MACHINE LEARNING: BÀI THỰC HÀNH SỐ 1.

1. PHƯƠNG PHÁP NÄIVE BAYES CHO DỮ LIỆU RỜI RẠC

Ví dụ 1. Ví dụ này chỉ mang tính minh họa cho việc phân loại văn bản.

Giả sử tập training của chúng ta có 04 "văn bản", được chia làm hai lớp là nội dung về miền bắc (B) và nội dung về miền nam (N), cụ thể như sau:

Văn bản	Nội dung	Thuộc lớp
D1	Hà-Nội phở cháo-lòng Hà-Nội	B
D2	Hà-Nội bún-chả phở ô-mai	B
D3	Phở bánh-giò ô-mai	B
D4	Sài-gòn hủ-tiếu bánh-bò phở	N

Dựa vào tập training, cần dự đoán văn bản dưới đây thuộc lớp Bắc (B) hay Nam (N):

D5: *Hà-Nội Hà-Nội bún-chả hủ-tiếu* ---- và D6: *phở hủ-tiếu bánh-bò*.

Giải : Bài toán này có đầu ra là rời rạc (hai lớp) và dạng nhị phân (hoặc bắc hoặc nam) nên có thể được giải quyết bằng một trong hai mô hình: Multinomial Naive Bayes hoặc Bernoulli Naive Bayes.

a. Dùng Multinomial Naive Bayes:

- Tìm p(B) và p(N): Trong tập training, 3/4 số văn bản thuộc lớp B; 1/4 thuộc lớp N, vậy p(B) = 3/4; p(N) = 1/4.
- Xác định từ điển, tức tập hợp toàn bộ các từ trong các văn bản, đánh chỉ số thứ tự các từ như sau:

  V = { Hà-Nội, phở, cháo-lòng, bún-chả, ô-mai, bánh-giò, Sài-gòn, hủ-tiếu, bánh-bò}

  Tổng số phần tử trong từ điển là |V| = 9.
- Trong bước train, chúng ta sẽ phải lập các vector đặc trưng (feature vector) ứng với mỗi văn bản theo cách: Vector sẽ có số chiều bằng số phần tử từ điển (là 9); Ứng với mỗi văn bản Dk trong tập training, xét từ có chỉ số j trong từ điển xuất hiện x_j lần trong văn bản (j=1, 2, …, 9), ta lập được vector đặc trưng tương ứng là: $\boldsymbol{x^k} = (x_1, x_2, \dots, x_9)$. Cụ thể với tập training D1, D2, D3, D4 ta có
  - Với văn bản D1: $\boldsymbol{x^1} = (2, 1, 1, 0, 0, 0, 0, 0, 0)$; (B)
  - Với văn bản D2: $\boldsymbol{x^2} = (1, 1, 0, 1, 1, 0, 0, 0, 0)$; (B)
  - Với văn bản D3: $\boldsymbol{x^3} = (0, 1, 0, 1, 1, 0, 0, 0, 0)$; (B)
  - Với văn bản D4: $\boldsymbol{x^4} = (1, 0, 0, 0, 0, 0, 1, 1, 1)$; (N)
  - Với văn bản test
    - D5: $\boldsymbol{x^5} = (2, 0, 0, 1, 0, 0, 0, 1, 0)$.
    - D6: $\boldsymbol{x^6} = (0, 1, 0, 0, 0, 0, 0, 1, 1)$.
- Tổng số lượt từ (kể cả lặp lại) xuất hiện trong các văn bản thuộc lớp B là: $N_B = 11$;
- Tổng số lượt từ (kể cả lặp lại) xuất hiện trong các văn bản thuộc lớp N là: $N_N = 4$.
- Tính các λ^B_j, j=1, 2, …,9 – **tương ứng với các từ trong từ điển**. Ví dụ: $\lambda^B_1 = 3/20$ (Từ Hà-Nội xuất hiện 3 lần trong toàn bộ class B). Tương tự, $\lambda^B_2 = 3/20$; $\lambda^B_3 = \lambda^B_4 = \lambda^B_6 = 1/20$; $\lambda^B_5 = 2/20$; $\lambda^B_1 = 3/20$; $\lambda^B_7 = \lambda^B_8 = \lambda^B_9 = 0$.
- Tương ứng với các λ^N_j, j=1, 2, …,9: $\lambda^N_2 = \lambda^N_7 = \lambda^N_8 = \lambda^N_9 = 1$; $\lambda^N_1 = \lambda^N_3 = \lambda^N_4 = \lambda^N_5 = \lambda^N_6 = 0$.
- Chúng ta áp dụng Laplace smoothing với $\alpha = 1$. Bảng ở trang sau mô tả các tính toán tiếp theo.

TRAINING

class = B	Hà-Nội	phở	cháo-lòng	bún-chả	ô-mai	bánh-giò	Sài-Gòn	hủ-tiếu	bánh-bò	
d1: $\mathbf{x}_1$	2	1	1	0	0	0	0	0	0	
d2: $\mathbf{x}_2$	1	1	0	1	1	0	0	0	0	
d3: $\mathbf{x}_3$	0	1	0	0	1	1	0	0	0	$d = \|V\| = 9$
Total	3	3	1	1	2	1	0	0	0	$\Rightarrow N_{\mathrm{B}} = 11$
$\Rightarrow \hat{\lambda}_{\mathrm{B}}$	4/20	4/20	2/20	2/20	3/20	2/20	1/20	1/20	1/20	$(20 = N_{\mathrm{B}} + \|V\|)$

class = N										
d4: $\mathbf{x}_4$	0	1	0	0	0	0	1	1	1	$\Rightarrow N_{\mathrm{N}} = 4$
$\Rightarrow \hat{\lambda}_{\mathrm{N}}$	1/13	2/13	1/13	1/13	1/13	1/13	2/13	2/13	2/13	$(13 = N_{\mathrm{N}} + \|V\|)$

TEST

d5: $\mathbf{x}_5 = [2, 0, 0, 1, 0, 0, 0, 1, 0]$

$$p(\mathrm{B}|\mathrm{d5}) \propto p(\mathrm{B}) \prod_{i=1}^{d} p(x_i|\mathrm{B})$$

$$= \tfrac{3}{4} \left(\tfrac{4}{20}\right)^2 \tfrac{2}{20} \tfrac{1}{20} \approx 1.5 \times 10^{-4}$$

$$p(\mathrm{N}|\mathrm{d5}) \propto p(\mathrm{N}) \prod_{i=1}^{d} p(x_i|\mathrm{N})$$

$$= \tfrac{1}{4} \left(\tfrac{1}{13}\right)^2 \tfrac{1}{13} \tfrac{2}{13} \approx 1.75 \times 10^{-5}$$

$\Rightarrow p(\mathbf{x}_5|\mathrm{B}) > p(\mathbf{x}_5|\mathrm{N}) \Rightarrow \mathrm{d5} \in \mathrm{class(B)}$

Chú ý các giá trị tìm được ở trên không phải là xác suất để D5 rơi vào lớp B hay N, mà chỉ là một đại lượng đồng biến với xác suất trên (do đó ta có cùng kết luận phân loại). Xác suất đúng sẽ là

$$p(\mathrm{B}|\mathrm{d5}) = \frac{1.5 \times 10^{-4}}{1.5 \times 10^{-4} + 1.75 \times 10^{-5}} \approx 0.8955, \quad p(\mathrm{N}|\mathrm{d5}) = 1 - p(\mathrm{B}|\mathrm{d5}) \approx 0.1045$$

Tương tự với D6, ta có kết quả:

$$p(\mathrm{B}|\mathrm{d6}) \approx 0.29, \quad p(\mathrm{N}|\mathrm{d6}) \approx 0.71$$

Tức là D6 được dự đoán thuộc về class N.

b. Sử dụng phương pháp NB Bernouli

- Do đầu ra của dự đoán chỉ là 02 lớp (B hay N) tương đương dạng boolean (Đúng-Sai) nên có thể sử dụng phương pháp Bernoulli trong nhưng bài toán dạng này.
- Sử dụng NB Bernouli, chúng ta thay đổi vector đặc trưng (xem lại lý thuyết), tức là từ thứ j trong từ điển xuất hiện trong tài liệu D thì ta đặt thành phần thứ j của vector là 1 (không quan tâm xuất hiện bao nhiêu lần). Vậy
    - Với văn bản D1: $\boldsymbol{x}^1$ = (1, 1, 1, 0, 0, 0, 0, 0, 0); (B)
    - Với văn bản D2: $\boldsymbol{x}^2$ = (1, 1, 0, 1, 1, 0, 0, 0, 0); (B)
    - Với văn bản D3: $\boldsymbol{x}^3$ = (0, 1, 0, 0, 1, 1, 0, 0, 0); (B)
    - Với văn bản D4: $\boldsymbol{x}^4$ = (1, 0, 0, 0, 0, 0, 1, 1, 1); (N)
    - Với văn bản test
        - D5: $\boldsymbol{x}^5$ = (1, 0, 0, 1, 0, 0, 0, 1, 0).
        - D6: $\boldsymbol{x}^6$ = (0, 1, 0, 0, 0, 0, 0, 1, 1).

- Tương tự phần trước, ta có p(B) = 3/4; p(N) = 1/4. Ta cần tính các P(i|C), tức là tỷ lệ từ thứ *i* trong từ điển xuất hiện trong class C (C = N hoặc C = B)
    - Class B: P(1|B) = P(5|B) = 2/3 (2 văn bản xuất hiện từ Hà-Nội và ô-mai trong số 3 văn bản); P(2|B) = 3/3; P(3|B) = P(4|B) = P(6|B) = 1/3; còn lại là 0.
    - Class N: P(2|N) = P(7|N) = P(8|N) = P(9|N) = 1/1; P(j|N) còn lại là 0.
- Cuối cùng, ta tính xác suất D6 với $\boldsymbol{x^6}$ = (0, 1, 0, 0, 0, 0, 0, 1, 1) thuộc về các class như sau
    - P(B|D6) = P(B) . $\prod_{i=1}^{9}\{P(i|B)x_i^6 + [1 - P(i|B)][1 - x_i^6]\} \propto 0.16948581$
    - P(N|D6) = P(N) . $\prod_{i=1}^{9}\{P(i|N)x_i^6 + [1 - P(i|N)][1 - x_i^6]\} \propto 0.83051419$
- Vậy D6 thuộc class N.

c. Lập trình

- Có thể lập trình bằng ngôn ngữ bất kỳ (Python/Java/C…) theo các bước như trên.
- Nếu dùng Python, có thể sử dụng thư viện **sklearn**. Một số phương thức quan trọng phục vụ giải bài toán phân loại theo phương pháp dạng Naïve Bayes:
    - Cần import thư viện sklearn.naive_bayes :

```
from sklearn.naive_bayes import MultinomialNB
```

    - Với tập dữ liệu (training_data, label) tương ứng đã khởi tạo, gọi các phương thức giải bài toán phân loại với vector dữ liệu test đã có như sau

```
## call MultinomialNB
clf = MultinomialNB()
# training
clf.fit(train_data, label)
```

    - Các phương thức để in kết quả gồm
        - str(clf.predict(test_data)[j])) để in ra class thứ j, class [0] sẽ là class được dự báo;
        - clf.predict_proba(test_data)) in ra xác suất để test thuộc vào mỗi class.

- **Chương trình trong python theo phương pháp Multinomial Naïve Bayes:**

```
from __future__ import print_function
from sklearn.naive_bayes import MultinomialNB
import numpy as np

# train data
d1 = [2, 1, 1, 0, 0, 0, 0, 0, 0]
d2 = [1, 1, 0, 1, 1, 0, 0, 0, 0]
d3 = [0, 1, 0, 0, 1, 1, 0, 0, 0]
d4 = [0, 1, 0, 0, 0, 0, 1, 1, 1]
```

```
train_data = np.array([d1, d2, d3, d4])
label = np.array(['B', 'B', 'B', 'N'])

# test data
d5 = np.array([[2, 0, 0, 1, 0, 0, 0, 1, 0]])
d6 = np.array([[0, 1, 0, 0, 0, 0, 0, 1, 1]])

## call MultinomialNB
clf = MultinomialNB()
# training
clf.fit(train_data, label)

# test
print('Predicting class of d5:', str(clf.predict(d5)[0]))
print('Predicting class of d6:', str(clf.predict(d6)[0]))
print('Probability of d5 in each class:', clf.predict_proba(d5))
print('Probability of d6 in each class:', clf.predict_proba(d6))
```

- **Chương trình trong python theo phương pháp Bernoulli Naïve Bayes:**

```
from __future__ import print_function
from sklearn.naive_bayes import BernoulliNB
import numpy as np

# train data
d1 = [1, 1, 1, 0, 0, 0, 0, 0, 0]
d2 = [1, 1, 0, 1, 1, 0, 0, 0, 0]
d3 = [0, 1, 0, 0, 1, 1, 0, 0, 0]
d4 = [0, 1, 0, 0, 0, 0, 1, 1, 1]

train_data = np.array([d1, d2, d3, d4])
label = np.array(['B', 'B', 'B', 'N']) # 0 - B, 1 - N

# test data
d5 = np.array([[1, 0, 0, 1, 0, 0, 0, 1, 0]])
d6 = np.array([[0, 1, 0, 0, 0, 0, 0, 1, 1]])

## call MultinomialNB
```

```
clf = BernoulliNB()
# training
clf.fit(train_data, label)
# test
print('Predicting class of d5:', str(clf.predict(d5)[0]))
print('Predicting class of d6:', str(clf.predict(d6)[0]))
print('Probability of d5 in each class:', clf.predict_proba(d5))
print('Probability of d6 in each class:', clf.predict_proba(d6))
```

Ví dụ 2. Phân loại email SPAM – NOT.SPAM bằng Naïve Bayes

- Dữ liệu là tập con của bộ dữ liệu có tại link: https://metatext.io/datasets/ling-spam-dataset . Tập dữ liệu này bao gồm tổng cộng 960 emails tiếng Anh, được tách thành tập training và test theo tỉ lệ 700:260, 50% trong mỗi tập là các spam emails.
- Bộ dữ liệu đã được tiền xử lý theo các bước:
    - Loại bỏ stop words: Những từ xuất hiện thường xuyên như 'and', 'the', 'of', … được loại bỏ.
    - Lemmatization: Những từ có cùng 'gốc' được đưa về cùng loại. Ví dụ, 'include', 'includes', 'included' đều được đưa chung về 'include'. Tất cả các từ cũng đã được đưa về dạng ký tự thường (không phải HOA).
    - Loại bỏ non-words: Số, dấu câu, ký tự 'tabs', ký tự 'xuống dòng' đã được loại bỏ.
- Bộ dữ liệu sau xử lý có trong link sau:

  http://openclassroom.stanford.edu/MainFolder/courses/MachineLearning/exercises/ex6materials/ex6DataPrepared.zip
- Các tệp sau khi giải nén:

  test-features.txt

  test-labels.txt

  train-features-50.txt

  train-features-100.txt

  train-features-400.txt

  train-features.txt

  train-labels-50.txt

  train-labels-100.txt

  train-labels-400.txt

  train-labels.txt

tương ứng với các file chứa dữ liệu của tập training và tập test. File train-features-50.txt chứa dữ liệu của tập training thu gọn với chỉ có tổng cộng 50 training emails.

Mỗi file *labels*.txt chứa nhiều dòng, mỗi dòng là một ký tự 0 hoặc 1 thể hiện email là non-spam hoặc spam.

Mỗi file *features*.txt chứa nhiều dòng, mỗi dòng có 3 số, ví dụ:

1 564 1

1 19 2

trong đó số đầu tiên là chỉ số của email, bắt đầu từ 1; số thứ hai là thứ tự của từ trong từ điển (tổng cộng 2500 từ); số thứ ba là số lượng của từ đó trong email đang xét. Dòng đầu tiên nói rằng trong email thứ nhất, từ thứ 564 trong từ điển xuất hiện 1 lần. Cách lưu dữ liệu như thế này giúp tiết kiệm bộ nhớ vì 1 email thường không chứa hết tất cả các từ trong từ điển mà chỉ chứa một lượng nhỏ, ta chỉ cần lưu các giá trị khác không.

Nếu ta biểu diễn feature vector của mỗi email là một vector hàng có độ dài bằng độ dài từ điển (2500) thì dòng thứ nhất nói rằng thành phần thứ 564 của vector này bằng 1. Tương tự, thành phần thứ 19 của vector này bằng 1. Nếu không xuất hiện, các thành phần khác được mặc định bằng 0.

Dựa trên các thông tin này, chúng ta có thể tiến hành lập trình với thư viện sklearn.

Khai báo thư viện và đường dẫn tới files:

```
## packages
from __future__ import division, print_function, unicode_literals
import numpy as np
from scipy.sparse import coo_matrix # for sparse matrix
from sklearn.naive_bayes import MultinomialNB, BernoulliNB
from sklearn.metrics import accuracy_score # for evaluating results


# data path and file name
path = 'ex6DataPrepared/'
train_data_fn = 'train-features.txt'
test_data_fn = 'test-features.txt'
train_label_fn = 'train-labels.txt'
test_label_fn = 'test-labels.txt'
```

Hàm số đọc dữ liệu từ file data_fn với labels tương ứng label_fn. Chú ý rằng số lượng từ trong từ điển là 2500.

Dữ liệu sẽ được lưu trong một ma trận mà mỗi hàng thể hiện một email. Ma trận này là một ma trận thưa nên chúng ta sẽ sử dụng hàm scipy.sparse.coo_matrix

```
nwords = 2500


def read_data(data_fn, label_fn):
    ## read label_fn
    with open(path + label_fn) as f:
        content = f.readlines()
    label = [int(x.strip()) for x in content]


    ## read data_fn
```

```
    with open(path + data_fn) as f:
        content = f.readlines()
    # remove '\n' at the end of each line
    content = [x.strip() for x in content]

    dat = np.zeros((len(content), 3), dtype = int)

    for i, line in enumerate(content):
        a = line.split(' ')
        dat[i, :] = np.array([int(a[0]), int(a[1]), int(a[2])])

    # remember to -1 at coordinate since we're in Python
    # check this: https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.coo
_matrix.html
    # for more information about coo_matrix function
    data = coo_matrix((dat[:, 2], (dat[:, 0] - 1, dat[:, 1] - 1)),\
             shape=(len(label), nwords))
    return (data, label)
```

Sau khi đọc training data và test data, sử dụng class MultinomialNB trong sklearn để xây dựng mô hình và dự đoán đầu ra cho test data.

```
(train_data, train_label)  = read_data(train_data_fn, train_label_fn)
(test_data, test_label)  = read_data(test_data_fn, test_label_fn)

clf = MultinomialNB()
clf.fit(train_data, train_label)
y_pred = clf.predict(test_data)
```

Thử tự in kết quả ra để kiểm tra!

2. PHƯƠNG PHÁP GAUSSIAN NAÏVE BAYES VỚI DỮ LIỆU LIÊN TỤC

Ví dụ 3. Phân loại bộ dữ liệu hoa IRIS

- Bộ dữ liệu hoa IRIS có thể lấy từ link sau (tệp excel csv): http://archive.ics.uci.edu/ml/datasets/Iris (hoặc tệp Iris.data – dạng văn bản kèm theo bài thực hành).
- Các thuộc tính của dữ liệu gồm:
    1. sepal length in cm (độ dài cánh hoa)
    2. sepal width in cm (độ rộng cánh hoa)
    3. petal length in cm (độ dài cánh đài)

4. petal width in cm (độ rộng cánh đài)

5. class: Dữ liệu trên là số đo của 03 loại hoa

-- Iris Setosa

-- Iris Versicolour

-- Iris Virginica

- Dữ liệu gồm 150 dòng (ứng với 150 mẫu), chia đều cho 03 loại hoa trên. Chúng ta lấy ngẫu nhiên mỗi loại hoa 40 mẫu dữ liệu dùng làm training data, phần còn lại dùng làm testing data.
- Do dữ liệu vào là số thực, vì vậy chúng ta sẽ phải dùng phương pháp Gaussian Naïve Bayes (tự xem lại phần lý thuyết).
- Để đơn giản trước hết ta chỉ căn cứ vào độ dài cánh hoa để phân loại. Các bước tính toán theo lý thuyết (sử dụng phương pháp ước lượng hợp lý cực đại – MLE):

- Setosa, Versicolor, Virginica ~ label as $y \in \{0, 1, 2\}$
- X_1 : sepal length ; X_2 : sepal width.

- **Means:** $\mu_{x_1|y=0}, \mu_{x_1|y=1}, \mu_{x_1|y=2}$ và $\mu_{x_2|y=0}, \mu_{x_2|y=1}, \mu_{x_2|y=2}$
- **Variance:** $\sigma^2_{x_1|y=0}, \sigma^2_{x_1|y=1}, \sigma^2_{x_1|y=2}$ và $\sigma^2_{x_2|y=0}, \sigma^2_{x_2|y=1}, \sigma^2_{x_2|y=2}$
- **Prior:** $p(y=0), p(y=1), p(y=2)$

$$\begin{aligned}\text{class}\,(\vec{x}) &= argmax_y \prod_{\alpha=1}^{2} p(x_\alpha|y)p(y) \\ &= argmax_y\{p(x_1|y)p(y) \cdot p(x_2|y)p(y)\} \\ &= argmax_y\{\phi(x_1|\mu_{x_1|y}, \sigma^2_{x_1|y})p(y) \cdot \{\phi(x_2|\mu_{x_2|y}, \sigma^2_{x_2|y})p(y)\}\end{aligned}$$

Trong đó,

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Chúng ta có thể giải bằng 02 thư viện, numpy và scikit-learning. Chú ý bộ dữ liệu IRIS khá chuẩn mực trong việc sử dụng làm minh họa cho bài toán classification, do đó có thể load từ thư viện của python.

a. Sử dụng Numpy & Pandas

Đoạn chương trình gọi thư viện và khởi tạo dữ liệu

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import matplotlib.colors as colors
import seaborn as sns
import itertools
from scipy.stats import norm
import scipy.stats
from sklearn.naive_bayes import GaussianNB

%matplotlib inline
sns.set()
```

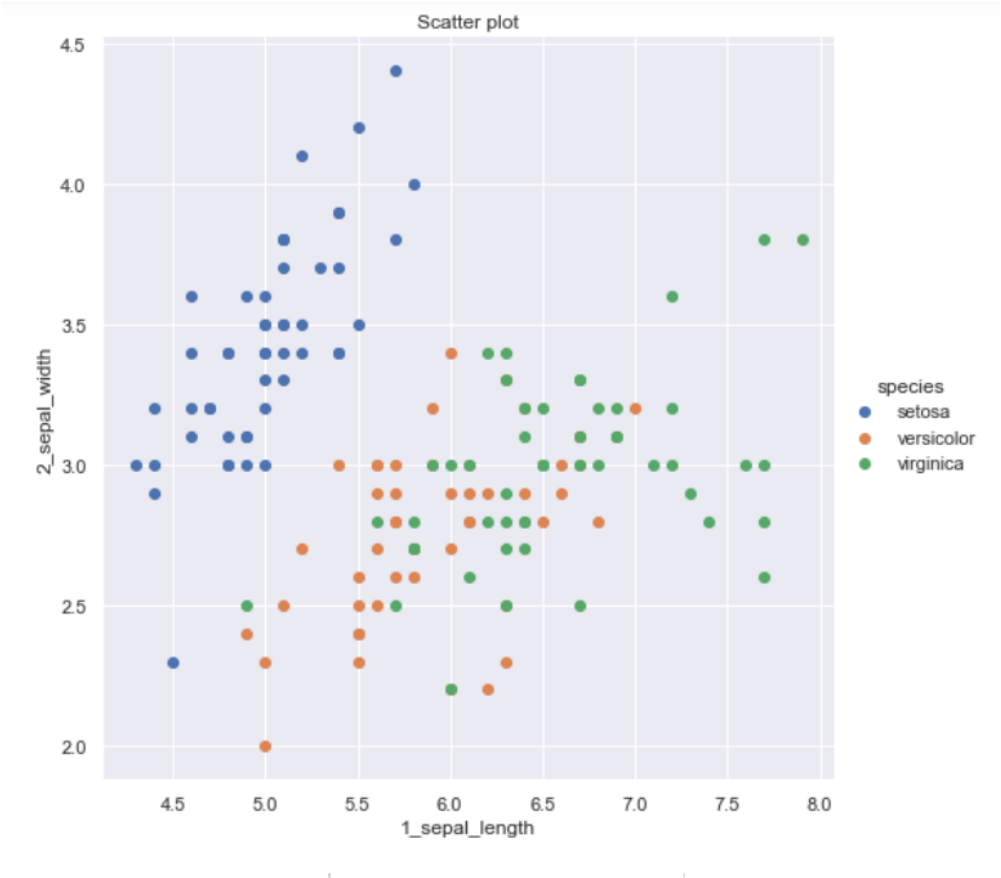
Đoạn chương trình vẽ phân bố các điểm dữ liệu để dễ hình dung (đoạn này có thể lỗi khi chạy onl do thư viện)

```
#Load the data set
```

```
iris = sns.load_dataset("iris")
iris = iris.rename(index = str, columns = {'sepal_length':'1_sepal_length','sepal_wid
th':'2_sepal_width', 'petal_length':'3_petal_length', 'petal_width':'4_petal_width'})

#Plot the scatter of sepal length vs sepal width
sns.FacetGrid(iris, hue="species", size=7) .map(plt.scatter,"1_sepal_length", "2_sepa
l_width", )  .add_legend()
plt.title('Scatter plot')
df1 = iris[["1_sepal_length", "2_sepal_width",'species']]
```

chạy trên Jupyter Notebook cho kết quả sau



Xử lý bằng phương pháp Gaussian Naïve Bayes dùng thư viện Numpy và Pandas

```
def predict_NB_gaussian_class(X,mu_list,std_list,pi_list):
    #Returns the class for which the Gaussian Naive Bayes objective function has greatest v
alue
    scores_list = []
    classes = len(mu_list)

    for p in range(classes):
        score = (norm.pdf(x = X[0], loc = mu_list[p][0][0], scale = std_list[p][0][0] )
                * norm.pdf(x = X[1], loc = mu_list[p][0][1], scale = std_list[p][0][1] )
                * pi_list[p])
        scores_list.append(score)

    return np.argmax(scores_list)

def predict_Bayes_class(X,mu_list,sigma_list):
    #Returns the predicted class from an optimal bayes classifier - distributions must be k
nown
    scores_list = []
    classes = len(mu_list)

    for p in range(classes):
        score = scipy.stats.multivariate_normal.pdf(X, mean=mu_list[p], cov=sigma_list[p])
        scores_list.append(score)

    return np.argmax(scores_list)
```

In và vẽ kết quả ra màn hình (chú ý đoạn này chạy trên Cocal.com có thể lỗi do thiếu thư viện)

```
#Estimating the parameters
mu_list = np.split(df1.groupby('species').mean().values,[1,2])
std_list = np.split(df1.groupby('species').std().values,[1,2], axis = 0)
pi_list = df1.iloc[:,2].value_counts().values / len(df1)

# Our 2-dimensional distribution will be over variables X and Y
N = 100
X = np.linspace(4, 8, N)
Y = np.linspace(1.5, 5, N)
X, Y = np.meshgrid(X, Y)

#fig = plt.figure(figsize = (10,10))
#ax = fig.gca()
color_list = ['Blues','Greens','Reds']
my_norm = colors.Normalize(vmin=-1.,vmax=1.)

g = sns.FacetGrid(iris, hue="species", size=10, palette = 'colorblind') .map(plt.scatter, "
1_sepal_length", "2_sepal_width",)  .add_legend()
my_ax = g.ax


#Computing the predicted class function for each value on the grid
zz = np.array(  [predict_NB_gaussian_class( np.array([xx,yy]).reshape(-1,1), mu_list, std_l
ist, pi_list)
                     for xx, yy in zip(np.ravel(X), np.ravel(Y)) ] )


#Reshaping the predicted class into the meshgrid shape
Z = zz.reshape(X.shape)


#Plot the filled and boundary contours
my_ax.contourf( X, Y, Z, 2, alpha = .1, colors = ('blue','green','red'))
my_ax.contour( X, Y, Z, 2, alpha = 1, colors = ('blue','green','red'))

# Addd axis and title
my_ax.set_xlabel('Sepal length')
my_ax.set_ylabel('Sepal width')
my_ax.set_title('Gaussian Naive Bayes decision boundaries')

plt.show()
```

b. Sử dụng scikit-learning

Đoạn chương trình dưới đây sử dụng thư viện sklearn của python

```
from sklearn.naive_bayes import GaussianNB

#Setup X and y data
X_data = df1.iloc[:,0:2]
y_labels = df1.iloc[:,2].replace({'setosa':0,'versicolor':1,'virginica':2}).copy()

#Fit model
model_sk = GaussianNB(priors = None)
model_sk.fit(X_data,y_labels)


# Our 2-dimensional classifier will be over variables X and Y
N = 100
X = np.linspace(4, 8, N)
Y = np.linspace(1.5, 5, N)
X, Y = np.meshgrid(X, Y)
```

```
#fig = plt.figure(figsize = (10,10))
#ax = fig.gca()
color_list = ['Blues','Greens','Reds']
my_norm = colors.Normalize(vmin=-1.,vmax=1.)

g = sns.FacetGrid(iris, hue="species", size=10, palette = 'colorblind') .map(plt.scatter, "
1_sepal_length", "2_sepal_width",)  .add_legend()
my_ax = g.ax


#Computing the predicted class function for each value on the grid
zz = np.array(  [model_sk.predict( [[xx,yy]])[0] for xx, yy in zip(np.ravel(X), np.ravel(Y)
) ] )


#Reshaping the predicted class into the meshgrid shape
Z = zz.reshape(X.shape)


#Plot the filled and boundary contours
my_ax.contourf( X, Y, Z, 2, alpha = .1, colors = ('blue','green','red'))
my_ax.contour( X, Y, Z, 2, alpha = 1, colors = ('blue','green','red'))

# Addd axis and title
my_ax.set_xlabel('Sepal length')
my_ax.set_ylabel('Sepal width')
my_ax.set_title('Gaussian Naive Bayes decision boundaries')

plt.show()
```

BÀI TẬP TỰ GIẢI. MÔ HÌNH PHÂN LOẠI NAÏVE BAYES

Xét tập dữ liệu phân loại bệnh nhân ung thư vú của Đại học Wisconsin–Madison, Hoa Kỳ tại link: https://ftp.cs.wisc.edu/math-prog/cpo-dataset/machine-learn/cancer/cancer1/

Để dự đoán một bệnh nhân có khả năng ung thư hay không, người ta thu thập các mẫu sinh thiết, xét nghiệm bằng việc lấy một lát cắt của mẫu để soi dưới kính hiển vi, quét và ghi lại mẫu dưới dạng các khung ảnh số của các nhân tế bào. Từ đó, người ta tiến hành đo kích thước, hình dạng, kết cấu của chúng và tính toán 10 đặc tính của nhân. Cụ thể trong tệp dữ liệu các trường sẽ là

1. Mã số mẫu
2. Loại: 2 cho lành tính, 4 cho ác tính (2 for benign, 4 for malignant)
3. Độ dày hạch (Clump Thickness)
4. Sự đồng nhất của kích thước tế bào (Uniformity of Cell Size)
5. Tính đồng nhất của hình dạng tế bào (Uniformity of Cell Shape)
6. Độ kết dính lề - biên (Marginal Adhesion)
7. Kích thước tế bào biểu mô đơn (Single Epithelial Cell Size)
8. Phần nhân ngoài (Bare Nuclei)
9. Chất nhiễm sắc thể (Bland Chromatin)
10. Nhân trong thường (Normal Nucleoli)
11. Vỏ (Mitoses)

Tìm hiểu thêm về dữ liệu bằng cách đọc tệp data.doc trong cùng thư mục. Trong số 698 mẫu thu thập (tệp datacum), người ta chia thành hai nhóm, u lành tính (*B*–benign) và u ác tính (*M*–malignant). Hãy tách dữ liệu thành các phần như sau:

- Chọn 80 mẫu lành tính (Trường phân loại là 2) và 40 mẫu ác tính (Trường phân loại là 4) làm dữ liệu Test, chúng ta sẽ không sử dụng trường phân loại. Phần còn lại sẽ là dữ liệu training.
- Sử dụng mô hình phân loại Naïve Bayes (chú ý dữ liệu quan sát là biến liên tục – cần chọn phương pháp phù hợp) để lập chương trình thực hiện phân loại dựa trên dữ liệu training.
- Sau đó thực hiện việc kiểm chứng và so sánh độ chính xác theo các chỉ số Acurracy, Precision và Recall. Có thể sử dụng thư viện, nhưng cần có giải thích rõ các phần chương trình.