-1]
        df_y = df.iloc[:, -1]


        # Chia bộ dữ liệu thành 2 phần
        X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size = 0.25, random_state = 42)
        print(X_train.shape)
        print(y_train.shape)

        (222, 13)
        (222,)
```

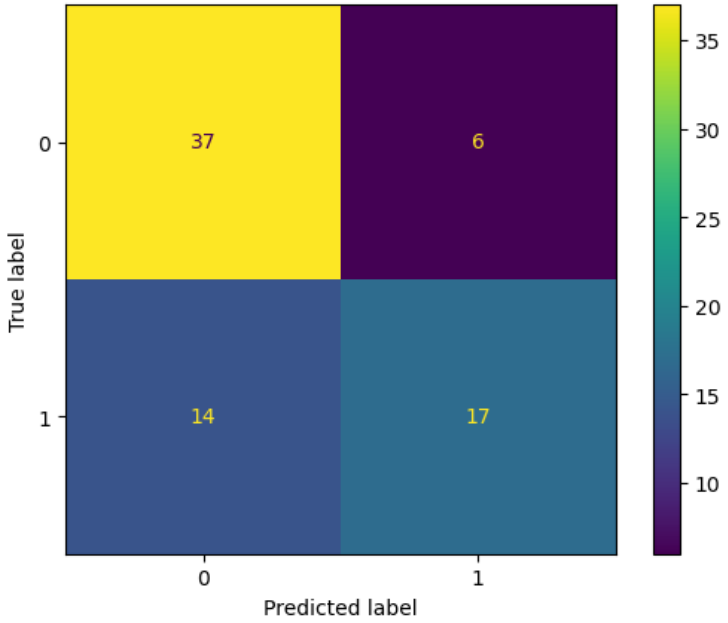
❖ Dữ liệu cho phép chia mô hình theo hai tập training và testing

❖ Lấy dữ liệu từ DataFrame

Xây dựng mô hình

- Khởi tạo và huấn luyện mô hình trên dữ liệu training:

```
In [7]: # Khởi tạo mô hình SVM
        svc = SVC()

        # Training với dữ liệu train
        svc.fit(X_train, y_train)

Out[7]: SVC()
```



- Đánh giá mô hình:

```
In [8]: # Đánh giá mô hình
        from sklearn.metrics import classification_report, ConfusionMatrixDisplay

        y_pred = svc.predict(X_test)
        print(classification_report(y_test, y_pred))
        ConfusionMatrixDisplay.from_estimator(svc, X_test, y_test);
```

```
                precision    recall  f1-score   support

           0       0.73      0.86      0.79        43
           1       0.74      0.55      0.63        31

    accuracy                           0.73        74
   macro avg       0.73      0.70      0.71        74
weighted avg       0.73      0.73      0.72        74
```

Đánh giá mô hình phân lớp

TP (true positive): Tổng số trường hợp báo khớp positive
FP (false positive): Tổng số trường hợp dự đoán sai nhãn negative thành positive
TN (true negative): Tổng số trường hợp báo khớp negative
FN (False negative): Tổng số trường hợp dự đoán sai nhãn positive thành negative

Precision: là tỷ lệ giữa những người thật sự có bệnh so với tất cả những người được dự đoán là có bệnh.

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

Recall: trong những người thực sự có bệnh, bao nhiêu trong số họ được dự đoán đúng bởi mô hình ? Nói cách khác, có bao nhiêu dự đoán Positive đúng là do mô hình đưa ra ?

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

F1-score: Chỉ số trung bình điều hòa giữa Precision và Recall

$$F_1 = \frac{2\,\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Accuracy: Chỉ số đánh giá hiệu quả dự báo của mô hình trên một bộ dữ liệu

$$\text{Accuracy} = \frac{TP + TN}{\text{total sample}}$$

Đánh giá mô hình phân lớp

- Tính các chỉ số TP, FP, TN, FN, Precision, Recall, F1-score, Accuracy trong ví dụ sau:

y_true	y_pred
Cat	Dog
Dog	Dog
Dog	Cat
Cat	Cat
Dog	Dog
Cat	Cat
Cat	Dog
Cat	Dog
Dog	Cat
Cat	Cat

TP: 2, FP: 3, TN: 3, FN: 2
➔ Precision = 2 / (2 +3) = 0.4
➔ Recall = 2 / (2+2) = 0.5
➔ F1-score = (2 * 0.4 * 0.5) / (0.4 + 0.5) = 0.44
➔ Accuracy = (2 + 3) / 10 = 0.5

❖ Kiểm tra bằng thư viện:

```
In [3]: from sklearn.metrics import classification_report, ConfusionMatrixDisplay

y_true = ["Cat", "Dog", "Dog", "Cat", "Dog", "Cat", "Cat", "Cat", "Dog", "Cat"]
y_pred = ["Dog", "Dog", "Cat", "Cat", "Dog", "Cat", "Dog", "Dog", "Cat", "Cat"]
print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

         Cat       0.60      0.50      0.55         6
         Dog       0.40      0.50      0.44         4

    accuracy                           0.50        10
   macro avg       0.50      0.50      0.49        10
weighted avg       0.52      0.50      0.51        10
```

Cải thiện mô hình SVM

- Sử dụng thuật toán GridSearch để tìm kiếm các tham số tốt nhất cho mô hình

In [9]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.pipeline import Pipeline


parameters = {
    'clf__kernel':['linear', 'rbf', 'poly', 'sigmoid'],  # Các dạng hàm kernel
    'clf__C':[0.05, 1, 100], # Trọng số của phạt phân loại sai
    'clf__coef0': [2, 3, 4], # Tương ứng với tham số gamma của đa thức
    'clf__degree': [1, 2, 3] # Bậc d của đa thức
}

pipeline = Pipeline([
                    ('clf', SVC())
])

cv = GridSearchCV(pipeline, parameters, cv=5, n_jobs=12, scoring='accuracy', verbose=2, refit=True)
cv.fit(X_train, y_train)
```

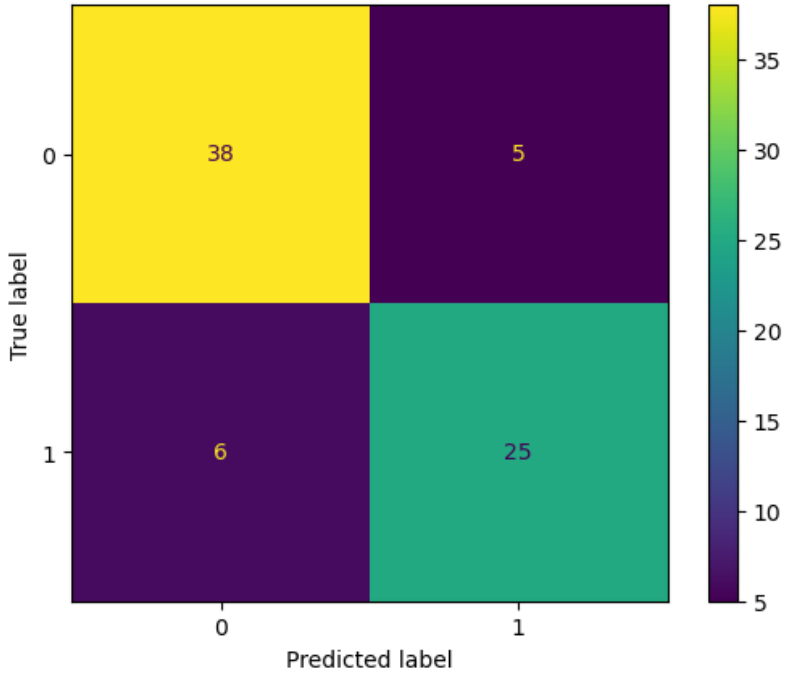
```
Fitting 5 folds for each of 108 candidates, totalling 540 fits
```

Cải thiện mô hình SVM

- Đưa ra tham số mô hình và đánh giá lại mô hình

```
In [10]: # Đưa ra tham số mô hình
         print(cv.best_params_)

         # Đánh giá lại mô hình
         y_pred_2 = cv.predict(X_test)
         print(classification_report(y_test, y_pred_2))
         ConfusionMatrixDisplay.from_estimator(cv, X_test, y_test);
```

```
{'clf__C': 100, 'clf__coef0': 2, 'clf__degree': 1, 'clf__kernel': 'linear'}
              precision    recall  f1-score   support

           0       0.86      0.88      0.87        43
           1       0.83      0.81      0.82        31

    accuracy                           0.85        74
   macro avg       0.85      0.85      0.85        74
weighted avg       0.85      0.85      0.85        74
```



- Độ chính xác của mô hình đã tăng từ 73% → 85%

Cải thiện mô hình SVM

- Tiếp tục cải thiện khi chuẩn hóa dữ liệu (dựa trên chuẩn hóa min-max):

```
In [11]: from sklearn.preprocessing import StandardScaler, MinMaxScaler

pipeline = Pipeline([
    ('scaler', MinMaxScaler()),
    ('clf', SVC()),

])

cv = GridSearchCV(pipeline, parameters, cv=5, n_jobs=12, scoring='accuracy', verbose=2, refit=True)
cv.fit(X_train, y_train)
```

Chuẩn hóa min-max

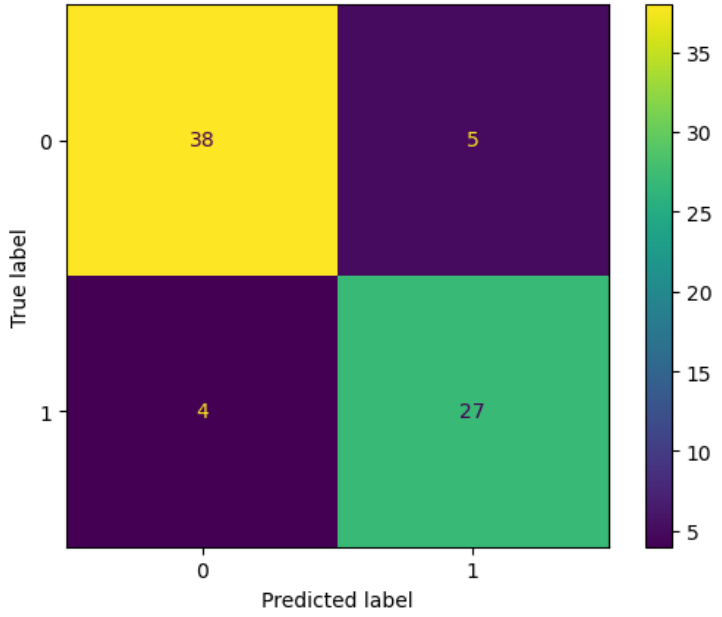
```
Fitting 5 folds for each of 108 candidates, totalling 540 fits

Out[11]: GridSearchCV(cv=5,
                      estimator=Pipeline(steps=[('scaler', MinMaxScaler()),
                                                ('clf', SVC())]),
                      n_jobs=12,
                      param_grid={'clf__C': [0.05, 1, 100], 'clf__coef0': [2, 3, 4],
                                  'clf__degree': [1, 2, 3],
                                  'clf__kernel': ['linear', 'rbf', 'poly', 'sigmoid']},
                      scoring='accuracy', verbose=2)
```

Cải thiện mô hình SVM

❑ Đánh giá lại mô hình:

```
In [12]: # Đưa ra tham số mô hình
         print(cv.best_params_)

         # Đánh giá lại mô hình
         y_pred_3 = cv.predict(X_test)
         print(classification_report(y_test, y_pred_3))
         ConfusionMatrixDisplay.from_estimator(cv, X_test, y_test);
```

```
{'clf__C': 1, 'clf__coef0': 4, 'clf__degree': 2, 'clf__kernel': 'poly'}
              precision    recall  f1-score   support

           0       0.90      0.88      0.89        43
           1       0.84      0.87      0.86        31

    accuracy                           0.88        74
   macro avg       0.87      0.88      0.88        74
weighted avg       0.88      0.88      0.88        74
```



❖ Độ chính xác của mô hình đã tăng từ 85% → 88%

Cải thiện mô hình SVM

- Cải thiện mô hình khi sử dụng kỹ thuật trích chọn thuộc tính (feature selection):

```
In [13]: df.corr()['dial'].sort_values(ascending = False)

Out[13]: dial        1.000000
         thal        0.529841
         ca          0.462007
         oldpeak     0.428842
         exang       0.420130
         cp          0.405980
         slope       0.343609
         sex         0.281261
         age         0.230677
         restecg     0.170041
         trestpbs    0.156210
         chol        0.079546
         fbs         0.010889
         thalach    -0.424377
         Name: dial, dtype: float64
```

```
In [14]: from sklearn.model_selection import train_test_split

         # Loại bỏ đi 2 thuộc tính dư thừa
         df_X = df.drop(columns=['chol', 'fbs'],  axis = 1)
         df_y = df.iloc[:, -1]
         df_X

         # Chia bộ dữ liệu thành 2 phần
         X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size = 0.25, random_state = 42)
         print(X_train.shape)
         print(y_train.shape)

         (222, 12)
         (222,)
```

Các thuộc tính không đóng góp vai trò lớn so với thuộc tính quyết định

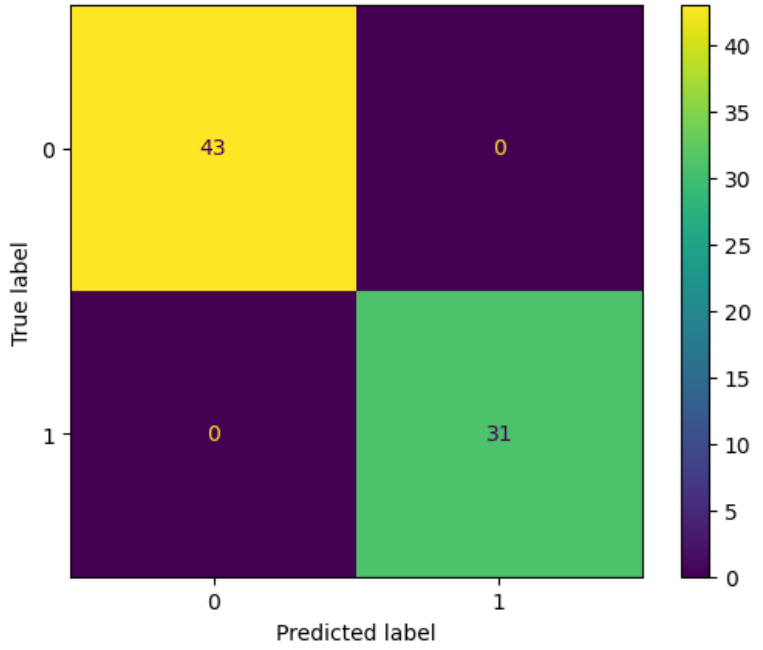
Loại bỏ đi hai thuộc tính chol và fbs trong dữ liệu.

Cải thiện mô hình SVM

❑ Đánh giá lại mô hình:

```
In [16]: # Đưa ra tham số mô hình
         print(cv.best_params_)

         # Đánh giá lại mô hình
         y_pred_4 = cv.predict(X_test)
         print(classification_report(y_test, y_pred_4))
         ConfusionMatrixDisplay.from_estimator(cv, X_test, y_test);
```

```
{'clf__C': 0.05, 'clf__coef0': 2, 'clf__degree': 1, 'clf__kernel': 'linear'}
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        43
           1       1.00      1.00      1.00        31

    accuracy                           1.00        74
   macro avg       1.00      1.00      1.00        74
weighted avg       1.00      1.00      1.00        74
```



❖ Độ chính xác của mô hình đã tăng từ 88% → 100%

Bài tập về nhà

- Xem lại toàn bộ bài giảng trên lớp, chú ý các kiến thức trong việc xây dựng và cải thiện mô hình SVM.
- Tìm hiểu thêm một số phương pháp giúp cải thiện tốt cho mô hình phân lớp.
- Tìm hiểu một số kiến thức cơ bản về học sâu và mạng nơ ron nhân tạo.
- Yêu cầu mang máy tính để phục vụ quá trình thực hành vào buổi sau.


thank
you