MỤC LỤC

BÀI THỰC HÀNH SỐ 2

I. Mục đích:

Để sử dụng và hiện thực các hàm cơ bản ứng dụng trong thống kê và các classifier đơn giản với 1 đặc trưng và 2 lớp. Histogram được sử dụng để xác định biệt số (discriminant) sao cho tối thiểu misclassification.

II. Nội dung:

Bài tập 1*:* Xây dựng một phân lớp dựa trên histogram, với tập dữ liệu sau:

- in_time = [(0, 27), (1, 25), (2, 16), (3, 19), (4, 26), (5, 20), (6, 19), (7, 17), (8, 10), (9, 5), (10, 4), (11, 4), (12,2)]
- cls_late = [(5,3), (6, 5), (7, 8), (8, 15), (9, 17), (10, 18), (11, 19), (12,16), (13, 9), (14, 8), (15, 8)]

Kiểm tra kết quả nếu rời nhà lúc 6:34, 6:35, 6:36, 6:37, 6:38.

Bước 1: Import thư viện cần sử dụng:

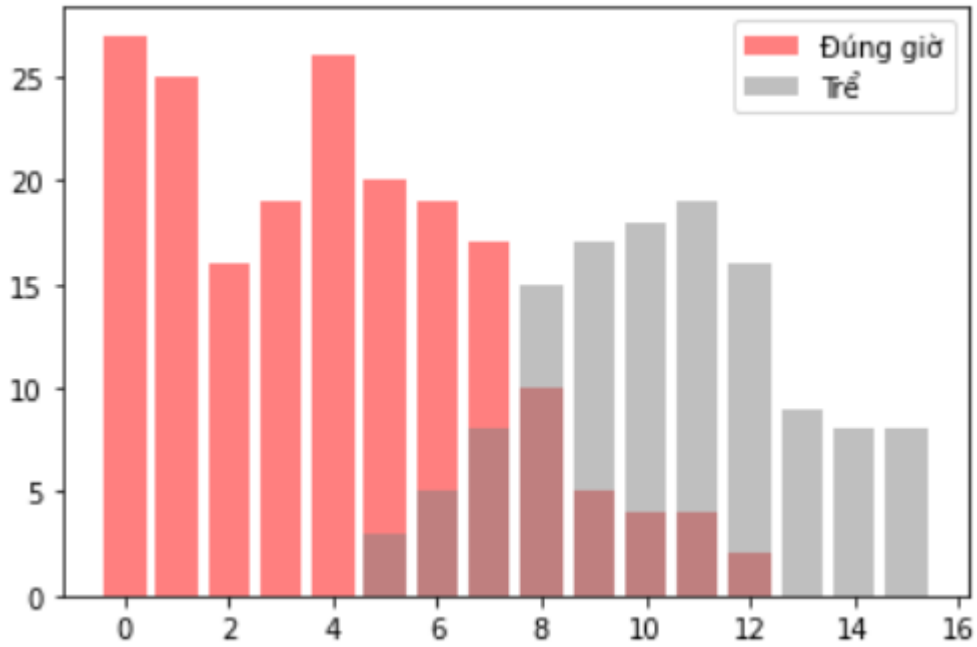
```
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
```

Bước 2: Khai báo 2 danh sách các điểm dữ liệu in_time và cls_late theo yêu cầu đề bài:

```
in_time=[(0,27),(1,25),(2,16),(3,19),(4,26),(5,20),(6,19),(7,17)
                              ,(8,10),(9,5),(10,4),(11,4),(12,2)]
cls_late = [(5,3), (6, 5), (7, 8), (8, 15), (9, 17), (10, 18),
                    (11, 19), (12,16), (13, 9),(14, 8), (15, 8)]
```

Bước 3: Sử dụng thư viện **matplotlib** biểu diễn dữ liệu bằng biểu đồ histogram nhằm trực quan hóa dữ liệu:

```
X, Y = zip(*in_time)
X2, Y2 = zip(*cls_late)
bar_width = 0.8
plt.bar(X, Y, bar_width, color="red", alpha=0.5, label="Đúng giờ")
plt.bar(X2, Y2, bar_width, color="gray", alpha=0.5, label="Trể")
plt.legend(loc='upper right')
plt.show()
```



Bước 4: Sau khi quan sát biểu đồ histogram, dễ thấy ngưỡng để phân loại dữ liệu vào 2 tập "đúng giờ" và "trễ giờ" là 1 điểm.

```
def pre_sum(A):
    cur=0
    res=np.zeros((len(A)))
    for i in range(len(A)):
        cur+=A[i]
        res[i]=cur
    return res

def back_sum(A):
    cur=0
    n=len(A)
    res=np.zeros((n))
    for i in range(n):
        cur+=A[n-i-1]
        res[n-i-1]=cur
    return res
```

```
pre_sum(vi_dat)
```

```
Out:array([1.        , 2.        , 3.        , 4.        , 5.
,
       5.86956522, 6.66123188, 7.34123188, 7.74123188, 7.9685046
1,
       8.15032279, 8.32423584, 8.43534695, 8.43534695, 8.4353469
5])
```

```
back_sum(vi_dat)
```

```
array([8.43534695, 7.43534695, 6.43534695, 5.43534695, 4.4353469
  5, 3.43534695, 2.56578173, 1.77411506, 1.09411506, 0.69411506,
                   0.46684234, 0.28502415, 0.11111111, 0., 0.])
```

```
vi_dat = np.array(vi_dat)
def find_thresh_holding(A,B):
    sumA = np.sum(A)
    sumB = np.sum(B)
    AA = back_sum(A)
    BB = pre_sum(B)
    C = np.zeros((len(A)))
    print(C)
    for i in range(len(A)-1):
        C[i] = (sumA-AA[i+1]+sumB-BB[i])
    for i in range(len(A)-1):
        if(C[i+1]<C[i]):
            return [i,C[i]]
```

```
thresh,__ = find_thresh_holding(vi_dat,1-vi_dat)
```

```
def testt(a):
    if a>thresh:
        print('Ket qua luc 6:{} la tre'.format(30+a))
    else:
        print('Ket qua luc 6:{} la dung gio'.format(30+a))
```

```
test = [4,5,6,7,8]
for i in test:
    testt(i)
```

```
Ket qua luc 6:34 la dung gio
Ket qua luc 6:35 la dung gio
Ket qua luc 6:36 la dung gio
Ket qua luc 6:37 la dung gio
Ket qua luc 6:38 la tre
```

Bài 3: Tính mean và variance của các vector đặc trưng sau:

a. [1 2 4 6 9 10 20 7]
b. [0 2 4 6 8 … 100]; tất cả các số chẵn từ 0 đến 100.
c. [1 3 25 … 9801]; tất cả bình phương các số lẻ từ 1 đến 100.
d. $\begin{bmatrix}\begin{bmatrix}2\\4\end{bmatrix} & \begin{bmatrix}3\\7\end{bmatrix} & \begin{bmatrix}4\\6\end{bmatrix} & \begin{bmatrix}5\\5\end{bmatrix} & \begin{bmatrix}2\\3\end{bmatrix}\end{bmatrix}$

```
  def mean(x):
    s = 0
    for i in x:
      s += i
    return s/len(x)

  def variance(x):
    mu = mean(x)
    s = 0
    for i in x:
      s += (i - mu)**2
    return s/len(x)
```

```
import numpy as np
a = [1,2,3,4,6,9,10,20,7]

print("Trung bình của a là:")
print(mean(a))
print("Variance của a là:")
print(variance(a))
```

```
Trung bình của a là:
6.888888888888889
Variance của a là:
29.876543209876544
```

```
b = list(range(2,101,2))
print("Trung bình của b là:")
print(mean(b))
print("Variance của b là:")
print(variance(b))
```

```
Trung bình của b là:
50.0
Variance của b là:
850.0
```

```
c = list(range(1, 100, 2))
def power(my_list):
    return [x**2 for x in my_list]
c = power(c)
print("Trung bình của c là:")
print(mean(c))
print("Variance của c là:")
print(variance(c))
```

```
Trung bình của c là:
51.0
Variance của c là:
833.0
```

```
def mean_2D(x):
  s = 0
  l = 0
  for i in x:
    for j in i:
      s += j
      l += 1
  return s/l

def variance_2D(x):
  mu = mean_2D(x)
  s = 0
  l = 0
  for i in x:
    for j in i:
      s += (j - mu)**2
      l += 1
  return s/l
```

```
d = [[2,4], [3,7], [4,6], [5,5], [2,3]]
print("Trung bình của d là:")
print(mean_2D(d))
print("Variance của d là:")
print(variance_2D(d))
```

```
Trung bình của d là:
4.1
Variance của d là:
2.4900000000000007
```

Bài 4: Tính covariance matrix của các vector đặc trưng sau:

X=[2 3 6 3 7 8] và Y=[5 7 9 6 7 8].

```
X = [2, 3, 6, 3, 7, 8]
Y = [5, 7, 9, 6, 7, 8]
matrix_1 = [X, Y]
print("Gộp 2 ma trận:")
print(matrix_1)
```

```
Gộp 2 ma trận:
[[2, 3, 6, 3, 7, 8], [5, 7, 9, 6, 7, 8]]
```

```
print("Tính trung bình ma trận:")
avg_matrix = [sum(matrix_1[0])/len(matrix_1[0]), sum(matrix_1[1])/l
en(matrix_1[1])]
print(avg_matrix)
```

```
Tính trung bình ma trận:
[4.833333333333333, 7.0]
```

```
print("Hiệu ma trận ban đầu với trung bình:")
matrix_2 = []
for i in range(len(matrix_1)):
  line = []
  for j in range(len(matrix_1[i])):
    line.append(matrix_1[i][j] - avg_matrix[i])
  matrix_2.append(line)
print(matrix_2)
```

```
Hiệu ma trận ban đầu với trung bình:
[
    [-2.833333333333333, -1.833333333333333, 1.166666666666667,
     -1.833333333333333, 2.166666666666667, 3.166666666666667],
    [-2.0, 0.0, 2.0, -1.0, 0.0, 1.0]
]
```

```
matrix_2 = np.asarray(matrix_2)
```

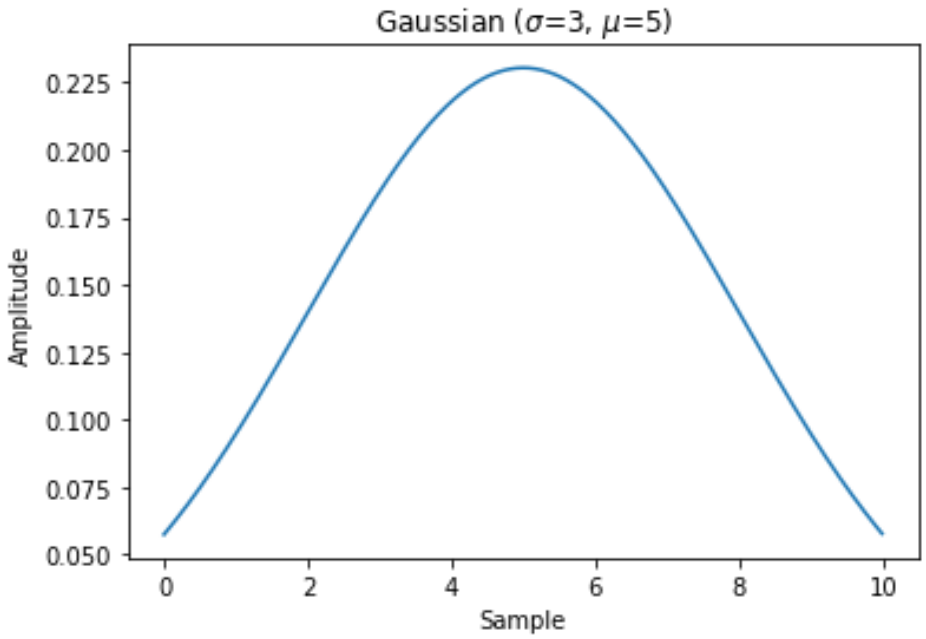
```
N = 6
print("Ma trận hiệp phương sai với N = " + str(N))
print((1/(N-1))*np.dot(matrix_2, matrix_2.T))
```

```
Ma trận hiệp phương sai với N = 6
[[6.16666667 2.6       ]
 [2.6        2.        ]]
```

Bài 5: Tạo hàm mật độ của phân bố Gauss với mean là 5 và variance là 3. Plot hàm kết quả.

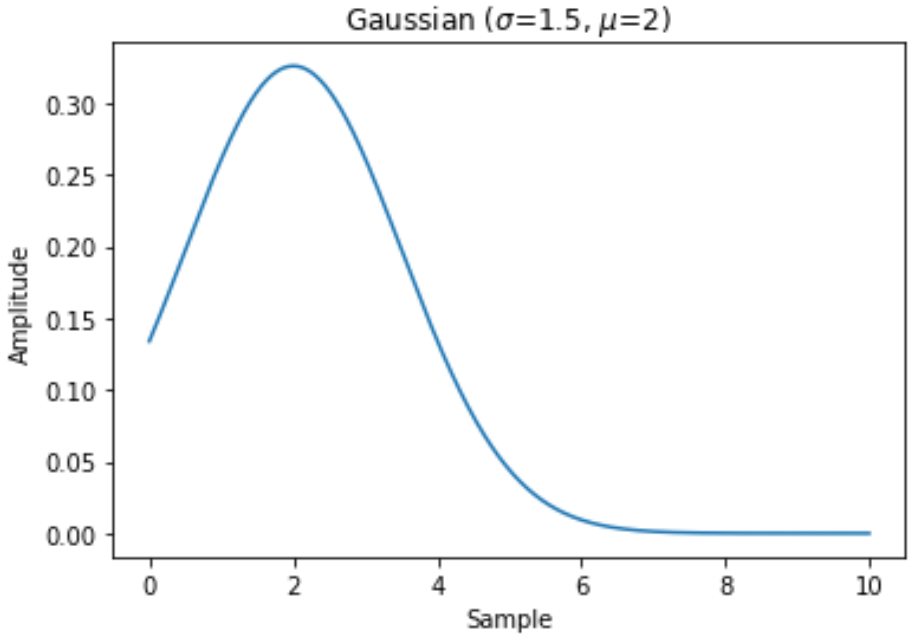
```
def gauss(x, mean, variance):
  return (1/(math.sqrt(2*math.pi*variance)))*math.exp((-1/2)*((x-
mean)/variance)**2)
```

```
import numpy as np
import matplotlib.pyplot as plt
import math

mean = 5
variance = 3
list_x = np.arange(0, 10, 0.01)
list_y = []
for x in list_x:
  list_y.append(gauss(x, mean, variance))
plt.plot(list_x, list_y)
plt.title(r"Gaussian ($\sigma$="+ str(variance) + ", $\mu$=" + str(
mean) + ")")
plt.ylabel("Amplitude")
plt.xlabel("Sample")
plt.show()
```



Bài 6: Tạo hàm mật độ của phân bố Gauss khác với mean là 2 và variance là 1.5. Plot hàm này trong cùng cửa sổ với hàm được tạo ra ở câu 3. Cho nhận xét.

```
mean = 2
variance = 1.5
list_x = np.arange(0, 10, 0.01)
list_y = []
for x in list_x:
  list_y.append(gauss(x, mean, variance))
plt.plot(list_x, list_y)
plt.title(r"Gaussian ($\sigma$="+ str(variance) + ", $\mu$=" + str(
mean) + ")")
plt.ylabel("Amplitude")
plt.xlabel("Sample")
plt.show()
```

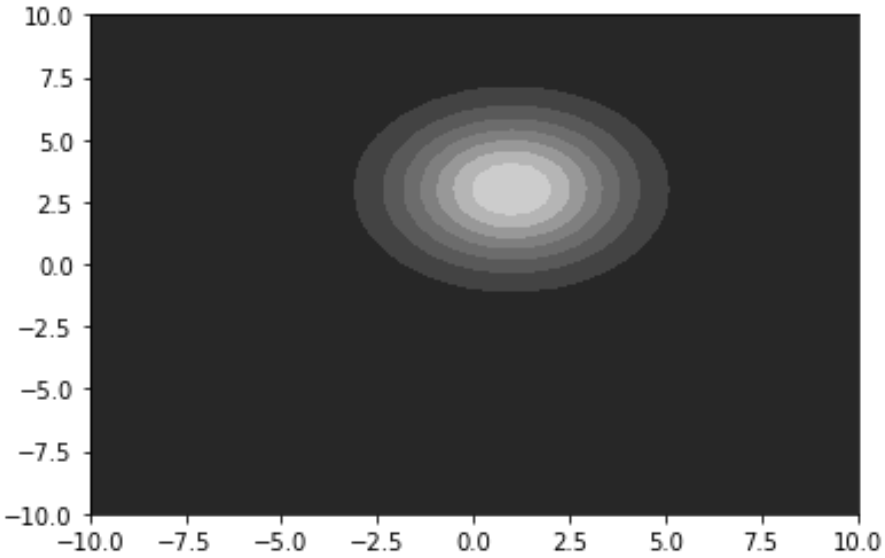


Nhận xét: biểu đồ bị lệch sang trái so với biểu đồ có mean = 3 và variance = 5

Bài 7: Tạo hàm mật độ phân bố Gauss 2 chiều với mean [1 3] và variance [2 2]. Plot hàm này trên lưới [-10 10] x [-10 10] và tính khoảng cách Mahalanobis đối với các mẫu [0 0], [3 4], và [1 2].

```
import numpy as np

def gaus2d(x=0, y=0, mx=0, my=0, sx=1, sy=1):
    return 1. / (2. * np.pi * sx * sy) * np.exp(-((x - mx)**2.
             / (2. * sx**2.) + (y - my)**2. / (2. * sy**2.)))

x = np.linspace(-10, 10)
y = np.linspace(-10, 10)
x, y = np.meshgrid(x, y)
```

```
z = gaus2d(x, y, 1, 3, 2, 2)
plt.contourf(x,y,z)
plt.show()
```




```
sample_1 = [0,0]
list_x1 = []
for i in range(50):
  list_x1.append(sample_1)
m1 = np.asarray(list_x1)
print("Khoảng cách mahalanobis của điểm [0,0]:")
print(np.dot(np.dot(m1.T,np.cov(z)), m1))
print()

sample_1 = [3,4]
list_x1 = []
for i in range(50):
  list_x1.append(sample_1)
m1 = np.asarray(list_x1)
print("Khoảng cách mahalanobis của điểm [3,4]:")
print(np.dot(np.dot(m1.T,np.cov(z)), m1))
print()

sample_1 = [1,2]
list_x1 = []
for i in range(50):
  list_x1.append(sample_1)
m1 = np.asarray(list_x1)
print("Khoảng cách mahalanobis của điểm [1,2]:")
print(np.dot(np.dot(m1.T,np.cov(z)), m1))
print()
```

```
Khoảng cách mahalanobis của điểm [0,0]:
[[0. 0.]
 [0. 0.]]

Khoảng cách mahalanobis của điểm [3,4]:
[[0.24855543 0.33140724]
 [0.33140724 0.44187632]]

Khoảng cách mahalanobis của điểm [1,2]:
[[0.02761727 0.05523454]
 [0.05523454 0.11046908]]
```

Bài 8: Xây dựng bộ classifier sử dụng 1 đặc trưng có sẵn.

- ***Load data:***

- Load 2 file tương ứng cho 2 class là: *class1.txt* và *class2.txt*.
- Cho biết số mẫu và số đặc trưng của mỗi class.
- Tính mean, variance, và covariance của các vector đặc trưng.
- Trích chọn 1 đặc trưng.

```
def covariance(x):
  N = len(x)
  TB = mean(x)
  M = []
  for i in x:
    M.append(i - TB)
  D = np.asarray(M)
  return  (1/(N-1))*D*D.T
```

```
datContent = [i.strip().split()
                  for i in open("twoclass.dat", "r").readlines()]
print("Số mẫu là: " + str(len(datContent)))
print("Số đặc trưng của mỗi class là: "
                                    + str(len(datContent[0][:-2])))
for i in range(len(datContent[0][:-2])):
  print("Trung bình của đặc trưng " + str(i+1) + " là: "
              + str(mean([float(row[i]) for row in datContent])))
  print("Phương sai của đặc trưng " + str(i+1) + " là: "

+ str(variance([float(row[i]) for row in datContent])))
  print("Hiệp phương sai của đặc trưng " + str(i+1) + " là: "

+ str(np.cov([float(row[i]) for row in datContent])))
  print()

# Chọn đặc trưng 1
feature_1 = [float(row[0]) for row in datContent]
feature_2 = [float(row[1]) for row in datContent]
feature_3 = [float(row[2]) for row in datContent]
feature_4 = [float(row[3]) for row in datContent]
class_1 = [int(row[4]) for row in datContent]

# Tạo list index
idx = np.arange(0, len(feature_1))
# Randơm lại list index
np.random.shuffle(idx)
# Lấy ra phân nửa list index đã random
idx_train = idx[:len(feature_1)//2]
idx_test = idx[len(feature_1)//2:]

# Tách tập train
feature_1_train = [feature_1[i] for i in idx_train]
feature_2_train = [feature_2[i] for i in idx_train]
feature_3_train = [feature_3[i] for i in idx_train]
feature_4_train = [feature_4[i] for i in idx_train]

class_train = [class_1[i] for i in idx_train]

# Tách tập test
feature_1_test = [feature_1[i] for i in idx_test]
feature_2_test = [feature_2[i] for i in idx_test]
feature_3_test = [feature_3[i] for i in idx_test]
feature_4_test = [feature_4[i] for i in idx_test]

class_test = [class_1[i] for i in idx_test]
```

```
Số mẫu là: 242
Số đặc trưng của mỗi class là: 4
Trung bình của đặc trưng 1 là: 0.06021218181818183
Phương sai của đặc trưng 1 là: 0.8578042003907271
Hiệp phương sai của đặc trưng 1 là: 0.8613635539193196

Trung bình của đặc trưng 2 là: 0.055756892561983476
Phương sai của đặc trưng 2 là: 1.0511796321486822
Hiệp phương sai của đặc trưng 2 là: 1.055541373360918

Trung bình của đặc trưng 3 là: 1.4545952520661152
Phương sai của đặc trưng 3 là: 4.234982801030213
Hiệp phương sai của đặc trưng 3 là: 4.252555343773077

Trung bình của đặc trưng 4 là: -1.388338768595042
Phương sai của đặc trưng 4 là: 4.359071929443203
Hiệp phương sai của đặc trưng 4 là: 4.377159364835083
```

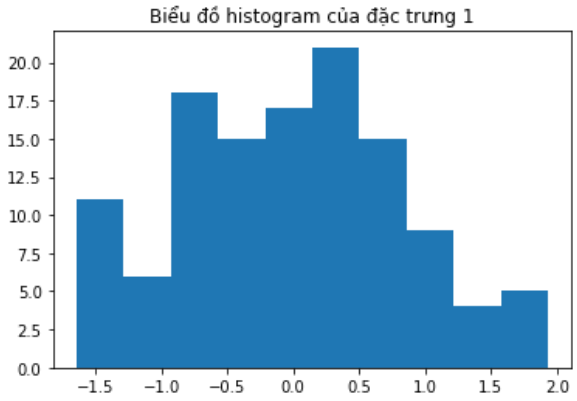
- ***Xây dựng classifier ứng với 1 đặc trưng được chọn:***

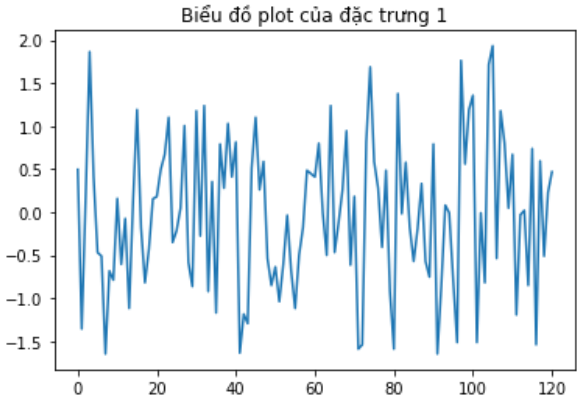
- Chia tập dữ liệu thành 2 tập con, tập huấn luyện gồm 50%.
- Tính histogram và plot:
- Xác định giá trị biệt số (ngưỡng). Ứng với histogram được tính, chọn biệt số sao cho giá trị misclassification là nhỏ nhất.
- Đánh giá trên tập dữ liệu test: ứng với giá trị ngưỡng được chọn (biệt số), chúng ta cần phải đánh giá trên 1 tập dữ liệu khác (dữ liệu không được dùng trong huấn luyện); tập dữ liệu test. Dựa vào các giá trị đặc trưng tưng ứng cho các mẫu test và giá trị ngưỡng được xác định, tính phần trăm lỗi.

Tương tự, thực nghiệm với các cách phân chia tập dữ liệu khác nhau (tập dữ liệu huấn luyện là 60%, 70%, và 80%) và các bin của histogram khác nhau.

Cho biết đặc trưng có khả năng phân loại tốt nhất.

```
import matplotlib.pyplot as plt

plt.hist(feature_1_train)
plt.title("Biểu đồ histogram của đặc trưng 1")
plt.show()
plt.plot(feature_1_train)
plt.title("Biểu đồ plot của đặc trưng 1")
plt.show()
```






```
def find_thresh(x, label):
  thresh = None
  num_errors = 999999
  for i in x:
    errors = 0
    for d, l in zip(x, label):
      if d > i and l == 1:
        errors += 1
      elif d <= i and l == 0:
        errors += 1
    if errors < num_errors:
      num_errors = errors
      thresh = i
  return (thresh, num_errors)

def find_error_with_thresh(x, label, thresh):
  num_errors = 0
  for d, l in zip(x, label):
    if d > thresh and l == 1:
      num_errors += 1
    elif d <= thresh and l == 0:
      num_errors += 1
  return num_errors
```

```
# Đặc trưng 1
fea_1_test = find_thresh(feature_1_train, class_train)
print("Ngưỡng tìm được từ tập train đặc trưng 1 là: "
                                    + str(fea_1_test[0]))
print("Số lỗi tìm được từ tập train đặc trưng 1 là: "
                                    + str(fea_1_test[1]))
print("Số lỗi tìm được từ tập test đặc trưng 1 là: "

        + str(find_error_with_thresh(feature_1_test, class_test,
                                                fea_1_test[0])))
print()
```

```
# Đặc trưng 2
fea_2_test = find_thresh(feature_2_train, class_train)
print("Ngưỡng tìm được từ tập train đặc trưng 2 là: "
                                          + str(fea_2_test[0]))
print("Số lỗi tìm được từ tập train đặc trưng 2 là: "
                                          + str(fea_2_test[1]))
print("Số lỗi tìm được từ tập test đặc trưng 2 là: "
          + str(find_error_with_thresh(feature_2_test, class_test,
          fea_2_test[0])))
print()

# Đặc trưng 3
fea_3_test = find_thresh(feature_3_train, class_train)
print("Ngưỡng tìm được từ tập train đặc trưng 3 là: "
                                         + str(fea_3_test[0]))
print("Số lỗi tìm được từ tập train đặc trưng 3 là: "
                                         + str(fea_3_test[1]))
print("Số lỗi tìm được từ tập test đặc trưng 3 là: "
          + str(find_error_with_thresh(feature_3_test, class_test,
            fea_3_test[0])))
print()

# Đặc trưng 4
fea_4_test = find_thresh(feature_4_train, class_train)
print("Ngưỡng tìm được từ tập train đặc trưng 4 là: "
                                        + str(fea_4_test[0]))
print("Số lỗi tìm được từ tập train đặc trưng 4 là: "
                                        + str(fea_4_test[1]))
print("Số lỗi tìm được từ tập test đặc trưng 4 là: "
          + str(find_error_with_thresh(feature_4_test, class_test,
            fea_4_test[0])))
print()
```

```
Ngưỡng tìm được từ tập train đặc trưng 1 là: 1.761705
Số lỗi tìm được từ tập train đặc trưng 1 là: 55
Số lỗi tìm được từ tập test đặc trưng 1 là: 66

Ngưỡng tìm được từ tập train đặc trưng 2 là: 2.284433
Số lỗi tìm được từ tập train đặc trưng 2 là: 57
Số lỗi tìm được từ tập test đặc trưng 2 là: 64

Ngưỡng tìm được từ tập train đặc trưng 3 là: 7.060067
Số lỗi tìm được từ tập train đặc trưng 3 là: 57
Số lỗi tìm được từ tập test đặc trưng 3 là: 66

Ngưỡng tìm được từ tập train đặc trưng 4 là: 2.397017
Số lỗi tìm được từ tập train đặc trưng 4 là: 57
Số lỗi tìm được từ tập test đặc trưng 4 là: 65
```

Từ kết quả trên ta có thể thấy là đặc trưng 1 và 4 là cho kết quả tốt nhất. Vì số điểm lỗi trên tập train với thực tế chỉ cách nhau thấp nhất.

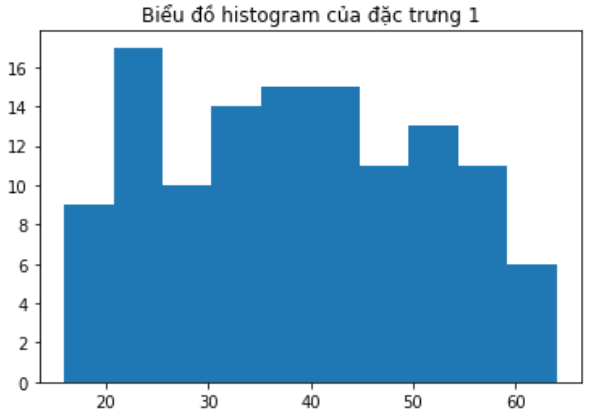
Bài 9: Thực hiện tương tự như bài 8 với 2 tập dữ liệu là cross (cross.dat) và twoclass (twoclass.dat). Chú ý dữ liệu cho các class đều gom chung 1 file và 2 đặc trưng cuối để chỉ ra class, ví dụ [1 0] tương ứng cho class1 và [0 1] tương ứng cho class2.

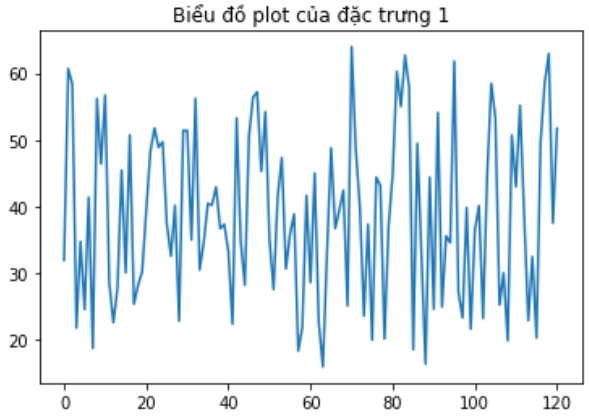
```
datContent_cross = [i.strip().split()
                    for i in open("cross.dat", "r").readlines()]
print("Số mẫu là: " + str(len(datContent_cross)))
print("Số đặc trưng của mỗi class là: "
                           + str(len(datContent_cross[0][:-2])))
for i in range(len(datContent_cross[0][:-2])):
  print("Trung bình của đặc trưng " + str(i+1) + " là: "
                                    + str(mean([float(row[i])
for row in datContent_cross]))):
  print("Phương sai của đặc trưng " + str(i+1) + " là: "
                           + str(variance([float(row[i])
for row in datContent_cross]))):
  print("Hiệp phương sai của đặc trưng " + str(i+1) + " là: "
            str(np.cov([float(row[i])
for row in datContent_cross])))

# Chọn đặc trưng 1
feature_1_cross = [float(row[0]) for row in datContent_cross]
feature_2_cross = [float(row[1]) for row in datContent_cross]
class_1_cross = [int(row[2]) for row in datContent_cross]

# Tạo list index
idx = np.arange(0, len(feature_1_cross))

# Random lại list index
np.random.shuffle(idx)

# Lấy ra phân nửa list index đã random
idx_train = idx[:len(feature_1)//2]
idx_test = idx[len(feature_1)//2:]

# Tách tập train
feature_1_train_cross = [feature_1_cross[i] for i in idx_train]
feature_2_train_cross = [feature_2_cross[i] for i in idx_train]

class_train_cross = [class_1_cross[i] for i in idx_train]

# Tách tập test
```

```
feature_1_test_cross = [feature_1_cross[i] for i in idx_test]
feature_2_test_cross = [feature_2_cross[i] for i in idx_test]

class_test_cross = [class_1_cross[i] for i in idx_test]
```

```
Số mẫu là: 200
Số đặc trưng của mỗi class là: 2
Trung bình của đặc trưng 1 là: 40.38131579000002
Phương sai của đặc trưng 1 là: 157.35237854889428
Hiệp phương sai của đặc trưng 1 là: 158.14309401898927

Trung bình của đặc trưng 2 là: 40.393948170000016
Phương sai của đặc trưng 2 là: 171.44175718153042
Hiệp phương sai của đặc trưng 2 là: 172.30327354927695
```

```
import matplotlib.pyplot as plt

plt.hist(feature_1_train_cross)
plt.title("Biểu đồ histogram của đặc trưng 1")
plt.show()
plt.plot(feature_1_train_cross)
plt.title("Biểu đồ plot của đặc trưng 1")
plt.show()
```






```
# Đặc trưng 1
fea_1_cross_test = find_thresh(feature_1_train_cross,
                                                class_train_cross)
print("Ngưỡng tìm được từ tập train đặc trưng 4 là: "
                                          + str(fea_1_cross_test[0]))
print("Số lỗi tìm được từ tập train đặc trưng 4 là: "
                                          + str(fea_1_cross_test[1]))
print("Số lỗi tìm được từ tập test đặc trưng 4 là: "
                  +str(find_error_with_thresh(feature_1_test_cross,
                  class_test_cross, fea_1_cross_test[0])))
print()

# Đặc trưng 2
```

```
fea_2_cross_test = find_thresh(feature_2_train_cross,
                                                class_train_cross)
print("Ngưỡng tìm được từ tập train đặc trưng 4 là: "
                                            + str(fea_2_cross_test[0]))
print("Số lỗi tìm được từ tập train đặc trưng 4 là: "
                                            + str(fea_2_cross_test[1]))
print("Số lỗi tìm được từ tập test đặc trưng 4 là: "
                  str(find_error_with_thresh(feature_2_test_cross,
                  class_test_cross, fea_2_cross_test[0])))
```

```
Ngưỡng tìm được từ tập train đặc trưng 4 là: 33.264324
Số lỗi tìm được từ tập train đặc trưng 4 là: 34
Số lỗi tìm được từ tập test đặc trưng 4 là: 32

Ngưỡng tìm được từ tập train đặc trưng 4 là: 32.043167
Số lỗi tìm được từ tập train đặc trưng 4 là: 35
Số lỗi tìm được từ tập test đặc trưng 4 là: 26
```

Từ kết quả trên ta có thể thấy là đặc trưng 2 là cho kết quả tốt nhất. Vì số điểm lỗi trên tập test thấp.

BÀI THỰC HÀNH SỐ 3

I. Mục đích:

Xây dựng classifier và biệt thức dựa trên luật Bayes.

II. Báo cáo:

Bài 1: Cho 2 tập dữ liệu class A (classA.mat) và class B (classB.mat). Xây dựng bộ classifier với 2 đặc trưng. Giả sử hai tập dữ liệu có dạng phân bố Gauss có cùng ma trận hiệp phương sai là SIGMA=[1 0; 0 1].

- *Bước 1:* Import thư viện:

```
import numpy as np
from scipy.io import loadmat
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
import scipy.io
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

- *Bước 2:* Load dữ liệu: Load 2 file tương ứng cho 2 class là: *classA.mat* và *classB.mat*.

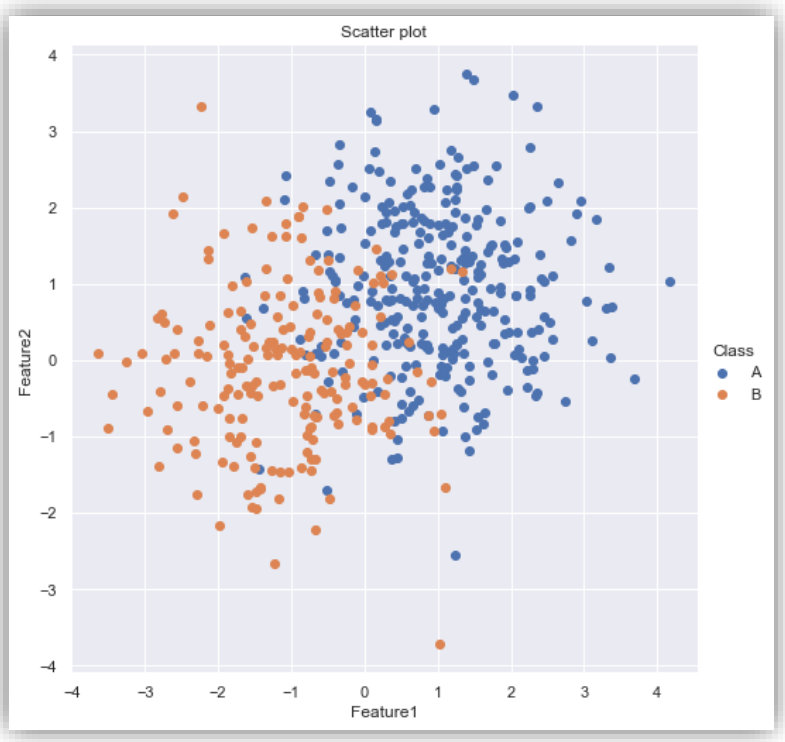
```
columnsName = ['Feature1','Feature2']
classA = pd.DataFrame(loadmat("classA.mat")['classA'],columns =

columnsName)
classB = pd.DataFrame(loadmat("classB.mat")['classB'],columns =

columnsName)
```

```
#Merge data
classAB = pd.concat([classA,classB], keys=['A',
'B']).reset_index()
    .drop('level_1', axis=1).rename(columns = {'level_0':
'Class'})
classAB.head()
```

	Class	Feature1	Feature2
0	A	0.572257	2.179579
1	A	0.420627	1.752968
2	A	1.925968	1.484078
3	A	1.005510	1.775438
4	A	0.365509	0.943339

- *Bước 3:* Plot dữ liệu.

```
sns.FacetGrid(classAB, hue="Class", height=7).map(plt.scatter,
                          "Feature1","Feature2",).add_legend()
plt.title('Scatter plot')
plt.show()
```



- *Bước 4:* Xác định số mẫu của mỗi class.

```
#Estimating the data
nA = len(classA)
nB = len(classB)
print("Số mẫu của class A là :",nA)
print("Số mẫu của class B là :",nB)
```

```
Số mẫu của class A là : 300
Số mẫu của class B là : 195
```

- *Bước 5:* Phân chia tập dữ liệu thành 2 tập con: tập huấn luyện (60%) và tập kiểm thử (40%).

```
train_set, test_set = train_test_split(classAB, train_size=0.7)
print("Số lượng của tập train là :",len(train_set))
print("Số lượng của tập train là :",len(test_set))
```

```
Số lượng của tập train là : 346
Số lượng của tập train là : 149
```

Huấn luyện

- Tính mean tương ứng cho từng class.

```
#Estimating the parameters
mu_list = np.split(train_set.groupby('Class').mean().values,[1])

cov_list = np.split(train_set.groupby('Class').cov().values,[2])

pi_list = train_set.iloc[:,0].value_counts().values /
len(train_set)

print("Mean của từng class A trong train set là :
",mu_list[0][0])
print("Mean của từng class B trong train set là :
",mu_list[1][0])
```

```
Mean của từng class A trong train set là :  [1.02409221 0.8953582
4]
Mean của từng class B trong train set là : [-1.02802323 -0.117050
38]
```

- *Bước 6:* Xây dựng biệt thức (discriminant function).

Áp dụng công thức cho trường hợp $\sum_i$ bất kỳ:

$$g_i(\mathbf{x}) = -\frac{1}{2}\left[\mathbf{x}^T\boldsymbol{\Sigma}_i^{-1}\mathbf{x} - 2\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\mathbf{x} + \boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i\right] - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i)$$
$$= \mathbf{x}^T\mathbf{W}_i\mathbf{x} + \mathbf{w}_i^T\mathbf{x} + w_{i0}$$

trong đó

$$\mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1}$$

$$\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i$$

và

$$w_{i0} = -\frac{1}{2}\boldsymbol{\mu}_i^T\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i)$$

```
#Build discriminant functions

def DF(X,mu_list,cov_list,pi_list):
  scores_list = []
  classes = len(mu_list)
  for p in range(classes):
    Wi = (-1/2)*np.linalg.inv(cov_list[p])
    wi = np.linalg.inv(cov_list[p])@mu_list[p][0]
    wi0 = (-
1/2)*np.transpose(mu_list[p][0])@np.linalg.inv(cov_list[p])@mu_list
[p][0]
    + (-1/2)*np.log(np.linalg.norm(cov_list[p]))
    + np.log(pi_list[p])
    score = np.transpose(X)@Wi@X + np.transpose(wi)@X + wi0
    # print(np.transpose(X)@Wi@X)
    scores_list.append(score)
  # print(scores_list)
  return np.argmax(scores_list)
```

- *Bước 7:* Đánh giá:

Đánh giá trên tập dữ liệu test. Tính độ chính xác cho từng tập và xây dựng confusion matrix.

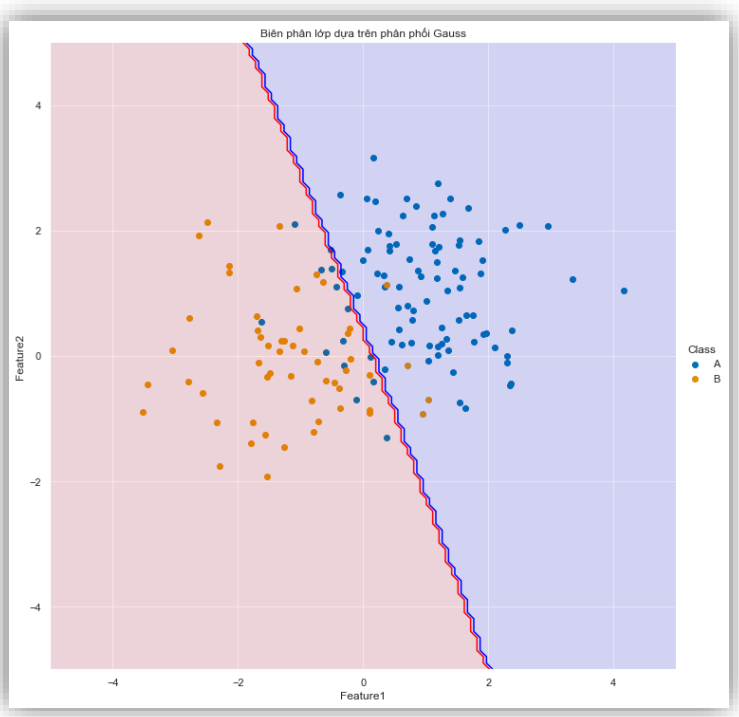
```
prediction = ["A" if DF(np.array([x,y]).reshape(-1,1),mu_list,
                                cov_list, pi_list)==0 else "B"
         for x, y in test_set[["Feature1","Feature2"]].values]
label = list(test_set['Class'].values)
print(pd.DataFrame(confusion_matrix(label, prediction),index=
           ['Class A', 'Class B'], columns=['Class A', 'Class B']))
```

```
         Class A  Class B
Class A       82       13
Class B        4       50
```

- *Bước 8:* Plot dữ liệu testing của 2 lớp và đường biên phân lớp trên cùng một hình.

```
#Plot with boundary contours
N = 100
X = np.linspace(-5, 5, N)
Y = np.linspace(-5, 5, N)
X, Y = np.meshgrid(X, Y)

#Configure plot
color_list = ['Blues','Reds']
g = sns.FacetGrid(test_set, hue="Class", height=10, palette =
```

```
                'colorblind', hue_order=["A","B"]).map(plt.scatter,
                "Feature1","Feature2",).add_legend()
my_ax = g.ax

#Computing the predicted class function for each value on the grid
zz = np.array( [DF( np.array([xx,yy]).reshape(-1,1),mu_list,
                                                cov_list, 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, 1, alpha = .1, colors = ('blue','red'))
my_ax.contour( X, Y, Z, 1, alpha = 1, colors = ('blue','red'))

# Addd axis and title
my_ax.set_xlabel('Feature1')
my_ax.set_ylabel('Feature2')
my_ax.set_title('Biên phân lớp dựa trên phân phối Gauss')
plt.show()
```



Trường hợp tập dữ liệu huấn luyện là 75%:

- *Bước 1:* Chia tập dữ liệu vào train và test:

```
#Split train and test
train_set, test_set = train_test_split(classAB, train_size=0.75)
print("Số lượng của tập train là :",len(train_set))
```

```
print("Số lượng của tập train là :",len(test_set))
```

```
Số lượng của tập train là : 371
Số lượng của tập train là : 124
```

- *Bước 2:* Tính mean tương ứng cho từng class.

```
#Estimating the parameters
mu_list = np.split(train_set.groupby('Class').mean().values,[1])
cov_list = np.split(train_set.groupby('Class').cov().values,[2])
pi_list = train_set.iloc[:,0].value_counts().values /
len(train_set)
print("Mean của từng class A trong train set là : ",mu_list[0][0])
print("Mean của từng class B trong train set là : ",mu_list[1][0])
```

```
Mean của từng class A trong train set là :  [0.96513234 0.91431234]
Mean của từng class B trong train set là :  [-1.09776943 -0.1086282
6]
```

- *Bước 3:* Đánh giá trên tập dữ liệu test. Tính độ chính xác cho từng tập và xây dựng confusion matrix.

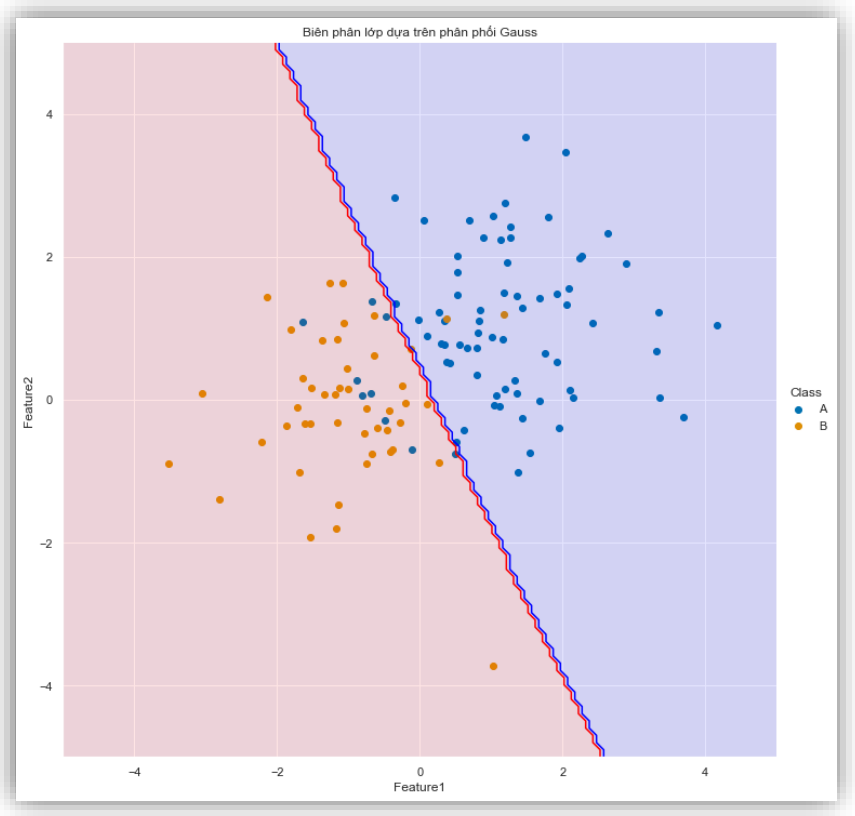
```
#Confusion matrix
prediction = ["A" if DF(np.array([x,y]).reshape(-1,1),mu_list,
cov_list, pi_list)==0 else "B"
                                        for x, y in
test_set[["Feature1","Feature2"]].values]
label = list(test_set['Class'].values)
print(pd.DataFrame(confusion_matrix(label, prediction), index=[
             'Class A', 'Class B'], columns=['Class A', 'Class
B']))
```

```
         Class A  Class B
Class A       68        9
Class B        3       44
```

- *Bước 4:* Plot dữ liệu testing của 2 lớp và đường biên phân lớp trên cùng một biểu đồ.

```
#Plot with boundary contours
N = 100
```

```
X = np.linspace(-5, 5, N)
Y = np.linspace(-5, 5, N)
X, Y = np.meshgrid(X, Y)

#Configure plot
color_list = ['Blues','Reds']
g = sns.FacetGrid(test_set, hue="Class", height=10, palette =
'colorblind', hue_order=["A","B"]).map(plt.scatter,

"Feature1","Feature2",).add_legend()
my_ax = g.ax

#Computing the predicted class function for each value on the grid
zz = np.array( [DF( np.array([xx,yy]).reshape(-1,1),mu_list,
                                          cov_list, 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, 1, alpha = .1, colors = ('blue','red'))
my_ax.contour( X, Y, Z, 1, alpha = 1, colors = ('blue','red'))

# Addd axis and title
my_ax.set_xlabel('Feature1')
my_ax.set_ylabel('Feature2')
my_ax.set_title('Biên phân lớp dựa trên phân phối Gauss')
plt.show()
```



Trường hợp tập dữ liệu huấn luyện là 80%:

- *Bước 1:* Chia tập dữ liệu thành 2 tập train và test:

```
#Split train and test
train_set, test_set = train_test_split(classAB, train_size=0.8)
print("Số lượng của tập train là :",len(train_set))
print("Số lượng của tập train là :",len(test_set))
```

```
Số lượng của tập train là : 396
Số lượng của tập train là : 99
```

- *Bước 2:* Tính mean tương ứng cho từng class.

```
#Estimating the parameters
mu_list = np.split(train_set.groupby('Class').mean().values,[1])
cov_list = np.split(train_set.groupby('Class').cov().values,[2])
pi_list = train_set.iloc[:,0].value_counts().values /
len(train_set)
print("Mean của từng class A trong train set là : ",mu_list[0][0])
print("Mean của từng class B trong train set là : ",mu_list[1][0])
```

```
Mean của từng class A trong train set là :  [1.00020499 0.9389721 ]
Mean của từng class B trong train set là :  [-1.10655257 -0.0981167
8]
```

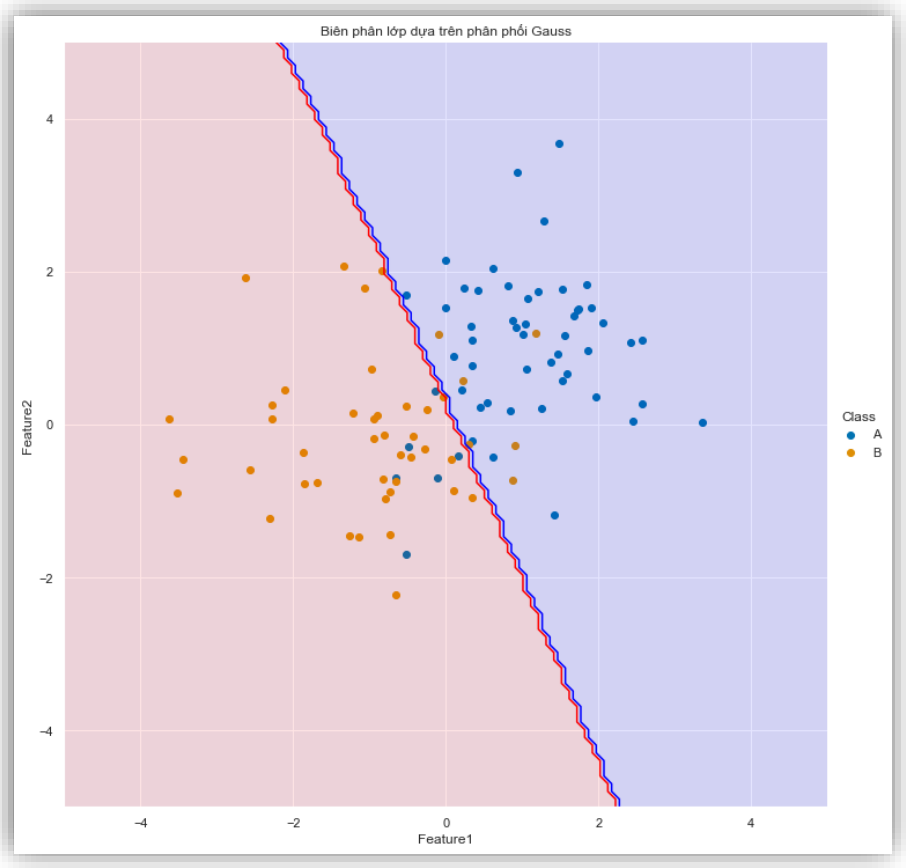
- *Bước 3:* Đánh giá trên tập dữ liệu test. Tính độ chính xác cho từng tập và xây dựng confusion matrix.

```
#Confusion matrix
prediction = ["A" if DF(np.array([x,y]).reshape(-1,1),mu_list,
                                cov_list, pi_list)==0 else "B"
              for x, y in
test_set[["Feature1","Feature2"]].values]
label = list(test_set['Class'].values)
print(pd.DataFrame(confusion_matrix(label, prediction), index=
       ['Class A', 'Class B'], columns=['Class A', 'Class B']))
```

```
         Class A  Class B
Class A       48        6
Class B        7       38
```

- *Bước 4:* Plot dữ liệu testing của 2 lớp và đường biên phân lớp trên cùng một hình.

```
#Plot with boundary contours
N = 100
X = np.linspace(-5, 5, N)
Y = np.linspace(-5, 5, N)
X, Y = np.meshgrid(X, Y)

#Configure plot
color_list = ['Blues','Reds']
g = sns.FacetGrid(test_set, hue="Class", height=10, palette =
'colorblind', hue_order=["A","B"]).map(plt.scatter,

"Feature1","Feature2",).add_legend()
my_ax = g.ax

#Computing the predicted class function for each value on the grid
zz = np.array( [DF( np.array([xx,yy]).reshape(-1,1),mu_list,
                                          cov_list, 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, 1, alpha = .1, colors = ('blue','red'))
my_ax.contour( X, Y, Z, 1, alpha = 1, colors = ('blue','red'))

# Addd axis and title
my_ax.set_xlabel('Feature1')
my_ax.set_ylabel('Feature2')
my_ax.set_title('Biên phân lớp dựa trên phân phối Gauss')
plt.show()
```



Bài 3: Xây dựng bộ classifier với 2 lớp, 2 đặc trưng. Giả sử tập dữ liệu có dạng phân bố Gauss. Tập dữ liệu huấn luyện là class1_train.mat và class2_train.mat. Tập dữ liệu đánh giá là class1_test.mat và class2_test.mat.

• ***Load data:***

- *Bước 1:* Import thư viện:

```
import numpy as np
from scipy.io import loadmat
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
import scipy.io
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

- *Bước 2:* Load 2 file tương ứng cho 2 class là: *class1_train.mat* và *class2_train.mat*. Sử dụng lệnh:

```
#Load data
columnsName = ['Feature1','Feature2']
class1_train =
pd.DataFrame(loadmat('class1_train.mat')['class1_train'],
                                          columns = columnsName)
class2_train =
pd.DataFrame(scipy.io.loadmat('class2_train.mat')['class2_train'],
                                          columns = columnsName)
class1_test =
pd.DataFrame(scipy.io.loadmat('class1_test.mat')['class1_test'],
                                          columns = columnsName)
class2_test =
pd.DataFrame(scipy.io.loadmat('class2_test.mat')['class2_test'],
                                          columns = columnsName)
```

```
class1_train.head()
```

	Feature1	Feature2
0	0.743738	2.258677
1	5.027590	5.702984
2	1.021112	2.899441
3	2.140428	3.794555
4	4.029366	3.811860

```
#Merge data
trainMerge = pd.concat([class1_train,class2_train],
                  keys=['1', '3']).reset_index().drop('level_1',
                   axis=1).rename(columns = {'level_0': 'Class'})
testMerge = pd.concat([class1_test,class2_test],
                  keys=['1','2']).reset_index().drop('level_1',
                  axis=1).rename(columns = {'level_0': 'Class'})
```

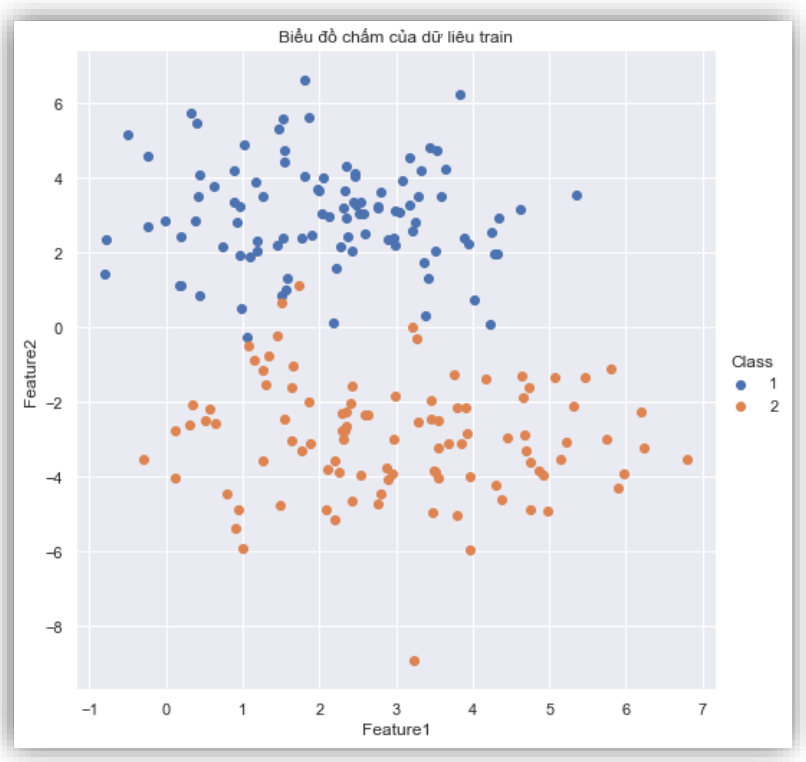
```
trainMerge.head()
```

	Class	Feature1	Feature2
0	1	0.743738	2.258677
1	1	5.027590	5.702984
2	1	1.021112	2.899441
3	1	2.140428	3.794555
4	1	4.029366	3.811860

- *Bước 3:* Plot dữ liệu.

```
#Plot data
sns.set()
sns.FacetGrid(trainMerge, hue="Class",
height=7).map(plt.scatter,"Feature1","Feature2",).add_legend()
plt.title('Biểu đồ chấm của dữ liêu train')
plt.show()
```

```
#Plot data
sns.set()
sns.FacetGrid(testMerge, hue="Class",
height=7).map(plt.scatter,"Feature1","Feature2",).add_legend()
plt.title('Biểu đồ chấm của dữ liệu train')
plt.show()
```



- Bước 4: Xây dựng classifier:

Tính mean, và covariance tương ứng cho từng class.

```
#Estimating the parameters
mu_list = np.split(trainMerge.groupby('Class').mean().values,[1])
# cov_list = np.split(np.array([1,0,0,1]*2).reshape(-1,2),[2])
cov_list = np.split(trainMerge.groupby('Class').cov().values,[2])
pi_list = trainMerge.iloc[:,0].value_counts().values /
len(trainMerge)
print("Mean của từng class A trong train set là :
",mu_list[0][0])
print("Mean của từng class B trong train set là :
",mu_list[1][0])
```

```
Mean của từng class A trong train set là : [1.97909027 2.97947776
]
Mean của từng class B trong train set là : [ 3.01056198 -2.871673
77]
```

- *Bước 5:* Xây dựng biệt thức (discriminant function).

```
#Build discriminant functions
def DF(X,mu_list,cov_list,pi_list):
  scores_list = []
  classes = len(mu_list)
  for p in range(classes):
    Wi = (-1/2)*np.linalg.inv(cov_list[p])
    wi = np.linalg.inv(cov_list[p])@mu_list[p][0]
    wi0 = (-1/2)*np.transpose(mu_list[p][0])
                  @np.linalg.inv(cov_list[p])@mu_list[p][0]
                  - (-1/2)*np.log(np.linalg.norm(cov_list[p]))
                  + np.log(pi_list[p])
    score = np.transpose(X)@Wi@X + np.transpose(wi)@X + wi0
    scores_list.append(score)
  return np.argmax(scores_list)
```

- *Bước 6:* Đánh giá trên tập dữ liệu test, *class1_test.mat* và *class2_test.mat*. Tính độ chính xác cho từng tập và xây dựng confusion matrix.

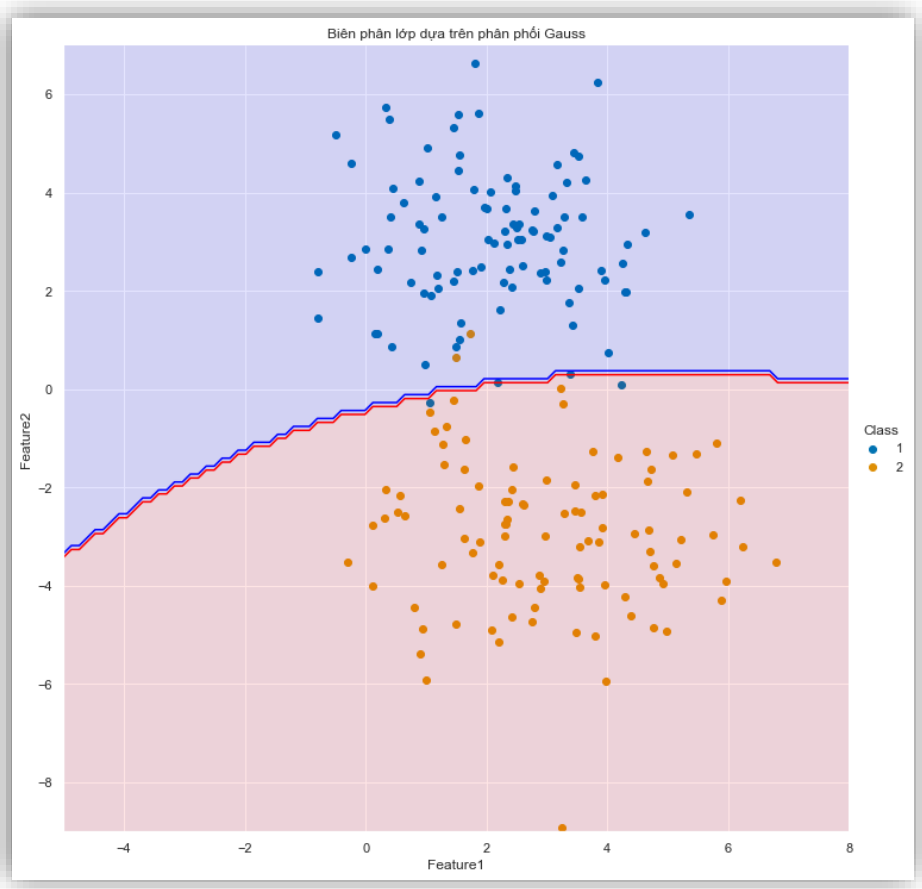
```
#Confusion matrix
prediction = ["1" if DF(np.array([x,y]).reshape(-1,1),mu_list,
cov_list, pi_list)==0 else "2"
           for x, y in testMerge[["Feature1","Feature2"]].values]
label = list(testMerge['Class'].values)
print(pd.DataFrame(confusion_matrix(label, prediction),
index=['Class 1', 'Class 2'], columns=['Class 1', 'Class 2']))
```

```
         Class 1  Class 2
Class 1       98        2
Class 2        2       98
```

- *Bước 7:* Plot dữ liệu testing của 2 lớp và đường biên phân lớp trên cùng một hình.

```
#Plot with boundary contours
N = 100
X = np.linspace(-5, 8, N)
Y = np.linspace(-9, 7, N)
X, Y = np.meshgrid(X, Y)

#Configure plot
color_list = ['Blues','Reds']
g = sns.FacetGrid(testMerge, hue="Class", height=10, palette =
              'colorblind', hue_order=["1","2"]).map(plt.scatter,
              "Feature1","Feature2",).add_legend()
my_ax = g.ax

#Computing the predicted class function for each value on the
grid
zz = np.array( [DF( np.array([xx,yy]).reshape(-1,1),mu_list,
cov_list, 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, 1, alpha = .1, colors = ('blue','red'))
my_ax.contour( X, Y, Z, 1, alpha = 1, colors = ('blue','red'))

# Addd axis and title
my_ax.set_xlabel('Feature1')
my_ax.set_ylabel('Feature2')
my_ax.set_title('Biên phân lớp dựa trên phân phối Gauss')
plt.show()
```



BÀI THỰC HÀNH SỐ 4

I. Mục đích:

Xây dựng bộ classifier dựa trên ước lượng Parzen Window và các tiếp cận khác.

III. Nội dung:

Bài 1: Xây dựng bộ classifier dựa trên Parzen window:

- Load 2 file *classA.txt* và *classB.txt*.
- Phân chia tập dữ liệu thành 2 tập con training (70%) và testing (30%).
- Huấn luyện bộ classifier sử dụng dữ liệu 2 chiều, 2 lớp từ tập training. Áp dụng Parzen window với hàm cửa sổ hypercube

$$\varphi(\frac{\mathbf{x}-\mathbf{x}_i}{h}) = \begin{cases} 1 & if\ |\mathbf{x}-\mathbf{x}_i| \le \frac{h}{2} \\ 0 & otherwise \end{cases}$$

  và h=1.
- Sử dụng tập testing để đánh giá hiệu quả của bộ classifier.
  Lặp lại với h=0.5 và 1.5.

Giải:

- *Bước 1:* Import thư viện cần sử dụng

```
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
import scipy.io
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

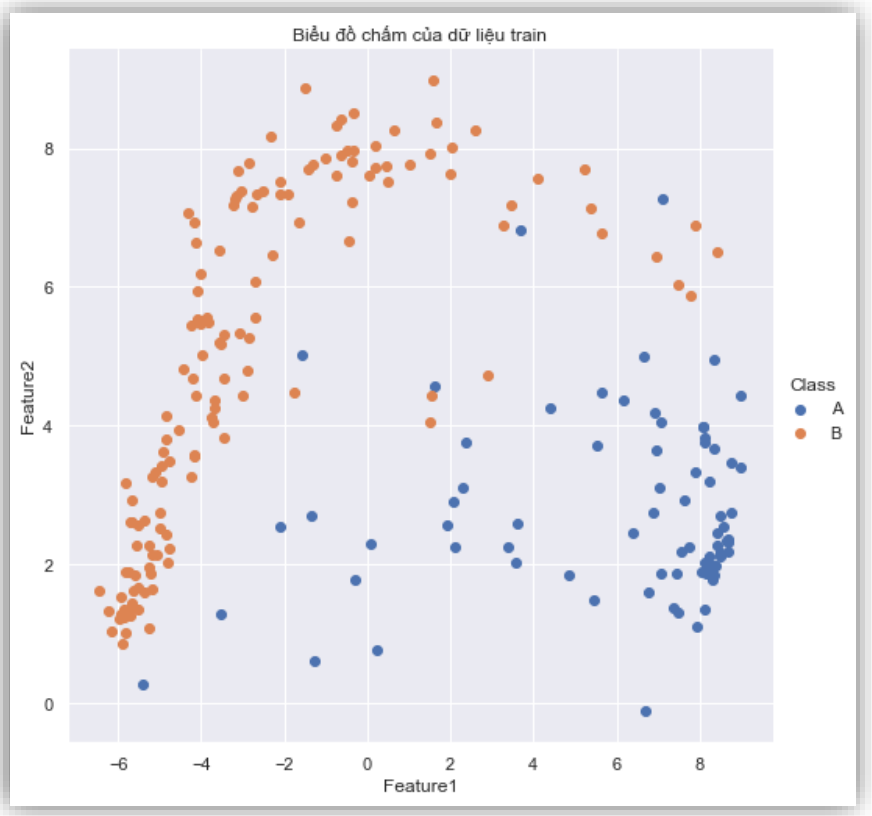
- *Bước 2:* Load 2 file *classA.txt* và *classB.txt*.

```
#Load data
columnsName = ['Feature1','Feature2']
f = open("classA.txt", "r")
classA = pd.DataFrame(np.array(f.read().split()).astype('float64')
                              .reshape(-1,2),columns = columnsName)
f = open("classB.txt", "r")
classB = pd.DataFrame(np.array(f.read().split()).astype('float64')
                              .reshape(-1,2),columns = columnsName)
```

```
#Merge data
classMerge = pd.concat([classA,classB], keys=['A',
'B']).reset_index().drop('level_1', axis=1).rename(columns =
{'level_0': 'Class'})
```

- *Bước 3*: Biểu diễn đồ thị bằng biểu đồ 2 chiều nhằm trực quan hóa các điểm dữ liệu.

```
#Plot data
sns.set()
sns.FacetGrid(classMerge, hue="Class",
height=7).map(plt.scatter,"Feature1","Feature2",).add_legend()
plt.title('Biểu đồ chấm của dữ liệu train')
plt.show()
```



- *Bước 4:* Phân chia tập dữ liệu thành 2 tập con training (70%) và testing (30%).

```
#Split train and test
train_set, test_set = train_test_split(classMerge, train_size=0.7)
print("Số lượng của tập train là :",len(train_set.index))
print("Số lượng của tập train là :",len(test_set.index))
```

```
Số lượng của tập train là : 151
Số lượng của tập train là : 65
```

- *Bước 5:* Huấn luyện bộ classifier sử dụng dữ liệu 2 chiều, 2 lớp từ tập training. Áp dụng Parzen window với hàm cửa sổ hypercube

$$\varphi(\frac{\mathbf{x}-\mathbf{x}_i}{h}) = \begin{cases} 1 & if\ |\mathbf{x}-\mathbf{x}_i| \le \frac{h}{2} \\ 0 & otherwise \end{cases}$$

và h=1.

```
#Build classifier functions
h = 1.5

def phi(x):
  if ((np.absolute(x)>1/2).any()):
    return 0
  else:
    return 1

def PW(X,data_train):
  scores_list = []
  for p in train_set.groupby('Class'):
    score = 0
    for x in p[1][['Feature1','Feature2']].to_numpy():
      score+=phi((X-x.reshape(-1,1))/h)
    scores_list.append(score)
  return np.argmax(scores_list)
```

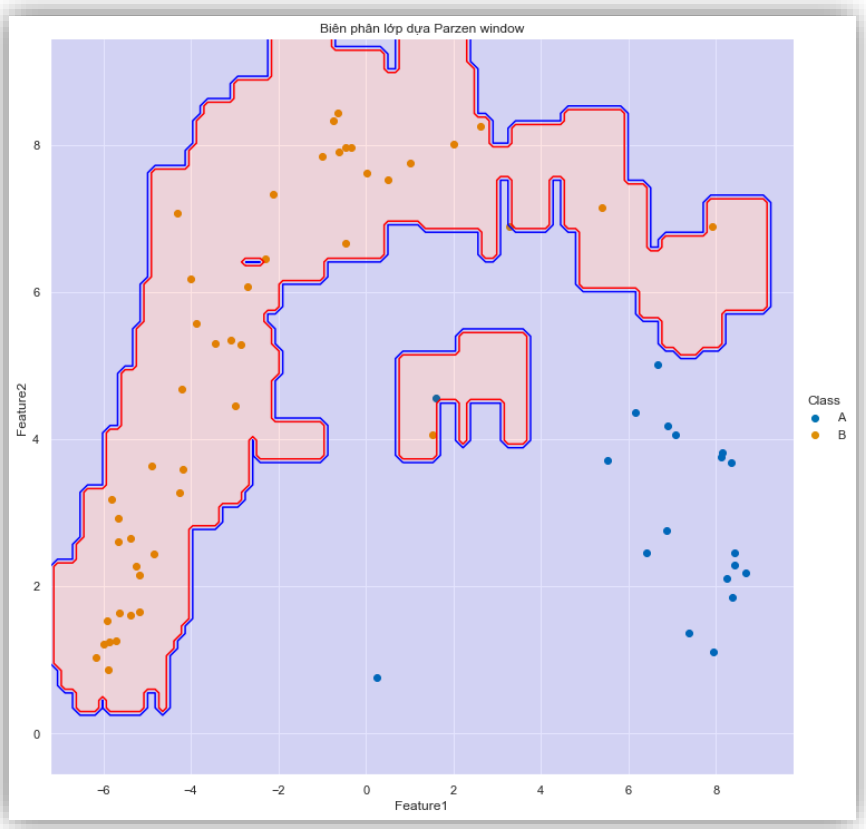
- *Bước 6:* Sử dụng tập testing để đánh giá hiệu quả của bộ classifier.

```
#Confusion matrix
prediction = ["A" if PW(np.array([x,y]).reshape(-1,1),train_set)==0
                  else "B"
              for x, y in test_set[["Feature1","Feature2"]].values]
label = list(test_set['Class'].values)
print(pd.DataFrame(confusion_matrix(label, prediction),
      index=['Class A', 'Class B'], columns=['Class A', 'Class
B']))
```

```
         Class A  Class B
Class A       18        1
Class B        1       45
```

- *Bước 7:* Vẽ biểu đồ thể hiện biên phân lớp của dữ liệu.

```
#Plot with boundary contours
N = 100
maxv = classMerge[["Feature1","Feature2"]].max().values
minv = classMerge[["Feature1","Feature2"]].min().values
len = maxv - minv
X = np.linspace(minv[0]-len[0]*0.05, maxv[0]+len[0]*0.05, N)
Y = np.linspace(minv[1]-len[1]*0.05, maxv[1]+len[1]*0.05, N)
X, Y = np.meshgrid(X, Y)

#Configure plot
color_list = ['Blues','Reds']
g = sns.FacetGrid(test_set, hue="Class", height=10,
      palette = 'colorblind', hue_order=["A","B"]).map(plt.scatter,
      "Feature1","Feature2",).add_legend()
my_ax = g.ax

#Computing the predicted class function for each value on the grid
zz = np.array( [PW(np.array([xx,yy]).reshape(-1,1),train_set)
                  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, 1, alpha = .1, colors = ('blue','red'))
my_ax.contour( X, Y, Z, 1, alpha = 1, colors = ('blue','red'))

# Addd axis and title
my_ax.set_xlabel('Feature1')
my_ax.set_ylabel('Feature2')
my_ax.set_title('Biên phân lớp dựa Parzen window')
plt.show()
```



Bài 2: Lặp lại bài 1 với hàm cửa sổ Gauss

$$\varphi(\frac{\mathbf{x}-\mathbf{x}_i}{h}) = \exp(-(\mathbf{x}-\mathbf{x}_i)^T(\mathbf{x}-\mathbf{x}_i)/(2h^2))$$

Giải:

- *Bước 1:* Import thư viện cần sử dụng:

```
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
import scipy.io
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
```

- *Bước 2:* Load tập dữ liệu từ file classA.txt và classB.txt

```
#Load data
```

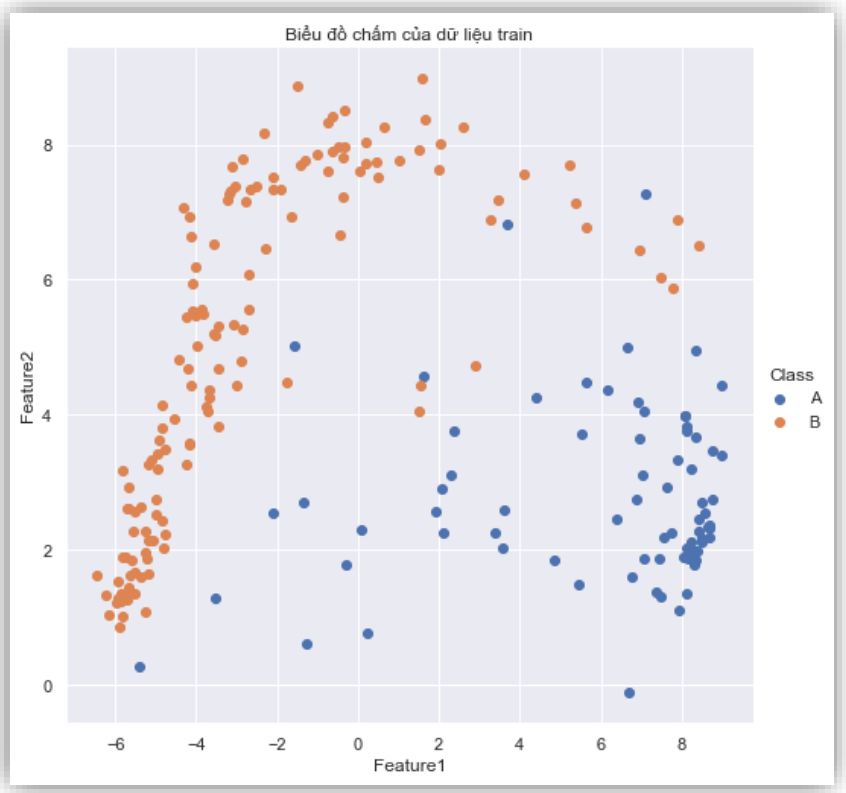
```
columnsName = ['Feature1','Feature2']
f = open("classA.txt", "r")
classA =
pd.DataFrame(np.array(f.read().split()).astype('float64').reshape(-
1,2),columns = columnsName)
f = open("classB.txt", "r")
classB =
pd.DataFrame(np.array(f.read().split()).astype('float64').reshape(-
1,2),columns = columnsName)
```

- *Bước 3:* Gộp 2 tập dữ liệu lại

```
#Merge data
classMerge = pd.concat([classA,classB], keys=['A',
'B']).reset_index().drop('level_1', axis=1).rename(columns =
{'level_0': 'Class'})
```

- *Bước 4:* Biểu diễn dữ liệu trên đồ thị

```
#Plot data
sns.set()
sns.FacetGrid(classMerge, hue="Class", height=7).map(plt.scatter,
                              "Feature1","Feature2",).add_legend()
plt.title('Biểu đồ chấm của dữ liệu train')
plt.show()
```



- *Bước 5:* Chia tập dữ liệu sau khi gộp thành 2 tập train và test.

```
#Split train and test
train_set, test_set = train_test_split(classMerge, train_size=0.7)
print("Số lượng của tập train là :",len(train_set.index))
print("Số lượng của tập train là :",len(test_set.index))
```

```
Số lượng của tập train là : 151
Số lượng của tập train là : 65
```

- *Bước 6:* Xây dựng window function dựa trên phân bố chuẩn với h=0.5:

```
#Build classifier functions
h = 0.5

def phi(x):
  return np.exp(-np.transpose(x)@x)/2
```

```
def PW(X,data_train):
  scores_list = []
  for p in train_set.groupby('Class'):
    score = 0
    for x in p[1][['Feature1','Feature2']].to_numpy():
      score+=phi((X-x.reshape(-1,1))/h)
    scores_list.append(score)
  return np.argmax(scores_list)
```

- *Bước 7:* Sử dụng tập testing để đánh giá hiệu quả của bộ classifier.

```
#Confusion matrix
prediction = ["A" if PW(np.array([x,y]).reshape(-1,1),train_set)==0
                  else "B"
            for x, y in test_set[["Feature1","Feature2"]].values]
label = list(test_set['Class'].values)
print(pd.DataFrame(confusion_matrix(label, prediction),
index=['Class A', 'Class B'], columns=['Class A', 'Class B']))
```

```
         Class A  Class B
Class A       22        3
Class B        3       37
```

- *Bước 8:* Biểu đồ hóa dữ liệu tập test:

```
#Plot with boundary contours
N = 100
maxv = classMerge[["Feature1","Feature2"]].max().values
minv = classMerge[["Feature1","Feature2"]].min().values
len = maxv - minv
X = np.linspace(minv[0]-len[0]*0.05, maxv[0]+len[0]*0.05, N)
Y = np.linspace(minv[1]-len[1]*0.05, maxv[1]+len[1]*0.05, N)
X, Y = np.meshgrid(X, Y)

#Configure plot
color_list = ['Blues','Reds']
g = sns.FacetGrid(test_set, hue="Class", height=10, palette =
                'colorblind', hue_order=["A","B"]).map(plt.scatter,
                "Feature1","Feature2",).add_legend()
my_ax = g.ax

#Computing the predicted class function for each value on the grid
zz = np.array( [PW(np.array([xx,yy]).reshape(-1,1),train_set)
                    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, 1, alpha = .1, colors = ('blue','red'))
my_ax.contour( X, Y, Z, 1, alpha = 1, colors = ('blue','red'))

# Addd axis and title
my_ax.set_xlabel('Feature1')
my_ax.set_ylabel('Feature2')
my_ax.set_title('Biên phân lớp dựa trên Parzen window với cửa sổ
Gauss')
plt.show()
```