ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

~~~~~~*~~~~~~

BÁO CÁO ĐỒ ÁN

Thuật toán Hierarchical clustering

Môn: Lập trình Python cho Máy học

Sinh viên thực hiện:

Lại Nguyễn Vĩnh Phú	19522020
Hoàng Anh Tú	19522450
Trương Thế Trương	19522247

Giảng viên hướng dẫn: Nguyễn Vinh Tiệp

Quảng Bình, ngày 23/12/2021
~~~~~~

Mục lục

I. Giới thiệu thuật toán

I.1 Bài toán phân cụm dữ liệu

Phân cụm dữ liệu là bài toán gom nhóm các đối tượng dữ liệu vào thành từng cụm sao cho:

- Các đối tượng trong cùng một cụm có sự tương đồng.
- Các cụm dữ liệu là khác nhau, tách biệt nhau hoàn toàn không có sự chồng lấp 2 cụm dữ liệu với nhau.

Input và Output của bài toán:

- Input: Tập dữ liệu không có nhãn.
- Output: Các cụm dữ liệu đã được phân chia.

I.2 Thuật toán Hierarchical clustering

Thuật toán Hierarchical clustering được sử dụng để ứng dụng trong việc giải bài toán phân cụm dữ liệu với:

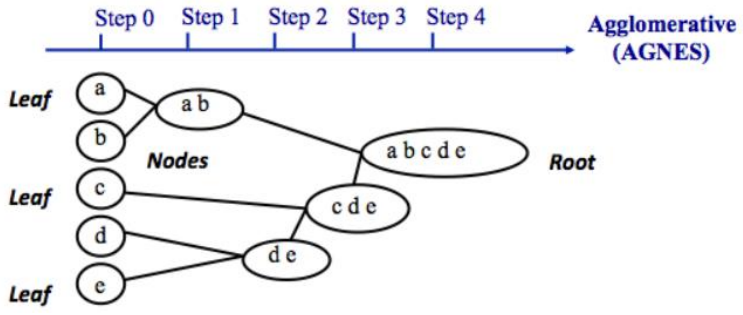
- Input:
  Mảng 2D có kích thước mxn, trong đó:
  - Một array chứa tập hợp m array con, các array con có cùng n phần tử và mỗi phần tử có cùng kiểu dữ liệu là số.

  Hay nói theo toán học thì input là tập hợp m điểm trong một cùng một không gian n chiều.
- Output:
  - Mảng 1D gồm các số nguyên, trong đó: Mỗi số nguyên là nhãn cho từng điểm dữ liệu tương ứng, nhãn xác định điểm dữ liệu thuộc về cụm nào.

Phân loại Hierarchical clustering

I.2.1 Agglomerative clustering

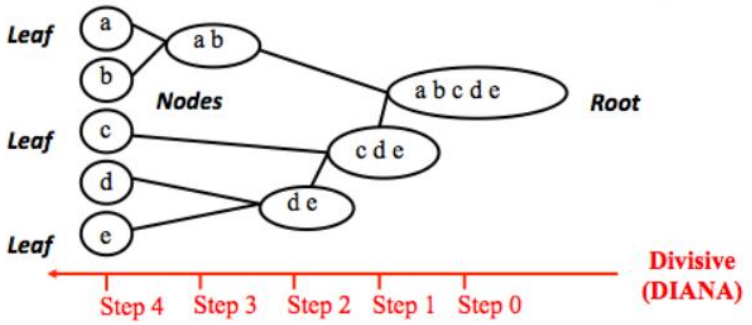
Ở thuật toán này, ban đầu, mỗi đối tượng được coi là một cụm (lá). Ở mỗi bước của thuật toán, hai cụm gần giống nhau nhất (khoảng cách giữa 2 cụm trong không gian n chiều là nhỏ nhất) sẽ được kết hợp lại thành một cụm lớn hơn (nút). Quy trình này được lặp lại cho đến khi tất cả các điểm chỉ là thành viên của một cụm lớn duy nhất (gốc).



Hình I.1 Quy trình thực hiện Agglometartive

I.2.2 Divisive clustering

Trái ngược với Agglomerative, ban đầu, tất cả các đối tượng sẽ được xem là một cụm (gốc). Ở mỗi bước của thuật toán, cụm không đồng nhất sẽ được tách ra làm 2. Quy trình được lặp lại cho đến khi tất cả đối tượng là một cụm riêng.



Hình I.2 Quy trình thực hiện Divisive

I.2.3 So sánh Agglomerative và Divisive clustering

Phương pháp Divisive phức tạp hơn Agglomerative về mặt khái niệm cũng như là cách hoạt động, đồng nghĩa với việc thuật toán phân cụm bằng phương pháp Agglomerative có độ phức tạp thực toán nhỏ hơn so với phương pháp Divisive. Vì thế, trong thực tế, Agglomerative được sử dụng phổ biến hơn.

Bù lại, phương pháp Divisive cho ra kết quả chính xác hơn phương pháp Agglomerative. Vì Agglomerative tiến hành phân cụm bằng cách xem xét từ các cụm dữ liệu nhỏ đến các cụm dữ liệu lớn hơn, nên phương pháp này sẽ quyết định chỉ dựa trên sự phân bố cục bộ của dữ liệu mà không quan tâm đến tính phân bố toàn cục của tất cả dữ liệu lúc đầu. Trong khi đó phương pháp Divisive đưa ra quyết định bằng cách xem xét một cách tổng quan về sự phân phối của dữ liệu, phân cụm từ cụm dữ liệu lớn nhất, nên sẽ đưa ra kết quả một cách khách quan hơn.

I.3 Sử dụng thuật toán Agglomerative clustering trong Python

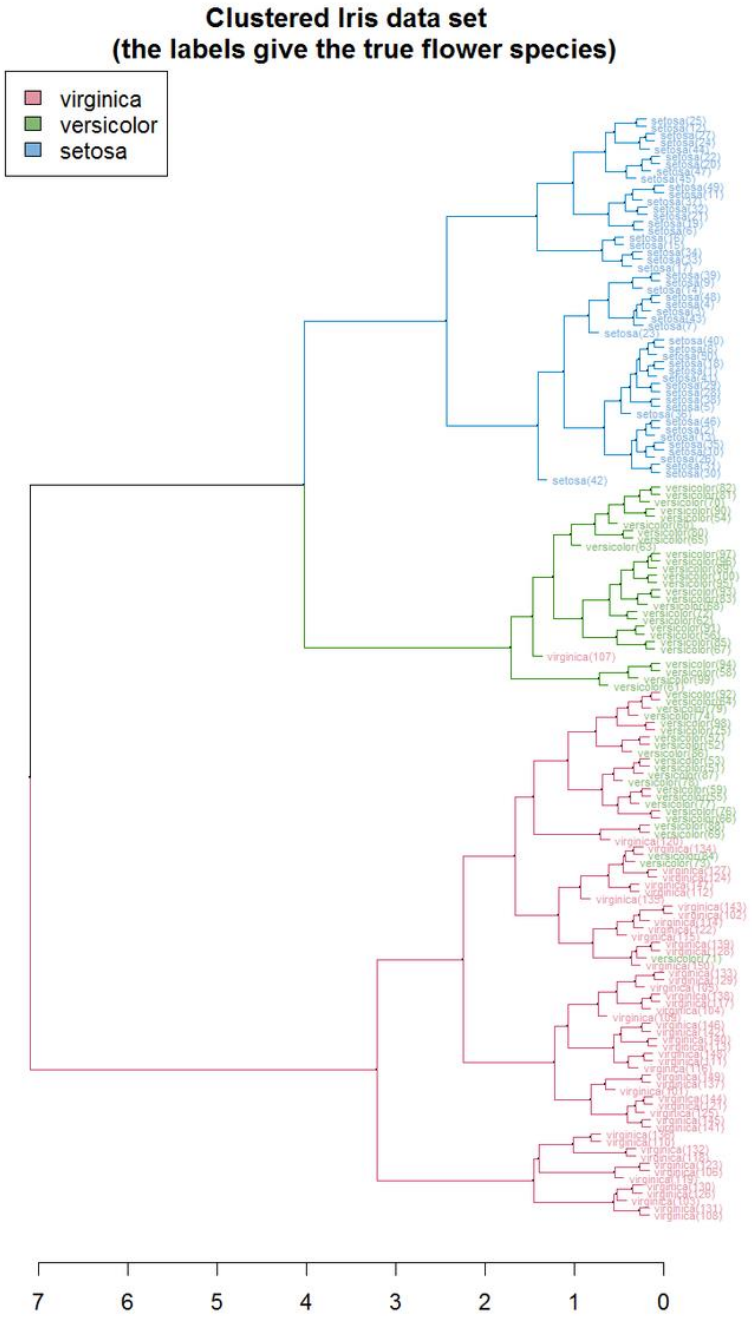
Do nhóm em không tìm thấy thư viện hỗ trợ thuật toán Diviseve clustering, đồng thời nếu tự mình code thuật toán thì cho dù chạy được cũng chưa chắc đã tốt và chính xác. Vì vậy, nhóm xin phép thầy sử dụng thuật toán Agglomerative clustering của thư viện sklearn.

Thư viện sklearn có hỗ trợ sklearn.cluster.AgglomerativeClustering, là một object hỗ trợ chạy thuật toán Hierarchical clustering trên Python. Thư viện hỗ trợ nhiều cài đặt của thuật toán như các cách tính khoảng cách khác nhau giữa các điểm và giữa các cụm, chế độ dừng gộp cụm khi các cụm có khoảng cách xa... Các cài đặt của thuật toán sẽ được truyền vào bằng các tham số như linkage, distance_threshord,… cụ thể các tham số sẽ được trình bày sau.

II. Cách hoạt động

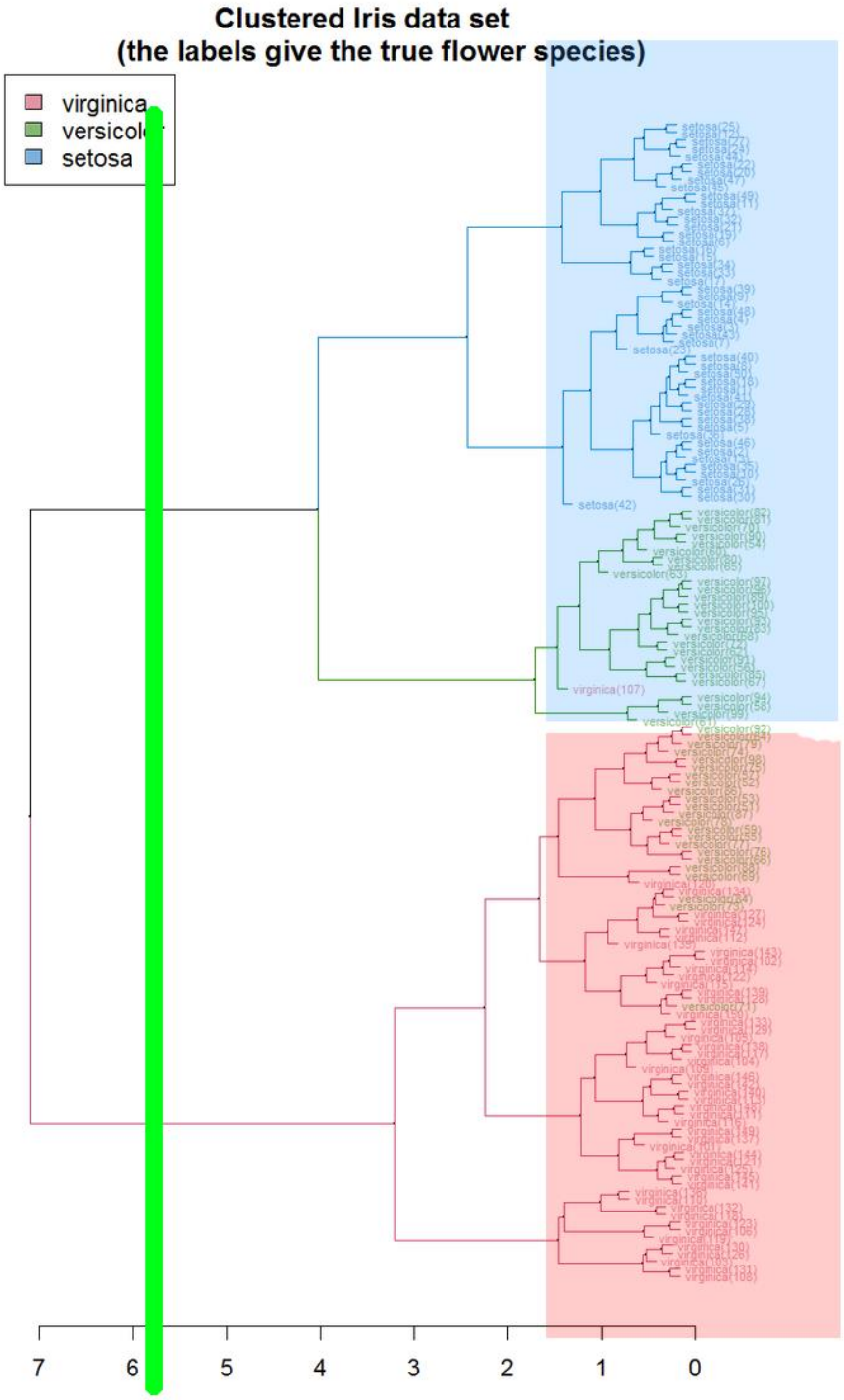
Dữ liệu đầu vào sẽ được coi như các **điểm** trong không gian. Thuật toán sẽ bắt đầu "**kết nối**" 2 điểm gần nhau nhất (có khoảng cách trong không gian nhỏ nhất), sau đó lại đến 2 điểm gần nhau thứ 2, … Thuật tóa sẽ tiếp tục đến khi tất cả các điểm đều được kết nối.

Để biểu diễn sự kết nối, người ta dùng sơ đồ cây (**Dendrogram**) , ví dụ như sau :



Hình II.1. Dendrogram của data Iris

Nhìn vào đồ thị cây bên trên, các lá là các điểm dữ liệu, các nhánh cây thể hiện sử kết nối. Nhánh cây càng nhỏ tức là khoảng cách giữa các điểm được kết nối trong nhánh càng nhỏ.



Hình II.2 Ví dụ phân cụm

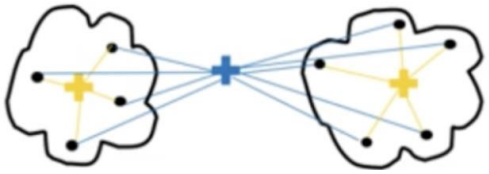
Ví dụ với dữ liệu trên ta kẻ một đường thẳng đứng cắt 2 nhánh cây lớn nhất, thì dữ liệu sẽ được phân thành 2 cum, với các điểm dữ liệu thuộc cùng 1 nhanh cây sẽ cùng một cum.

III. Siêu tham số

Linkage (liên kết)

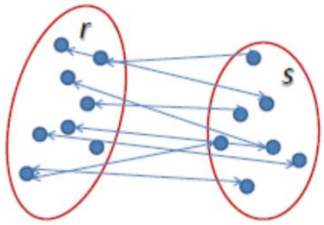
Về bản chất thuật toán là một thuật toán **đệ quy**, liên tục thực hiện liên kết các cụm có "**khoảng cách**" gần nhau nhất. Về các cách tính "**khoảng cách**" giữa các cụm, thư viện sklearn có hỗ trợ 4 cách:

- "**ward**": liên kết 2 cụm có độ "biến động nhỏ nhất" (min variance). Độ biến động có thể hiểu như thể tích(3d), diện tích(2d), độ dài(1d) của một cụm chiếm trong không gian.



Hình III.1. Ward's Method

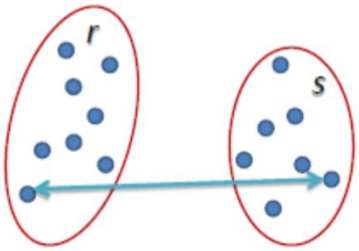
- "**average**": khoảng cách trung bình của các điểm thuộc về 2 cụm.



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

Hình III.2. Average Linkage

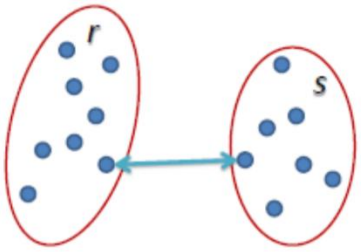
- "**complete**" hay "**maximum**": khoảng cách lớn nhất của các điểm thuộc về 2 cụm.



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

Hình III.3. Complete Linkage

- "**single**": khoảng cách nhỏ nhất của các điểm thuộc về 2 cụm.



$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Hình III.4. Single Linkage

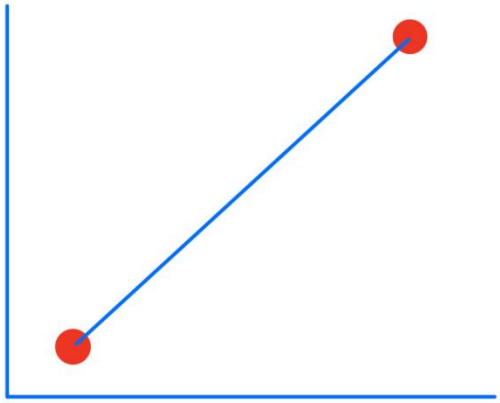
Affinity

Tham số này là chỉ số được sử dụng để tính toán liên kết. Sklearn có hỗ trợ 4 lựa chọn cho chỉ số này:

- **"euclidean" (L2)**: Đây là số liệu khoảng cách mặc định được sử dụng để đo khoảng cách giữa các cụm và chỉ đơn giản là khoảng cách giữa hai điểm. Điều này được biểu thị bằng toán học là:

$$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

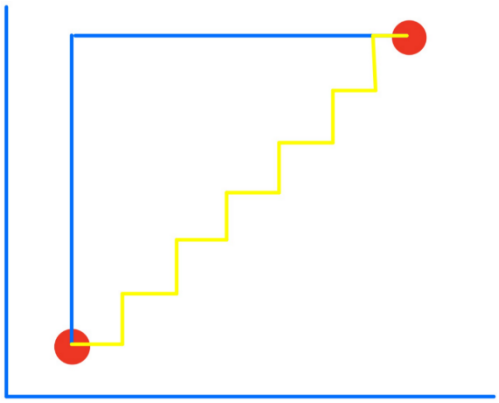
Được trực quan như sau:



- **"manhattan" (L1)**: Khoảng cách này được tính bằng tổng chiều dài của hình chiếu của đường thẳng nối hai điểm này trên các trục tọa độ, được biểu thị toán học là:

$$d(x, y) = \sum_{n=1} |x_i - y_i|$$

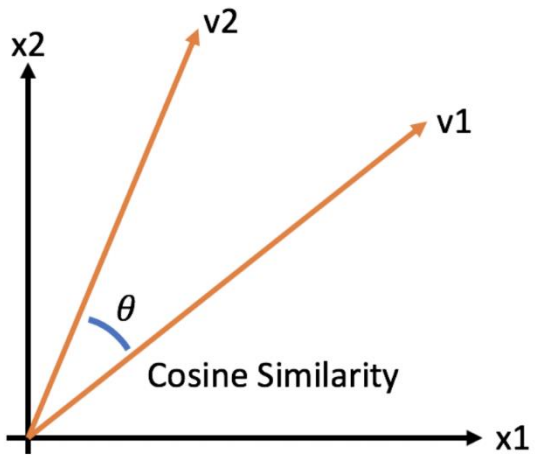
Được trực quan như sau:



- **“cosine”**: Số liệu khoảng cách này đo mức độ góc giữa hai vectơ. Góc càng nhỏ, độ tương tự cosin của bạn sẽ càng lớn và độ tương tự cosin của bạn càng lớn, thì các vectơ của bạn càng giống nhau. Điều này được đo lường như sau:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Được trực quan như sau:



- **“precomputed”**: Được sử dụng khi đã có ma trận khoảng cách (distance matrix), số liệu này nhằm mục đích tối đa hóa khoảng cách giữa các điểm khác lớp và giảm thiểu khoảng cách giữa các điểm cùng lớp.

N_clusters

Đầu vào của tham số này là một số nguyên thể hiển sộ cụm dữ liệu cần tìm.

Memory

Tham số này được sử dụng để lưu kết quả vào bộ nhớ cache. Mặc định sẽ là **None.**

Connectivity

Tham số này là ma trận kết nối (connectivity matrix), được dùng để xác định cho mỗi mẫu lân cận theo một cấu trúc dữ liệu nhất định. Mặc đinh của tham số này là **None**, có nghĩa là thuật toán sẽ là phân cụm không có cấu trúc.

Compute_full_tree

Tham số này có thể là 3 giá trị "auto", False, True. Tham số này quyết định thuật toán có dừng sớm hay sẽ tính toán như bình thường.

Distance_threshold

Kiểu dữ liệu của tham số này là **Float** hoặc **None**. Nếu 2 cụm có khoảng cách lớn hơn tham số này thì sẽ không được gộp thành 1 cụm. Cách tính khoảng cách của 2 cụm dựa vào tham số **linkage** (mặc định là "ward").

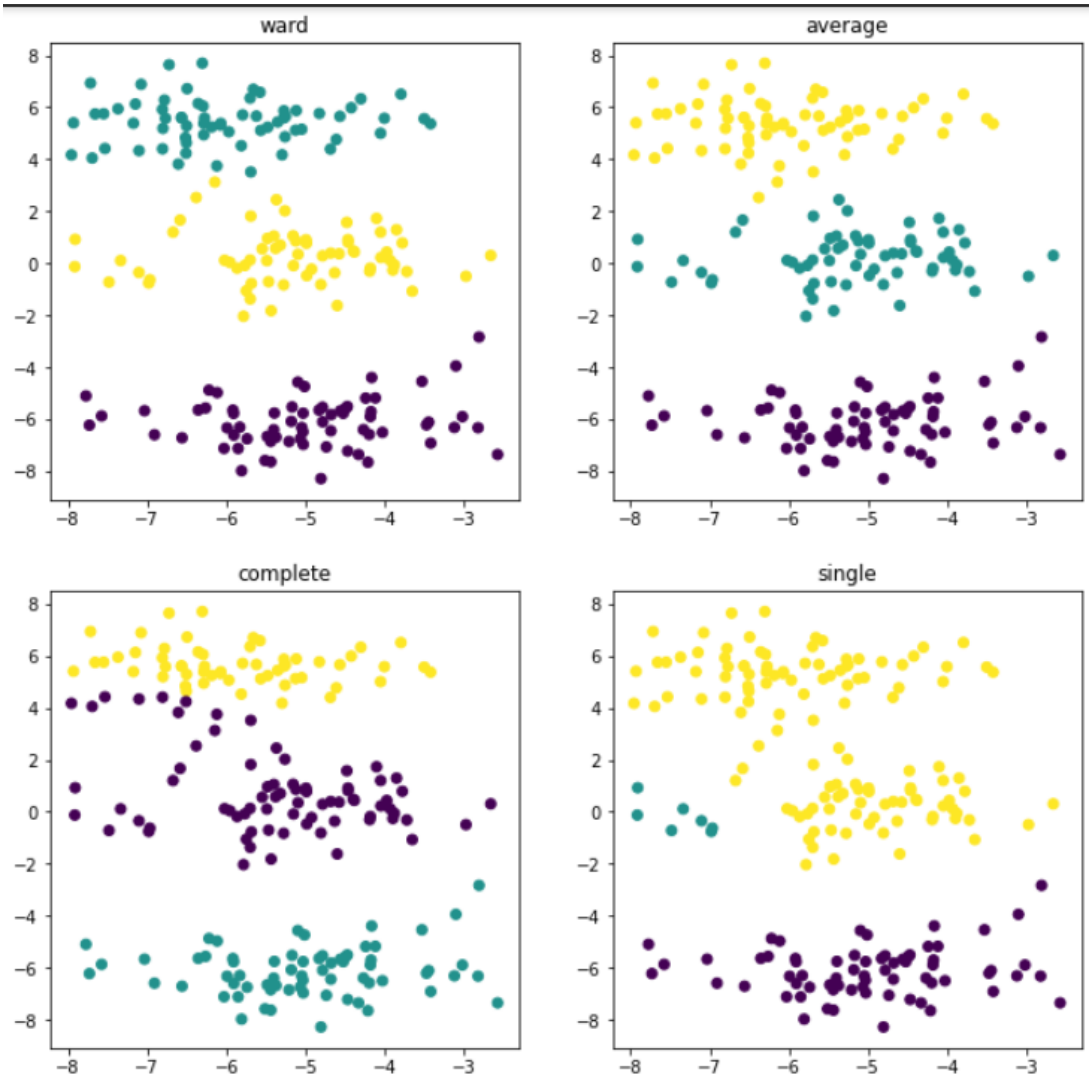
Compute_distances

Kiểu dữ liệu tham số này là bool. Tham số này quyết định có tính khoảng cách giữa các cụm hay không.

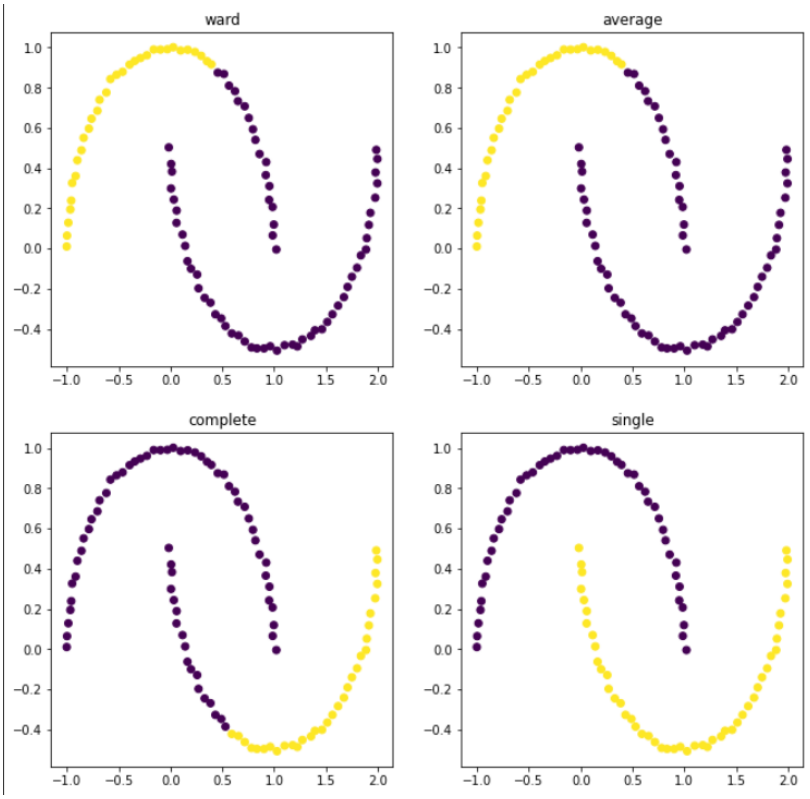
IV. Điều chỉnh siêu tham số

IV.1 Linkage (liên kết)

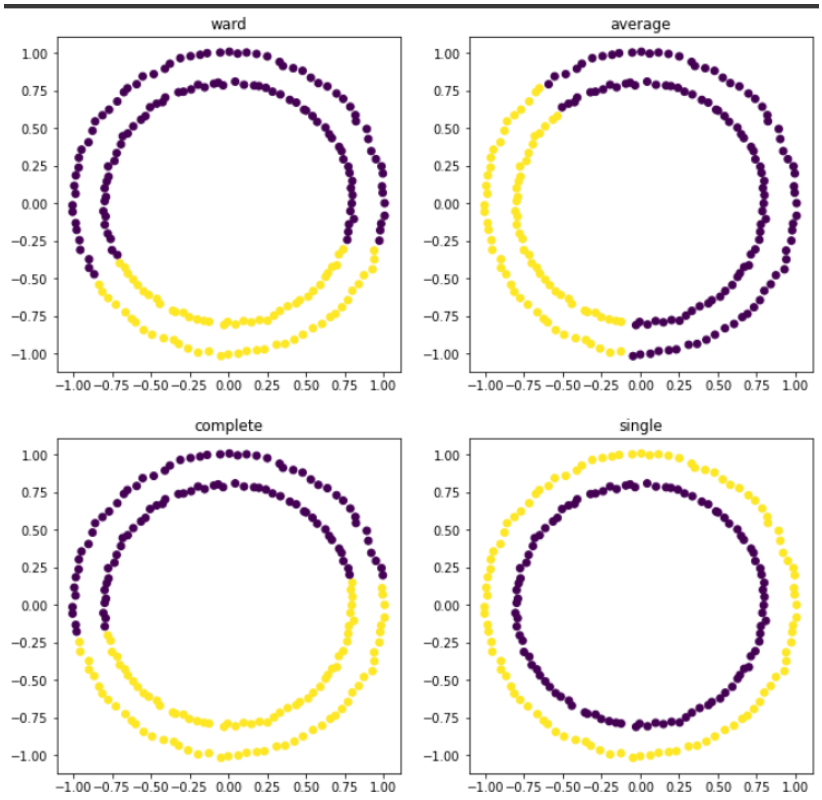
Tiến hành chạy thử 4 cách tính linkage với các mẫu dữ liệu:



Hình IV.1. Kết quả phân cụm của data ngẫu nhiên



Hình IV.2. Kết quả phân cụm của data vòng cung



Hình IV.3. Kết quả phân cụm của data đường tròn

Dựa vào kết quả phân cụm của ít bộ dữ liệu ở trên, ta thấy linkage "single" có kết quả khác biệt nhất với các linkage còn lại.

Linkage **"single"** phù hợp với các dữ liệu có "hình dáng" đặc biệt, ví dụ như các điểm tạo thành các đường tròn, đường vòng cung ở trên. Các dữ liệu đó có điểm chung đó là khoảng cách giữa các điểm trong cụm tương đối đều, các cụm tách biệt rõ ràng.

Linkage **"ward", "average", "complete"** phù hợp với các dữ liệu có tính ngẫu nhiên cao hơn, các cụm phân tách ít rõ ràng, các cụm cũng không có hình dáng đặc biệt. VD trong mẫu dữ liệu ở hình IV.1, bằng mắt thường ta có thể thấy các điểm tạo thành 3 dãy nằm ngang, trừ linkage "single" thì các linkage còn lại đã làm tốt.

IV.2 Affinity

Affinity **"euclidean"** và **"manhattan"** thường được sử dụng khi bộ dữ liệu có tính liên tục, khoảng cách **"manhattan"** được ưu tiên hơn khi bộ dữ liệu có số chiều lớn.

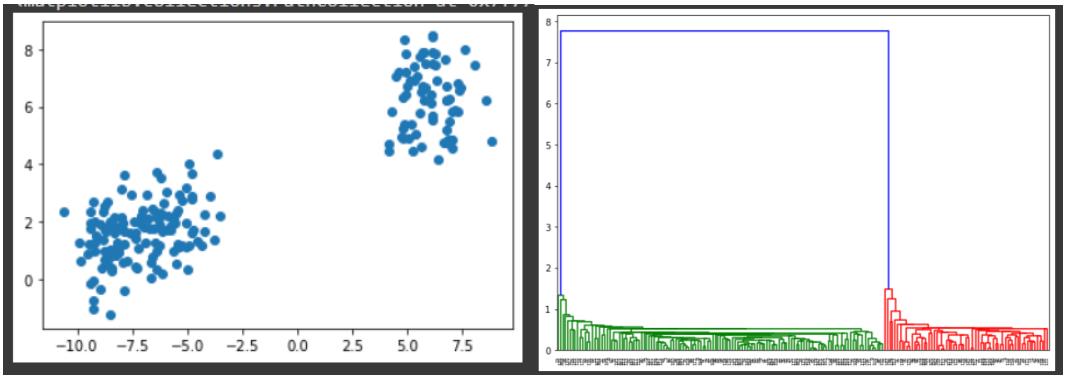
Còn **"cosine"** được dùng khi độ lớn khoảng cách giữa các điểm không quan trọng nhưng hướng thì có, tức là khi cần tìm điểm giống nhau giữa hai điểm dữ liệu, số liệu này thường được sử dụng trong các hệ thống đề xuất.

Còn **"precomputed"** được dùng khi đã có ma trận khoảng cách (distance matrix), số liệu này được sử dụng nhằm mục đích tối đa hóa khoảng cách giữa các điểm khác lớp và giảm thiểu khoảng cách giữa các điểm cùng lớp. Điều này giúp cho thuật toán có độ chính xác cao hơn.

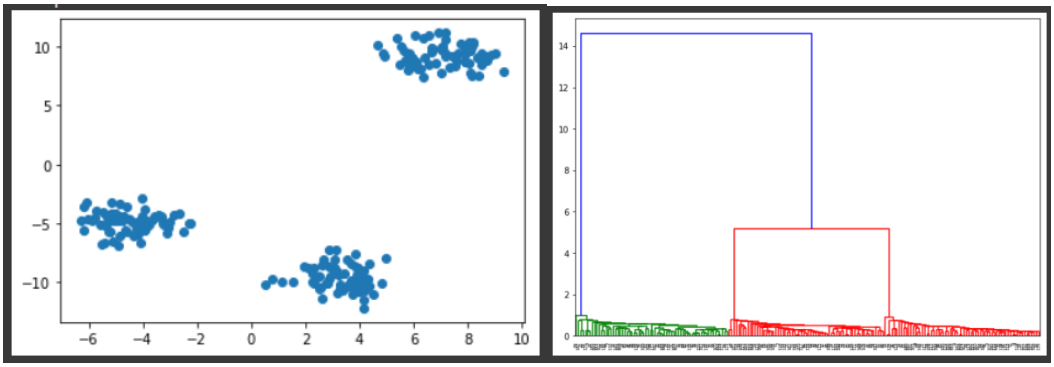
IV.3 N_cluster

Đối với thuật toán KMeans thì có thể sử dụng đồ thị Elbow để chọn số cụm, thì thuật toán Hierachical có đồ thị Dendrogram để chọn số cụm phù hợp nhất.

Khi thuật toán liên tục kết nối các điểm gần nhất lại với nhau, ta có thể biểu diễn sự kết đó bằng đồ thị cây, với các lá là các điểm dữ liệu, còn chiều cao cây đại diện cho khoảng cách của các điểm dữ liệu.

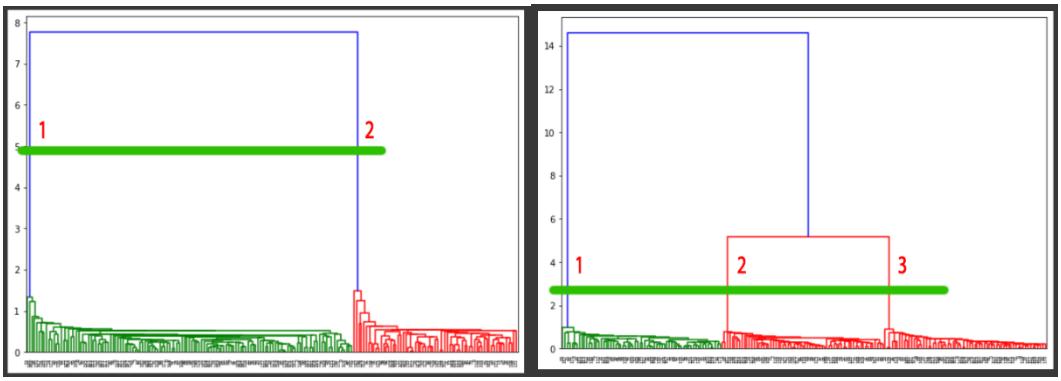


Hình IV.4 data và Dendrogram tương ứng



Hình IV.5 data và Dendrogram tương ứng

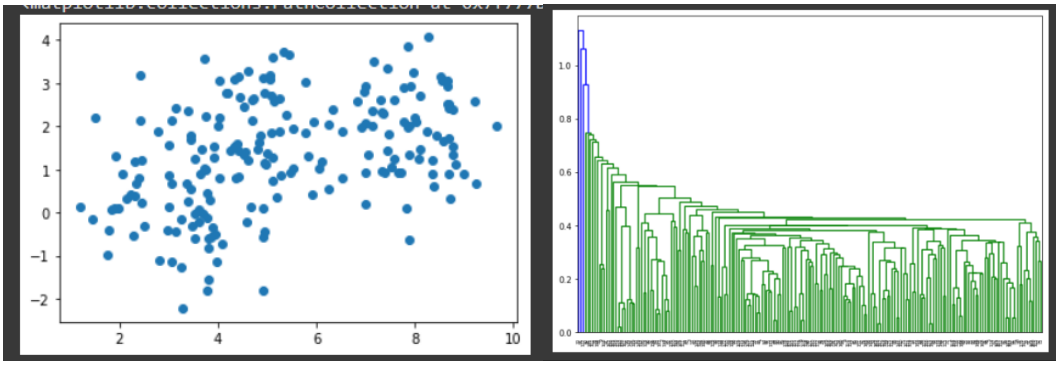
Ví dụ với data như hình trên bên trái, ta sẽ được Dendrogram như hình bên phải. Những nhánh cây nối các điểm cùng một cụm khá đồng đều, còn nhánh kết nối các cụm sẽ cao và rộng. Ta có thể cắt ngang các nhánh cao đó là có thể chọn được số cụm.



Hình IV.6 chọn số cụm cho các dữ liệu trên.

Bên trên là các dữ liệu dễ, các điểm tạo thành các cụm rõ ràng, nhưng nếu gặp các dữ liệu khó hơn, phân bố ngẫu nhiên hơn, thì nên dựa vào yêu cầu bài toán để chọn ra số lượng cụm dữ liệu cần tìm phù hợp nhất.

Ví dụ một mẫu dữ liệu khác.



Hình IV.7 data và Dendrogram tương ứng

IV.4 Compute_full_tree

Dựa vào bài toán để tùy chỉnh giá trị cho tham số này.

- Nếu là "**auto**" thì thư viện sẽ xem xét các tham số khác để xác định **False** hay **True**
- Nếu là **True**, thì thuật toán sẽ tính toán như bình thường.
- Nếu là **False**, thuật toán sẽ dừng sớm. Phù hợp để giảm khối lượng tính toán khi số lượng cụm không quá nhỏ so với data.

IV.5 Distance_threshold

Mặc định của tham số này là None, tức là không sử dụng lọc ngưỡng khoảng cách trong thuật toán. Nhưng trong trường hợp người dùng muốn các cụm có một khoảng cách tối thiểu là **d** thì có thể truyền vào tham số **distance_threshold = d** . Khi truyền vào tham số này vào thì các cụm có khoảng cách lớn hơn **d** sẽ không được gộp thành 1 cụm.

IV.6 Compute_distances

Giá trị mặc định của tham số này là False.

Việc tính khoảng cách các cụm có thể sử dụng cho lọc ngưỡng khoảng cách (distance_threshord) và vẽ đồ thị cây (dendrogram), nhưng lại tốn tài nguyên tính toán.

Nếu có 1 trong 2 nhu cầu trên thì hãy truyền vào giá trị **compute_distances = True** .

V. Thực nghiệm

V.1 Dữ liệu

Nhóm sử dụng 2 bộ dữ liệu:

Tên file	Nguồn	Kích thước
Iris.csv	https://www.kaggle.com/uciml/iris?fbclid=IwAR2t90CoL5s31kr-aPfI1WEeySeFOmDC3W59wzR-ymg1Go2rDvevDctwq1I	150 đối tượng x 4 thuộc tính
Bank.csv	https://archive.ics.uci.edu/ml/machine-learning-databases/00222/?fbclid=IwAR2we-Z8dCvdHcNkuJc2Q0J2JjD12zoFbGCQMdovA5JzE3kLp7my3Me7V74	7907 đối tượng x 16 thuộc tính

V.1.1 Khám phá bộ dữ liệu Iris.csv

Mô tả bộ dữ liệu

- Bộ dữ liệu bao gồm ba loài hoa diên vĩ với 50 mẫu mỗi loài cũng như một số đặc tính về mỗi loài hoa.

```
RangeIndex: 150 entries, 0 to 149
Data columns (total 4 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   SepalLengthCm  150 non-null    float64
 1   SepalWidthCm   150 non-null    float64
 2   PetalLengthCm  150 non-null    float64
 3   PetalWidthCm   150 non-null    float64
dtypes: float64(4)
```

Hình V.1. Thông tin chung

	0	1	2
SepalLengthCm	5.1	4.9	4.7
SepalWidthCm	3.5	3.0	3.2
PetalLengthCm	1.4	1.4	1.3
PetalWidthCm	0.2	0.2	0.2

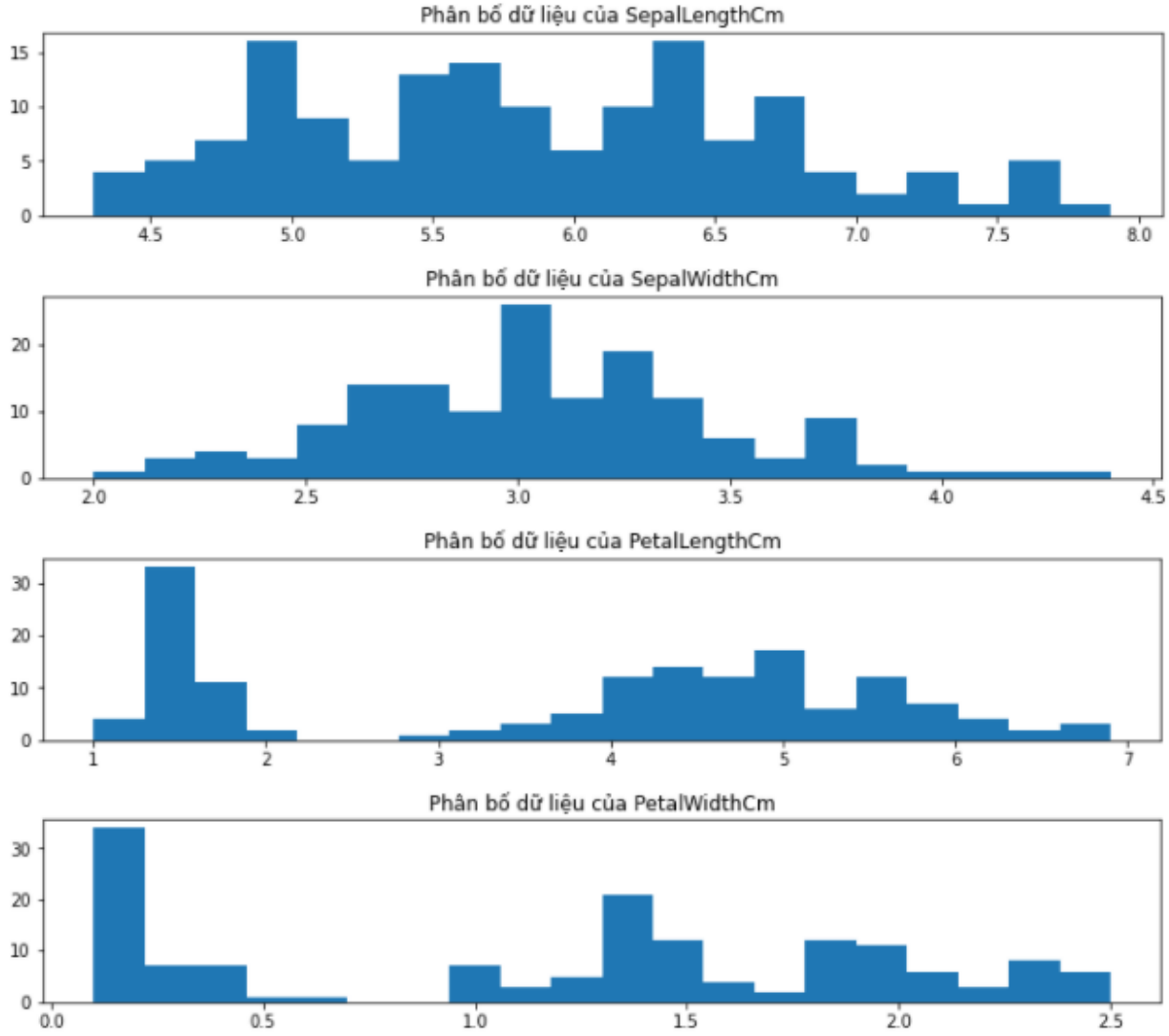
Hình V.2. Thông tin một vài đối tượng

Nhận xét:

(1) Bộ dữ liệu không bị thiếu khuyết.
(2) Tập giá trị của các cột trong bộ dữ liệu là số thực.
(3) Các đơn vị đo lường của từng cột là giống nhau, đều là “Cm”.

Mô hình hóa dữ liệu

Sự phân bố dữ liệu



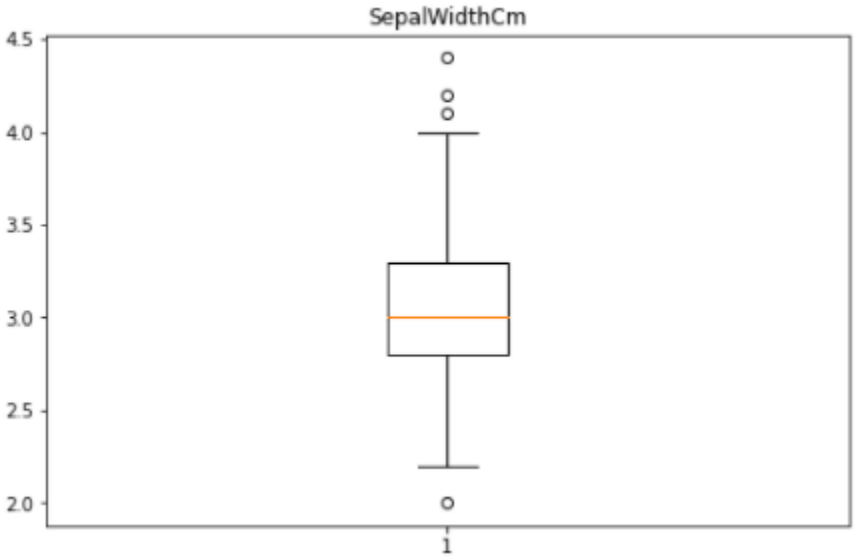
Hình V.3. Phân bố dữ liệu của một số cột

Nhận xét:

(4) Sự chênh lệch ở tập giá trị giữa các cột nhỏ, chưa đủ để làm khoảng cách giữa các "điểm trong không gian n chiều" do những thuộc tính có khoảng giá trị lớn hơn quyết định.

```
SepalLengthCm : [ 4.3 ;  7.9 ]
SepalWidthCm :  [ 2.0 ;  4.4 ]
PetalLengthCm : [ 1.0 ;  6.9 ]
PetalWidthCm :  [ 0.1 ;  2.5 ]
```

Sự tồn tại của các ngoại lệ (outliers)

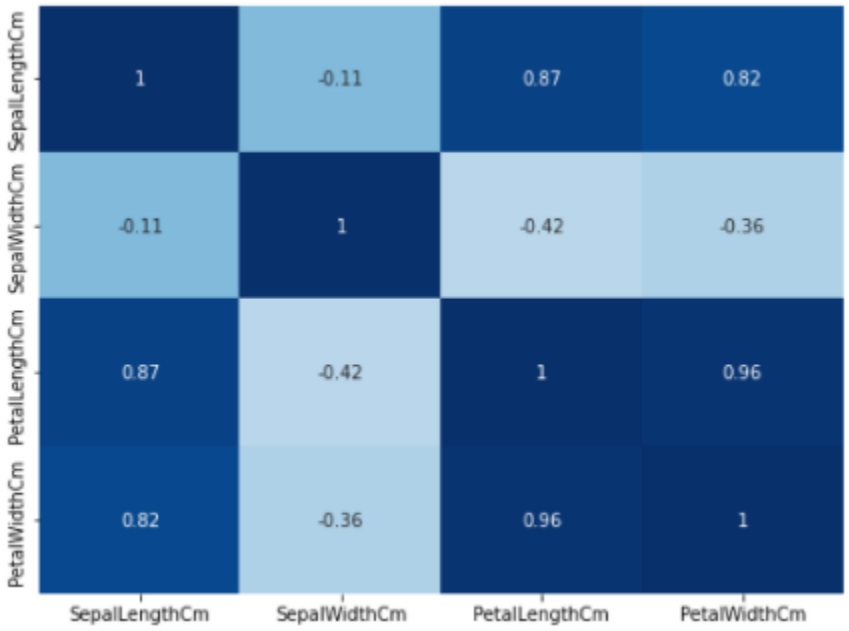


Hình V.4. Biểu đồ boxplot của một số cột chứa ngoại lệ

Nhận xét:

(5) Cột SepalWidthCm có chứa các giá trị ngoại lệ.

Sự tương quan tuyến tính giữa các thuộc tính



Hình V.5. Biểu đồ thể hiện sự tương quan tuyến tính giữa các thuộc tính

Theo nhận xét ở mục II:

- Từ ý (1), ta không cần **xử lý dữ liệu bị khuyết**.
- Từ ý (2), ta không cần **mã hóa đặc trưng dạng nhóm**.

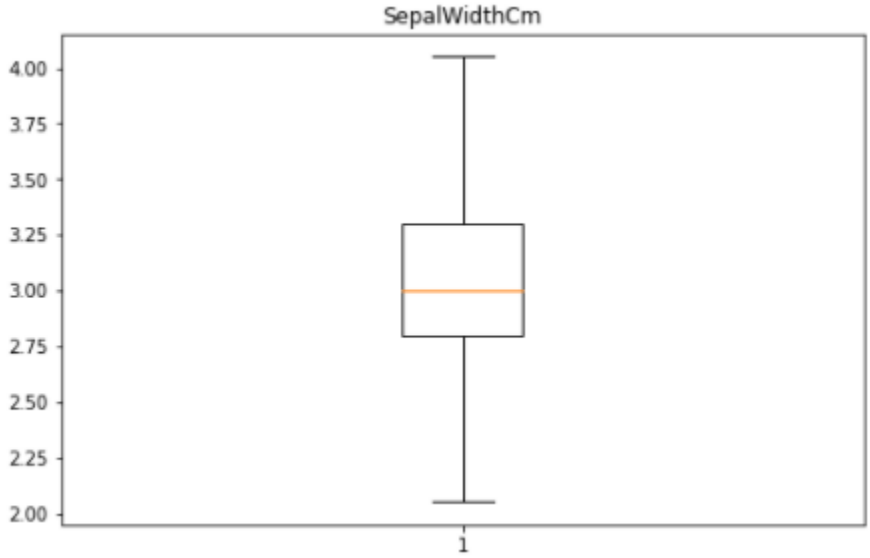
- Từ ý (3) và (4), ta không cần phải **chuẩn hóa dữ liệu**.
- Từ ý (5), nên **xử lý các điểm ngoại lệ (outlier)** để tránh việc model phân ra những cụm chỉ có 1-2 phần tử.

Vậy ở bước tiền xử lý dữ liệu, nhóm sẽ chỉ thực hiện **xử lý các điểm ngoại lệ (outlier)**.

V.1.2 Tiền xử lý bộ dữ liệu Iris.csv

Xử lý các điểm ngoại lệ (outlier)

Các điểm ngoại lệ sẽ được xử lý bằng cách clip về giá trị cực tiểu và cực đại của Boxplot.



Hình V.6. Các cột có outlier ở Hình V.4 sau khi loại bỏ outlier.

- Tập giá trị của cột SepalWidthCm từ [2.0; 4.4] thay đổi thành [2.05; 4.05].

V.1.3 Khám phá và xử lý bộ dữ liệu Bank.csv

- Dữ liệu có liên quan đến các chiến dịch tiếp thị trực tiếp của một tổ chức ngân hàng Bồ Đào Nha.
- Mục tiêu của bộ dữ liệu là dự đoán xem liệu khách hàng có đăng ký một khoản tiền gửi có kỳ hạn hay không.

```
RangeIndex: 7907 entries, 0 to 7906
Data columns (total 16 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        7907 non-null   int64
 1   job        7907 non-null   object
 2   marital    7907 non-null   object
 3   education  7907 non-null   object
 4   default    7907 non-null   object
 5   balance    7907 non-null   int64
 6   housing    7907 non-null   object
 7   loan       7907 non-null   object
 8   contact    7907 non-null   object
 9   day        7907 non-null   int64
 10  month      7907 non-null   object
 11  duration   7907 non-null   int64
 12  campaign   7907 non-null   int64
 13  pdays      7907 non-null   int64
 14  previous   7907 non-null   int64
 15  poutcome   7907 non-null   object
dtypes: int64(7), object(9)
```

	0	1	2
age	33	42	33
job	admin.	admin.	services
marital	married	single	married
education	tertiary	secondary	secondary
default	no	no	no
balance	882	-247	3444
housing	no	yes	yes
loan	no	yes	no
contact	telephone	telephone	telephone
day	21	21	21
month	oct	oct	oct
duration	39	519	144
campaign	1	1	1
pdays	151	166	91
previous	3	1	4
poutcome	failure	other	failure

Hình V.7. Thông tin ban đầu của bộ dữ liệu

Nhận xét:

- Bộ dữ liệu không bị khuyết thiếu giá trị.
- Nhiều thuộc tính có kiểu dữ liệu là chuỗi.
- Nhiều thuộc tính mang ý nghĩa dư thừa.
- Tập giá trị của các cột chênh lệnh nhau lớn.

Tiền xử lý dữ liệu:

1. Đầu tiên, nhóm sẽ xóa bớt các thuộc tính không cần thiết.
2. Tiếp theo là chuyển các đối tượng là chuỗi về số nguyên.
3. Sau đó chuẩn hóa dữ liệu để đưa các thuộc tính về cùng một thang đo.

```
RangeIndex: 7907 entries, 0 to 7906
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   age        7907 non-null   float64
 1   job        7907 non-null   float64
 2   marital    7907 non-null   float64
 3   education  7907 non-null   float64
 4   default    7907 non-null   float64
 5   balance    7907 non-null   float64
 6   housing    7907 non-null   float64
 7   loan       7907 non-null   float64
 8   duration   7907 non-null   float64
 9   campaign   7907 non-null   float64
 10  pdays      7907 non-null   float64
 11  previous   7907 non-null   float64
 12  poutcome   7907 non-null   float64
dtypes: float64(13)
```

	0	1	2
age	-0.237467	0.031598	-0.287475
job	-0.440294	-0.373460	-0.372031
marital	-0.269217	0.194698	-0.325911
education	-0.420349	0.092944	0.132653
default	-0.029048	-0.024639	-0.035166
balance	-0.074623	-0.170287	0.256184
housing	-0.448880	0.223539	0.319043
loan	-0.138337	0.725345	-0.167469
duration	-0.321776	0.320337	-0.204315
campaign	-0.232567	-0.197264	-0.281543
pdays	-0.220621	-0.149198	-0.483639
previous	-0.014334	-0.138323	0.072680
poutcome	-0.258903	0.156079	-0.313425

Hình V.8. Bộ dữ liệu sau khi xử lý

V.2 Phương pháp đánh giá model

Để đánh giá chất lượng mô hình phân cụm ta có thể đánh giá thông qua một số phương pháp như sau:

V.2.1 Error Sum of Squares (SSE)

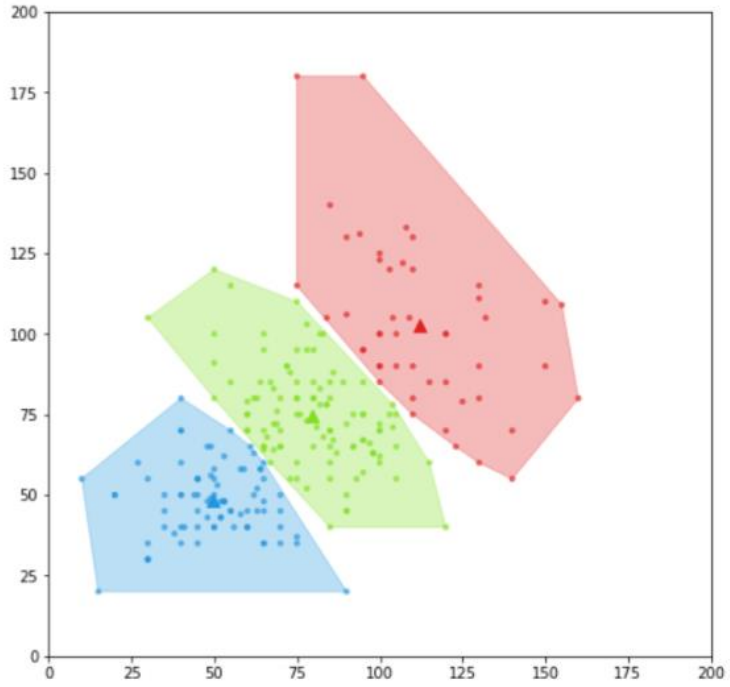
SSE là tổng bình phương khoảng cách từ một điểm tới tâm của cụm chứa điểm đó.

$$SSE = \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Hình V.1. Công thức tính SSE

SSE thể hiện sự khác biệt giữa các đối tượng trong cùng một cụm dữ liệu. Cùng một giá trị n_clusters, model nào cho SSE nhỏ hơn thì tốt hơn.

V.2.2 Biểu đồ đa giác lồi



Hình V.2. Biểu đồ đa giác lồi

Mỗi đa giác lồi là hình ảnh một cụm dữ liệu được biểu diễn trên 2 thuộc tính. Khoảng trống giữa hai đa giác bất kỳ càng lớn càng tốt.

V.2.3 Accuracy (Acc)

Sau khi phân cụm thì nhãn của các cụm không phụ thuộc vào nhãn trong data, nên có khả năng thuật toán đã phân cụm tốt nhưng **nhãn không tương ứng** với nhãn trong data. Ví dụ như sau.

```
clusters_labels: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
          truth: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Hình V.3. Kết quả khi dự đoán

Vì vậy, ta cần **gán lại nhãn** của các cụm. Việc gán lại nhãn của các cụm thì đơn giản ta **hoán đổi giá trị (swap)** nhãn của các cụm, hoán đổi hết các trường hợp và tìm trường hợp có Accuracy cao nhất.

```
clusters_labels: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2 2 2
 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2 1 2
 2 1]
          truth: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Hình V.4. Hoán đổi nhãn để tìm bộ nhãn khớp nhất với bộ nhãn thực tế

Cách tính Accuracy khi có clusters_lebels và truth thì đơn giản bằng cách lấy thương (số nhãn đúng)/(tổng số nhãn).

V.3 Tiến hành gridsearch

V.3.1 Tham số sử dụng

```
linkages = ['ward', 'average', 'complete','single']
affinities = ['euclidean', 'l1', 'l2', 'manhattan', 'cosine', 'precomputed']
distance_theshords = [None, 0.001, 0.01, 0.1, 1, 10]
```

V.3.2 Ví dụ sử dụng gridsearch với bộ dữ liệu Iris.csv

Kết quả:

```
model 0 ... 0.8933333333333333
model 1 ... ERROR parameters
model 2 ... ERROR parameters
model 3 ... ERROR parameters
model 4 ... ERROR parameters
model 5 ... ERROR parameters
model 6 ... 0.9066666666666666
model 7 ... 0.9
model 8 ... 0.9066666666666666
model 9 ... 0.9
model 10 ... 0.66
model 11 ... ERROR parameters
model 12 ... 0.84
model 13 ... 0.8933333333333333
model 14 ... 0.84
model 15 ... 0.8933333333333333
model 16 ... 0.84
model 17 ... ERROR parameters
model 18 ... 0.68
model 19 ... 0.6733333333333333
model 20 ... 0.68
model 21 ... 0.6733333333333333
model 22 ... 0.66
model 23 ... ERROR parameters
model 24 ... ERROR parameters
...
------------------FINAL MODEL------------------
linkage : average
affinity : euclidean
distance_threshord : None
acc : 0.9066666666666666
```

Có những bộ tham số cho kết quả là "**ERROR parameters**" là do thư viện **sklearn** quy định một số tham số **bắt buộc** phải dùng với nhau. Ví dụ, **linkage = "ward"** thì bắt buộc **affinity = "euclidean"**, nếu affinity được truyền vào giá trị khác thì sẽ lỗi.

Các tham số tốt nhất cho bài toán lần lượt là **linkage = "average", affinity = "euclidean", distance_threshord = None**, và Accuracy cao nhất đạt được là 0.906.

V.4 Kết quả

Bộ dữ liệu Iris.csv

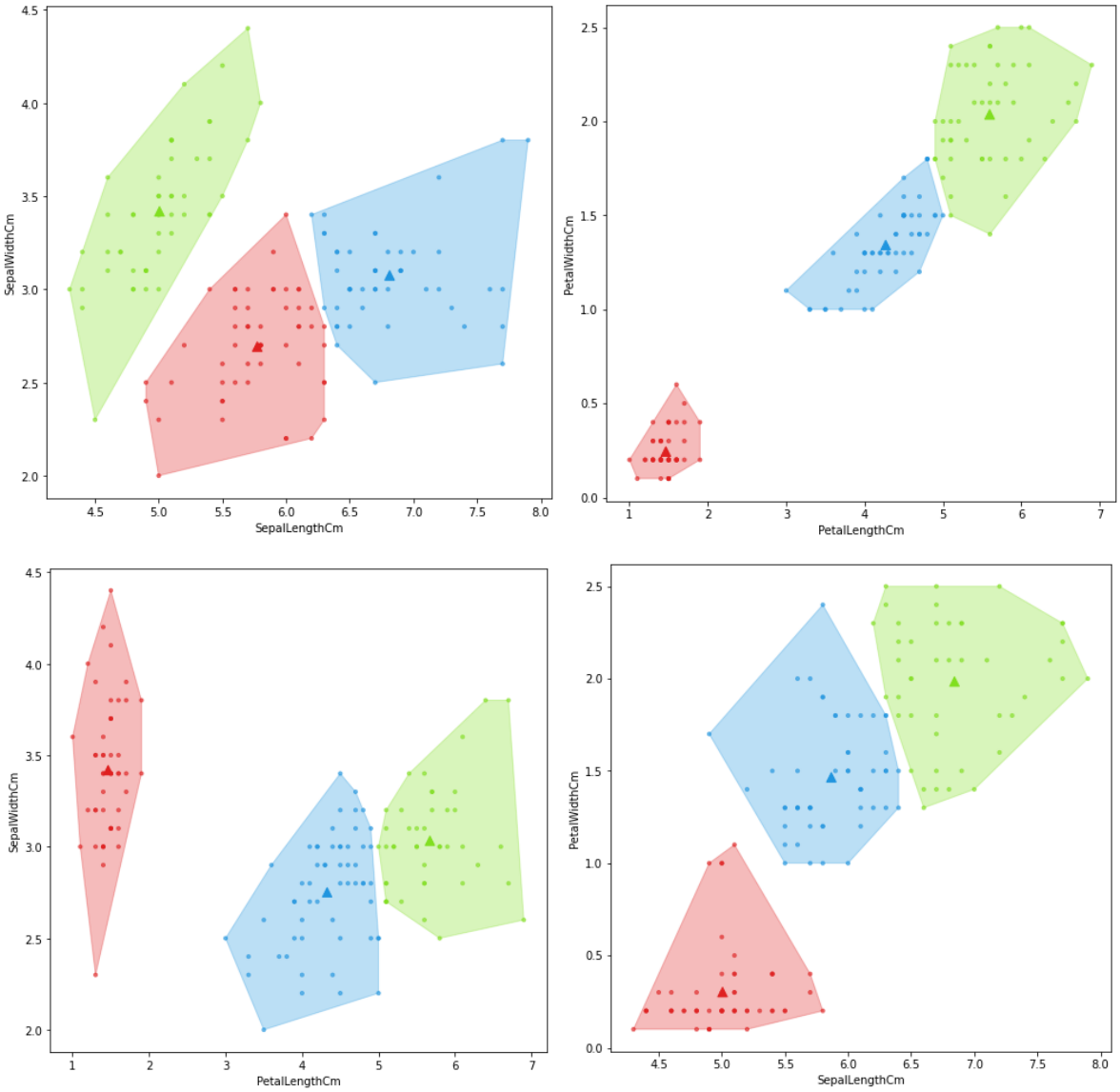
Bộ tham số: linkage = "*average*", affinity = "*euclidean*", distance_threshord = *None*.

Thời gian chạy thuật toán: 2.99 ms

Accuracy: 0.9067

SSE: [15.24; 42.26; 22.03]

Biểu đồ đa giác lồi:



Nhận xét:

Đây là một kết quả tốt:

- Acc khá cao 0.9067.
- SSE là nhỏ so với tập giá trị của các thuộc tính.
- Biểu thức đa giác lồi khi chiếu lên các cặp thuộc tính thì các đa giác không bị đè lên nhau.

Kết quả này đạt được là do bộ dữ liệu đã sạch, hầu như không cần xử lý; lựa chọn đúng giá trị của siêu tham số nhờ vào gridsearch.

Bộ dữ liệu Bank.csv

Bộ tham số: linkage = "*single*", affinity = "*euclidean*", distance_threshord = *None*.

Thời gian chạy thuật toán: 737.72 ms

Accuracy: 0.7667

SSE: [3.73; 7802.73]

Biểu đồ đa giác lồi:

Do bộ dữ liệu có 14 thuộc tính mà chạy lần lượt trên colab không tìm thấy biểu đồ nào có 2 đa giác lồi đè lên nhau. Vì vậy nhóm sẽ không chèn ảnh đồ thị vào phần này.

Nhận xét:

Khó khăn để nhận xét kết quả như thế nào, tuy nhiên em nhận xét kết quả này là trung bình, không tốt. Vì:

- Acc không cao 0.7667.
- SSE ở 2 cụm có sự chênh lệch rất lớn; SSE ở cụm thứ 2 cao hơn rất nhiều so với tập giá trị của các thuộc tính ([-1;1]).
    - Điều này cho thấy cụm 2 rất lớn và cụm 1 rất bé.

VI. So sánh với các thuật toán phân cụm khác

Trong phần này, nhóm sẽ so sánh kết quả phân cụm của thuật toán Hierachical Clustering và thuật toán Kmeans.

Bộ dữ liệu Iris.csv

Thuật toán	Thời gian chạy thuật toán	Acc	SSE
Hierarchical	2.99 ms	0.9067	[15.24; 42.26; 22.03]
Kmeans	50.83 ms	0.8933	[15.24; 39.82; 23.88]

Bộ dữ liệu Bank.csv

Thuật toán	Thời gian chạy thuật toán	Acc	SSE
Hierarchical	737.72 ms	0.7667	[3.73; 7802.73]
Kmeans	135.6 ms	0.6970	[2766.30; 4041.84]

Nhận xét

- Khi thực hiện bộ dữ liệu nhỏ thì tốc độ chạy thuật toán của Hierarchical là nhanh hơn (gấp 19 lần Kmeans nhưng do thời gian nhỏ nên chênh lệch sẽ không lớn). Tuy nhiên khi bộ dữ liệu trở nên lớn thời gian chạy thuật toán của Hierarchical lại trở nên rất lớn, gấp gần 6 lần Kmeans (lúc này chênh lệch đã trở nên lớn hơn rất nhiều).
- Độ chính xác (Acc) của Hierarchical trong cả 2 bộ dữ liệu là nhỉnh hơn (một phần lý do là việc sử dụng gridsearch với thuật toán Hierarchical trong khi chỉ sử dụng các tham số mặc định đối với thuật toán Kmeans).
- Sự đồng đều các cụm ở Kmeans nổi trội hơn so với Hierarchical. Trong bài toán phân cụm, theo em nghĩ việc các cụm đồng đều sẽ quan trọng hơn là độ chính xác (Acc).

VII. Ưu, nhược điểm

Ưu điểm:

- Đơn giản, dễ hiểu, tương đối hiệu quả.
- Các đối tượng tự động gán vào nhóm.
- Thường được tối ưu cục bộ.

Nhược điểm:

- Các thuộc tính không phải số.
- Cần xác định số nhóm (n_clusters) trước.
- Phụ thuộc vào việc chọn các nhóm đầu tiên.
- Gặp vấn đề khi các nhóm có kích thước, mật độ khác nhau hoặc hình dáng không phải hình cầu.
- Nhạy cảm với dữ liệu nhiễu.
- Thời gian thực thi khi giải quyết các bộ dữ liệu lớn cao hơn rất nhiều so với Kmeans.

VIII. Tài liệu tham khảo

- https://ndquy.github.io/posts/thuat-toan-phan-cum-kmeans/
- https://www.datanovia.com/en/lessons/agglomerative-hierarchical-clustering/
- https://towardsdatascience.com/visualizing-clusters-with-pythons-matplolib-35ae03d87489
- https://www.youtube.com/watch?v=QXOkPvFM6NU
- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html
- https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html
- https://stackabuse.com/hierarchical-clustering-with-python-and-scikit-learn/
- https://l.facebook.com/l.php?u=https%3A%2F%2Fwww.benchpartner.com%2Fq%2Fdifferentiate-agglomerative-and-divisive-hierarchical-clustering%3Ffbclid%3DIwAR0QDrI-F6IUpm9gRZiRYt8R8MO-INmhgsoRCN98M0f50dUMyDVacqOdl7w&h=AT11apW4BbQBNByDtbDoHaV-_WecGW6tG1iSw2gjHjrINUumh9PF9KVWX1AIjIa0XI1TOxcuFDCuD6qE-V94GCCC39KoMFRmEUOOVD09TrqLXGNEz8hinRNl7oWfDLzi9WDUTtq-UYZnelU