ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

MÔN HỌC MÁY VÀ ỨNG DỤNG

BÁO CÁO BÀI TẬP LỚN

TRIỂN KHAI GIẢI THUẬT PHÂN CỤM GIA TĂNG LEADER VÀ ÁP DỤNG VÀO PHÂN CỤM TẬP DỮ LIỆU IRIS

Giảng viên hướng dẫn: PGS.TS. Dương Tuấn Anh

Học viên: Trương Văn Thịnh - 2270092

Đinh Thanh Phong - 2270243

TP. Hồ Chí Minh, tháng 12 năm 2023

Mục lục

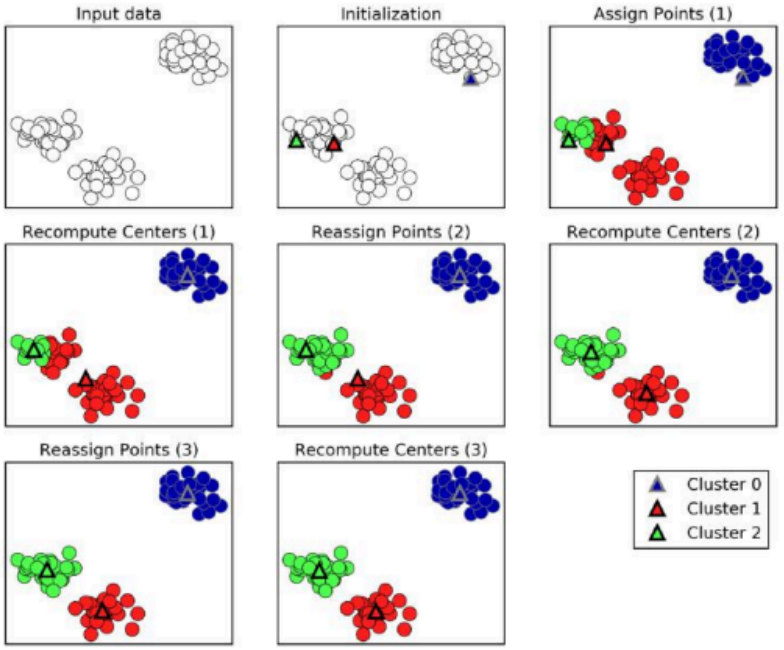
Danh mục hình ảnh và bảng

1. GIỚI THIỆU

1.1 Giới thiệu chung về phân cụm và thuật toán phân cụm gia tăng Leaders

1.1.1 Giới thiệu chung về phân cụm

Phân cụm là một phương pháp phân loại dữ liệu vào các nhóm dựa trên sự tương đồng giữa chúng. Mục tiêu của các giải thuật phân cụm là nhằm chia dữ liệu thành các cụm sao cho các điểm nằm trong cùng một cụm thì sẽ có giá trị các thuộc tính là tương tự nhau, và các điểm ở các cụm khác nhau sẽ có các thuộc tính giá trị các thuộc tính là khác biệt nhau. Tương tự các thuật toán phân loại, các thuật toán phân cụm sẽ gán (hoặc dự đoán) một giá trị độ đo cho mỗi điểm dữ liệu, sau đó chỉ ra điểm đó thuộc về cụm nào. Tuy nhiên, đối với thuật toán phân loại, chúng ta được biết trước các nhãn, còn thuật toán phân cụm, chúng ta không được biết trước các nhãn của dữ liệu[3].

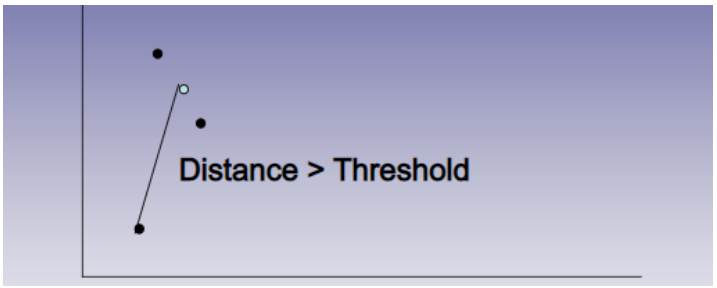


Hình 1. Minh hoạ về phân cụm

1.1.2 Thuật toán phân cụm gia tăng Leaders

Thuật toán phân cụm gia tăng là một dạng đặc biệt của phân cụm, được thiết kế để xử lý các tập dữ liệu có tính chất động, thay đổi theo thời gian hoặc cần được cập nhật

liên tục. Trong môi trường này, dữ liệu mới có thể được thêm vào hoặc dữ liệu cũ có thể bị thay đổi. Điều này đòi hỏi thuật toán phân cụm phải có khả năng tương thích với sự thay đổi này mà không cần phải tính toán lại toàn bộ dữ liệu.



Hình 2. Minh hoạ về thuật toán Leaders

phân cụm gia tăng dựa trên giả định rằng có thể xem xét từng mẫu một và gán chúng cho các cụm hiện có. Một mẫu dữ liệu mới được gán cho một cụm mà không ảnh hưởng đáng kể đến các cụm hiện có [1].

Thuật toán phân cụm gia tăng Leaders cung cấp một phương tiện để phân cụm một tập hợp các điểm dữ liệu. Không giống như nhiều thuật toán phân cụm khác, thuật toán phân cụm gia tăng không phân cụm theo chỉ định số lượng cụm từ người dùng, mà thuật toán dựa vào một mức ngưỡng (khoảng cách giữa một mẫu tới các leader) để làm cơ sở để phân cụm. Vì vậy để có được số mong muốn, người dùng điều chỉnh mức ngưỡng thích hợp.

Tuy nhiên, thuật toán phân cụm gia tăng phụ thuộc vào thứ tự xuất hiện của mẫu, leader trong quá trình phân cụm [1].

1.2 Mục tiêu của báo cáo

Mục tiêu của báo cáo là:

- Hiểu rõ về thuật toán phân cụm gia tăng Leaders.
- Triển khai hiện thực thuật toán phân cụm gia tăng Leaders.
- Áp dụng thuật toán vào một bài toán phân cụm với tập dữ liệu về hoa Iris.
- Thực hiện một số thử nghiệm để khảo sát thuật toán và so sánh phiên bản hiện thực với các phiên bản từ thư viện học máy (Sklearn).
- Đánh giá và so sánh kết quả

2. ĐỘ ĐO KHOẢNG CÁCH

Khoảng cách là một giá trị số học miêu tả 2 điểm cách nhau bao xa. Hàm khoảng cách giữa từ điểm x đến y là một hàm số d(x, y). Hàm này được coi là metric nếu nó thoả mãn các tính chất bao gồm một số tính chất nổi bật sau:

• Giá trị của hàm d(x, y) luôn lớn hơn hoặc bằng 0.

• Giá trị của hàm d(x, y) bằng 0 khi và chỉ khi x = y.

• Giá trị của khoảng cách từ x đến y bằng khoảng cách từ y đến x.

• Bất đẳng thức tam giác: Xét thêm một điểm thứ 3 z, khoảng cách từ x đến y luôn bé hơn tổng khoảng cách từ x đến z và từ y đến z.

Giả sử các điểm dữ liệu là một vector trong không gian nhiều chiều. Gọi:

• xi là một điểm dữ liệu với p tính năng (feature): (xi1,xi2,, ..., xip).

• n là tổng số điểm dữ liệu và do đó i = 1, 2, ..., n.

Có rất nhiều cách để định nghĩa hàm d(x, y), được chia thành các họ khác nhau. Ta sẽ xem xét một số khoảng cách thông dụng.

2.1 Khoảng cách Euclid

Khoảng cách Euclid (Euclidian distance) là trong những cách đo khoảng cách giữa các điểm dữ liệu liên tục thường được sử dụng. Khoảng cách Euclid giữa hai điểm dữ liệu xi và xj được tính bởi công thức:

$$d(x_i, j_i) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})} \quad (1)$$

2.2 Khoảng cách Manhattan

Khoảng cách Manhattan hay còn gọi là khoảng cách trong thành phố là một dạng khoảng cách giữa hai điểm trong không gian Euclid và được tính bởi công thức sau:

$$d(x_i, j_i) = \sum_{k=1}^{p} |(x_{ik} - x_{jk})| \quad (2)$$

3. XÂY DỰNG THUẬT TOÁN LEADERS

3.1 Nguyên tắc hoạt động

Thuật toán phân cụm Leaders khởi đầu quá trình của mình bằng cách chọn ra mẫu dữ liệu đầu tiên từ tập dữ liệu và gán nó làm leader đầu tiên của một cụm. Leader này không chỉ là một biểu diễn đại diện cho cụm mà còn được sử dụng để mở rộng cụm theo thời gian. Khi qua mỗi mẫu dữ liệu tiếp theo, thuật toán quyết định xem mẫu đó có thể tham gia vào một cụm hiện tại nào đó hay không.

Nếu mẫu dữ liệu mới có thể được hòa nhập vào một cụm đã tồn tại, nó sẽ được thêm vào cụm đó và cập nhật lại leader của cụm đó để đảm bảo tính đại diện. Ngược lại, nếu không có cụm nào phù hợp, một cụm mới sẽ được tạo ra, và mẫu dữ liệu mới sẽ được xác định làm leader của cụm mới này.

Quá trình này lặp đi lặp lại cho đến khi tất cả các mẫu dữ liệu đã được xem xét. Kết quả cuối cùng là một tập hợp các cụm, mỗi cụm có một leader là một trong các mẫu dữ liệu trong tập dữ liệu, đồng thời mỗi mẫu dữ liệu trong tập dữ liệu được gán vào một cụm cụ thể, tạo ra sự phân loại và tổ chức dữ liệu một cách hiệu quả.

Các bước cụ thể [1]:

Khởi tạo giá trị và ký hiệu:

- Lựa chọn giá trị ngưỡng T
- Số cụm C_i
- Danh sách leader L_i
- Mẫu dữ liệu thứ j (P_j) trong tập dữ liệu P.

Bước 1: Khởi tạo leader đầu tiên L_1 là mẫu dữ liệu đầu tiên P_1 và phân P_1 vào cụm C_1. Thêm nó vào danh sách các leader L.

Bước 2: Từ mẫu thứ hai (P_j, $j \geq 2$) trở đi:

- Tính khoảng cách từ điểm P_j đến các leader trong danh sách L.
- Tìm leader gần nhất.
- Nếu khoảng cách từ mẫu P_j đến leader gần nhất nhỏ hơn ngưỡng T, thì thêm mẫu P_j vào cụm của leader gần nhất đó. Nếu không thoả mãn với điều khiện, thì thiết lập i=i+1 và thêm mẫu P_j vào một cụm mới C_i. Thiết lập leader L_i là P_j.
- Lặp lại bước 3 cho tới khi các mẫu đã được phân và các cụm.

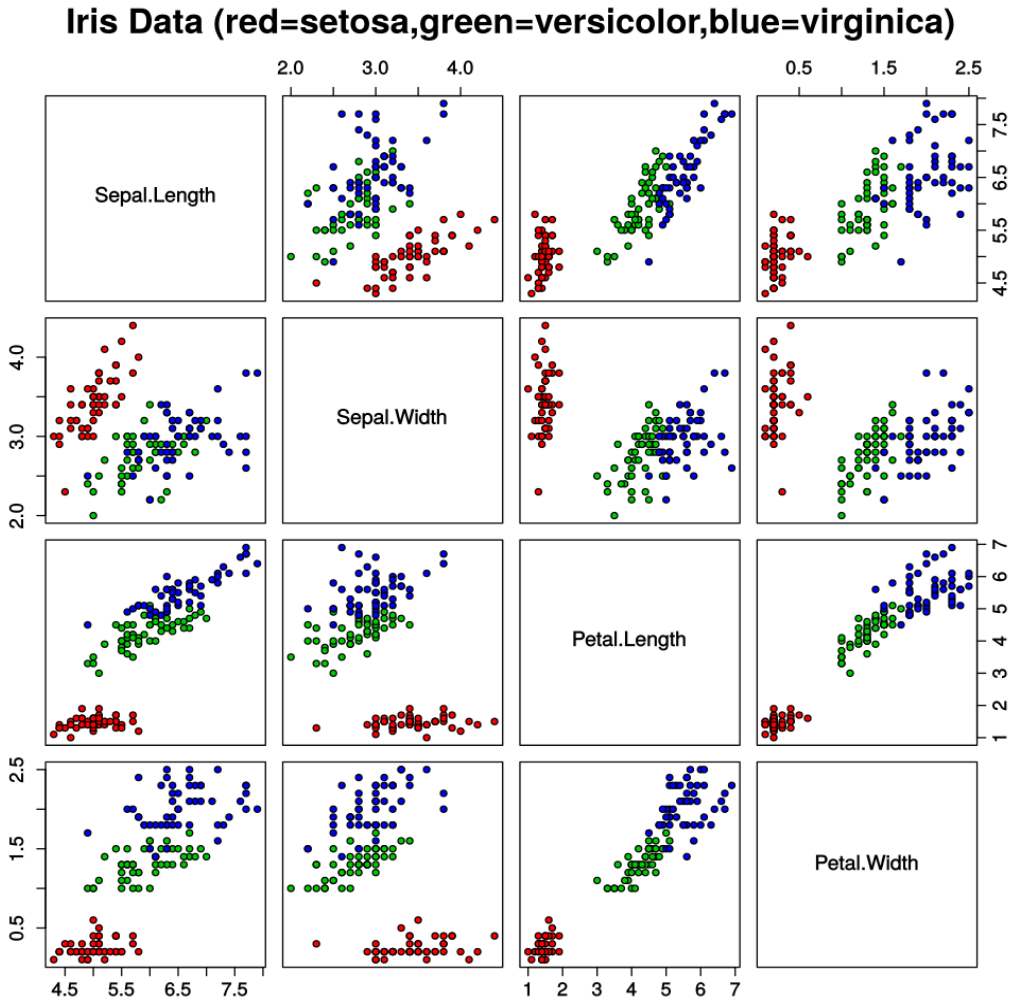
3.2. Phân tích độ phức tạp

Dựa vào cách tính toán, độ phức tạp của thuật toán Leaders có thể xác định là O(N^2), với N là số lượng điểm dữ liệu. Điều này xuất phát từ việc mỗi điểm dữ liệu cần được so sánh với tất cả các điểm khác trong quá trình phân cụm, tạo ra tổng số lần so sánh là dạng bậc hai của số lượng điểm dữ liệu, tức là O(N^2).

4. ÁP DỤNG THUẬT TOÁN LEADERS TRONG BÀI TOÁN phân CỤM TRÊN TẬP DỮ LIỆU IRIS

4.1 Định nghĩa bài toán

Bộ dữ liệu Iris là bộ dữ liệu phổ biến trong việc phân cụm và phân lớp trong lĩnh vực học máy. Bộ dữ liệu Iris, chứa thông tin về các loài hoa Iris. Mục tiêu của thuật toán trong ngữ cảnh này là phân cụm các mẫu dữ liệu thành các nhóm tương ứng, dựa trên đặc điểm và sự tương đồng giữa chúng.



(Hình ảnh phân bố dữ liệu của loài hoa Iris)

Các mẫu dữ liệu về loài hoa Iris trong bộ dữ liệu được phân thành các nhóm rời rạc, với điểm dữ liệu trong cùng một loài có xu hướng tương đồng và gần nhau hơn so với các

loài khác. Mục tiêu của thuật toán Leaders được áp dụng để hiệu quả phân cụm, với mỗi nhóm có một leader đại diện.

Để đánh giá hiệu suất của thuật toán, kết quả của nó được so sánh với nhãn mẫu đã biết và đánh giá bằng phương pháp trực quan từ hình ảnh. Việc này giúp xác định độ chính xác và sự hiệu quả của thuật toán Leaders trong việc phân cụm dữ liệu Iris, cung cấp thông tin chi tiết về khả năng phân loại và phân cụm trên bộ dữ liệu này. Điều này là quan trọng để đảm bảo thuật toán có thể hiệu quả trong việc xử lý và phân tích dữ liệu thực tế từ bộ dữ liệu Iris.

4.2 Mô tả tập dữ liệu Iris

Bộ dữ liệu về hoa Iris được thu thập bởi Edgar Anderson – nhà thực vật học, sau đó được thống kê và rút gọn lai bởi Ronald Aylmer Fisher. Bộ dữ liệu về hoa iris bao phân 50 mẫu từ mỗi loài trong số ba loài *Iris* (*Iris setosa* , *Iris virginica* và *Iris versicolor*) [2].

Bốn thuộc tính về hình dạng được đo từ mỗi mẫu (đơn vị cm):

- Chiều dài của lá đài.
- Chiều rộng của lá đài.
- Chiều dài của cánh hoa.
- Chiều rộng của cánh hoa

Một thuộc tính còn lại là tên của loài Iris: Iris Setosa, Iris Versicolor, Iris Virginica.

Ở đây, ta sử dụng dữ liệu cho bài toán phân cụm, vì vậy chỉ sử dụng bốn thuộc tính về hình dạng để thực hiện cho việc phân cụm và sử dụng thuộc tính loài Iris để đánh giá kết quả phân cụm.

Hình . Ba loài hoa Iris

4.3 Hiện thực thuật toán Leader

Nhóm sử dụng ngôn ngữ lập trình python để hiện thực thuật leader và áp dụng thực hiện toán cho việc phân cụm.

4.3.1 Đọc và khai phá dữ liệu

- Tải bộ dữ liệu Iris từ bộ dataset của Sklearn.

- Loại bỏ đặc trưng “target" chỉ sử dụng 4 đặc trưng phân sepal length (cm), sepal width (cm), petal length (cm), petal width (cm).

```
# Remove "target" column to get data for clustering
data = data_org.drop(columns=['target'])
```

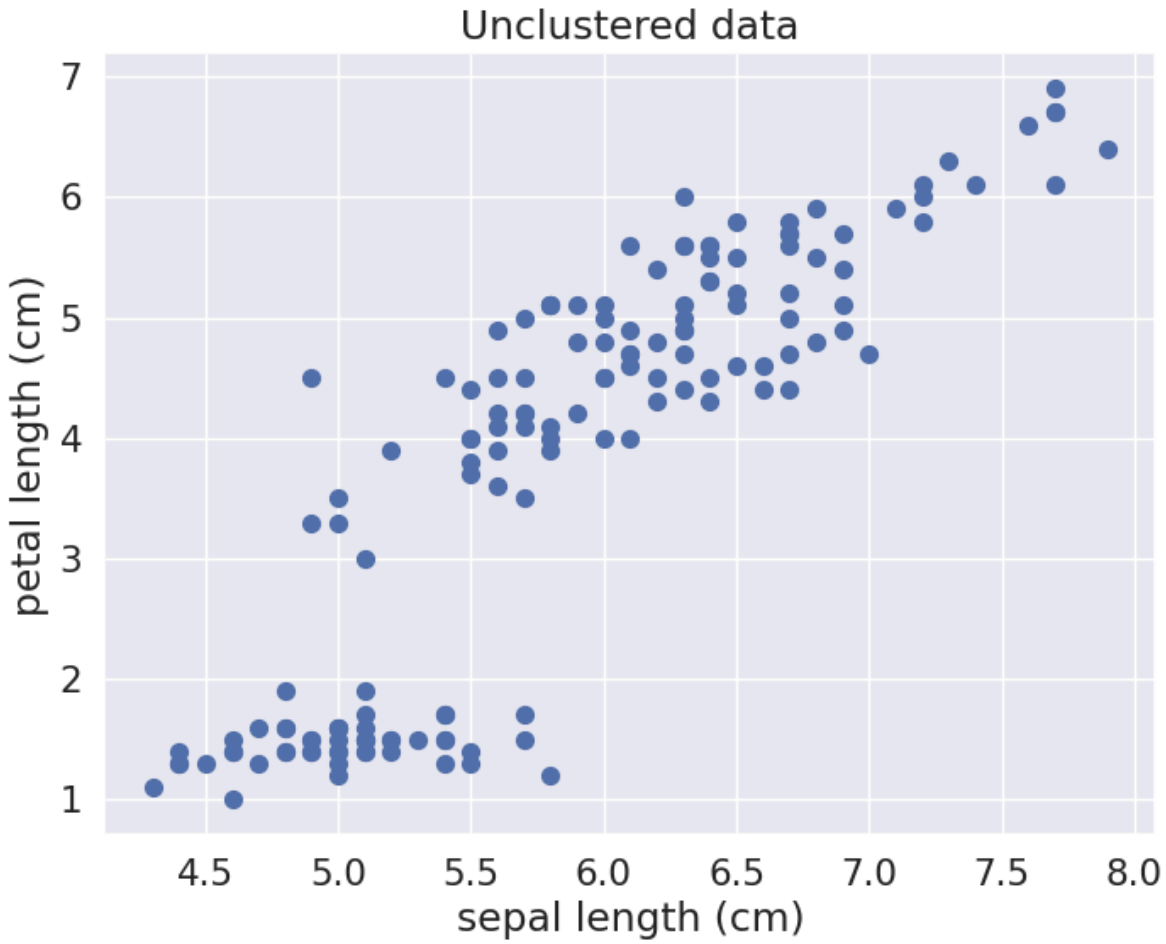
```
# Preview data used for clustering
print('Dataframe shape:', data.shape)
data.head(5)
```

```
Dataframe shape: (150, 4)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

- Vẽ biểu đồ minh hoạ bộ dữ liệu khi chưa phân cụm. Do tập dữ liệu có 4 thuộc tính, để đơn giản cho việc trực quan, ta lựa chọn 2 thuộc tính đại diện để thực hiện trực quan hoá dữ liệu là "sepal length (cm)" và "petal length (cm)".

```
plt.figure(figsize=(8,6))
plt.scatter(x=data['sepal length (cm)'], y=data['petal length (cm)'], s=50)
plt.xlabel('sepal length (cm)')
plt.ylabel('petal length (cm)')
plt.title('Unclustered data')
plt.show()
```

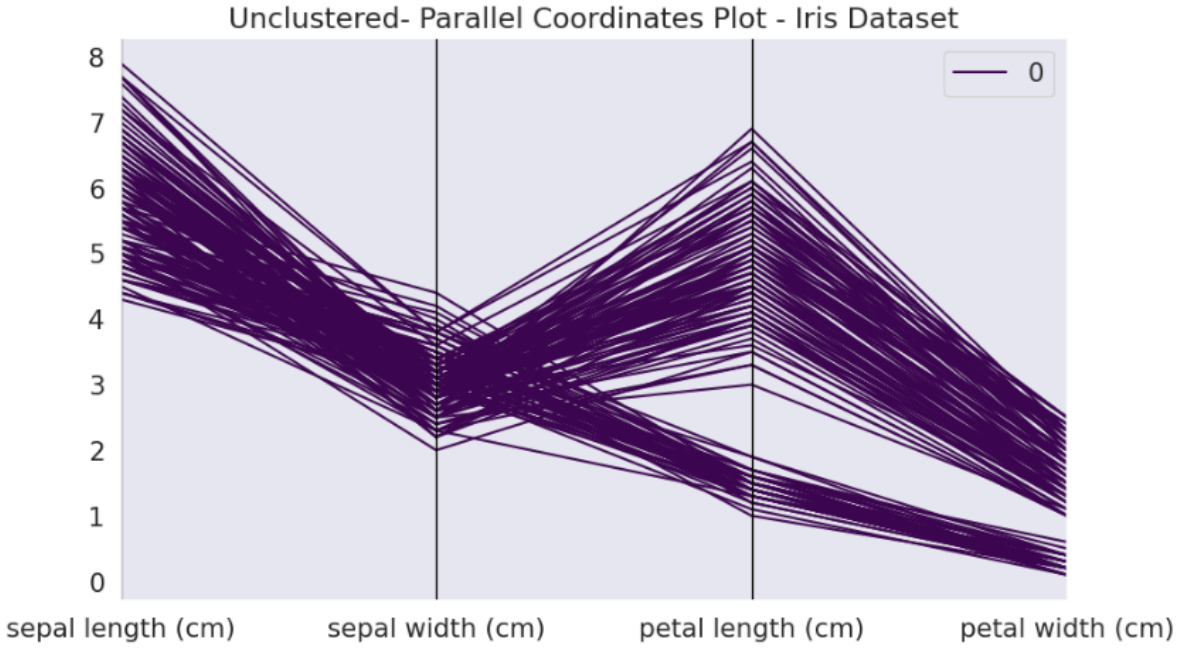


Hình 4. Minh hoạ bộ dữ liệu khi chưa phân cụm

- Hoặc ta có thể sử dụng phương pháp toạ độ song song (Parallel coordinates), là một phương pháp trực quan hóa dữ liệu thống kê và đa biến.

```
from pandas.plotting import parallel_coordinates
from sklearn.datasets import load_iris

# Load Iris dataset
data_org['target'] = 0
# Parallel coordinates plot
plt.figure(figsize=(10, 6))
parallel_coordinates(data_org, 'target', colormap='viridis')
plt.title('Unclustered- Parallel Coordinates Plot - Iris Dataset')
plt.show()
```



4.3.2 Hiện thực thuật toán phân cụm Leader

Tạo class *LeaderClustering* có nhiệm vụ thực hiện giải thuật phân cụm Leader. Trong class *LeaderClustering* sẽ xây các hàm để hiện thực giải thuật leader:

1. *__init__()*: Hàm khởi tạo có nhiệm vụ khởi tạo khoảng cách ngưỡng.
2. *euclidean()*: Hàm tính khoảng cách bằng công thức euclidean.
3. *leader()*: Hiện thực giải thuật leader và thực hiện phân cụm trên tập dữ liệu đã cho.
4. *appendSamples()*: Hiện thực cách gia tăng của thuật toán leader. Hàm sẽ thực hiện phân cụm các điểm dữ liệu mới dựa các cụm đã được phân cụm từ bộ dữ liệu ban đầu.

4.3.2.1 __init__(self, threshold=2) - Hàm khởi tạo đối tượng và ngưỡng khoảng cách

Hàm khởi tạo có một tham số là *threshold*, tham số này khởi tạo giá trị khoảng cách ngưỡng từ từ một điểm mẫu đến leader của cụm. Mặc định nếu không được khởi tạo, *threshold* sẽ bằng 2.

```
def __init__(self, threshold=2):
    """
    Initialize the LeadersClustering object.

    Parameters:
    - threshold (float): The distance threshold for clustering.
    """
    self.threshold = threshold
```

4.3.2.2 euclidean(self, v1, v2) - Hàm tính khoảng cách

Nhóm quyết định sử dụng khoảng cách Euclidean để tính toán khoảng cách giữa các mẫu và leader. Hàm *euclidean* có hai tham số, tương ứng với hai mẫu. Trong vận dụng của bài toán này, *v1* sẽ đại diện cho mẫu leader, *v2* là mẫu dữ liệu. Hàm trả về khoảng cách được dựa trên tính toán các chiều tuần tự của *v1*, *v2*.

```
def euclidean(self, v1, v2):
    """
    Calculate the Euclidean distance between two vectors.

    Parameters:
    - v1 (numpy array): First vector.
    - v2 (numpy array): Second vector.

    Returns:
    - float: Euclidean distance between v1 and v2.
    """
    return np.sqrt(np.sum((v1 - v2) ** 2))
```

4.3.2.3 leader(self, dataSet) - Giải thuật Leader

Hàm *leader* khai báo các thuộc tính:

- *self.DataSet*: lưu trữ tập dữ liệu
- *self.leaderList*: danh số các leader, mỗi leader là tương ứng với một mẫu.

- self.clusters: mảng ghi lại các cluster mà các mẫu thuộc về trong quá trình phân cụm.

Hàm *leader* sẽ hiện thực giải thuật Leader, hàm có một tham số là tập dữ liệu được gán vào biến *dataSet*. Hàm sẽ tiến hành phân cụm trên bộ dữ liệu đầu vào này dựa trên nguyên lý của thuật toán leader. Đầu ra là kết quả phân cụm của từng mẫu, sẽ được ghi vào thuộc tính *self.clusters*.

```
def leader(self, dataSet):
    """
    Perform leaders clustering on the given dataset.

    Parameters:
    - dataSet (pandas DataFrame): The input dataset.

    Returns:
    - clusters (numpy array): Array representing the assigned cluster for each data point.
    - leaderList (list): List of leader points.
    """
    self.dataSet = dataSet
    self.leaderList = [dataSet.iloc[0]]
    self.clusters = np.zeros(len(dataSet), dtype=int)
    self.clusters[0] = 0

    for i in range(1, len(dataSet)):
        distances = np.zeros(len(self.leaderList))
        for index, leader in enumerate(self.leaderList):
            distances[index] = self.euclidean(dataSet.iloc[i], leader)

        # Find the leader index that has the shortest distance to the sample.
        closest_leader_index = np.argmin(distances)

        # Assign sample to cluster of closest leader
        if distances[closest_leader_index] < self.threshold:
            self.clusters[i] = closest_leader_index
        # If not, create new cluster and self assign it into new cluster
        else:
            self.leaderList.append(dataSet.iloc[i])
            self.clusters[i] = len(self.leaderList) - 1
```

4.3.2.4 appendSamples(self, samples) - phân cụm gia tăng cho các mẫu mới:

Thuật toán leader là thuật toán phân cụm gia tăng nên cần có khả năng xử lý loại dữ liệu cập nhật liên tục.

Hàm *appendSample* có một tham số là danh sách các mẫu dữ liệu mới. Sau đó mẫu dữ liệu mới sẽ được thêm vào thuộc tính *self.DataSet* và phân các mẫu dữ liệu mới vào các cụm ở thuộc tính *self.leaderList* đã được khởi tạo ở hàm *leader*.

```
"""
Append new samples to the dataset and assign them to clusters.
Args:
    samples (list): List of new samples to be appended to the dataset.
Returns:
    None
"""
def appendSamples(self, samples):
# Iterate over each row in the input samples
  for sample_row in samples:
    # Add a new row to the dataset with the given samples
    self.dataSet.loc[len(self.dataSet)] = sample_row

    # Convert the new row to a NumPy array
    newPoint = self.dataSet.iloc[-1]

    # Calculate the Euclidean distance between the new point and existing leaders
    distances = {}
    for index, leader in enumerate(self.leaderList):
        distances[index] = self.euclidean(newPoint, leader)

    # Find the index of the leader with the minimum distance
    closest_leader_index = min(distances, key=distances.get)

    # Check if the closest leader is within the threshold distance
    if distances[closest_leader_index] < self.threshold:
        # If within threshold, assign the new point to the closest cluster
        self.clusters = np.append(self.clusters, closest_leader_index)
    else:
        # If outside the threshold, create a new cluster with the new point as the leader
        self.leaderList.append(newPoint)
        self.clusters = np.append(self.clusters, len(self.leaderList) - 1)
```

4.3.3 Áp dụng thuật toán đã hiện thực vào phân cụm

Ta sẽ phân cụm bộ dữ liệu đã được giới thiệu ở mục 4.3.1

- Bước 1: Khởi tạo đối tượng từ class *LeaderClustering* và thông số ngưỡng khoảng cách cho phân cụm.

```
model = LeaderClustering(threshold=2)
```

- Bước 2: Thực hiện phân cụm bằng thực toán leader. Từ đối tượng *model* đã được khởi tạo, sử dụng hàm leader và truyền bộ dữ liệu cần được phân cụm *data* vào hàm *leader*.

```
model.leader(data)
```

- Bước 3: Kiểm tra kết quả phân cụm
    - In kết quả phân cụm bằng lệnh:

```
print(model.clusters)
```

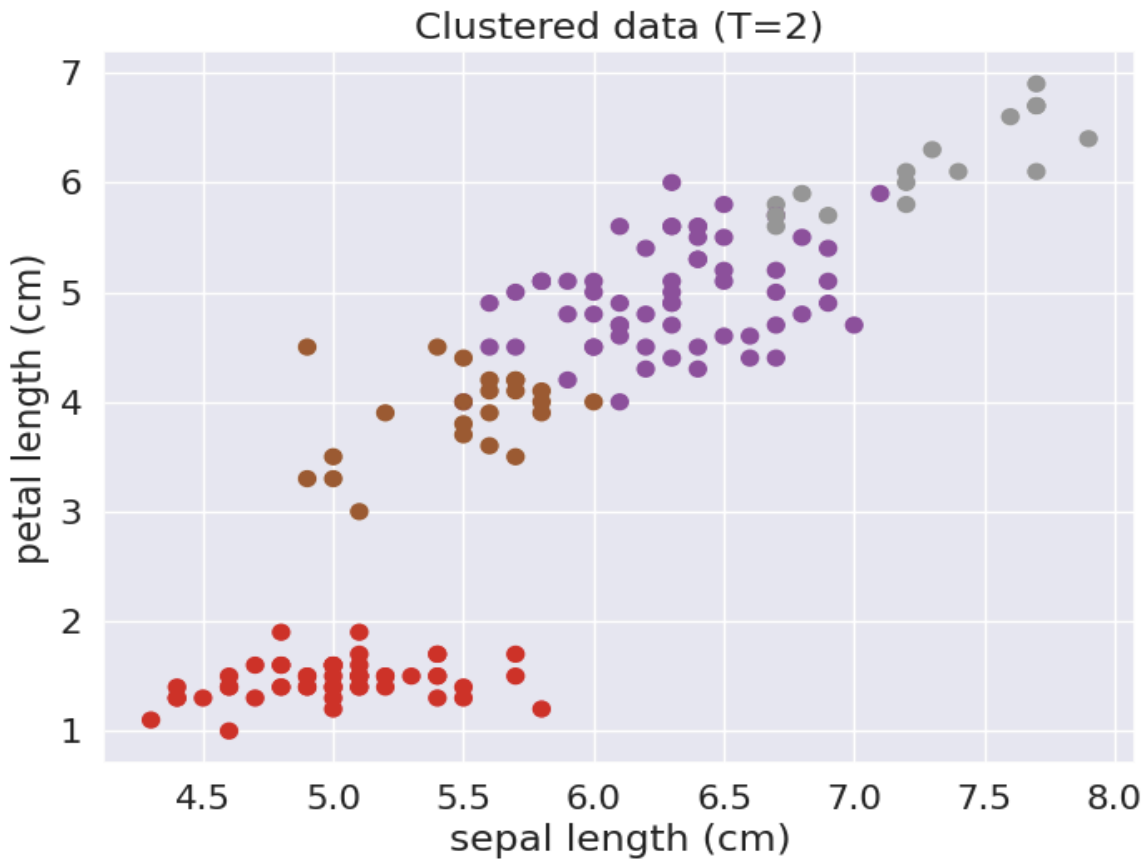
Ta sẽ thu được kết quả bên dưới là một mảng giá trị tương ứng với cụm được phân vào của 150 điểm dữ liệu ban đầu. Số cụm phân được là **4**, tương ứng với **0,1,2,3**.

```
[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 2 1 2 2 1 2 1 2 1 1 2 1 2 1 1 1 1
 1 1 1 1 1 2 2 2 2 1 2 1 1 1 2 2 2 1 2 2 2 2 2 1 2 2 1 1 1 1 1 3 2 3 3 3 1
 1 1 1 1 1 1 3 3 1 3 1 3 1 1 3 1 1 1 3 3 3 1 1 1 3 1 1 1 1 3 1 1 3 3 1 1 1
 1 1]
```

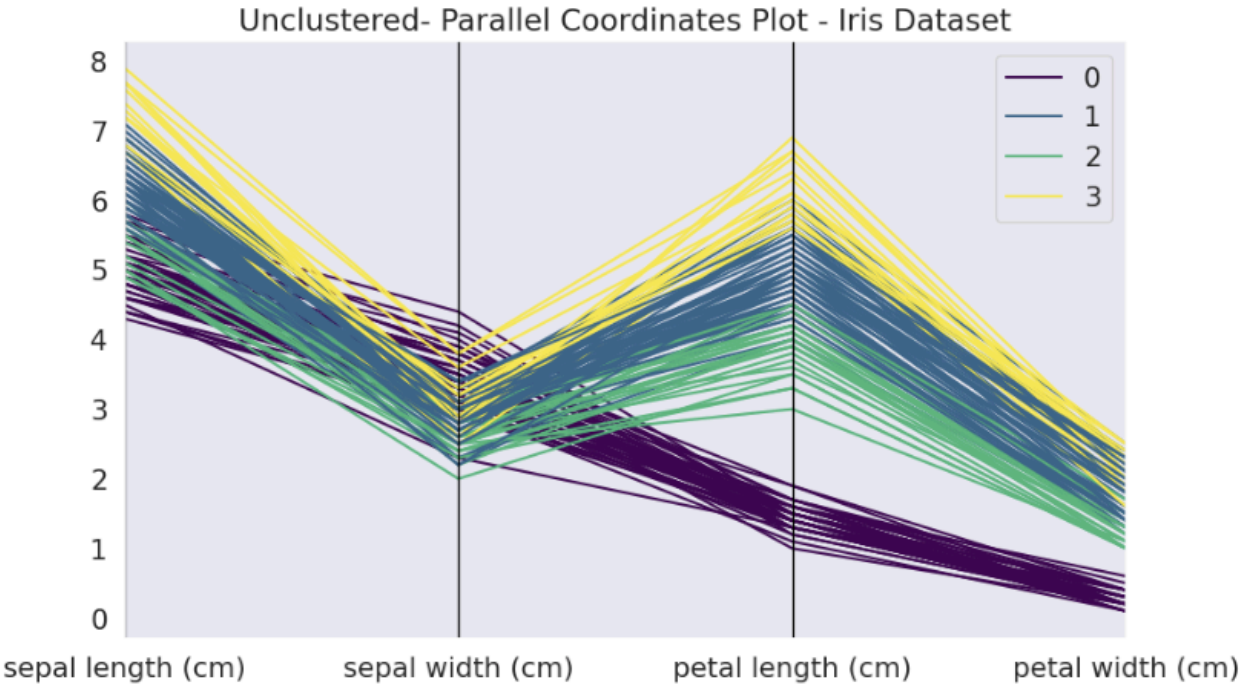
- Trực quan hoá bằng cách vẽ đồ thị kết quả phân cụm, ở đây ta 2 thuộc tính làm đại diện là "sepal length (cm)" và "petal length (cm)" như ở Hình 4.

```
# Plot clusters
plt.figure(figsize=(8,6))
plt.scatter(x=data['sepal length (cm)'], y=data['petal length (cm)'], c=model.clusters, cmap='Set1', s=50)

plt.xlabel('sepal length (cm)')
plt.ylabel('petal length (cm)')
plt.title('Clustered data (T=' + str(THR) + ')')
plt.show()
```



- Trực quan hoá bằng phương pháp toạ độ song song với 4 thuộc tính của tập dữ liệu: sepal length, sepal width, petal length, petal width.



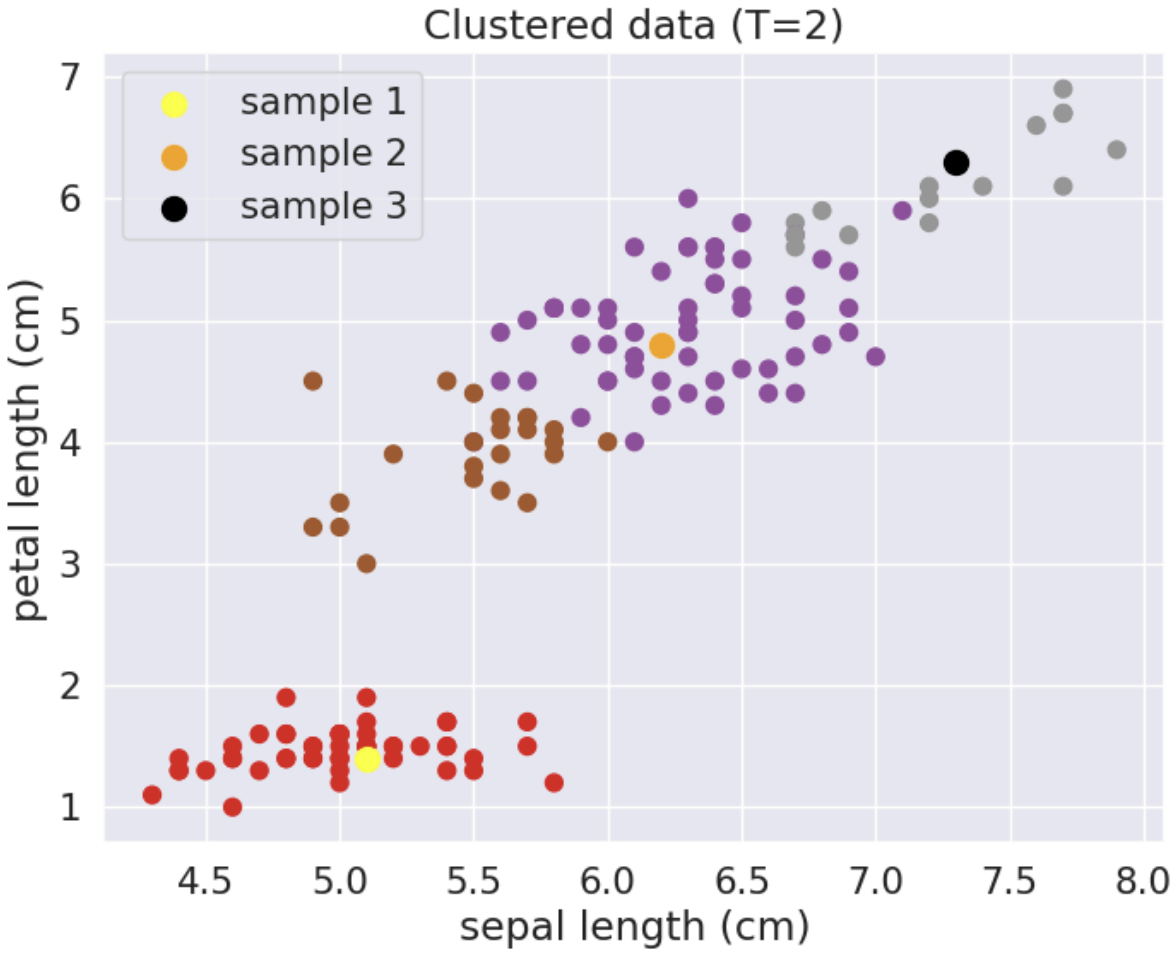
- Bước 4: Thực hiện phân cụm gia tăng. Khởi tạo các điểm dữ liệu mới new_points và từ đối tượng *model* đã được khởi tạo truyền *new_points* hàm *appendSamples* để thực hiện phân cụm cho các điểm dữ liệu mới.
- Khởi tạo 3 điểm dữ liệu mới.

```
new_points = [
    [5.1, 3.5, 1.4, 0.2],
    [6.2, 2.8, 4.8, 1.8],
    [7.3, 3.1, 6.3, 2.5]
]
model.appendSamples(new_points)
```

- In ra kết quả phân cụm, ta nhận thấy 3 điểm dữ liệu mới đã được phân cụm dựa trên các cụm ban đầu. Thuật toán không hiện phân cụm lại toàn bộ dữ liệu và chỉ phân cục cho 3 điểm dữ liệu mới.

```
[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 2 1 2 2 1 2 1 2 1 1 2 1 2 1 1 1 1
 1 1 1 1 1 2 2 2 2 1 2 1 1 1 2 2 2 1 2 2 2 2 2 1 2 2 1 1 1 1 1 3 2 3 3 3 1
 1 1 1 1 1 1 3 3 1 3 1 3 1 1 3 1 1 1 3 3 3 1 1 1 3 1 1 1 1 3 1 1 3 3 1 1 1
 1 1 0 1 3]
```

- Trực quan các điểm dữ liệu mới được phân vào cụm.



5. ĐÁNH GIÁ KẾT QUẢ PHÂN CỤM

5.1 Môi trường đánh giá

Nhóm sử dụng môi trường Google Colab. Google Colab là một nền tảng hợp tác trực tuyến của Google, xây dựng trên Jupyter Notebook, cho phép người dùng làm việc chung với mã nguồn Python. Với khả năng sử dụng miễn phí GPU và TPU từ Google, Colab giúp tăng tốc quá trình machine learning. Tích hợp thư viện và công cụ phổ biến, cũng như khả năng chia sẻ dự án, Colab là công cụ mạnh mẽ cho nghiên cứu và hợp tác trực tuyến trong lĩnh vực khoa học dữ liệu và machine learning.

5.2 So sánh kết quả với thư viện Sklearn

5.2.1 Đánh giá kết quả phân cụm

Sử dụng thuật toán phân cụm K-means để so sánh với kết quả đã thuận toán gia tăng leader đã được hiện thực. Nhóm chúng em chọn cách sử dụng K-means (k=4) của thư viện *Sklearn* làm dữ liệu tham chiếu vì ở bày trình trên với mức ngưỡng T=2, thuật toán leader cho ra số cụm bằng 4.

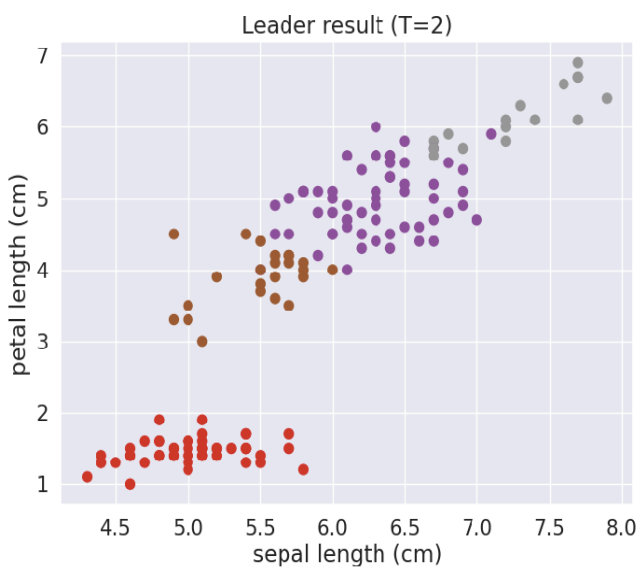
Đoạn mã thử nghiệm để chạy thuật toán từ thư viện sklearn như sau.

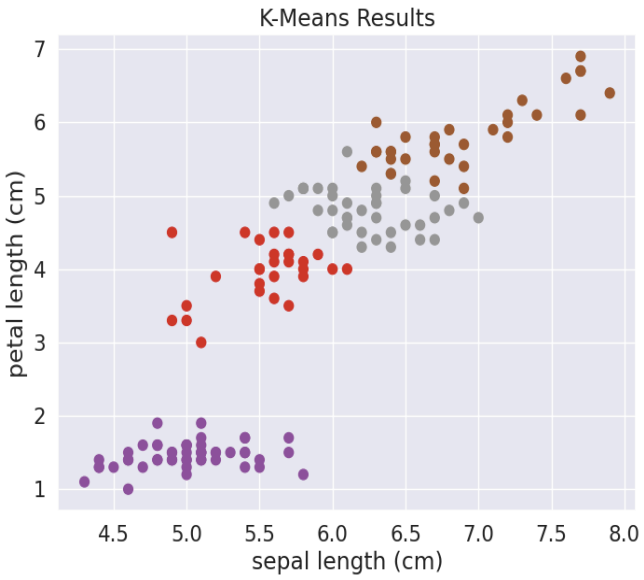
```
from sklearn.cluster import KMeans

kMeansNCluster = 4
km = KMeans(n_clusters=kMeansNCluster, n_init=123, random_state=14)
y_km = km.fit_predict(data)

plt.figure(figsize=(8,6))
plt.scatter(x=data['sepal length (cm)'], y=data['petal length (cm)'], c=km.labels_, cmap='Set1', s=50)
plt.xlabel('sepal length (cm)')
plt.ylabel('petal length (cm)')
plt.title('K-Means Results')
plt.show()
```

Kết quả chạy K-means (k=4) và Leader (T=2):





Nhận xét:

- Về cơ bản, hai thuật toán cho ra kết quả là tương đồng với nhau. Do đó, ta có thể kết luận phiên bản hiện thực thuật toán Leader đã thành công.
- Tuy nhiên, bằng cách quan sat hai đồ thị kết qủa, ta nhận thấy một điểm bị phân cụm sai bởi thuật toán leader (điểm màu tím toạ độ ở (6,7)), trong khi K-means thì không. Sau khi tìm hiểu nguyên nhân nhóm đưa ra một số nhận xét như:
    + Thuật toán K-means không bị điểm nhiễu bởi vì, thuật toán phân cụm dựa trên điểm trọng tâm (centroid) của cụm và điểm trọng tâm này được cập nhập liên tục vì vậy mà không làm mất cân bằng khi phân cụm.
    + Đối với thuật toán Leader, như đã đề cập ở mục 1.1.2, việc phân cụm của thuật toán ảnh hưởng bởi thứ tự xuất hiện của mẫu. Mẫu bị phân cụm sai là bởi vì khi xét mẫu đó, leader đúng của mẫu chưa xuất hiện nhưng mẫu vẫn thoả điều kiện phân cụm (nhở hơn mức khoảng cách ngưỡng).

5.2.1 Đánh giá thời gian thực thi

Thời gian thực thi của thuật toán Leader không chỉ phụ thuộc vào số lượng mẫu và thứ tự của các điểm mà còn phụ thuộc vào giá trị ngưỡng khoảng cách T. Giá trị T nhỏ sẽ dẫn đến việc tạo ra nhiều cụm hơn, điều này có thể làm tăng số lần phải tính toán khoảng cách từ điểm đang xét đến các leader. Ví dụ, nếu T=2, thuật toán có thể tạo ra kết quả với 4 cụm, khá tương đồng với thuật toán với K-Means khi lựa chọns số cụm ban đầu K=4. Trong trường hợp này, thời gian thực thi của thuật toán sẽ được ảnh hưởng theo cách nhất định, và cần được quan sát và đánh giá cẩn thận để hiểu rõ ảnh hưởng của giá trị T đối với hiệu suất của thuật toán.

Thuật toán	Thời gian thực thi
Leaders (T=2)	127 ms
K-Means (K=4)	182 ms

Bảng 1. So sánh thời gian thực thi

Nhìn chung, khi số lượng cụm bằng nhau thì thời gian thực thi của thuật toán Leaders tốt hơn K-Means.

5.3 Thực hiện một số thử nghiệm đánh giá

5.3.1 Thử nghiệm thay thế công thức tính khoảng cách từ euclidean sang manhattan

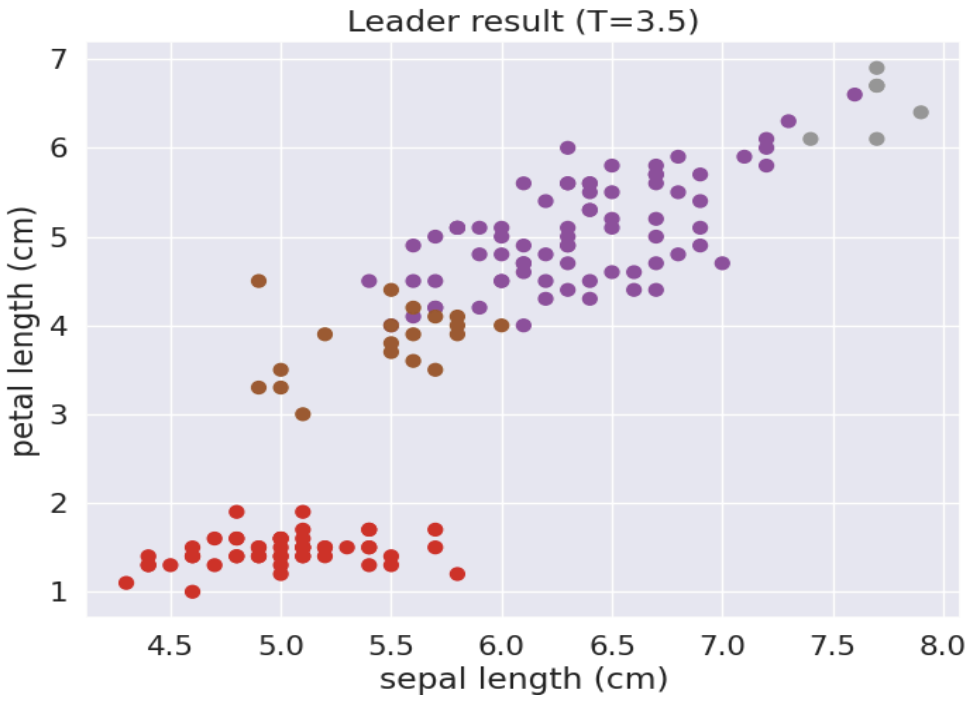
Trong quá trình phát triển và cải thiện thuật toán gia tăng leader, nhóm quan tâm đến việc thử nghiệm một phương pháp thay thế trong việc tính toán khoảng cách giữa các điểm dữ liệu. Thay vì sử dụng khoảng cách Euclidean, ta có thể thực hiện thử nghiệm với phương pháp Manhattan. Bằng cách thử nghiệm các biến thay đổi như loại công thức khoảng cách, nhóm hy vọng có thể đưa ra quyết định thông minh về lựa chọn công thức phù hợp nhất với bộ dữ liệu cụ thể và cải thiện khả năng tổng quan của thuật toán gia tăng leader trong việc xử lý và gom cụm dữ liệu.

```
def manhattan(self, v1, v2):
    """
    Calculate the Manhattan distance between two vectors.

    Parameters:
    - v1 (numpy array): First vector.
    - v2 (numpy array): Second vector.

    Returns:
    - float: Manhattan distance between v1 and v2.
    """
    return np.sum(np.abs(v1 - v2))
```

- Để tương đồng về mặt so sánh, ta lựa chọn mức ngưỡng phù hợp đối với manhattan để có kết quả phân cụm là 4.



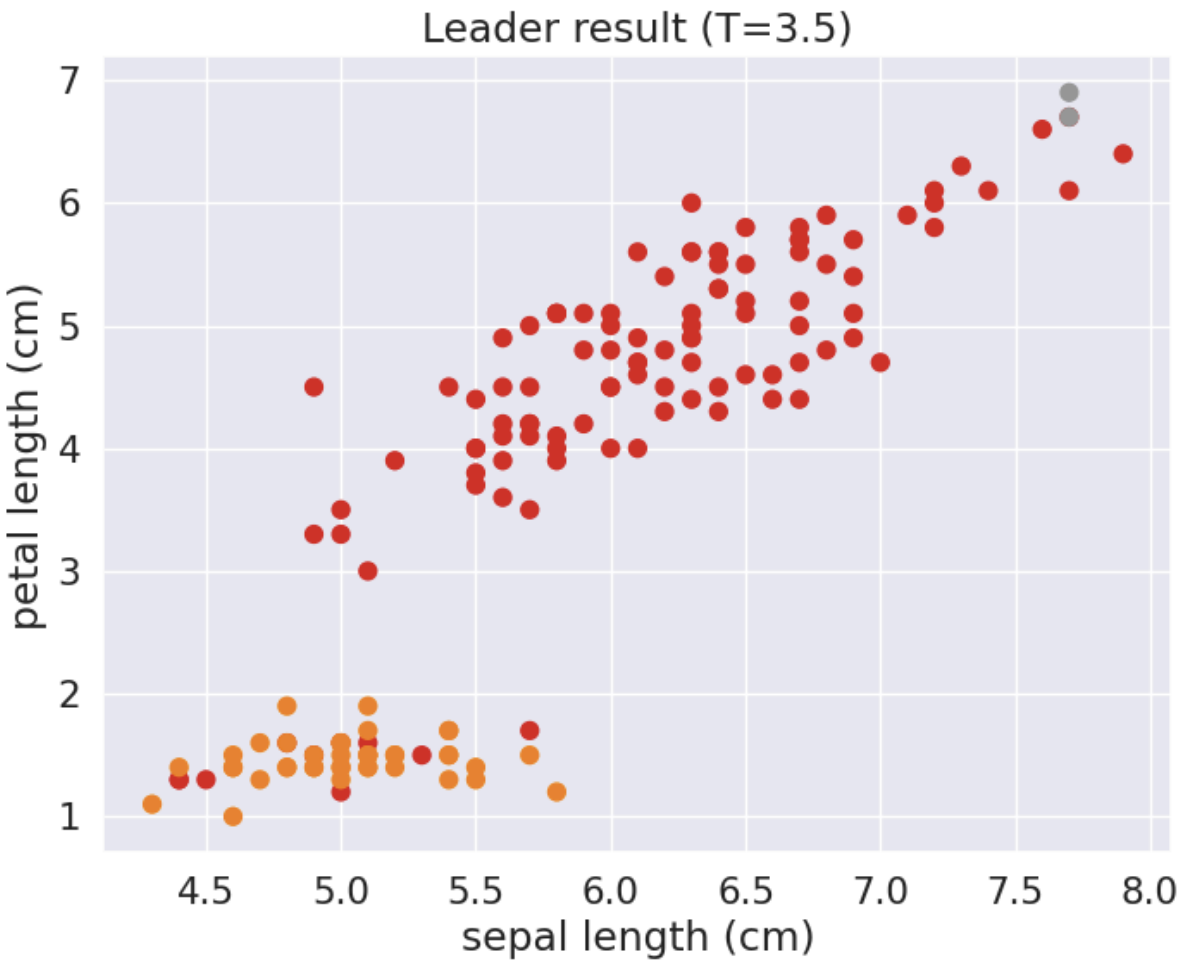
- Từ kết qủa trên, ta thấy rằng các cụm càng có giá trị lớn phân bố không tốt và bị nhiễu khá nhiều. Ta rút ra nhận xét rằng áp dụng công thức khoảng cách Euclidean tốt hơn so với manhattan trên bộ dữ liệu Iris.

5.3.2 Xáo trộn thứ tự của các điểm dữ liệu

```
#@title Shuffle data by row?
shuffle_df = shuffle(data)
shuffle_df = data.sample(frac=1).reset_index(drop=True)
shuffle_df.head(10)
data = shuffle_df
```

```
data.head(5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.5	2.6	4.4	1.2
1	5.7	3.8	1.7	0.3
2	6.3	3.4	5.6	2.4
3	6.2	2.9	4.3	1.3
4	4.4	3.2	1.3	0.2



5.4 Nhận xét thuật toán phân cụm Leaders

Thuật toán Leaders là một phương pháp phân cụm tăng tiến hiệu quả và có những ưu điểm đáng kể. Nó cho phép xử lý hiệu quả các tập dữ liệu lớn và mở rộng linh hoạt khi có sự thay đổi, không đòi hỏi phải sử dụng lại toàn bộ dữ liệu. Hiệu suất tính toán của thuật toán Leaders được đánh giá cao, đặc biệt là trong việc xử lý các tập dữ liệu có kích thước lớn. Với khả năng giảm thiểu thời gian tính toán và tài nguyên bằng cách không yêu cầu việc xử lý toàn bộ dữ liệu, thuật toán này trở thành một lựa chọn hữu ích trong nhiều ứng dụng phân cụm.

Tuy nhiên, thuật toán này cũng có nhược điểm:

- Đối với thuật toán Leaders, sự phụ thuộc vào tham số threshold đóng vai trò quan trọng trong quá trình quyết định. Việc lựa chọn giá trị threshold phù hợp đóng vai trò quan trọng, có thể ảnh hưởng đến cả hiệu suất và chất lượng của kết quả phân cụm.
- Thêm vào đó, phụ thuộc vào thứ tự của các điểm dữ liệu là một yếu tố đáng chú ý đối với thuật toán Leaders. Thuật toán có thể dễ bị ảnh hưởng bởi thứ tự của dữ liệu, điều này có thể tạo ra sự biến động trong quá trình phân cụm. Do đó, việc xem xét và kiểm soát thứ tự của dữ liệu có thể là quan trọng để đảm bảo tính ổn định và chính xác của kết quả.

6. KẾT LUẬN

Trong bài báo này, chúng em đã thực hiện nghiên cứu về giải thuật phân cụm tăng tiến Leaders và áp dụng nó vào việc phân cụm tập dữ liệu Iris. Các mục tiêu đã đề ra đã được đạt được thành công. Qua quá trình nghiên cứu, chúng em đã hiểu rõ cách hoạt động của thuật toán Leaders và đạt được kết quả tích cực khi triển khai thuật toán.

Chúng em đã thực hiện đánh giá và so sánh kết quả của thuật toán Leaders với một phương pháp phân cụm khác. Kết quả cho thấy thuật toán Leaders có hiệu suất thực thi tốt và khả năng xử lý dữ liệu gia tăng. Tuy nhiên, cũng cần lưu ý rằng thuật toán này có nhược điểm khi phải phụ thuộc vào ngưỡng khoảng cách và thứ tự của dữ liệu.

Tổng kết, báo cáo này làm rõ rằng phương pháp phân cụm Leaders đang thể hiện hiệu quả đối với việc xử lý dữ liệu gia tăng, và có tiềm năng phát triển và áp dụng rộng rãi trong môi trường dữ liệu lớn và sẽ được ứng dụng nhiều hơn trong tương lai.

TÀI LIỆU THAM KHẢO

1. PGS. TS Dương Tuấn Anh, Chương 6 - Học không giám sát - Phân cụm, Học Máy và Ứng dụng, khoa khoa Học và Kỹ Thuật Máy Tính, Đại Học bách Khoa TPHCM.
2. Edgar Anderson, Iris flower data set, Scikit-Learn. https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html
3. A.C. Muller, S. Guido, 2016, Introduction to Machine Learning with Python – A Guide for Data Scientists, O'Reilly