ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HỒ CHÍ MINH

KHOA KHOA HỌC và KỸ THUẬT MÁY TÍNH

-------o0o-------

MÔN: NHẬN DẠNG MẪU và HỌC MÁY

(Pattern Recognition and Machine learning**)**

CHỦ ĐỀ:

HIỆN THỰC GIẢI THUẬT kNN CẢI TIẾN VỚI KỸ THUẬT NHÁNH VÀ CẬN

(The improved kNN algorithm with branch-and-bound technique**)**

GV Hướng dẫn: **PGS. TS Dương Tuấn Anh**

Học viên thực hiện: **Nguyễn Quốc Dũng**

MSHV: **1770470**

TP HCM, Tháng 12/2018

Mục Lục

MỞ ĐẦU

Trong lĩnh vực Data Mining (khai phá dữ liệu) và Machine learning (máy học), thuật toán tìm k điểm lân cận gần nhất là một trong mười thuật toán phổ biến được sử dụng. Thuật toán này được giới thiệu vào những năm đầu của thập niên 1950, nhưng đến những năm thập niên 60's mới thật sự được áp dụng vì khả năng tính toán và hiệu suất làm việc của máy tính vào thời điểm đó chưa được mạnh, thuật toán không đạt hiệu quả như mong muốn với tập dữ liệu lớn và chi phí tính toán quá nhiều.

Ngày nay, với sự phát triển của phần cứng máy tính, thì hiệu suất tính toán đã được cải thiện rất lớn, thuật toán này với độ phức tạp thấp, dễ hiệu chỉnh vì vậy được nhiều người chọn lựa sử dụng.

KNN có thể yêu cầu rất nhiều bộ nhớ hoặc không gian để lưu trữ tất cả dữ liệu, nhưng chỉ thực hiện tính toán (hoặc học) khi một dự báo là cần thiết, còn được biết đến là một trong những thuật toán trong lazy learners. Thuật toán kNN được sử dụng hiệu quả trong việc phân lớp, và gom cụm dữ liệu, khi ta không thực hiện nhiều tính toán trong quá trình thể hiện mẫu huấn luyện, và thực hiện việc tính toán nhiều hơn trong quá trình phân lớp mẫu thử.

Thuật toán KNN rất đơn giản và rất hiệu quả, mô hình đại diện cho KNN là toàn bộ dữ liệu tập huấn luyện. Dự báo sẽ được thực hiện cho một điểm dữ liệu mới bằng cách tìm kiếm thông qua toàn bộ tập huấn luyện cho hầu hết các ví dụ K giống nhau (hàng xóm lân cận) và tóm tắt biến đầu ra cho các ví dụ K.

Để xác định sự giống nhau giữa các trường hợp dữ liệu, kỹ thuật đơn giản nhất nếu các thuộc tính có cùng kích cỡ, là sử dụng khoảng cách Euclide, một con số mà ta có thể tính toán trực tiếp dựa trên sự khác biệt giữa mỗi biến đầu vào.

Qua yêu cầu bài tập lớn của môn học: " Hiện thực hóa thuật toán ứng dụng k lân cận gần nhất cải tiến vào việc phân lớp mẫu thử, có kết hợp kĩ thuật nhánh và cận", kết hợp tham khảo bài báo khoa học tựa đề "A branch and bound algorithm for computing k-Nearest Neighbors" của 02 tác giả Keinosuke Fukunaga và Patrenahalli M.Narendra, các nội dung sau đây sẽ cần phải tìm hiểu:

- Giải thuật một lân cận gần nhất
- Giải thuật k lân cận gần nhất có cải tiến
- Giải thuật k lân cận gần nhất cải tiến có sử dụng kỹ thuật nhánh – cận.

Học viên thực hiện

Nguyễn Quốc Dũng (1770470)

1. GIẢI THUẬT K LÂN CẬN GẦN NHẤT

1.1. LÂN CẬN GẦN NHẤT

Một trong những cách hỗ trợ ra quyết định đơn giản nhất trong việc phân lớp, đó là sử dụng giải thuật tìm lân cận gần nhất. Việc phân loại này dựa trên cơ sở so sánh mức độ tương tự của mẫu thử đối với những mẫu trong tập huấn luyện. Mức độ tương tự giữa hai mẫu được tính bằng sự đo đạc về khoảng cách, ở đây là sử dụng khoảng cách Euclide.

Ví dụ, để tính khoảng cách Euclide 2 mẫu X_1, X_2 , n thuộc tính. Ta tính dựa vào công thức sau:

$$dist(\boldsymbol{X_1}, \boldsymbol{X_2}) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}.$$

Trong đó, n: số thuộc tính (đặc trưng) của X_1, X_2.

Như vậy, để tìm lân cận gần nhất của mẫu thử, ta tính khoảng cách Euclide tới một vài mẫu hoặc tất cả các mẫu trong tập huấn luyện. Lân cận gần nhất của mẫu thử chính là mẫu có khoảng cách ngắn nhất với mẫu thử nằm trong tập huấn luyện. Lớp của mẫu thử phụ thuộc vào lớp của mẫu lân cận gần nhất.

Ví dụ: Cho tập huấn luyện, gồm 25 mẫu như sau

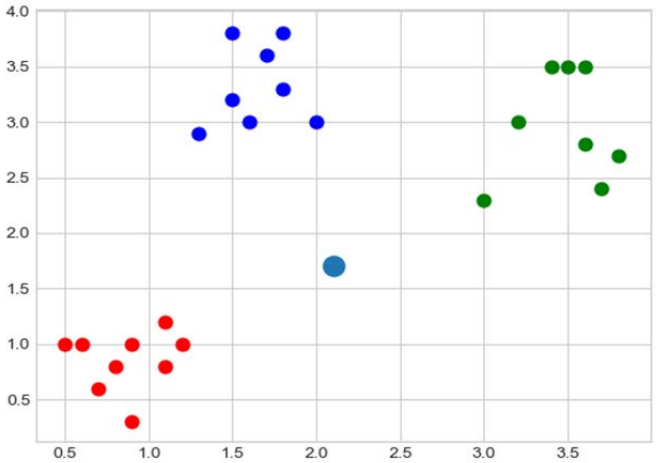
X1 =[0.9, 1.0, 1];	X6 =[0.7, 0.6, 1];	X10 =[1.5, 3.2, 2];	X15 =[1.8, 3.3, 2];
X2 =[1.1, 1.2, 1];	X7 =[0.5, 1.0, 1];	X11 =[1.6, 3.0, 2];	X16 =[1.8, 3.8, 2];
X3 =[0.9, 0.3, 1];	X8 =[1.2, 1.0, 1];	X12 =[1.5, 3.8, 2];	X17 =[2.0, 3.0, 2]
X4 =[0.6, 1.0, 1];	X9 =[1.1, 0.8, 1];	X13 =[1.7, 3.6, 2];	
X5 =[0.8, 0.8, 1];		X14 =[1.3, 2.9, 2];	

X18 =[3.2, 3.0, 3];	X23 =[3.8, 2.7, 3]
X19 =[3.7, 2.4, 3];	X24 =[3.4, 3.5, 3]
X20 =[3.5, 3.5, 3];	X25 =[3.6, 3.5, 3]
X21 =[3.6, 2.8, 3];	
X22 =[3.0, 2.3, 3] ;	

Trong đó thuộc tính thứ 1 và thứ 2 là tọa độ x,y của tập các mẫu và thuộc tính thứ 3 là phân lớp của các mẫu đó, sử dụng thuật toán một lân cận gần nhất để phân lớp mẫu thử P (2.1, 1.7).

Ta có phân lớp của các mẫu như hình 1. sau đây :

Đỏ : lớp 1
Xanh dương : lớp 2
Xanh lục : lớp 3



Hình 1: Sự phân bố dữ liệu các lớp

Ta sử dụng công thức tính khoảng cách Euclide để tính khoảng cách của mẫu thử P đến tất cả các mẫu trong tập huấn luyện, kết quả như sau:

d(X1 P) = 1.389	d(X6 P) = 1.780	d(X11 P) = 1.392	d(X16 P) = 2.121
d(X2 P) = 1.118	d(X7 P) = 1.746	d(X12 P) = 2.184	d(X17 P) = 1.303
d(X3 P) = 1.843	d(X8 P) = 1.140	d(X13 P) = 1.941	d(X18 P) = 1.702
d(X4 P) = 1.655	d(X9 P) = 1.345	d(X14 P) = 1.442	d(X19 P) = 1.746
d(X5 P) = 1.581	d(X10 P) = 1.615	d(X15 P) = 1.627	d(X20 P) = 2.280

d(X21 P)= 1.860
d(X22 P)= 1.081
d(X23 P)= 1.972
d(X24 P)= 2.220
d(X25 P)= 2.343

Dựa vào kết quả các khoảng cách đã được tính ở trên thì lân cận gần nhất của P chính là X22 thuộc nhóm 3 (nhóm màu xanh lục). Suy ra **mẫu thử P sẽ thuộc nhóm 3**.

Chúng ta cũng có thể sử dụng các phép chuẩn hóa dữ liệu để chuẩn hóa các giá trị thuộc tính của các mẫu trước khi chúng ta tính khoảng cách Euclide. Việc chuẩn hóa này được thực hiện nhằm mục đích tránh cho các giá trị khởi tạo thuộc tính nằm trong những khoảng quá lớn gây khó khăn cho quá trình tính toán.

Chúng ta có thể sử dụng chuẩn hóa Min-Max để chuyển đổi từ một giá trị v của thuộc tính A đến v' trong khoảng [new_min_A, new_max_A] bằng công thức tính:

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A$$

Hoặc cũng có thể sử dụng chuẩn hóa z-score: giá trị của thuộc tính A được chuẩn hóa dựa trên giá trị trung bình mean của A và độ lệch chuẩn của A

Giá trị của A là v, chuẩn hóa v' của A được tính theo công thức:

$$v' = (v - Ä)/\sigma_A$$

Với Ä là mean của A

σ_A: độ lệch chuẩn.

1.2.GIẢI THUẬT K LÂN CẬN GẦN NHẤT

Giải thuật k lân cận gần nhất được xây dựng dựa trên cơ sở của tìm điểm lân cận gần nhất. Tuy nhiên, thay vì chỉ tìm 1 điểm là lân cận gần nhất, thì ở đây ta tìm k điểm lân cận gần nhất đối với mẫu thử. Do đó, khi phân lớp cho mẫu thử, lớp của mẫu thử sẽ phụ thuộc vào lớp mà các dữ liệu mẫu chiếm đa số. Dựa trên cơ sở này sẽ làm giảm mức độ ảnh hưởng của dữ liệu xấu gây ra sự bị nhiễu trong việc phân lớp.

Tổng quát hơn, khi k=1 thì ta sẽ trở về bài toán tìm điểm lân cận gần nhất của mẫu thử, và phân lớp mẫu thử sẽ dựa vào lớp của điểm lân cận gần nhất.

Tuy nhiên, làm sao để ta có thể tính toán khoảng cách giữa 2 điểm với những thuộc tính là rời rạc, không phải dạng số, ví dụ như là màu sắc? Phương pháp đơn giản đó là so sánh từng giá trị tương ứng của thuộc tính từ mẫu X, Y.

Nếu giá trị chúng khác nhau, thì khoảng cách là 1 ngược lại là 0. Công thức tính như sau:

$$d(X, Y)= \sum_{k=0}^{n} \sigma(x_k, y_k)$$

với: $$\sigma(x_k, y_k) = \begin{cases} 0 \ nếu\ x_k = y_k \\ 1\ nếu\ x_k \neq y_k \end{cases}$$

Nhưng làm cách nào để xác định giá trị tốt nhất của k? Câu hỏi này có thể được xác định bằng thực nghiệm, bắt đầu với giá trị k=1, chúng ta kiểm tra tập mẫu để ước lượng hệ số lỗi của phân lớp, quá trình này được lặp đi lặp lại mỗi khi tăng giá trị từ từ lên cho k. Giá trị k tốt nhất được chọn là giá trị k làm cho hệ số lỗi nhỏ nhất. Có thể nói số phần tử trong tập huấn luyện càng lớn thì giá trị k càng lớn.

Ta trở lại ví dụ trên với k = 3, nghĩa là chúng ta sẽ lấy ra 3 điểm lân cận gần nhất với mẫu thử đã cho P. Dựa trên kết quả tính khoảng cách Euclide giữa P và 25 điểm trong mẫu huấn luyện, ta nhận thấy 3 điểm gần nhất với P là:

X2(1.1, 1.2) thuộc lớp 1; X8(1.2, 1.0) thuộc lớp 1; X22(3.0, 2.3) thuộc lớp 3, trong đó có hai điểm thuộc lớp 1 và một điểm thuộc lớp 3, vì vậy ta có thể phân lớp cho **mẫu thử P thuộc lớp 1** do là lớp chiếm đa số trong láng giềng lân cận.

Phân lớp bằng giải thuật k lân cận gần nhất dựa trên sự so sánh về khoảng cách không bị ảnh hưởng bởi trọng số các thuộc tính (trọng số cho mỗi thuộc tính được gán những giá trị bằng nhau). Do đó, thuật toán không thể phân lớp chính xác đối với những thuộc tính nhiễu hoặc không thích hợp. Vì vậy, thuật toán giải thuật lân cận cải tiến sẽ giải quyết vấn đề này, khi thuật toán có đề cập đến tính toán trọng số từng thuộc tính trong quá trình phân lớp dữ liệu thử.

2. GIẢI THUẬT K LÂN CẬN GẦN NHẤT CẢI TIẾN

Giải thuật k lân cận gần nhất cải tiến là một giải thuật được đưa ra để cải thiện hiệu suất của giải thuật k lân cận gần nhất khi nó có sự linh hoạt của những điểm lân cận gần nhất trong dữ liệu huấn luyện. Sự linh hoạt này được thể hiện qua trọng số của từng mẫu trong tập k mẫu lân cận với mẫu thử đang xét.

Dựa vào ý tưởng của giải thuật k lân cận gần nhất, sự phân lớp của mẫu thử dựa vào lớp của những điểm lân cận của nó. Tuy nhiên, ở giải thuật cải tiến này, có thêm tính toán mức độ ảnh hưởng của từng điểm lân cận (tính toán trọng số) lên mẫu thử. Mẫu thử sẽ được phân lớp dựa vào sự so sánh giữa tổng trọng số của các mẫu có chung nhãn lớp trong k điểm lân cận gần nhất.

Cách tính trọng số dựa vào khoảng cách :

Ý tưởng của giải thuật này chính là việc tính toán khoảng cách giữa k điểm lân cận với mẫu thử, sau đó sắp xếp giá trị khoảng cách này theo giá trị giảm dần.

$d_1 < d_2 < \ldots < d_j < \ldots < d_k$ trong đó k >= 1.

Như vậy để tính mức độ ảnh hưởng w_j của k điểm lân cận, ta có thể tính dựa trên công thức sau:

$$w_j = (d_k - d_j) / (d_k - d_1)$$

trong trường hợp d_k khác d_1 (khoảng cách từ k điểm lân cận đến mẫu thử là không bằng nhau)

Hoặc wj = 1, trong trường hợp $d_k = d_1$ (khoảng cách từ k điểm lân cận đến mẫu thử là bằng nhau).

Như vậy, phân lớp của mẫu thử dựa vào tổng trọng số của các điểm lân cận mẫu thử mà chung lớp với nhau. Lớp của mẫu thử là lớp có tổng trọng số là lớn nhất.

Xét lại ví dụ trên, với P(2.1, 1.7) giả sử k=5 thì khoảng cách từ P đến 5 điểm gần nhất lần lượt theo thứ tự tăng dần là:

*d(X22 P)= **1.081**, d(X2 P) = **1.118**, d(X8 P) = **1.140**, d(X17 P) = **1.303**, d(X9 P) = **1.345***

Ta có được trọng số tương ứng với 5 điểm trên như sau:

w_{22} = (1.345 – 1.081)/(1.345-1.081) = 1.0

w_2 = (1.345 – 1.118)/(1.345-1.081) = 0.862

w_8 = (1.345 – 1.140)/(1.345-1.081) = 0.778

w_{17} = (1.345 – 1.303)/(1.345-1.081) = 0.157

w_9 = (1.345 – 1.345)/(1.345-1.081) = 0

Từ đó ta tính các tổng trọng số cho các lớp như sau:

Tổng trọng số cho lớp 1: $w_2 + w_8 + w_9 = 0.862 + 0.778 + 0 = 1.64$

Tổng trọng số cho lớp 2: $w_{17} = 0.157$

Tổng trọng số cho lớp 3: $w_{22} = 1.0$

Với kết quả tính toán trên, điểm **P sẽ thuộc lớp 1** do có tổng trọng số lớn nhất.

3. GIẢI THUẬT K LÂN CẬN GẦN NHẤT CẢI TIẾN CÓ ỨNG DỤNG NHÁNH CẬN.

Theo các tác giả của bài báo [1], giải thuật là sự kết hợp giữa giải thuật tìm k điểm lân cận gần nhất cải tiến có sử dụng kĩ thuật nhánh và cận, để tránh cho việc tìm kiếm vét cạn tất cả các mẫu trong tất cả các lớp xuất hiện trong tập huấn luyện.

Giả sử ta có tập mẫu N: $\{X_1,X_2,\ldots,X_n\}$, n là số chiều và với mẫu thử X thuộc tập mẫu N.

Ý tưởng chính của thuật toán chính được chia làm 2 giai đoạn:

- Giai đoạn 1: Chia tập N thành các tập con , hình thành cấu trúc cây tìm kiếm.
- Giai đoạn 2: Tìm kiếm trên cây bằng giải thuật nhánh và cận kết hợp kNN

Ở giai đoạn 1, chúng ta dùng bất kỳ kỹ thuật gom cụm nào để phân rã tập mẫu, tập mẫu sẽ được phân rã vào ℓ tập con, mỗi tập con lại tiếp tục phân rã thành ℓ tập con khác, cứ thế tiếp tục. Trong bài báo, các tác giả dùng kỹ thuật k-Mean để gom cụm với $\ell = 3$ trên tập dữ liệu thử nghiệm và đã được chia thành 4 mức level, kết quả được minh họa qua cấu trúc cây như hình 2.

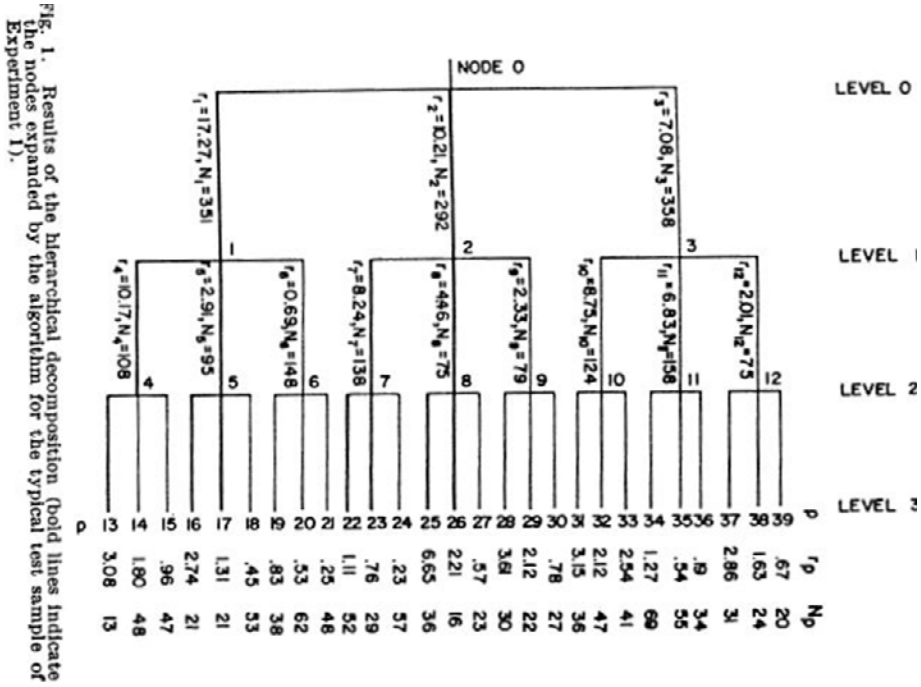


Fig. 1. Results of the hierarchical decomposition (bold lines indicate the nodes expanded by the algorithm for the typical test sample of Experiment 1).

Hình 2: Kết quả gom cụm tập dữ liệu với l =3

Với mỗi node p trên cây thì dại diện cho một nhóm các mẫu, và có những tham số đặc trưng được định nghĩa như sau;

S_p : tập các mẫu tương ứng với node p.
N_p : số lượng mẫu tương ứng với node p.
M_p : mẫu trung bình của S_p.
$r_p = \max_{Xi \in Sp} d(X_i, M_p)$ là khoảng cách lớn nhất từ M_p đến mẫu X thứ i ($X_i \in S_p$).

Ở giai đoạn 2, sau khi tập dữ liệu đã được gom cụm , và các đại lương S_p, N_p, M_p, r_p đã được định trị, mỗi node p sẽ được test có phải là lân cận tới X hay không bằng cách áp dụng các luật sau:

Luật số 1: Không có mẫu X thứ i ($X_i \in S_p$) có thể là lân cận với X nếu thỏa mãn công thức sau:

$B + r_p < d(X, M_p)$ với B là lân cân hiện hành của X theo tập mẫu, ban đầu gán $B = \infty$.

Với các node p tại level 3 (hình 2.), nếu không thỏa luật số 1 thì phải tính toán khoảng cách từ X đến từng các mẫu trong S_p, tuy nhiên luật số 2 sau sẽ giúp tránh việc tính toán nhiều khoảng cách không cần thiết:

Luật số 2: X_i không thể là lân cận của X, nếu

$B + d(X_i, M_p) < d(X, M_p)$ với $X_i \in S_p$

Các bước thực hiện giải thuật tìm kiếm trên cây được các tác giả đưa ra như sau:

Bước 0: Ban đầu , gán $B = \infty$, CURRENT LEVEL L= 1, CURRENT NODE = 0

Bước 1 (Mở rộng CURRENT NODE):
Đưa toàn bộ các nodes con của CURRENT NODE vào ACTIVE LIST của CURRENT LEVEL. Tính toán và lưu lại khoảng cách $d(X, M_p)$ của những nodes này.

Bước 2 (Test luật số 1):
Nếu với mỗi node p trong danh sách ACTIVE LIST của CURRENT LEVEL mà thỏa điều kiện bất đẳng thức $d(X, M_p) > B + r_p$ thì gỡ bỏ node p khỏi danh sách ACTIVE LIST của CURRENT LEVEL.

Bước 3 (Backtracking):
Nếu như không có nodes nào bị gỡ bỏ khỏi danh sách ACTIVE LIST của CURRENT LEVEL, thì quay lại level trước đó, gán L = L – 1, nếu L = 0 thì kết thúc giải thuật, nếu $L \neq 0$ thì quay lại bước 2, nếu vẫn còn node ACTIVE LIST của CURRENT LEVEL thì tiếp tục qua bước số 4.

Bước 4 (Chọn node gần nhất để mở rộng):
Chọn node p gần nhất (khoảng cách $d(X_i, M_p)$ là nhỏ nhất) trong số những node của danh sách ACTIVE LIST tại CURRENT LEVEL và gán CURRENT NODE cho node này , gỡ bỏ p khỏi danh sách ACTIVE LIST tại CURRENT LEVEL. Nếu CURRENT LEVEL là final level (mức kết thúc) thì qua bước 5, ngược lại gán L = L + 1 và quay về bước 1.

Bước 5 (Test luật số 2):
Với mỗi X_i của CURRENT NODE p, thực hiện tính toán sau:
Nếu $B + d(X_i, M_p) < d(X, M_p)$ thì X_i không thể là lân cận tới X, do đó không cần tính $d(X, X_i)$
Ngược lại , tính $d(X, X_i)$, nếu $d(X, X_i) < B$, gán CURRENT NN = I và $B = d(X, X_i)$.
Sau khi tất cả các X_i của CURRENT NODE quay lại bước 2,

Lưu ý ta có thể cải thiện thêm giải thuật tại bước 1 khi gán $B = \min[B, d(X, M_p) + r_p]$ khi khoảng cách $d(X, M_p)$ được tính toán.

Hình 3. dưới đây là lưu đồ (flow chart) của giải thuật:

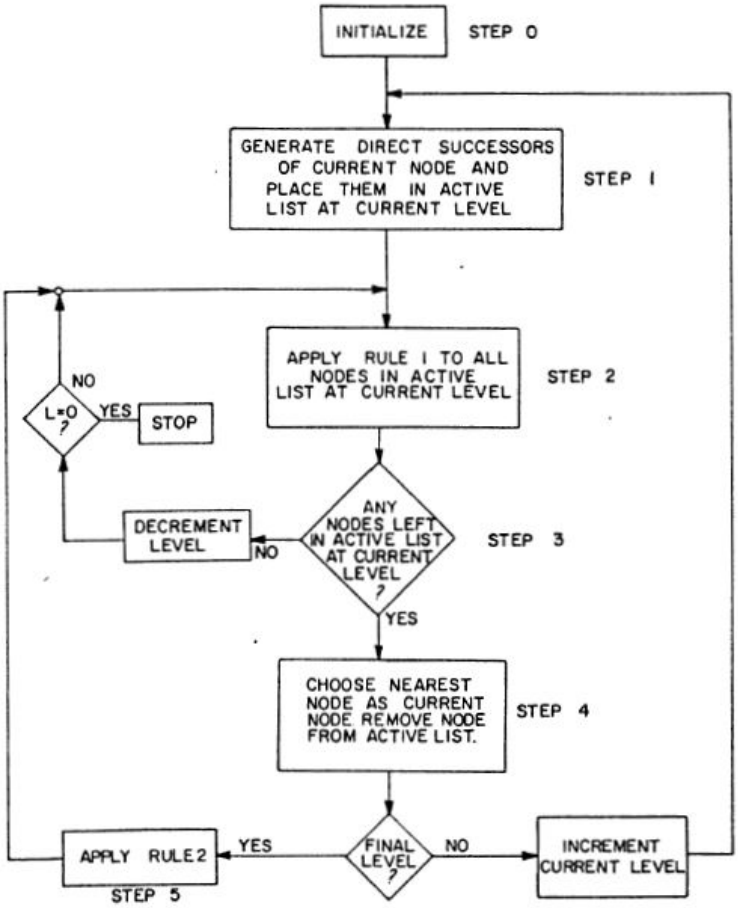


Fig. 3. Flowchart for the search algorithm.

Hình 3:Lưu đồ của giải thuật tìm kiếm

Mở rộng tính toán qua k-NN:

Việc mở rộng tính toán lân cận gần nhất trở nên đơn giản hơn, với B là khoảng cách tới lân cận thứ k, ở bước 5 khi khoản cách được tính toán, nó được so sánh với khoảng cách từ X đến lân cận hiện hành, và bảng lân cận được cập nhật với lân cận thứ k+1 của X.

4. KẾT QUẢ HIỆN THỰC

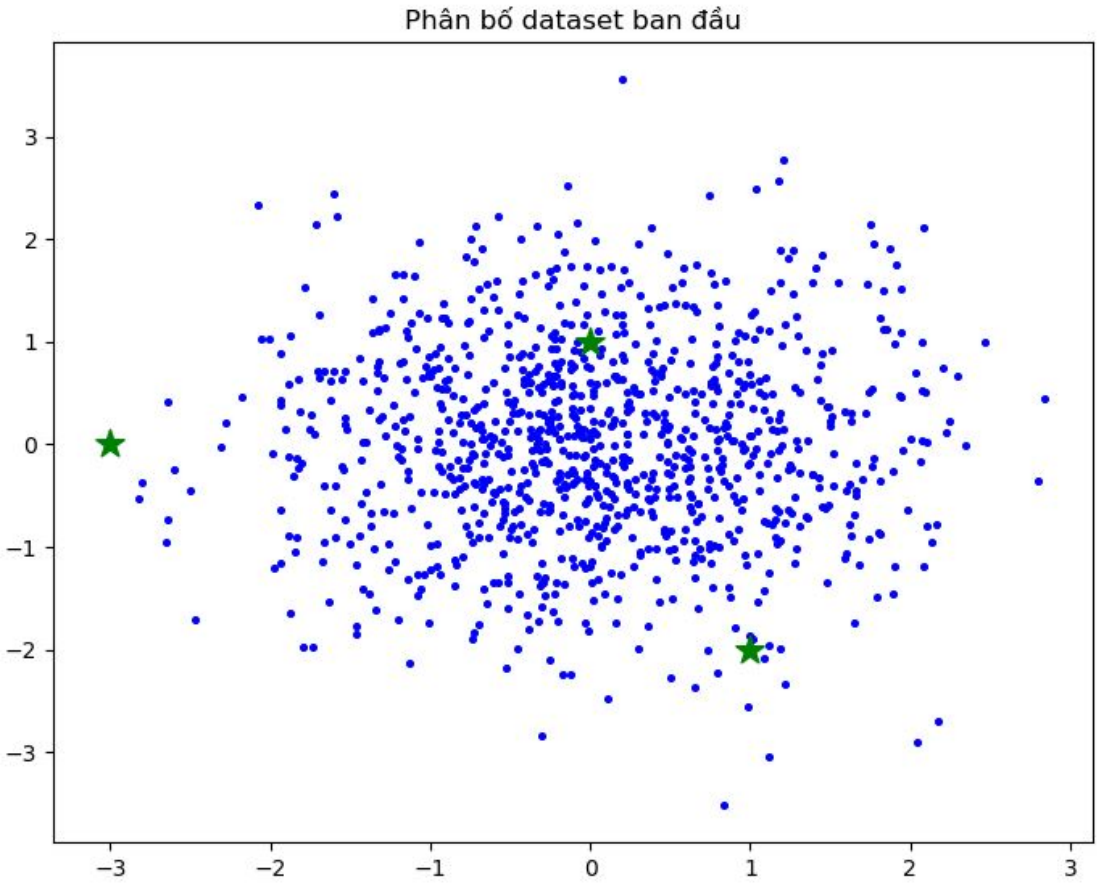
Theo giải thuật trình bày của bài báo [1] nêu trên, em đã cố gắng hiện thực hóa bằng ngôn ngữ lập trình Python và cũng tiến hành qua hai giai đoạn.

4.1 Giai đoạn 1 :

Với thư viện hỗ trợ của Python, Gaussian Distribution Dataset được tạo qua đoạn code sau đây:

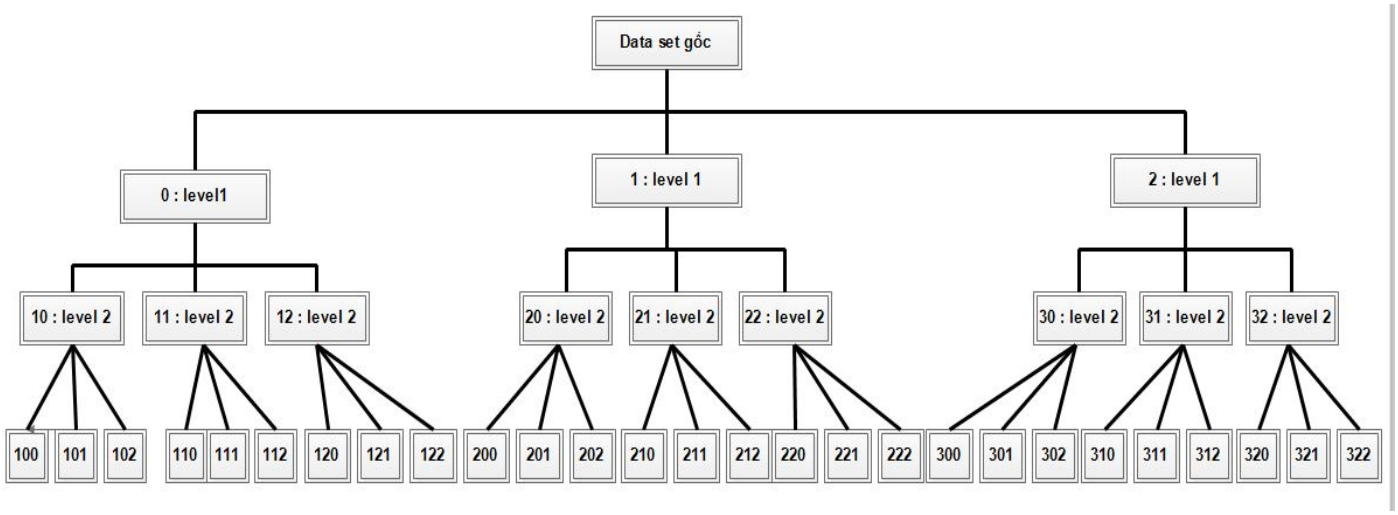
```
mean = [0, 0]
cov =  [[1, 0], [0, 1]]
x, y = np.random.multivariate_normal(mean, cov, 1000).T
X = np.array(list(zip(x, y)))
```

Giản đồ phân bố của dataset được tạo như hình 4 với 3 centroid ngẫu nhiên:



Hình 4:Giản đồ phân bố của tập dữ liệu mẫu

Sau đó dùng giải thuật k-Mean với k=3, lặp 3 lần trên dataset để phân cụm để có kết quả cây phân cụm như hình 5 dưới đây:



Hình 5: Cây phân cấp của dataset sau khi gom cụm

Python code:

```
# Phân cụm đầu tiên cho dataset
level1 = KMeans(num_clusters, random_state=0)   # gọi giải thuật kMean
rs_lv1 = level1.fit(X)                          # kết quả trả ra của kMean
lbl = rs_lv1.labels_                            # lấy nhãn từng điểm
ctr = rs_lv1.cluster_centers_                   # centroid của cụm
```

4.1.1 Kết quả của việc phân cụm đầu tiên bằng giải thuật kMean:

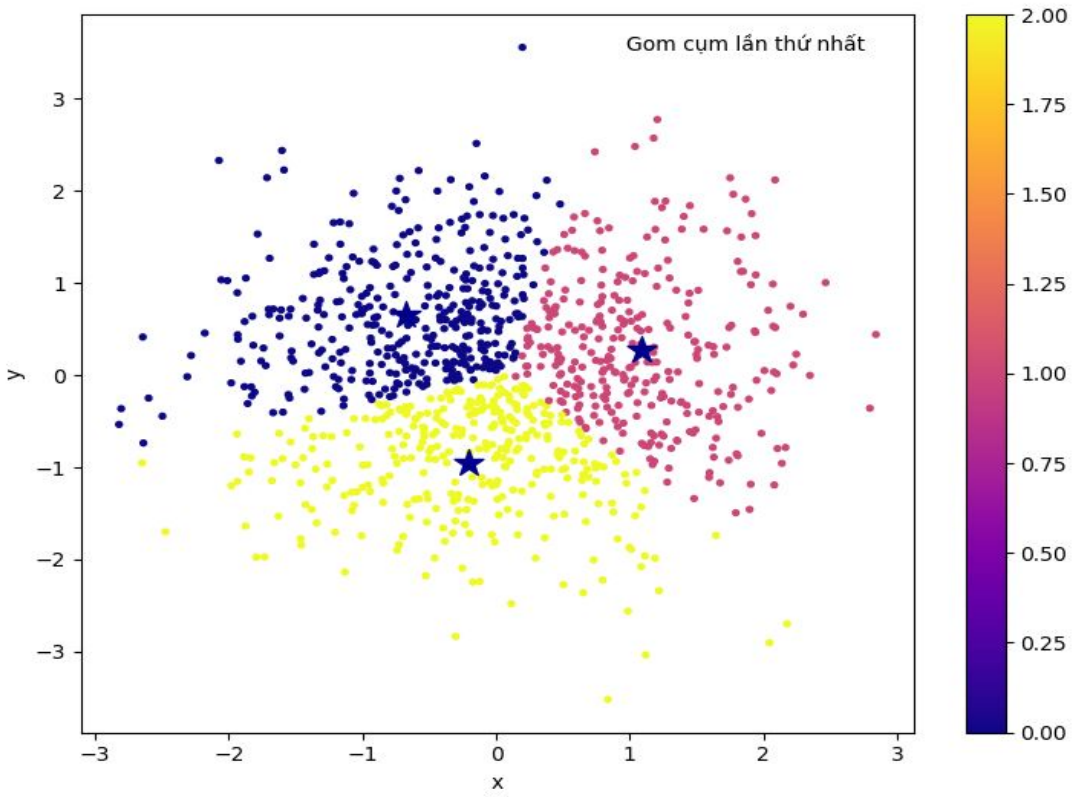
Thông tin kết quả được trả về như sau:

```
Phân cụm lần thứ nhất:
Cụm 0, số hàng: 364,  số cột: 3
Cụm 1, số hàng: 311,  số cột: 3
Cụm 2, số hàng: 325,  số cột: 3
```

Trong đó:

- Số cụm được gom là 3 cụm với nhãn cụm lần lượt là 0,1,2.
- Số cột gồm x , y và nhãn (label), số hàng là số các phần tử của cụm tương ứng.

Hình 6 dưới đây là giản đồ phân cụm bộ dữ liệu lần thứ nhất :



Hình 6:Giản đồ gom cụm dataset lần thứ nhất với k=3

4.1.2 Kết quả của việc phân cụm lần lặp thứ 2:

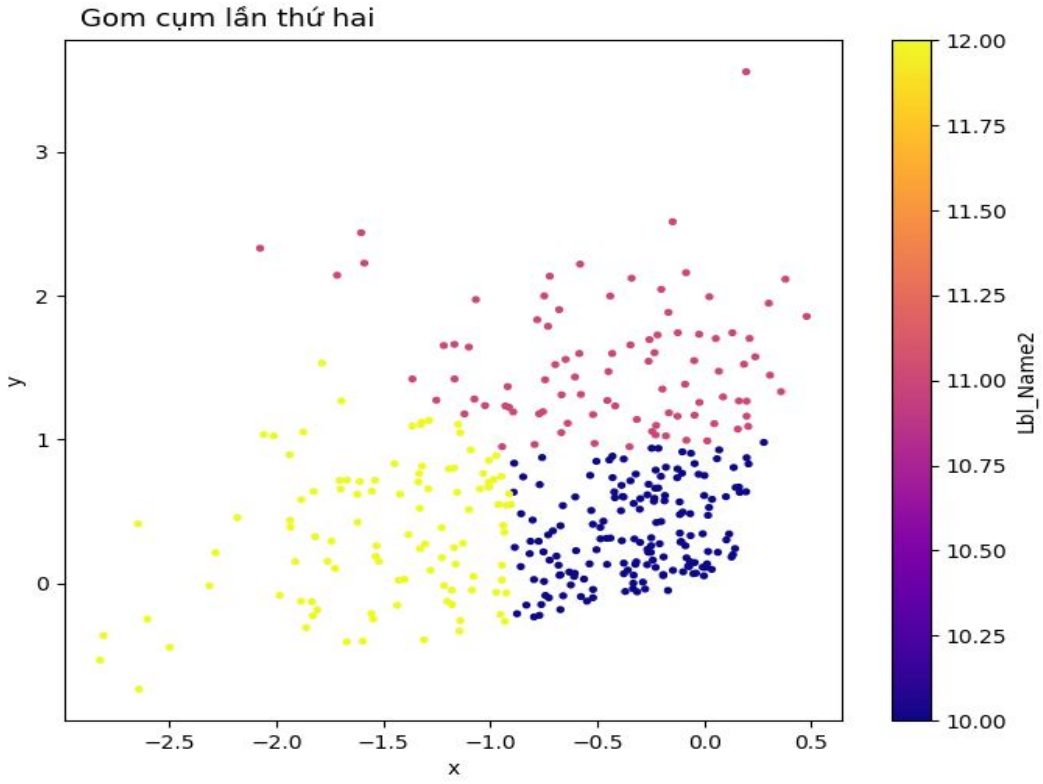
Thông tin kết quả được trả về như sau:

```
Phân cụm lần thứ 2:
Cụm 12, số hàng: 105,  số cột: 3
Cụm 10, số hàng: 168,  số cột: 3
Cụm 11, số hàng: 91,  số cột: 3
Cụm 21, số hàng: 69,  số cột: 3
Cụm 20, số hàng: 99,  số cột: 3
Cụm 22, số hàng: 143,  số cột: 3
Cụm 31, số hàng: 92,  số cột: 3
Cụm 30, số hàng: 158,  số cột: 3
Cụm 32, số hàng: 75,  số cột: 3
```

Trong đó:

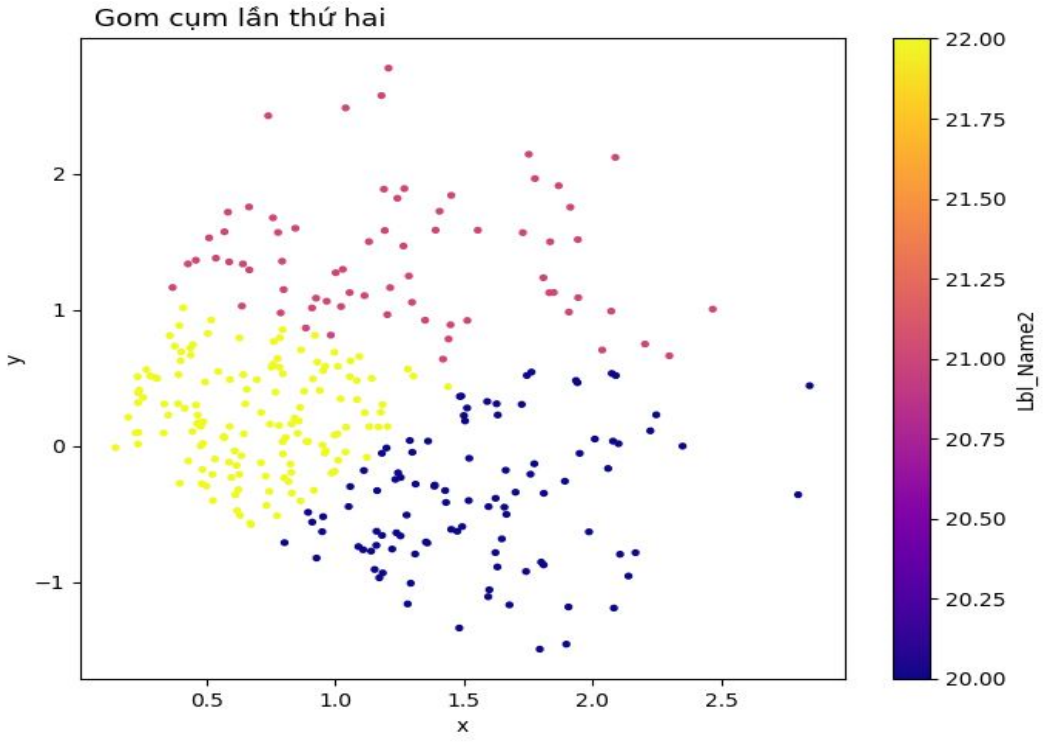
- Số cụm được gom là 3x3 = 9 cụm với nhãn cụm lần lượt như hình trên.
- Số cột gồm x , y và nhãn (label), số hàng là số các phần tử của cụm tương ứng.

a. Giản đồ thể hiện cụm có nhãn 0 (của lần lặp phân cụm thứ I _cụm xanh dương) được tiếp tục chia thành 3 cụm nhỏ:



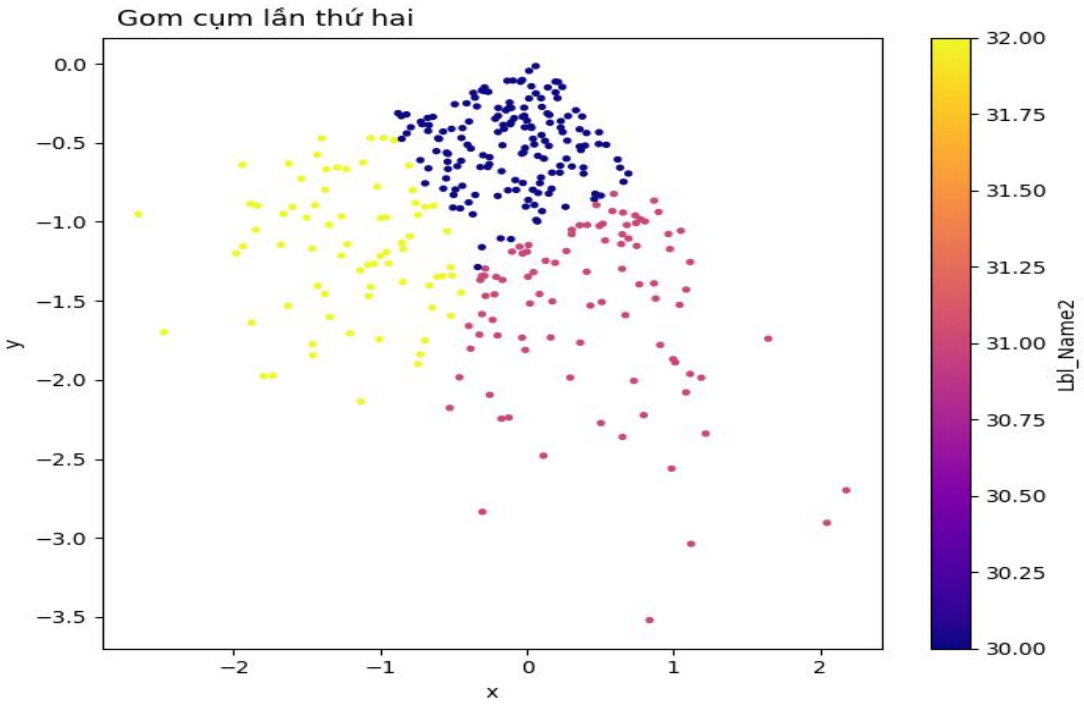
Hình 7: Giản đồ gom cụm lần thứ hai của cụm nhãn 0 với k=3

b. Cụm có nhãn 1 (của lần lặp phân cụm thứ I _cụm hồng) được tiếp tục chia thành 3 cụm nhỏ



Hình 8:Giản đồ gom cụm lần thứ hai của cụm nhãn 1 với k=3

c. Cụm có nhãn 2 (của lần lặp phân cụm thứ I _cụm vàng) được tiếp tục chia thành 3 cụm nhỏ



Hình 9: Giản đồ gom cụm lần thứ hai của cụm nhãn 2 với k=3

4.1.3 Kết quả của việc lặp lần thứ 3:

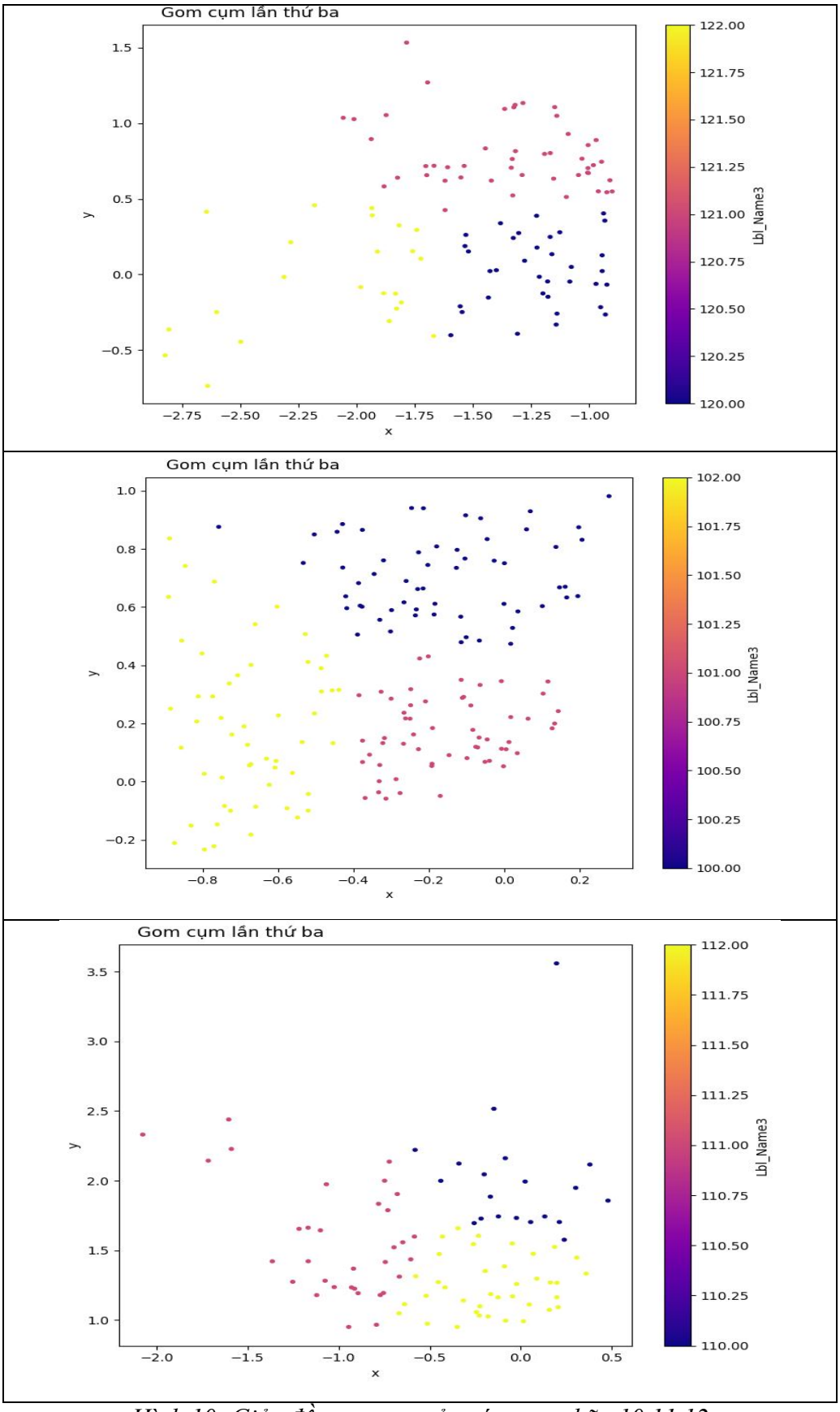
Thông tin kết quả được trả về như sau:

```
Phân cụm lần thứ 3:
Cụm 122, số hàng: 23,   số cột: 3
Cụm 120, số hàng: 35,   số cột: 3
Cụm 121, số hàng: 47,   số cột: 3
Cụm 100, số hàng: 58,   số cột: 3
Cụm 102, số hàng: 53,   số cột: 3
Cụm 101, số hàng: 57,   số cột: 3
Cụm 111, số hàng: 33,   số cột: 3
Cụm 112, số hàng: 38,   số cột: 3
Cụm 110, số hàng: 20,   số cột: 3
Cụm 210, số hàng: 35,   số cột: 3
Cụm 211, số hàng: 17,   số cột: 3
Cụm 212, số hàng: 17,   số cột: 3
Cụm 201, số hàng: 32,   số cột: 3
Cụm 202, số hàng: 47,   số cột: 3
Cụm 200, số hàng: 20,   số cột: 3
Cụm 221, số hàng: 57,   số cột: 3
Cụm 222, số hàng: 45,   số cột: 3
Cụm 220, số hàng: 41,   số cột: 3
Cụm 310, số hàng: 37,   số cột: 3
Cụm 312, số hàng: 17,   số cột: 3
Cụm 311, số hàng: 38,   số cột: 3
Cụm 302, số hàng: 58,   số cột: 3
Cụm 301, số hàng: 48,   số cột: 3
Cụm 300, số hàng: 52,   số cột: 3
Cụm 321, số hàng: 33,   số cột: 3
Cụm 320, số hàng: 27,   số cột: 3
Cụm 322, số hàng: 15,   số cột: 3
```

Trong đó:

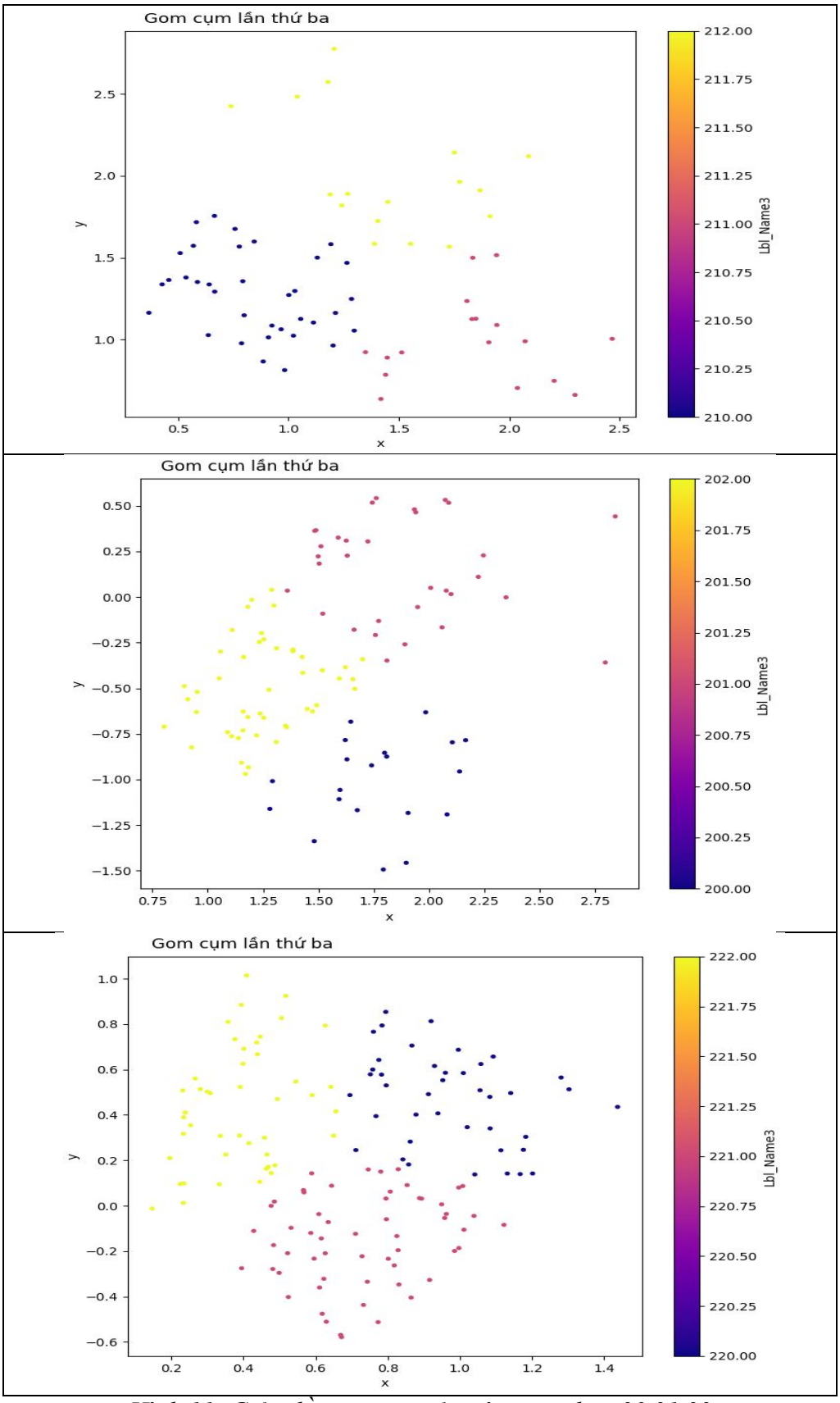
- Số cụm được gom là 9x3 = 27 cụm với nhãn cụm lần lượt như hình trên.
- Số cột gồm x , y và nhãn (label), số hàng là số các phần tử của cụm tương ứng.

a. Các cụm có nhãn 10, 11, 12 của lần gom cụm trước lại được tiếp tục chia với k =3



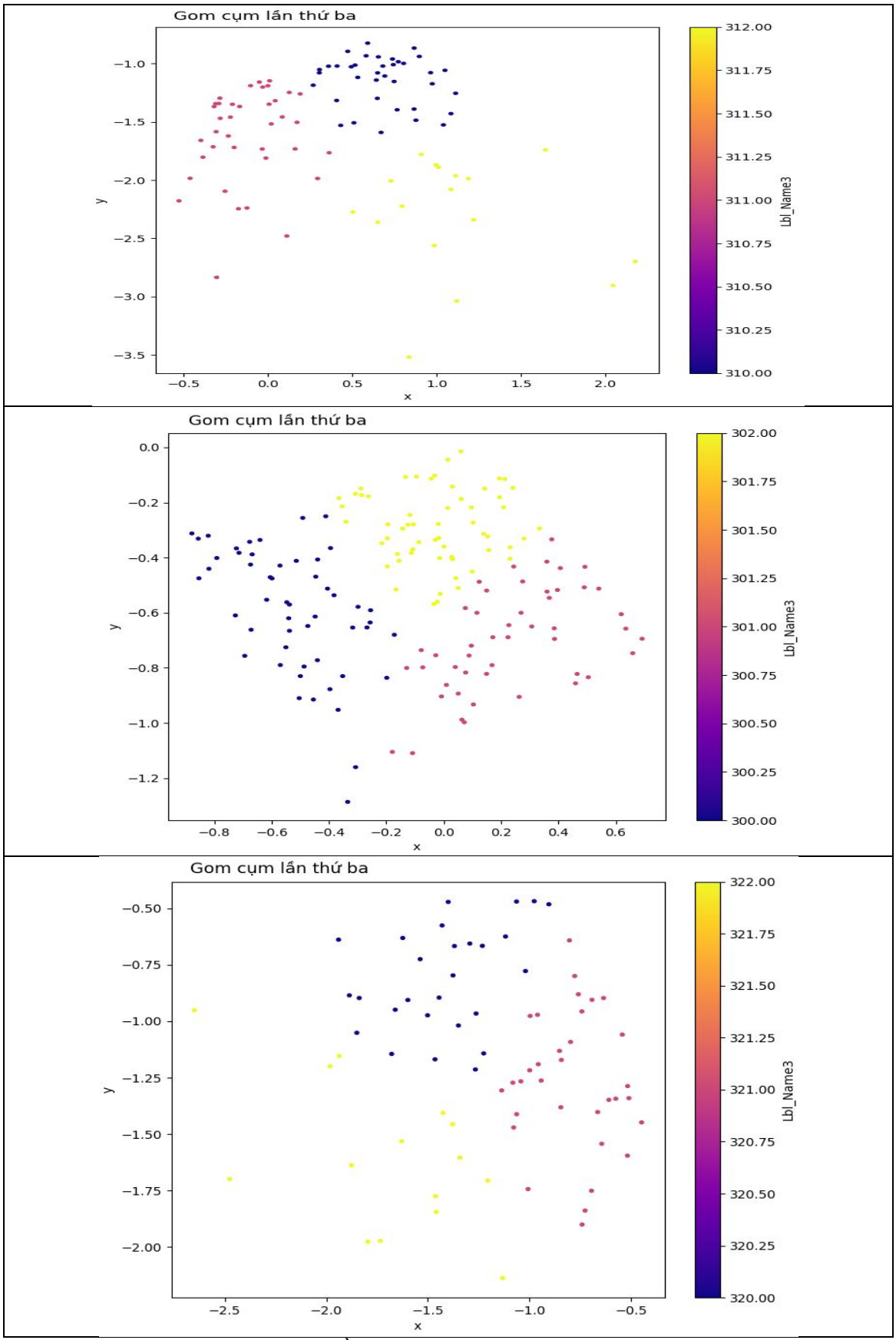
Hình 10: Giản đồ gom cụm của các cụm nhãn 10,11,12

b. Các cụm có nhãn 20, 21, 22 của lần gom cụm trước lại được tiếp tục chia với k =3



Hình 11: Giản đồ gom cụm của các cụm nhãn 20,21,22

c. Các cụm có nhãn 30, 31, 32 của lần gom cụm trước lại được tiếp tục chia với k =3



Hình 12: Giản đồ gom cụm của các cụm nhãn 30,31,32

4.2 Giai đoạn 2

5. KẾT LUẬN

TÀI LIỆU THAM KHẢO:

1. A Branch and Bound Algorithm for Computing k-Nearest Neighbors _ Keinosuke Fukunaga and Patrenahalli M. Narendra _ IEEE transaction on computer, July 1975)
2. Slides bài giảng môn học "Nhận dạng mẫu & học máy", của thầy PGS.TS Dương Tuấn Anh
3. https://matplotlib.org/api/
4. https://docs.python.org/3.6/library/index.html
5. https://docs.scipy.org/doc/numpy-1.15.1/reference/
6. https://scikit-learn.org/stable/modules/clustering.html#clustering
7. https://scikit-learn.org/stable/modules/neighbors.html#classification

PHỤ LỤC