SỞ GIÁO DỤC & ĐÀO TẠO THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG THPT CHUYÊN TRẦN ĐẠI NGHĨA

ĐỀ TÀI :

THUẬT TOÁN PHÂN CỤM K-MEANS

GIÁO VIÊN HƯỚNG DẪN : Th.S Hồ Ngọc Lâm
HỌC SINH THỰC HIỆN :

1. Nguyễn Thị Phương Anh – 02
2. Trần Việt Hoàng – 07
3. Phạm Ngọc Tường Nghi – 14
4. Nguyễn Trương Bảo Như – 18

MÔN: TIN CHUYÊN

Thành phố Hồ Chí Minh, tháng 10 năm 2023

MỤC LỤC

1. GIỚI THIỆU

Khai phá dữ liệu thuộc quá trình khám phá tri thức. Về bản chất là giai đoạn duy nhất tìm ra được thông tin mới, tiềm ẩn có trong cơ sở dữ liệu chủ yếu phục vụ cho mô tả và dự đoán.

Phân cụm dữ liệu là một kỹ thuật trong khai phá dữ liệu với mục đích chính là phân tách một tập dữ liệu gồm n phần tử thành k cụm sao cho các phần tử trong mỗi cụm là tương tự nhau và các phần tử nằm khác cụm thì sẽ không tương tự nhau theo một nghĩa nào đó, nhằm khám phá và tìm kiếm các thông tin tiềm ẩn, hữu ích. Được ứng dụng trong nhiều lĩnh vực khác nhau như: thiên văn học, tin sinh học, thương mại điện tử, phát hiện lừa đảo, quảng cáo, quản lý quan hệ khách hàng, chăm sóc sức khỏe, viễn thông, đầu tư.

Trong phân cụm dữ liệu, thuật toán K-means là thuật toán kinh điển nhưng không thể giải quyết tập dữ liệu lớn. Để khắc phục một số nhược điểm của K-means khi xử lý dữ liệu lớn, các cải tiến thường sử dụng mô hình lập trình MapReduce để tăng hiệu suất thuật toán.

2. KIẾN THỨC LIÊN QUAN

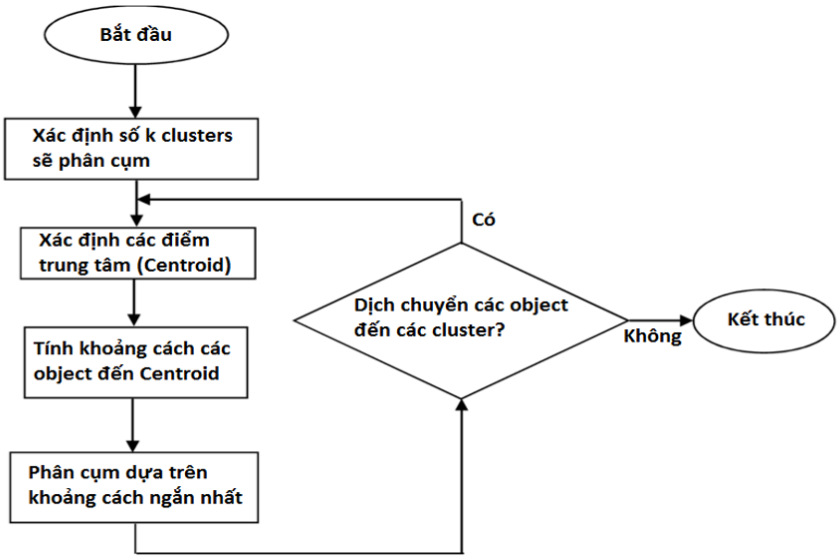
2.1 Khái niệm

Thuật toán K-means là một thuật toán quan trọng và được sử dụng phổ biến trong kỹ thuật phân cụm, thuật toán này được giới thiệu vào những năm 50 của thế kỷ 20. Ý tưởng của thuật toán K-Means là dựa trên k trọng tâm tại bước đầu tiên, thuật toán tiến hành gán các điểm vào các trọng tâm gần nhất và lặp lại quá trình trên cho đến khi các trọng tâm không thay đổi được nữa. Thuật toán K-Means thực hiện đơn giản và có độ phức tạp tính toán là O(k x n). K-means chia một tập D đối tượng thành k cụm sao cho kết quả tương đồng trong cùng cụm tương đối cao, ngược lại là sự tương đồng thấp giữa các cụm khác nhau.

2.2 Thuật toán phân cụm K-means

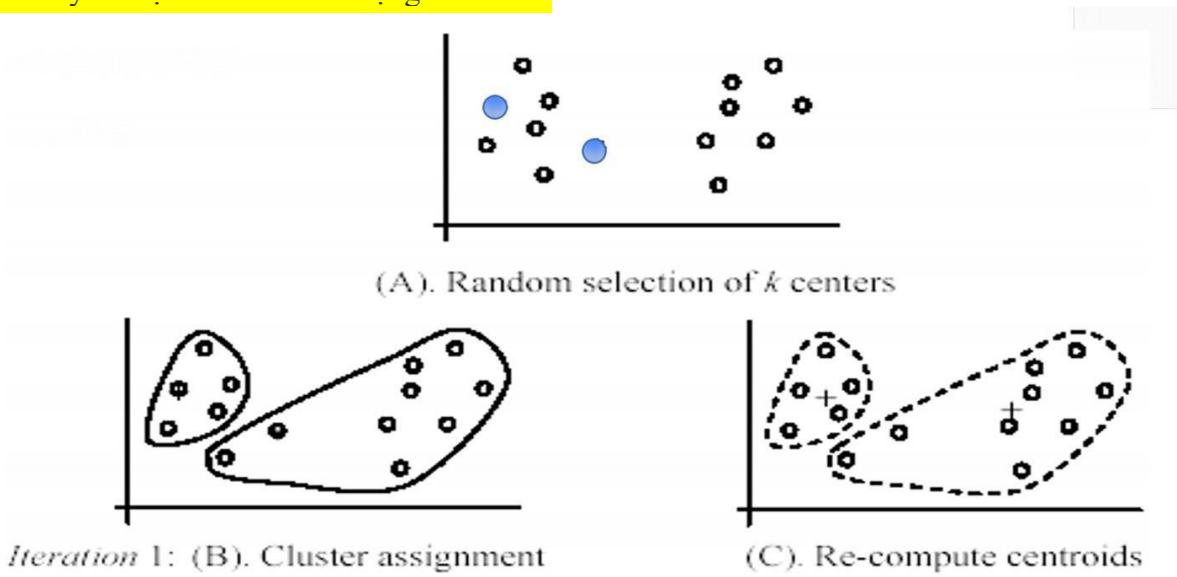
Các bước trong thuật toán

Thuật toán K-Means;
Input: Tập dữ liệu X, số lượng cụm cần phân tách k
Output: k cụm của X
+ Khởi tạo các trọng tâm cho các cụm ngẫu nhiên.
Repeat
+ Gán các điểm của X vào cụm gần nó nhất
+ Tính toán lại trọng tâm bằng trung bình cộng các điểm trong mỗi cụm tương ứng
Until (Các trọng tâm không thay đổi được nữa)



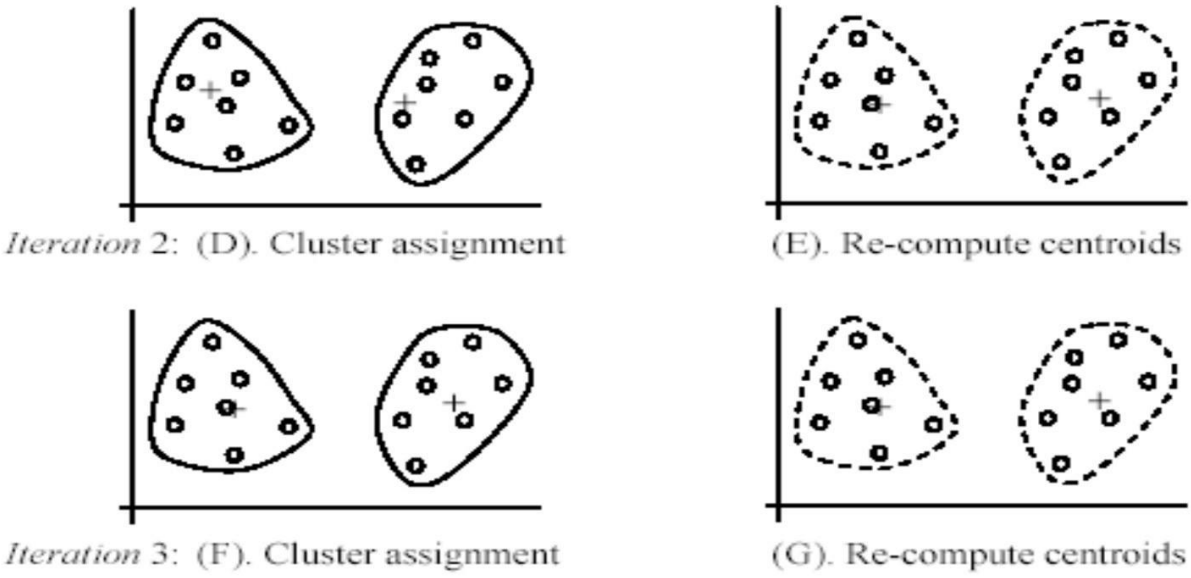
Hình 2.2.1 Sơ đồ thuật toán K-means

Sau đây là một số bước dưới dạng hình ảnh:



Hình 2.2.2 Thuật toán dưới dạng hình ảnh

Tại bước này thuật toán sẽ khởi tạo k điểm dữ liệu trung tâm ban đầu, sau đó qua iteration 1 để thực hiện bước 1: gán các điểm dữ liệu vào cụm gần nó nhất và bước 2: Xác định lại điểm trung tâm



Hình 2.2.3 Thuật toán dưới dạng hình ảnh

Các vòng lặp iteration 2 và iteration 3 tiếp tục thực hiện như vậy đến khi nào thuật toán hội tụ thì dừng lại.

Điều kiện hội tụ

Ta sẽ xác định điều kiện dừng thuật toán theo một số cách như sau:

- Tại 1 vòng lặp: có ít các điểm dữ liệu được gán sang cluster khác hoặc
- Điểm trung tâm (centroid) không thay đổi nhiều hoặc
- Giá trị hàm mất mát không thay đổi nhiều:

$$Error = \sum_{i=1}^{k} \sum_{\mathbf{x} \in C_i} d(\mathbf{x}, \mathbf{m_i})^2$$

Hình 2.2.4 Công thức điều kiện hội tụ

Trong đó Ci là cluster thứ i, Mi là điểm trung tâm của cluster Ci tương ứng.

Nhìn chung về điều kiện hội tụ có thể thấy mối liên hệ giữa các điều kiện là gần tương đồng như nhau. Khi có ít điểm dữ liệu được gán sang cluster khác có thể khiến điểm trung tâm không thay đổi nhiều và từ đó hàm mất mát cũng sẽ ít bị ảnh hưởng. Vậy nên chúng ta có thể sử dụng 1 trong 3 cách trên để xác định điều kiện dừng của thuật toán.

Xác định điểm trung tâm của cluster

Để xác định điểm trung tâm của cluster ta sử dụng công thức như sau:

$$\mathbf{m_i} = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

Hình 2.2.5 Công thức xác định điểm trung tâm

Trong đó Ci là cluster thứ i, Mi là điểm trung tâm của cluster Ci tương ứng.

Phép đo khoảng cách

Trong K-means để đánh giá mức độ giống nhau hay khoảng cách giữa 2 điểm dữ liệu ta có thể sử dụng các phép đo khoảng cách khác nhau. Ngoài khoảng cách Euclidean, tuỳ thuộc vào từng bài toán có thể sử dụng phương pháp đo khác (cosine, manhattan…)

$$d(\mathbf{x}, \mathbf{m_i}) = \|\mathbf{x} - \mathbf{m_i}\| = \sqrt{(x_1 - m_{i1})^2 + (x_2 - m_{i2})^2 + \ldots + (x_n - m_{in})^2}$$

Hình 2.2.6 Công thức phép đo khoảng cách

3. MINH HỌA GIẢI THUẬT

3.1 Bài 1

Đề bài:

Có 1 tập hợp các điểm trên không gian tọa độ Oxy. Mỗi điểm sẽ có tọa độ (x, y) xác định. Bài toán cần giải quyết là chia các điểm này thành K cụm khác nhau phân biệt.

Triển khai code:

Khởi tạo ngẫu nhiên các điểm dữ liệu, để thể hiện tính chất cụm của dữ liệu, ta sẽ khởi tạo các mẫu dữ liệu này xung quanh 3 tâm cụm (2, 2), (9, 2) và (4,9).

Ứng với mỗi tâm cụm, ta sẽ khởi tạo 500 điểm dữ liệu xung quanh nó. Ở đây lần lượt là X0, X1, X2.

```
means = [[2, 2], [9, 2], [4, 9]]
cov = [[2, 0], [0, 2]]
n_samples = 500
n_cluster = 3
X0 = np.random.multivariate_normal(means[0], cov, n_samples)
X1 = np.random.multivariate_normal(means[1], cov, n_samples)
X2 = np.random.multivariate_normal(means[2], cov, n_samples)
X = np.concatenate((X0, X1, X2), axis = 0)
```

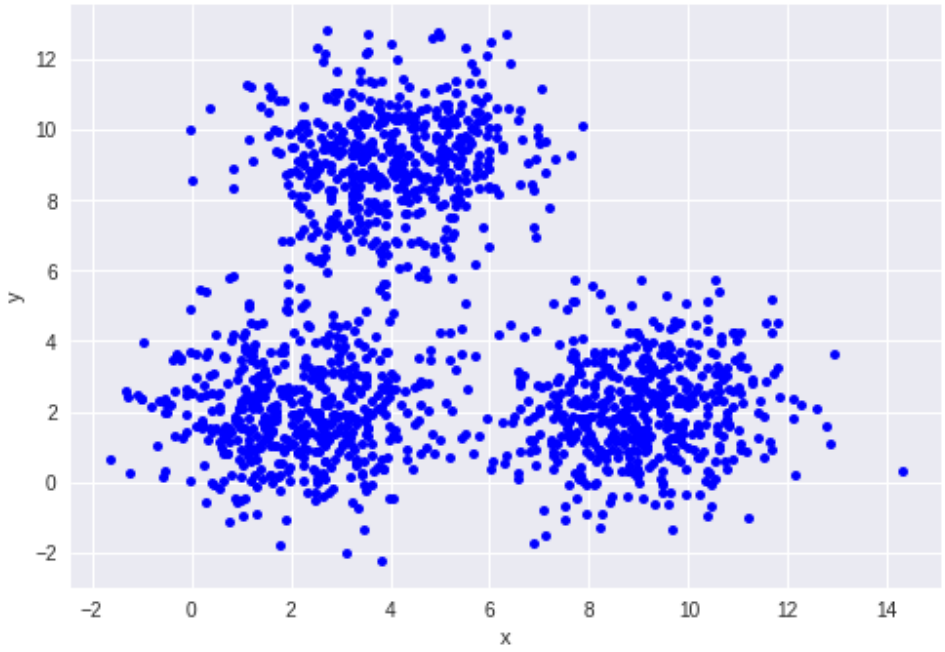
Để đơn giản, ta sẽ chỉ định số tâm cụm cho bài toán này là n_cluster = 3 luôn – chính là K đó ạ. Đúng với số cụm chúng ta vừa khởi tạo. Bạn có thể thử và xem với các giá trị n_cluster khác nhau để xem sự thay đổi.

```
plt.xlabel('x')
plt.ylabel('y')
plt.plot(X[:, 0], X[:, 1], 'bo', markersize=5)
plt.plot()
plt.show()
```

Phân bố của dữ liệu chúng ta vừa tạo

Như vậy, nếu thuật toán phân cụm k-means hoạt động tốt, nó sẽ phải lọc ra 3 tâm cụm có tọa độ sát với 3 tâm cụm (2, 2), (9, 2) và (4,9). Và ban đầu tọa độ của các tâm này sẽ được lấy ngẫu nhiên.

Bây giờ, chúng ta hãy viết hàm khởi tạo n_cluster tâm cụm



Hình 3.1.1 Phân bố của dữ liệu chúng ta vừa tạo

```
def kmeans_init_centers(X, n_cluster):
# random k index beetween 0 and shape(X) without duplicate index.
# Then return X[index] as cluster
return X[np.random.choice(X.shape[0], n_cluster, replace=False)]
```

Ở đây, ta khởi tạo n_cluster tâm cụm bằng cách lấy ngẫu nhiên n_cluster điểm dữ liệu của tập dữ liệu.

Xác định các tâm cụm:

```
def kmeans_predict_labels(X, centers):
D = cdist(X, centers)
# return index of the closest center
return np.argmin(D, axis = 1)
```

Với mỗi điểm dữ liệu trong tập dữ liệu, tâm cụm của nó sẽ là 1 trong số n_cluster tâm cụm gần với nó nhất. Việc tính toán khoảng cách giữa 2 điểm trong bài này sử dụng Euclidean distance.

Cập nhật lại vị trí của các tâm cụm:

```
def kmeans_update_centers(X, labels, n_cluster):
centers = np.zeros((n_cluster, X.shape[1]))
for k in range(n_cluster):
# collect all points assigned to the k-th cluster
Xk = X[labels == k, :]
# take average
centers[k,:] = np.mean(Xk, axis = 0)
return centers
```

Việc tính toán lại tọa độ của mỗi tâm cụm được thực hiện đơn giản bằng cách lấy trung bình cộng tọa độ của tất cả các điểm dữ liệu của cụm. Sau khi tính toán xong, vị trí mới của tâm cụm sẽ nằm chính giữa cụm của nó.

Kiểm tra tính hội tụ:

```
def kmeans_has_converged(centers, new_centers):
# return True if two sets of centers are the same
return (set([tuple(a) for a in centers]) ==
```

```
set([tuple(a) for a in new_centers]))
```

Nếu việc cập nhật lại vị trí của các tâm cụm không có thay đổi gì thì có nghĩa là chúng ta đã có thể dừng và đưa ra kết quả. Nghĩa là tọa độ cũ và tọa độ sau khi cập nhật của các tâm cụm là như nhau. Trong trường hợp này, nếu bạn có tiếp tục chạy nữa thì cũng sẽ không thay đổi gì.

Để các bạn có thể thấy rõ cách hoạt động của thuật toán, hãy bổ sung thêm hàm visualize này.

```
# Hàm này dùng để vẽ dữ liệu lên đồ thị
# Random color chỉ làm việc với k <= 4
# Nếu bạn thay đổi k > 4, hãy sửa lại phần random color nhé
# Chỉ sử dụng trong bài toán này thôi nhé.
def kmeans_visualize(X, centers, labels, n_cluster, title):
plt.xlabel('x') # label trục x
plt.ylabel('y') # label trục y
plt.title(title) # title của đồ thị
plt_colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'w'] # danh sách các màu hỗ trợ
for i in range(n_cluster):
data = X[labels == i] # lấy dữ liệu của cụm i
plt.plot(data[:, 0], data[:, 1], plt_colors[i] + '^', markersize = 4, label = 'cluster_' + str(i)) # Vẽ cụm i lên đồ thị
plt.plot(centers[i][0], centers[i][1],  plt_colors[i+4] + 'o', markersize = 10, label = 'center_' + str(i)) # Vẽ tâm cụm i lên đồ thị
plt.legend() # Hiện bảng chú thích
plt.show()
```

Sau mỗi lần cập nhật tâm cho từng điểm dữ liệu, hoặc cập nhật lại tọa độ của các tâm. Hãy gọi hàm này để xem sự thay đổi của vị trí tâm và tâm của các điểm dữ liệu. Kết quả sẽ được vẽ lên đồ thị để bạn dễ quan sát.

Toàn bộ thuật toán k-means:

```
def kmeans(init_centes, init_labels, X, n_cluster):
centers = init_centes
labels = init_labels
times = 0
while True:
labels = kmeans_predict_labels(X, centers)
kmeans_visualize(X, centers, labels, n_cluster, 'Assigned label for data at time = ' + str(times + 1))
new_centers = kmeans_update_centers(X, labels, n_cluster)
if kmeans_has_converged(centers, new_centers):
break
centers = new_centers
kmeans_visualize(X, centers, labels, n_cluster, 'Update center possition at time = ' + str(times + 1))
times += 1
return (centers, labels, times)
```

Bạn đầu, chúng ta sẽ khởi tọa độ cho các tâm cụm. Khởi tạo tất cả các điểm dữ liệu thuộc cụm 0 (tùy chọn).

Lặp lại việc tìm tâm cụm cho các điểm dữ liệu, cập nhật tọa độ tâm cụm cho tới khi tọa độ của các tâm cụm không còn thay đổi nữa.

Do bài toán này khá đơn giản, nên việc giải chỉ mất vài lần lặp.

Chúng ta sẽ gọi hàm kmeans_visualize sau mỗi lần gán lại tâm cụm cho từng điểm dữ liệu hoặc tính toán lại tọa độ các tâm cụm.

Bây giờ, việc ta cần làm là khởi tạo tọa độ tâm các cụm. Và sau đó gọi hàm kmeans phía trên để thực thi.

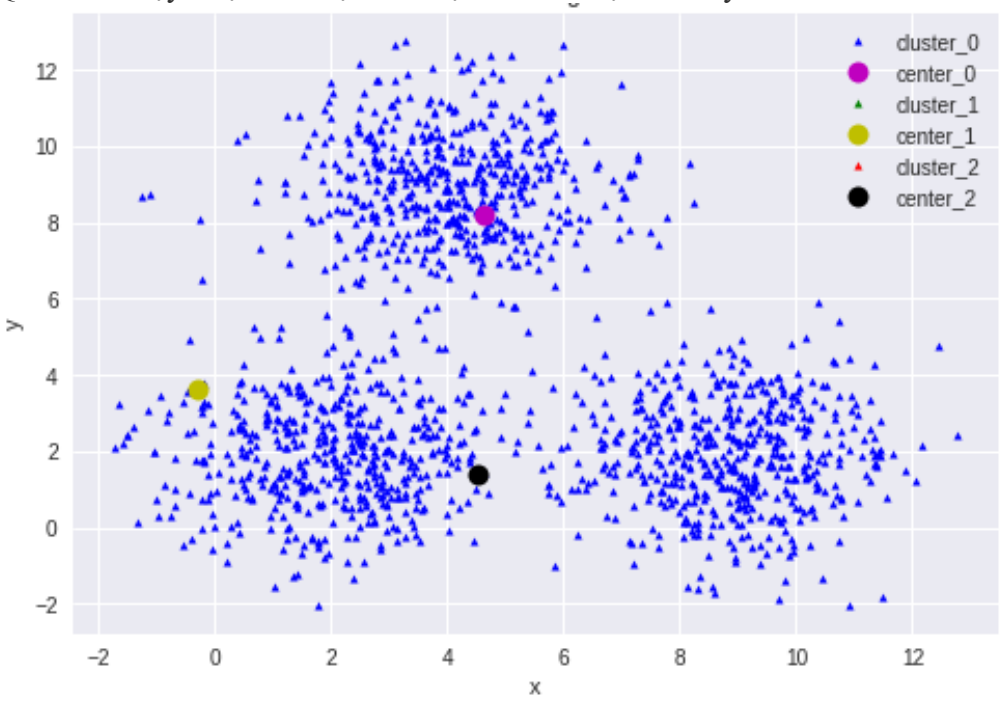
```
init_centers = kmeans_init_centers(X, n_cluster)
print(init_centers) # In ra tọa độ khởi tạo ban đầu của các tâm cụm
init_labels = np.zeros(X.shape[0])
kmeans_visualize(X, init_centers, init_labels, n_cluster, 'Init centers in the first run. Assigned all data as cluster 0')
centers, labels, times = kmeans(init_centers, init_labels, X, n_cluster)
print('Done! Kmeans has converged after', times, 'times')
```

Bạn có thể xem sự thay đổi ở các lần chạy như sau:

Đây là tọa độ tâm các cụm khởi tạo ngẫu nhiên

```
[[ 4.63795825  8.18623907]
[-0.30216026  3.63816318]
[ 4.53141139  1.41293449]]
```

Quá trình chạy được thể hiện lần lượt theo đồ thị dưới đây:



Hình 3.1.2 Gán tất cả dữ liệu dưới dạng cụm 0

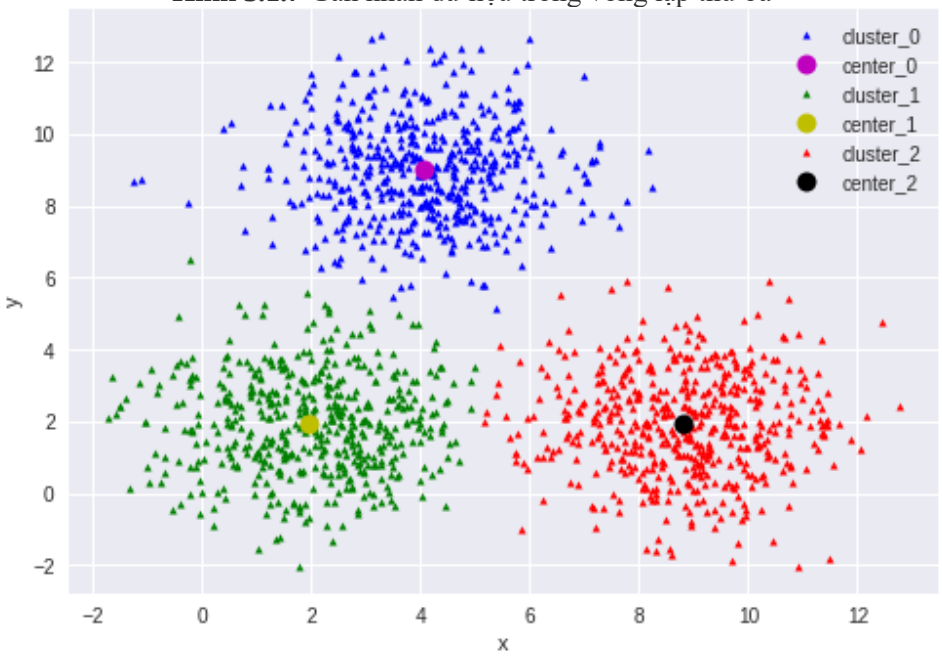
Hình 3.1.3 Gán nhãn cho dữ liệu trong vòng lặp đầu tiên

Hình 3.1.4 Cập nhật vị trí trung tâm trong vòng lặp đầu tiên

Hình 3.1.5 Gán nhãn dữ liệu trong vòng lặp thứ hai

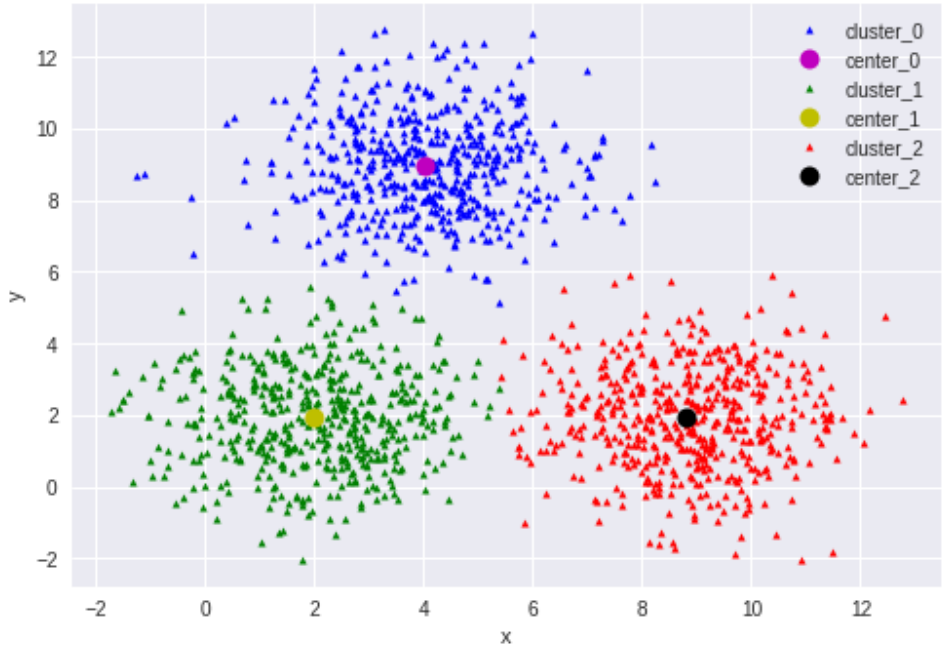
Hình 3.1.6 Cập nhật trung tâm trong vòng lặp thứ hai

Hình 3.1.7 Gán nhãn dữ liệu trong vòng lặp thứ ba



Hình 3.1.8 Cập nhật trung tâm trong vòng lặp thứ ba

Hình 3.1.9 Gán nhãn dữ liệu trong vòng lặp thứ tư



Hình 3.1.10 Cập nhật trung tâm trong vòng lặp thứ tư

Sau khi chạy xong thuật toán k-means. Chúng ta hãy thử in gia tọa độ của các tâm cụm:

```
print(centers)
>>
[[4.03457895 8.99262915]
[1.96728805 1.9546642 ]
[8.81688307 1.9386987 ]]
```

Kết quả cho ra tọa độ của 3 tâm cụm khá sát vớt các tâm mà chúng ta khởi tạo là (2, 2), (9, 2) và (4,9). Điều đó cho thấy k-means đã làm rất tốt trong bài toán này.

3.2 Bài 2

Đề bài:

Trước khi bắt đầu, hãy hiểu về loại vấn đề chúng ta sẽ giải quyết ở đây. Tập dữ liệu Mall_Customers chứa dữ liệu về những khách hàng ghé thăm và mua sắm ở Mall.

Trong tập dữ liệu đã cho, chúng ta có ID của khách hàng, Giới tính, Tuổi, Thu nhập hàng năm và Điểm chi tiêu (là giá trị tính toán số tiền mà khách hàng đã chi tiêu trong trung

tâm mua sắm, giá trị càng nhiều thì họ càng chi tiêu nhiều hơn). Từ tập dữ liệu này, ta cần tính toán một số mẫu, nhưng vì đây là phương pháp học không giám sát nên ta không biết chính xác những gì cần tính.

Các bước triển khai được đưa ra dưới đây:

- Xử lý dữ liệu
- Tìm số cụm tối ưu bằng phương pháp khuỷu tay (elbow method)
- Huấn luyện thuật toán K-means trên tập dữ liệu huấn luyệns
- Trực quan hóa các cụm

Bước 1: Xử lý dữ liệu

Bước đầu tiên là xử lý dữ liệu, nhưng những gì ta đã làm trong các thuật toán trước đây về Hồi Quy tuyến tính và Phân loại. Nhưng với bài toán Phân cụm thì sẽ khác với những bài toán trước. Hãy nói về nó:

Khai báo thư viện

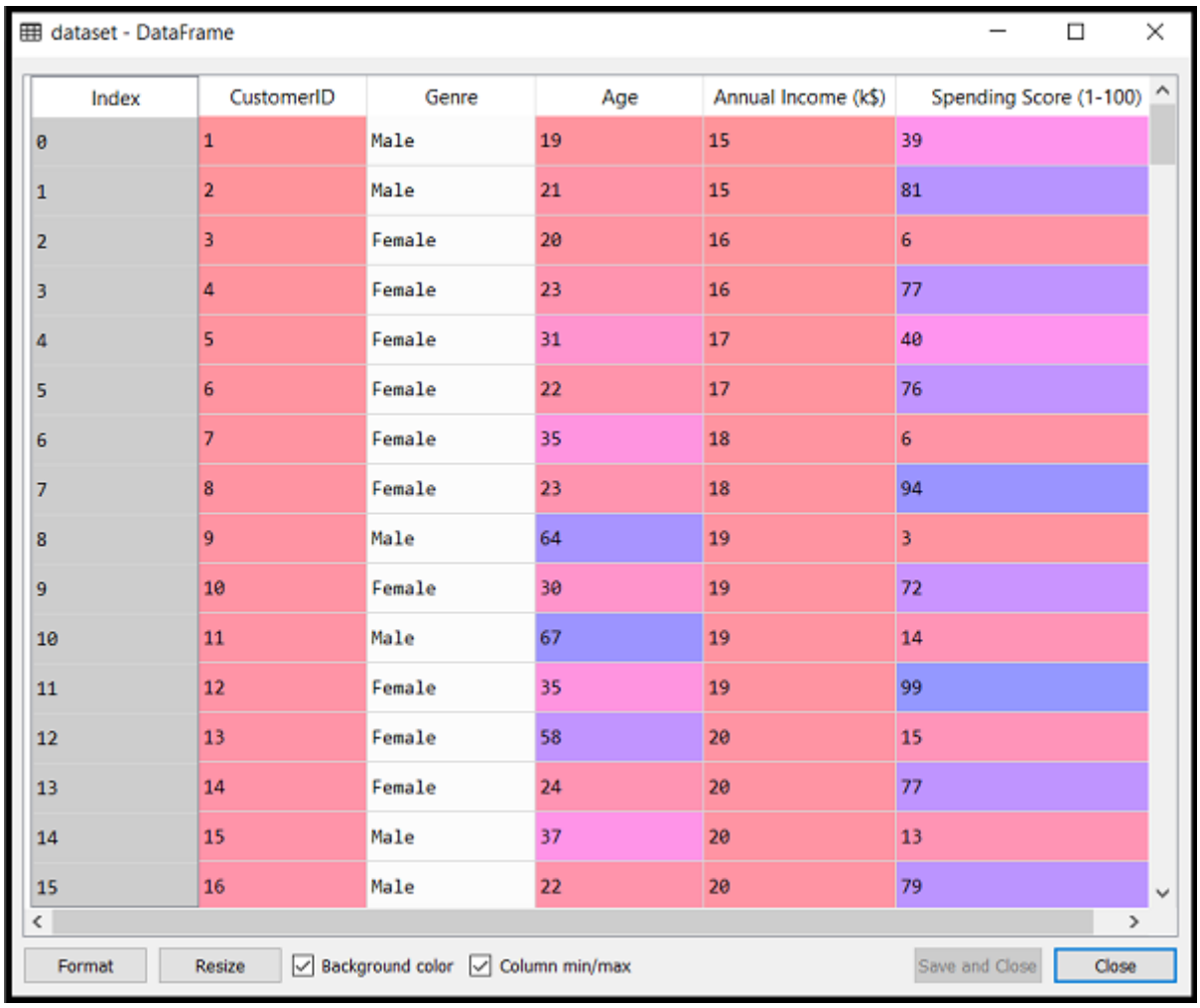
Như chúng ta đã làm trong các chủ đề trước, trước tiên, chúng ta sẽ nhập các thư viện cho mô hình của mình, đây là một phần của quá trình xử lý trước dữ liệu. Code được đưa ra dưới đây:

```
#importing libraries
import numpy as np
import matplotlib.pyplot as mtp
import pandas as pd
```

Trong đoạn code trên, ta nhập numpy để thực hiện phép tính toán học, tplotlib để vẽ đồ thị và pandas để quản lý tập dữ liệu.

Nhập tập dữ liệu: tiếp theo, ta sẽ nhập tập dữ liệu cần thiết, ở đây là tập dữ liệu Mall_Customer_data.csv. Ta có thể nhập bằng code dưới đây:

```
# Importing the dataset
dataset = pd.read_csv('Mall_Customers_data.csv')
```



Index	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72
10	11	Male	67	19	14
11	12	Female	35	19	99
12	13	Female	58	20	15
13	14	Female	24	20	77
14	15	Male	37	20	13
15	16	Male	22	20	79

Hình 3.2.1 Bảng dữ liệu

Từ tập dữ liệu trên, chúng ta cần tìm một số mẫu trong đó.

Trích xuất các biến độc lập

Chúng ta không cần bất kỳ biến phụ thuộc nào cho bước tiền xử lý dữ liệu vì đây là vấn đề về phân cụm và chúng ta không biết phải xác định điều gì. Vì vậy, chúng ta sẽ chỉ thêm một dòng mã cho ma trận các tính năng.

x = dataset.iloc[:, [3, 4]].values

Như chúng ta có thể thấy, chúng tôi chỉ trích xuất tính năng thứ 3 và thứ 4. Đó là bởi vì chúng ta cần biểu đồ 2d để trực quan hóa mô hình và một số tính năng không bắt buộc, chẳng hạn như customer_id.

Bước 2: Tìm số cụm tối ưu sử dụng phương pháp khuỷu tay

Trong bước thứ hai, chúng ta sẽ cố gắng tìm số cụm tối ưu cho bài toán phân cụm của mình. Vì vậy, như đã thảo luận ở trên, ở đây chúng ta sẽ sử dụng phương pháp khuỷu tay cho mục đích này.

Như chúng ta đã biết, phương pháp khuỷu tay sử dụng khái niệm WCSS để vẽ đồ thị bằng cách vẽ các giá trị WCSS trên trục Y và số cụm trên trục X. Vì vậy, chúng ta sẽ tính giá trị cho WCSS cho các giá trị k khác nhau trong khoảng từ 1 đến 10. Dưới đây là mã cho nó:

```
#finding optimal number of clusters using the elbow method
from sklearn.cluster import KMeans
wcss_list= []  #Initializing the list for the values of WCSS

#Using for loop for iterations from 1 to 10.
for i in range(1, 11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state= 42)
```

```
        kmeans.fit(x)
        wcss_list.append(kmeans.inertia_)
    mtp.plot(range(1, 11), wcss_list)
    mtp.title('The Elobw Method Graph')
    mtp.xlabel('Number of clusters(k)')
    mtp.ylabel('wcss_list')
mtp.show()
```

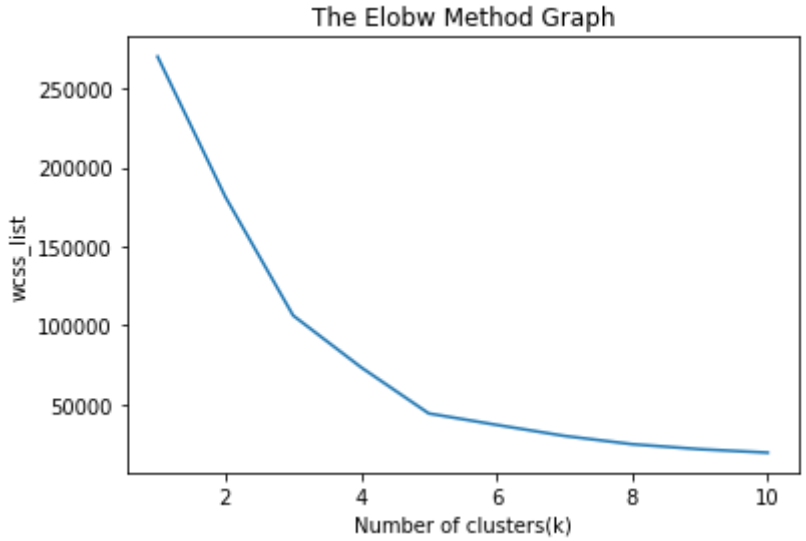
Như chúng ta có thể thấy trong đoạn mã trên, chúng ta đã sử dụng lớp sklearn của KMeans. thư viện cụm để tạo thành các cụm.

Tiếp theo, chúng ta đã tạo biến wcss_list để khởi tạo một danh sách trống, được sử dụng để chứa giá trị wcss được tính cho các giá trị khác nhau của k trong khoảng từ 1 đến 10.

Sau đó, chúng tôi đã khởi tạo vòng lặp for để lặp lại trên một giá trị khác của k trong khoảng từ 1 đến 10; vì vòng lặp for trong Python loại trừ giới hạn gửi đi, do đó, nó được lấy là 11 để bao gồm giá trị thứ 10.

Phần còn lại của mã tương tự như chúng tôi đã làm trong các chủ đề trước đó, vì chúng tôi đã điều chỉnh mô hình trên một ma trận các đặc điểm và sau đó vẽ đồ thị giữa số lượng cụm và WCSS.

Đầu ra: Sau khi thực thi đoạn mã trên, chúng ta sẽ nhận được kết quả đầu ra bên dưới:



Hình 3.2.2 Biểu đồ phương pháp khuỷu tay

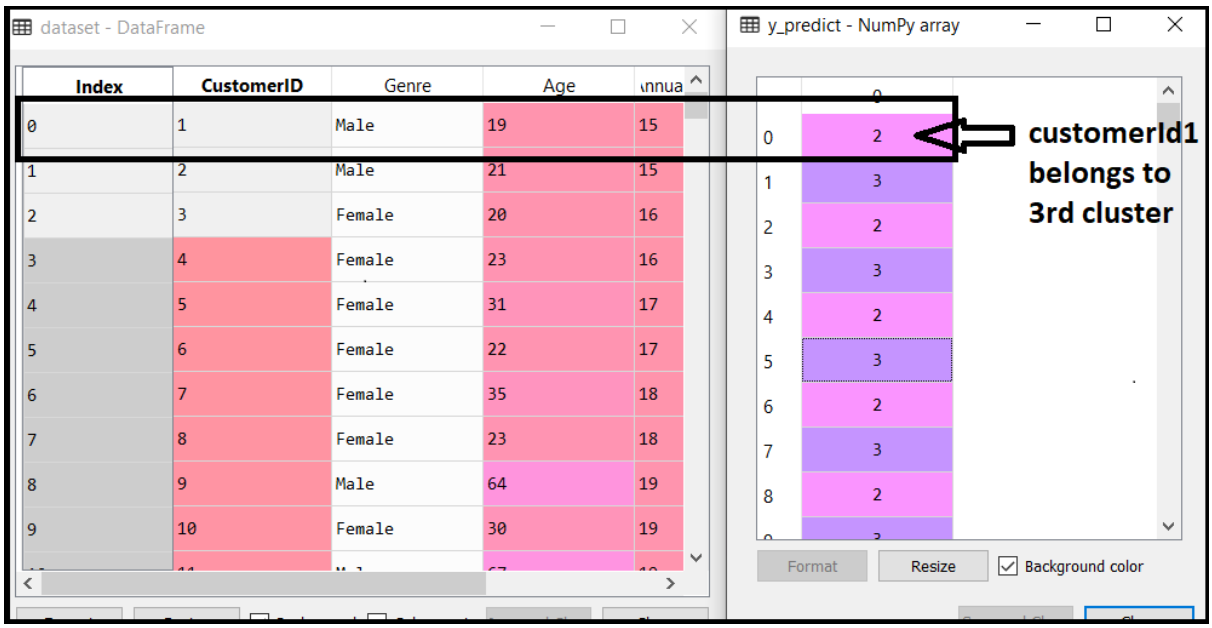
Bước 3: Kiểm tra thuật toán K-means bằng dữ liệu cụ thể

Bởi vì chúng ta đã có số lượng cụm tối ưu, sau đó ta sẽ bắt đầu thuật toán dựa vào dữ liệu đầu vào

Ta sẽ sử dụng 2 dòng code dưới đây, thay vì sử dụng biến i, ta sẽ sử dụng 5(số lượng cụm tối ưu đã tìm được ở trên)

```
    #Training the K-means model on a dataset
    kmeans = KMeans(n_clusters=5, init='k-means++', random_state= 42)
y_predict= kmeans.fit_predict(x)
```

Dòng thứ 2 ta tạo 1 biến phụ thuộc y_predict và so sánh giá trị của biến đó với tập giá trị gốc. Ta xem ảnh dưới đây:



Hình 3.2.3 Phân phối dữ liệu vào từng cụm

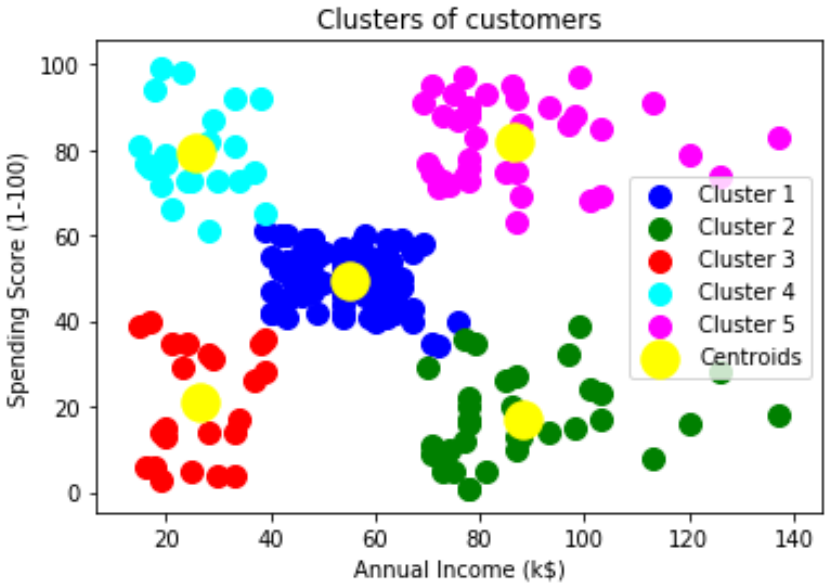
Ta thấy rằng Khách hàng 1 thuộc cụm 3, và tương tự khách hàng 2 thuộc cụm 4,...

Bước 4: Trực quan hóa các cụm

Bởi vì ta đã có 5 cụm cho mô hình, ta sẽ trực quan hóa nó từng bước một bằng việc sử dụng hàm mtp.scatter()

```
#visualizing the clusters
mtp.scatter(x[y_predict == 0, 0], x[y_predict == 0, 1], s = 100, c = 'ble', label = 'Cluster 1')
#for first cluster
mtp.scatter(x[y_predict == 1, 0], x[y_predict == 1, 1], s = 100, c = 'green', label = 'Cluster 2') #for second cluster
mtp.scatter(x[y_predict== 2, 0], x[y_predict == 2, 1], s = 100, c = 'red', label = 'Cluster 3')
#for third cluster
mtp.scatter(x[y_predict == 3, 0], x[y_predict == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
#for fourth cluster
mtp.scatter(x[y_predict == 4, 0], x[y_predict == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5') #for fifth cluster
mtp.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c = 'yellow', label = 'Centroid')
mtp.title('Clusters of customers')
mtp.xlabel('Annual Income (k$)')
mtp.ylabel('Spending Score (1-100)')
mtp.legend()
mtp.show()
```

Sự phối hợp của mtp.scatter, i.e., x[y_predict == 0, 0] chứa biến x cho ta thấy giá trị, và biến phụ thuộc y_predict chạy từ 0 đến 1.



Hình 3.2.4 Kết quả phân cụm

Ảnh trên cho ta thấy 5 cụm phân biệt rõ rệt với 5 màu khác nhau với trục x(giá trị thu nhập bình quân) và trục y(số tiền mỗi khách hàng sử dụng trong siêu thị).

4. KẾT LUẬN

Thuật toán K-means là một trong những thuật toán đơn giản của phân cụm nhưng có hiệu quả cao và được ứng dụng rộng rãi. Thuật toán dựa hoàn toàn vào bộ dữ liệu đầu vào, và việc lựa chọn số cụm hiệu quả là thực sự khó khăn vì số lượng lựa chọn quá lớn, điều đó dẫn đến hạn chế là độ phức tạp cao.

Tuy nhiên, giá trị nó mang lại thể hiện rõ rệt đặc điểm từng cụm, phân loại và trực quan hóa tổng thể những số liệu khổng lồ. Và kết quả ấy tạo nên những biểu đồ nhân khẩu học đến những nhóm nhỏ đơn giản nhất để những nhà chiến lược thị trường đưa ra giải pháp phù hợp nhất để đạt mục tiêu kinh doanh.

TÀI LIỆU THAM KHẢO

1. Hoàng Huyền Trang (2016), "Phương pháp phân cụm dựa trên tập thô và giải thuật di truyền" (PDF), từ http://lib.uet.vnu.edu.vn/bitstream/123456789/841/2/t%C3%B3m%20t%E1%BA%AFt%20LV_Ho%C3%A0ng%20Huy%E1%BB%81n%20Trang-K20HTTT.pdf, truy cập vào ngày 15/10/2023.
2. Trần Hoàng Việt, Nguyễn Thị Tuyết, Trần Thiên Thành (2017), "Một số cải tiến thuật toán K-means song song sử dụng phương pháp lấy mẫu" (PDF), từ http://elib.vku.udn.vn/bitstream/123456789/238/1/20181208091659.pdf, truy cập vào ngày 15/10/2023.
3. Cao Ngọc Ánh, Vũ Việt Vũ, Phùng Thị Thu Hiền (2017), "Thuật toán phân cụm K-means*" (PDF), từ http://jst.tnu.edu.vn/jst/article/viewFile/1087/pdf, truy cập vào ngày 16/10/2023.
4. Website, "K-means Clustering Algorithm", từ https://www.javatpoint.com/k-means-clu2ustering-algorithm-in-machine-learning, 16/10/2023.
5. Website, "Thuật toán K-means (K-means Clustering) và ví dụ", từ https://blog.luyencode.net/thuat-toan-phan-cum-k-means/?fbclid=IwAR3kGQKgXTHravgZVxfCEjquYuoZ8YDl1UtyQwfar9dlrOUhGNIPFDCznXI#thuc-hanh-voi-k-means, truy cập vào ngày 17/10/2023.
6. Nguyễn Đình Quý, "Thuật toán phân cụm K-Means", từ https://ndquy.github.io/posts/thuat-toan-phan-cum-kmeans/?fbclid=IwAR22zffmZD4LjQTYrcYBAy8D8tIVVhDX-gFEWWeUW1Fs_VCttu77OYbh760, truy cập vào ngày 17/10/2023.