TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA TOÁN - ỨNG DỤNG

BÁO CÁO THỰC HÀNH

CÁC GIẢI THUẬT PHÂN LOẠI CƠ BẢN

SVTH: NGUYỄN ĐĂNG TIẾN

MSSV: 3123580050

GVHD: TS. ĐỖ NHƯ TÀI

Năm học: 2025 - 2026

Mục lục

Mục tiêu chung

Mục tiêu của phàn thực hành này được thiết kế để giúp sinh viên có thể phát triển các kỹ năng áp dụng các giải thuật phân loại vào việc giải quyết các vấn đề bài toán thực tế trong khoa học dữ liệu, Cụ thể, phần thwucj hfanh nhằm đạt đươc các mục tiêu sau

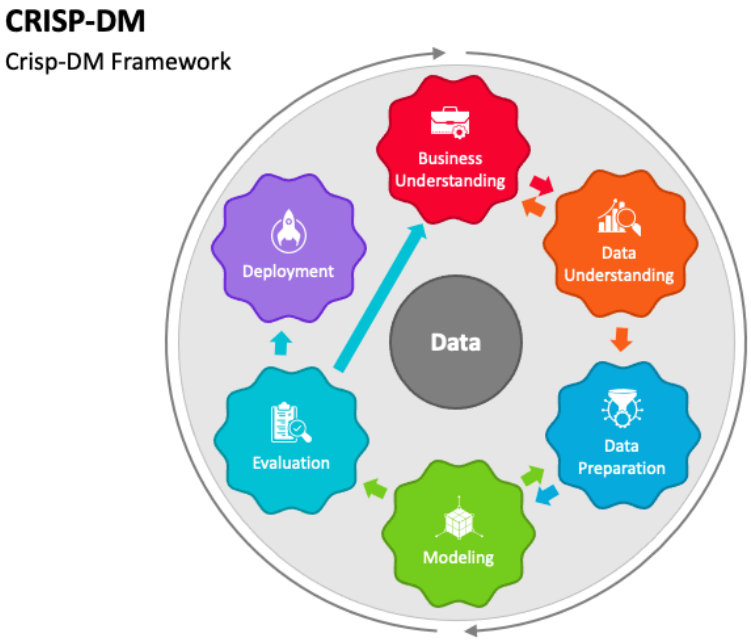
- Hiểu và triển khai được các thuật toán phân loại: Cây quyết định, rừng cây ngẫu nhiên, SVM và Bayes ngây thơ thông qua ngôn ngữ lập trình như Python và sử dụng các thư viện Scikit-learn, SciPy,.. Hướng dẫn thực hiện triển khai mô hình thông qua các bước như
    - Tiền xử lý dữ liệu: Chuẩn bị dữ liệu, bao gồm xử lý giá trị thiếu, chuẩn hóa dữ liệu và mã hóa các biến phân loại sao cho phù hợp với mỗi loại thuật toán
    - Đánh giá tối ưu mô hình thông qua các chỉ só như: Accuracy, Precision, Recall, …
    - Phân tích và diễn giải kết quả phân loại, nhận diện về vấn đề quá khớp và đề xuất cải tiến
- Nội dung bài thực hành bao gồm 3 phần ứng với 3 chương :
    - Chương 1: Giải thuật Cây quyết định và rừng cây
    - Chương 2: Giải thuật SVM (Support Vector Machine)
    - Chương 3: Giải thuật Navie Bayes (Bayes ngây thơ)

CHƯƠNG I: GIẢI THUẬT CÂY QUYẾT ĐỊNH VÀ RỪNG CÂY

1.1. Ôn tập lý thuyết

Quy trình khai phá dữ liệu CRISP-DM và SEMMA là gì?

Quy trình khai phá dữ liệu CRISP-DM là quy trình "kinh điển" 6 bước:



---

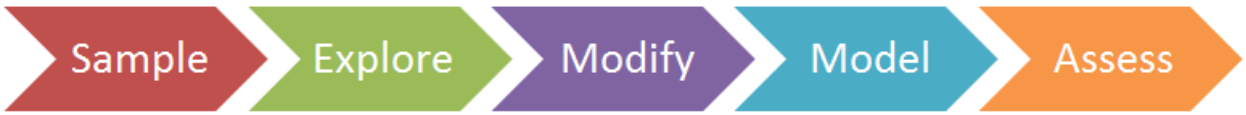
1. **Hiểu bài toán nghiệp vụ**
   - Xác định mục tiêu kinh doanh
   - Chuyển mục tiêu kinh doanh thành mục tiêu khai phá dữ liệu
2. **Hiểu dữ liệu**
   - Thụ thập dữ liệu ban đầu
   - Mô tả dữ liệu (số lượng mẫu và đặc trưng, kiểu dữ liệu)
   - Khám phá dữ liệu
   - Kiểm tra chất lượng dữ liệu
3. **Chuẩn bị dữ liệu**
   - Làm sạch dữ liệu
   - Biến đổi chuẩn hóa, mã hóa, tạo đặc trưng mới
   - Tích hợp dữ liệu
   - Chọn tập dữ liệu cuối cùng đưa vào mô hình
4. **Xây dựng mô hình**
   - Chọn thuật toán

- Chia tập huấn luyện và kiểm tra
- Huấn luyện, điều chỉnh tham số và so sánh nhiều mô hình

5. **Đánh giá**
   - Đánh giá mô hình theo mục tiêu nghiệp vụ
   - Kiểm tra có đạt yêu cầu kinh doanh không, có bias, hay quá khớp hay không
   - Quyết định có triểm khai hay cân quay lại bước trước
6. **Triển khai**
   - Đưa mô hình vào hệ thống thực
   - Theo dõi, bảo trì, huấn luyện lại khi dữ liệu thay đổi

Quy trình khai phá dữ liệu SEMMA là quy trình cảu SAS, tâp trung nhiều vào phần kỹ thuật dữ liệu

SEMMA Process

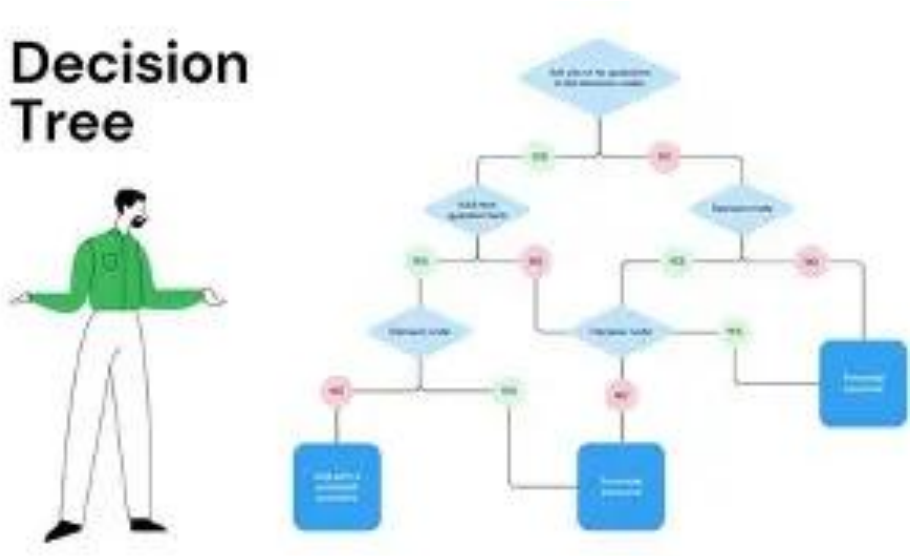


1. **Sample**
   - Lấy mẫu dữ liệu đại diện
   - Chia huấn luyện đánh giá và tập kiểm tra
2. **Explore**
   - Khám phá dữ liệu
   - Phát hiện quan hệ
3. **Modify**
   - Làm sạch, biến đổi, tạo feature, chọn feature.
   - Biến đổi phi tuyến, rời rạc hóa, chuẩn hóa…
4. **Model**
   - Xây dựng mô hình
   - Ước lượng tham số
5. **Access**
   - Đánh giá mô hình
   - So sánh mô hình chọn mô hình tốt nhất

Cây quyết định hoạt động như thế nào? Hãy giải thích các thành phần chính (nút gốc, nút lá, nhánh) và cách cây đưa ra dự đoán.

- Bắt đầu từ nút gốc với một mẫu mới
- Kiểm tra điều kiện tại nút hiện tại, đi theo nhánh tương ứng (Đúng/Sai hoặc theo khoảng giá trị)
- Lặp lại việc đi qua các nút trong đến khi gặp nút lá

- Kết quả dự đoán là nhãn/giá trị của nút lá đó



Các thành phần chính:

Nút gốc

- Nút đầu tiên của cây, chứa toàn bộ dữ liệu huấn luyện
- Thuộc tính được chọn tại nút gốc là thuộc tính giúp "chia tốt nhất" tập dữ liệu

Nút lá

- Nút cuối cùng của cây, không bị chia tiếp
- Với bài toán phân loại: chứa nhãn dự đoán
- Với bài toán hồi quy: giá trị số thực

Nhánh

- Đường dẫn tương ứng với két quả của điều kiện tại một nút

Các tiêu chí phân tách (splitting criteria) như Gini Index, Entropy, hay Information Gain được sử dụng trong cây quyết định là gì? Chúng khác nhau ra sao?

1. Entropy: Đo mức độ hỗn loạn/ không chắc chắn của nhãn lại một tập S

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2(p_i)$$

Trong đó:

- c: số lớp
- p_i: tỉ lệ mẫu thuộc lớp $i$trong tập dữ liệu S

2. Infomation Gain (IG): Đo mức giảm Entropy khi tách theo thuộc tính A

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Trong đó:

- A: thuộc tính dùng để chia
- S_v: tập con khi thuộc tính Anhận giá trị v
- $\frac{|S_v|}{|S|}$: trọng số theo kích thước tập con

3. Gini Index: Hay dùng trong CART (Classification And Regression Tree)

$$Gini(S) = 1 - \sum_{i=1}^{c} p_i^2$$

- p_i: xác suất của lớp i

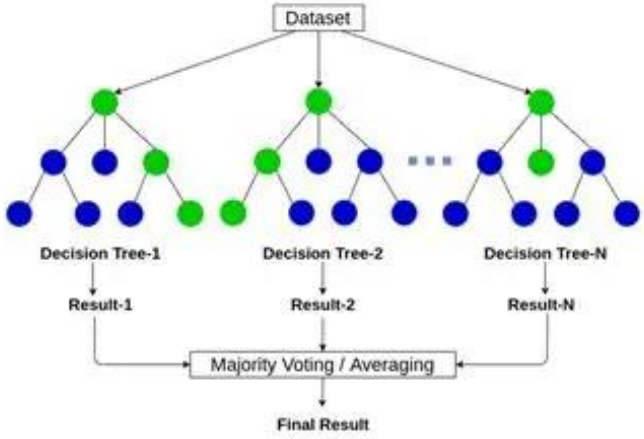
So sánh nhanh

- Entropy + Information Gain
  - Dựa trên lý thuyết thông tin.
  - ính toán hơi nặng hơn vì dùng log.
  - Nhạy hơn với sự thay đổi xác suất nhỏ.
- Gini
  - Tính nhanh hơn (chỉ bình phương).
  - Thường cho kết quả tương tự Entropy trong thực tế.
  - Scikit-learn mặc định dùng Gini cho phân loại.

Rừng cây (Random Forest) là gì? Nó khác gì so với một cây quyết định đơn lẻ? Tại sao Random Forest thường có hiệu suất tốt hơn cây quyết định trong các bài toán phân loại?

Rừng cây là mô hình ensemble gồm nhiều cây quyết định (thường là cây phân loại/hồi quy), mỗi cây được huấn luyện trên một mẫu con của dữ liệu và một tập con của thuộc tính nhằm tạo sự ngẫu nhiên Cách hoạt động:

Random Forest



1. **Với mỗi cây:**
   - Lấy ngẫu nhiên có lặp từ tập huấn luyện
   - Mỗi lần phân tách nút, chỉ xem xét một tập con ngẫu nhiên

2. Chọn đặc trưng ngẫu nhiên tại mỗi nút

   - Xem xét một tập con ngẫu nhiên

3. **Huấn luyện nhiều cây quyết định**
   - Tạo N cây tương ứng
   - Mỗi cây học trên nhiều mẫu dữ liệu khác nhau
4. **Tổng hợp dự đoán**
   - Nếu phân loại thì chọn nhãn có số phiếu bình chọn nhiều nhất
   - Hồi quy thì lấy trung bình

   Sự khác biệt so với một cây đơn lẻ
   - Tập hợp nhiều cây -> Khó overfitting
   - Hiệu suất cao hơn khi có 1 cây

Những ưu điểm và hạn chế của cây quyết định và Random Forest là gì? Trong trường hợp nào thì cây quyết định có thể hoạt động kém hiệu quả?

Cây quyết định

Ưu điểm

- Dễ hiểu, trực quan
- Không cần chuẩn hóa dữ liệu

Hạn chế

- Dễ overfitting
- kém ổn định

Rừng ngẫu nhiên

Ưu điểm

- Độ chính xác cao
- Chống overfitting tốt

Hạn chế

- Khó diễn giải
- Tốn tài nguyên

Khi nào nên sử dụng cây quyết định hoạt động hiểu quả

- Dữ liệu có cấu trúc rõ ràng các quy luật phân chia dễ thấy
- Bài toán có tính diễn giải quan trọng
- Dữ liệu có cả biên số và biến phân loại
- Chứa nhiều điều kiện
- Dữ liệu ít nhiễu
- Khi kích thước dữ liệu không quá lớn

Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình cây quyết định không? Hãy mô tả các bước thực hiện

```
# Khai báo thư viện
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True) # load dữ liệu Iris data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2) # Chia tập train test

model = DecisionTreeClassifier(max_depth=3) # Áp dụng mô hình với chiều sâu của các nút là 3
model.fit(X_train, y_train) # Fit

print("Accuracy:", model.score(X_test, y_test)) # Xuất độ chính xác cho mô hình
```

Làm thế nào để triển khai một mô hình Random Forest trong Python? Bạn thường thiết lập các tham số nào (ví dụ: n_estimators, max_depth)?

Sử dụng thư viện sklearn,ensemble bằng Python sau đó cài đặt thông số

```
from sklearn.ensemble import RandomForestClassifier
```

```
rf = RandomForestClassifier(
   n_estimators=200,     # số cây
   max_depth=8,          # độ sâu tối đa
   max_features="sqrt",  # số đặc trưng khi split
   random_state=42
)

rf.fit(X_train, y_train)
```

Làm thế nào để đánh giá tầm quan trọng của các đặc trưng (feature importance) trong Random Forest bằng Python?

```
# Dùng code
importances = rf.feature_importances_
print(importances)
```

Điều chỉnh siêu tham số (hyperparameter tuning) cho cây quyết định hoặc Random Forest chưa? Hãy mô tả cách bạn sử dụng GridSearchCV hoặc RandomizedSearchCV

GridSearchCV

Hoạt động bằng cách thử tất cả các tham số được đề cập và áp dụng vào mô hình đến khi tim được tham số tối ưu nhất cv chính là sử dụng cross-validation nhằm giảm độ overfitting cho mô hình

```
from sklearn.model_selection import GridSearchCV

params = {
   "n_estimators": [100, 200],
   "max_depth": [5, 10, None]
}

grid = GridSearchCV(RandomForestClassifier(), params, cv=3)
grid.fit(X_train, y_train)
print(grid.best_params_)
```

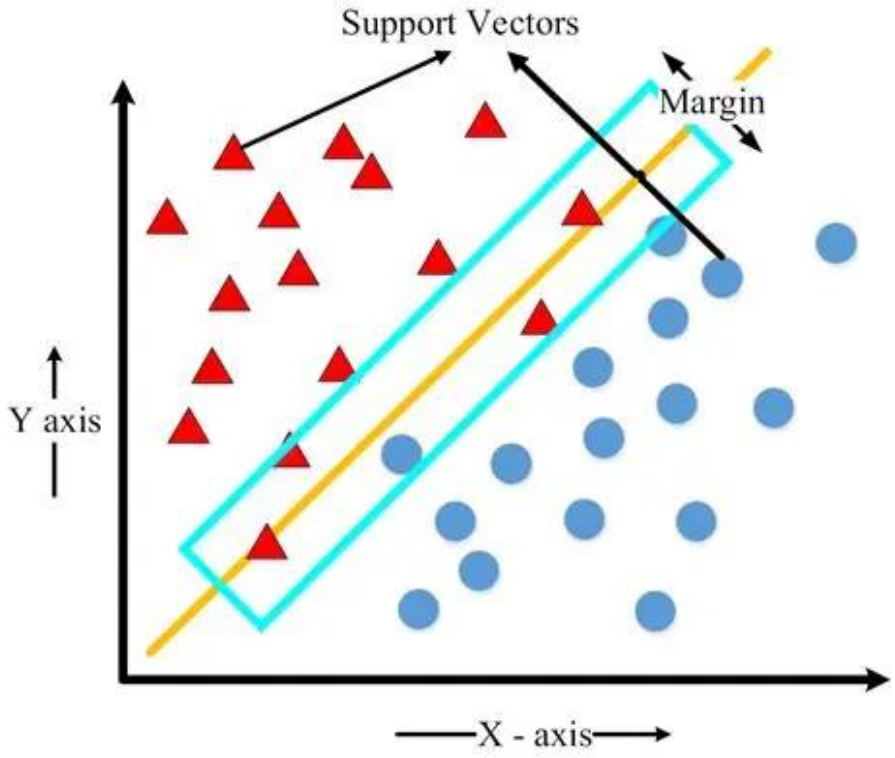
RandomizedSearchCV

Hoạt động dựa trên việc thử ngẫu nhiên các cặp tham số được đề cập bằng thuộc tính n-inter với tham số nào nhanh hơn thì lấy tham số đó sử dụng khi có quá nhiều tham số, thời gian huấn luyện tốn kém và cần kết quả tốt nhanh mà không cần thử hết mọi trường hợp

CHƯƠNG II: GIẢI THUẬT SVM (SUPPORT VECTOR MACHINE)

2.1. Ôn tập lý thuyết

Giải thuật Support Vector Machine hoạt động như thế nào? Hãy giải thích khái niệm về ranh giới phân tách (hyperplane) và lề (margin)



Ta có nhóm dữ liệu vẽ trên mặt phẳng. Nhiệm vụ của ta chính là thực hiện vẽ một doạn thẳng hoặc một mặt phẳng phân cách nhằm phân tách ra thành những nhóm khác nhau

Nhiệm vụ của SVM chính là tìm tối đa hóa margin và hạn chết lỗi phân loại

- Hyperplane (Siêu phẳng): Là đường thẳng hoặc mặt phẳng hoặc siêu phẳng dừng để tách các lớp dữ liệu

$$\mathbf{w}^{\mathrm{T}\mathbf{x}} + b = 0$$

- Margin (lề): Khoảng cách từ hyperplane đến các điểm gần nhất

$$\text{Margin} = \frac{2}{|\mathbf{w}|}$$

Các vector hỗ trợ (support vectors) có vai trò gì trong SVM? Tại sao chúng quan trọng?

Các vecto hỗ trợ chính là những điểm:

- Gần hyperplane nhất
- Nằm ở "biên" của mỗi lớp
- Quyết định hoàn toàn vị trí của hyperplane

Chúng có vai trò như những trụ cọt chống giữ cây - dịch chuyển cá trụ sẽ thay đổi cấu trúc cây, giúp SVM gọn nhẹ về mặt thông tin quan trọng, tập trung vào trường hợp khó phân tích nhất

Sự khác biệt giữa SVM với lề cứng (hard margin) và lề mềm (soft margin) là gì? Khi nào nên sử dụng lề mềm?

Hard Margin SVM

$$\min_{\mathbf{w},b} \quad \frac{1}{2}|\mathbf{w}|^2 \text{ ràng buộc } y_i(\mathbf{w}^{\top}\mathbf{x_i} + b) \geq 1, \quad \forall i$$

Giả sử dữ liệu hoàn toàn tashc được bằng một dường thẳng/mặt phẳng

Thuật toán này yêu cầu: Không được phép sai: tát cả các điểm phải nằm cùng một bên của hyperplane, tối đa hóa margin trong điều kiện phân loại hoàn hảo

Thích hợp khi: Dữ liệu ít nhiễu, tách tuyến tính tốt

Nhược điểm: Chỉ cần vài điểm nhiễu nằm lẫn sang phía bên khác thì thuật toán không thể tìm thấy hyperplane thích hợp

Soft Margin SVM

$$min_{\mathbf{w},b,\xi}\frac{1}{2}|\mathbf{w}|^2 + C\sum_{i=1}^{n}\xi_i \text{ rang buộc } y_i(\mathbf{w}^{\top}\mathbf{x_i} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

Cho phép có thể nằm sai phía hyperplane hoặc nằm bên trong vùng margin (rất gân ranh giới)

Bài toán lúc này có 2 mục tiêu chính là vừa tối đa hóa margin và giảm số lỗi mức độ vi phạm

Sử dụng khi dữ liệu có nhiễu, ngoại lai và tách không hoàn toàn

Hàm nhân (kernel) trong SVM là gì? Hãy giải thích các loại kernel phổ biến (linear, polynomial, RBF) và khi nào nên sử dụng chúng

Hàm nhân (kernel) thực hiện ánh xạ dữ liêu từ không gian gốc sang một không gian có số chieuf cao hơn nơi mà các lớp có thể tách tuyến được, thay vì tính toán trong không gian vừa tạo thì SVM dử dụng kernel để tính

Linear Kernel

$$K(\mathbf{x}, \mathbf{z}) = \mathbf{x}^{\top \mathbf{z}}$$

- Giống với SVM tuyến tính và không biến đổi không gian

Dùng khi:

- Dữ liệu gần tuyến tính
- Các đặc trưng nhiều so với số mẫu

Polynomial Kernel

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^{\top \mathbf{z}} + c)^d$$

- Mô hình học biên phân tách phi tuyến đa thức

Dùng khi:

- Quan hệ giữa các đặc trưng là một đa thức
- Bài toán không quá nhiều chiều và có tương tác bậc cao giữa các đặc trưng

RBF Kernel

$$K(\mathbf{x}, \mathbf{z}) = \exp(-\gamma |\mathbf{x} - \mathbf{z}|^2)$$

Dùng khi:

- Dữ liệu không tách được tuyến tính
- Không biết dạng quan hệ giữa các biến
- Số chiều không quá lớn
- Có hàng dạng biên phân chi phức tạp

Tham số C trong SVM có ý nghĩa gì? Nó ảnh hưởng như thế nào đến hiệu suất của mô hình?

- C chính là tham số phạt tham gia vào quá trình điều chỉnh độ quan trọng của việc phân loại đúng trên tập dữ liệu
- C càng lớn tức mô hình càng nghiêm khắc
    - Không cho phép nhiều điểm sai hơn
    - Mô hình cố gắng phân loại các điểm sao cho luôn đúng
    - Sẵn sàng bo cong điểm phân chia
    - Thu nhỏ margin sao cho có thể ôm sát từng điểm

Khi nào nên sử dụng C lớn và C nhỏ

- **C lớn:**
    - Khi dữ liệu ít nhiễu, sạch
    - Mô hình độ chính xác và training cực cao
    - Đòi hỏi phân loại nghiêm ngặt
- **C nhỏ:**
    - Dữ liệu có nhiều nhiễu, không sạch
    - Mô hình tổng quát tốt
    - Tránh overfitting

Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình SVM cho bài toán phân loại không? Hãy mô tả các bước thực hiện

```
# 1. Import thư viện
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report
# 2. Load dữ liệu (ví dụ: Iris)
iris = datasets.load_iris()
X = iris.data        # features
y = iris.target      # labels
# 3. Chia train / test
X_train, X_test, y_train, y_test = train_test_split(
X, y, test_size=0.3, random_state=42, stratify=y)
# 4. Tạo pipeline: Scaling -> SVM
model = Pipeline([
```

```
('scaler', StandardScaler()),          # chuẩn hóa
('svm', SVC(kernel='rbf', C=1.0, gamma='scale'))  # mô hình SVM
])
# 5. Huấn luyện
model.fit(X_train, y_train)
# 6. Dự đoán
y_pred = model.predict(X_test)
# 7. Đánh giá
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=iris.target_names))
```

Hàm nào trong Scikit-learn để chuẩn hóa dữ liệu (scaling) trước khi áp dụng SVM? Tại sao bước này quan trọng?

- **StandardScaler** chuẩn hóa theo z-score
- **MinMaxScaler** đưa đặc trưng vào khoảng 0 đến 1

Scaling quan trọng bởi vì SVM (nhất là với RBF, polynomial, linear) phụ thuộc mạnh vào khoảng cách giữa các điểm dữ liệu:

- Nếu một feature có giá trị rất lớn (ví dụ [0, 1000]) còn feature khác chỉ [0, 1], thì:
  - Feature lớn sẽ áp đảo khoảng cách.
  - Mô hình gần như "bỏ qua" những feature có thang đo nhỏ.

Chuẩn hóa giúp:

- Các feature có quy mô tương đương, không feature nào "nặng ký" quá mức.
- Tối ưu thuật toán (SMO, gradient) hội tụ ổn định hơn và nhanh hơn.
- Việc chọn tham số C, gamma trở nên hợp lý và dễ hơn.

CHƯƠNG III: GIẢI THUẬT NAÏVE BAYES (BAYES NGÂY THƠ)

3.1. Ôn tập lý thuyết

Giải thuật Naive Bayes hoạt động như thế nào? Hãy giải thích định lý Bayes và giả định "ngây thơ" trong thuật toán này?

$P(C|X) = \frac{P(X|C)\,P(C)}{P(X)}$ Naïve Bayes giả định độc lập $P(X|C) = \prod_{i=1}^{n} P(x_i|C)$

- Định lý Bayes cho biết xác suất xảy ra của một lớp C khi biết dữ liệu
- Trong phân loại, ta tìm lớp có xác suất hậu nghiệm lớn nhất
- Nguyên lý Naive Bayes hoạt động với dữ liệu có nhiều thuộc tính rất khó vì cần phân phối chung nhiều chiều

Các loại mô hình Naive Bayes (Gaussian, Multinomial, Bernoulli) khác nhau ra sao? Khi nào nên sử dụng từng loại?

1. **Gaussian Naive Bayes**

$$P(x_i|C_k) = \frac{1}{\sqrt{2\pi\sigma_{k,i}^2}} \exp\left(-\frac{(x_i - \mu_{k,i})^2}{2\sigma_{k,i}^2}\right)$$

- Gỉa định rằng mỗi đặc trưng theo từng lớp tuân theo phân phối chuẩn
    - Nguyên lý: Trung bình, phương sai của tất cả đặc trưng liên tục, khi có mẫu mới, chọn lớp có xác suất lớn nhất
- Dùng cho dữ liệu liên tục
- Gỉa định thuộc tính tuân theo phân phối chuẩn trong từng lớp

2. **Multinomial Naive Bayes**

$$\boldsymbol{P(X|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i\,!} \prod_i (\theta_{k,i})^{x_i}}$$

- Dùng cho dữ liệu đếm số lần xuất hiện:
    - Nguyên lý: Đếm số lần xuất hiện từng đặc trưng trong mỗi lớp, tính xác suất có điều kiện
- Dùng cho dữ liệu đếm
- Phổ biến trong phân loại văn bản

3. **Bernoulli Naive Bayes**

$$P(X|C_k) = \prod_{i=1}^{n} \mathrm{n}\left(\theta_{k,i}^{x_i}(1 - \theta_{k,i})^{1-x_i}\right)$$

- Dùng khi đặc trưng biểu thị có/không
    - Nguyên lý: Với mỗi đặc trưng, tính xác suất xuất hiện, Với mẫu mới, Chọn lớp có xác suất cao nhất
- Dùng cho thuộc tính nhị phân (0/1)
- Cũng dùng trong text những với dạng

Tại sao Naive Bayes được gọi là "ngây thơ"? Giả định về tính độc lập của các đặc trưng ảnh hưởng như thế nào đến hiệu suất của mô hình?

Trong thực tế, các đặc trưng phụ thuộc lẫn nhau những với Naive Bayes giả vở chúng độc lập có điều kiện theo lớp nên chúng được xem là ngây thơ

Gỉa định ảnh hưởng đến hiệu suất

- Nếu đặc trưng gần độc lập -> Hoạt động rất tốt
- Nếu đặc trưng phụ thuộc mạnh
    - Xác suất tính ra không "đúng" theo nghĩa xác suất thật
    - Tuy nhiện nhiều khi xếp hạng lớp vẫn đúng

Ưu điểm và hạn chế của Naive Bayes so với các thuật toán phân loại khác như SVM hoặc Random Forest là gì?

Ưu điểm

- Rất nhanh (huấn luyện & dự đoán), phù hợp dữ liệu lớn.
- Không cần nhiều dữ liệu để cho kết quả chấp nhận được.
- Hoạt động tốt với dữ liệu nhiều chiều (văn bản: hàng chục nghìn feature).
- Ít bị overfit hơn nhiều mô hình phức tạp.
- Dễ triển khai và giải thích (thống kê tần suất đơn giản).

Hạn chế

- Giả định độc lập thường không đúng, có thể làm giảm độ chính xác.
- Với dữ liệu phức tạp, SVM / Random Forest thường chính xác hơn.
- Khó nắm bắt quan hệ phi tuyến, tương tác giữa many-features.
- Gaussian NB phụ thuộc giả định phân phối chuẩn – nếu sai có thể ảnh hưởng kết quả.

So sánh

Naive Bayes

- Train rất nhanh
- Phù hợp với dữ liệu lơn nhiều chiều
- Không có khả năng mô hình hóa phức tạp
- Nhạy cảm với tương quan feature
- Dễ cài đặt nhất

SVM

- Train chậm
- Nếu chọn kernel phù hợp thì hoạt động tốt với dữ liệu lớn, nhiều chiều
- Khả năng mô hình hóa cao
- Ít nhạy cảm với tương quan giữa đặc trưng
- Mức cài đặt trung bình

Random Forest

- Tốc độ train vừa và chậm
- Dữ liệu lớn có thẻ chậm và tốn RAM
- Khả năng mô hình hóa phức tạp rất cao
- Nhảy cảm ít với tương quan giữa các đặc trưng
- Mức độ cài đặt trung bình

Viết đoạn code mẫu bằng Python (sử dụng Scikit-learn) để xây dựng một mô hình Naive Bayes (ví dụ: Gaussian Naive Bayes) không? Hãy mô tả các bước thực hiện

```
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score, classification_report


# 1. Load dữ liệu

iris = load_iris()
```

```
X = iris.data        # 4 đặc trưng liên tục
y = iris.target      # 3 lớp: 0,1,2

# 2. Chia train/test
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 3. Khởi tạo mô hình
model = GaussianNB()

# 4. Huấn luyện
model.fit(X_train, y_train)

# 5. Dự đoán
y_pred = model.predict(X_test)

# 6. Đánh giá
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=iris.target_names))

# 7. Thử dự đoán một mẫu mới
new_sample = [[5.0, 3.5, 1.5, 0.2]]
print("Dự đoán lớp cho mẫu mới:", iris.target_names[model.predict(new_sample)][0])
```

Làm thế nào để xử lý dữ liệu phân loại (categorical data) trước khi áp dụng Multinomial Naive Bayes trong Python?

Multinominal NB yêu cầu đặc trưng là số không âm ta có thể thực hiện One-Hot Encoding cho phiến phân loại -> Multinomial thực sự mạnh khi đặc trưng là biến **đếm/tần suất**. Nếu trong dataset có nhiều biến liên tục thì Gaussian NB thường được sử dụng nhiều hơn

Naive Bayes thường được sử dụng trong phân loại văn bản (text classification). Bạn có thể giải thích cách triển khai Naive Bayes cho bài toán này không?

Để thực hiện triển khai trong phân loại văn bản ta thực hiện như sau

1. **Thu thập dữ liệu văn bản**
    - Tập email, tin tức, review sản phẩm,…
2. **Tiền xử lý (tùy mức độ):**
    - Chuyển thường (lowercase)
    - Loại ký tự đặc biệt, số nếu không cần
    - Bỏ stopwords (the, is, a, …)
    - Stemming / Lemmatization (tuỳ bài toán)
3. **Vector hóa văn bản thành số**

Thường dùng:

- CountVectorizer (bag-of-words): mỗi cột = 1 từ, giá trị = số lần xuất hiện
- hoặc TfidfVectorizer (TF-IDF): giống trên nhưng có trọng số.

4. **Chọn mô hình Naive Bayes**
    - Thường dùng MultinomialNB
    - Nếu dùng vector nhị phân (từ có/không) → có thể dùng BernoulliNB
5. **Huấn luyện, đánh giá**
    - Chia train/test
    - Đánh giá bằng accuracy, precision, recall, F1,…
6. **Triển khai**
    - Đóng gói vào pipeline để pre-processing + model chạy liền mạch.
    - Save model (pickle, joblib) để deploy.
    -