BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC DUY TÂN**

**KHOA CÔNG NGHỆ THÔNG TIN**

🙡🕮🙣



**BÁO CÁO ĐỀ TÀI**

**MÔN HỌC: CDIO**

**Đề tài: *Tối ưu và tìm K tối ưu cho thuật toán phân cụm Kmeans bằng phương pháp Elbow và Silhouette.***

**Sinh viên thực hiện: Lê Văn Hiếu**

**Mã số sinh viên: 24211906618**

**MỤC LỤC**

[DANH MỤC CÁC KÍ HIỆU 3](#_Toc61560206)

[DANH MỤC CÁC HÌNH MÌNH HỌA 4](#_Toc61560207)

[Chương I. GIỚI THIỆU TỔNG QUAN ĐỀ TÀI 5](#_Toc61560208)

[I.1. Lý do chọn đề tài: 5](#_Toc61560209)

[I.2. Mục tiêu và phương pháp nghiên cứu 5](#_Toc61560210)

[I.3. Bố cục báo cáo 6](#_Toc61560211)

[Chương II. CƠ SỞ LÝ THUYẾT 7](#_Toc61560212)

[II.1. Giới thiệu lý thuyết về Machine Learning 7](#_Toc61560213)

[II.2. Giới thiệu về ngôn ngữ lập trình. 8](#_Toc61560214)

[II.3. Một số vấn đề về dữ liệu 10](#_Toc61560215)

[Chương III. THUẬT TOÁN PHÂN CỤM KMEANS 14](#_Toc61560216)

[III.1. Thuật toán phân cụm K-means 14](#_Toc61560217)

[III.2. Kmeans++ 18](#_Toc61560218)

[III.3. Vấn đề chọn số lượng tâm với Kmeans. 20](#_Toc61560219)

[Chương IV. XÂY DỰNG CÁC PHƯƠNG PHÁP TÌM K TỐI ƯU 21](#_Toc61560220)

[IV.1. Phương pháp Elbow 21](#_Toc61560221)

[IV.2. Phương pháp Silhouette 24](#_Toc61560222)

[Chương V. Kiểm Thử 29](#_Toc61560223)

[V.1. Thư viện sklearn và hàm make\_blops. 29](#_Toc61560224)

[V.2. Thực thi phân cụm bằng Kmean. 30](#_Toc61560225)

[V.3. Lựa chọn K trên phương pháp Elbow. 33](#_Toc61560226)

[V.4. Lựa chọn K trên phương pháp Silhouette. 34](#_Toc61560227)

[Chương VI. KẾT LUẬN 39](#_Toc61560228)

[VI.1. So sánh các thuật toán. 39](#_Toc61560229)

[VI.2. Các kết quả đạt được 39](#_Toc61560230)

[VI.3. Các hạn chế 39](#_Toc61560231)

[VI.4. Hướng phát triển: 40](#_Toc61560232)

[LIÊN KẾT VÀ TÀI LIỆU THAM KHẢO 40](#_Toc61560233)

DANH MỤC CÁC KÍ HIỆU

Các kí hiệu:

|  |  |
| --- | --- |
| Kí hiệu | Ý nghĩa |
| Kmeans | Thuật toán phân cụm dữ liệu Kmeans. |
| K/k | Số lượng tâm/cụm trong thuật toán phân cụm Kmeans |
| **C, c** | Lần lượt là tập ma trận các tâm của Kmeans và vector tâm thuộc **C** |
| **X, x** | Lần lượt là tập ma trận các điểm của tập dữ liệu đưa vào và vector điểm thuộc **X** |
| D(a,b) | Khoảng cách giữa ma trận hoặc vector a tới ma trân hoặc vector b. |
| d | Dimension. Số chiều hay cột của mảng/Tập dữ liệu |
| N | Numer. Số lượng phần tử của mảng/Tập dữ liệu |
| **L** | Labels. Tập ma trận các nhãn sinh từ Kmeans. |
| Min | Giá trị nhỏ nhất thuộc một tập nào đó. |
| Max | Giá trị lớn nhất thuộc một tập nào đó. |
| TWSS | Total Within Sum of Square, Giá trị dùng để đánh xem độ tượng đồng của một cụm như thế nào. |

DANH MỤC CÁC HÌNH MÌNH HỌA

[Hình 1. Bài toán phân cụm với số K=3 14](#_Toc61560191)

[Hình 2. Đầu ra của bài toán phân cụm với mỗi sự lựa chọn tâm khác nhau 18](#_Toc61560192)

[Hình 3. Ví dụ một mô hình Elbow method với tâm thích hợp là 5 22](#_Toc61560193)

[Hình 4. Biêu đồ biểu thị các giá trị Silhouette. 26](#_Toc61560194)

[Hình 5. Tạo dữ liệu 30](#_Toc61560195)

[Hình 6. Kmean init='rd' – 1 31](#_Toc61560196)

[Hình 7. Kmean init='rd' – 2 31](#_Toc61560197)

[Hình 8. Kmeans init = 'Kmeans++' – 1 32](#_Toc61560198)

[Hình 9. Kmeans init = 'Kmeans++' – 2 33](#_Toc61560199)

[Hình 10. Mô hình Silhouette K=2 35](#_Toc61560200)

[Hình 11. Mô hình Silhouette K=3 36](#_Toc61560201)

[Hình 12. Mô hình Silhouette K=4 36](#_Toc61560202)

[Hình 13. Mô hình Silhouette K=5 37](#_Toc61560203)

[Hình 14. Mô hình Silhouette K=6 37](#_Toc61560204)

[Hình 15. Mô hình Silhouette K=7 38](#_Toc61560205)

1. GIỚI THIỆU TỔNG QUAN ĐỀ TÀI
   1. Lý do chọn đề tài:

Kmeans là thuật toán phân cụm thuộc nhóm học không giám sát (Unsupervised Learning). Cách hình thành các cụm dựa trên việc tính khoảng cách các tâm và các điểm (Thường dùng khoảng cách Euclid). Sau đó điều chỉnh các tâm sao cho các tâm không còn thay đổi hoặc sự khác nhau không quá lớn. Sau khi tìm được các tâm thì ta có thể biết được các cụm.

Thuật toán Kmeans có thể giúp ích dược rất có thể rất hữu ích cho nhiều lĩnh vực khác nhau trong cuộc sống. Các lĩnh vực đó có thể là toán học, kinh tế, văn bản, quy hoạch đô thị, cứu nạn cứu trợ,… Một vài ứng dụng trong số đó có thể kể ra như là phân khúc khách hàng, nhận diện chữ viết, phân nhóm khu vực dựa vào phân bố dân cư, tách vật thể trong ảnh,…

Tuy vậy để Kmeans hoạt động được thì phải cần biết trước số cụm đưa vào(K). Đây là vấn đề thường gặp trong bài toán phân cụm nói chung. Với các bài toán đơn gian thì ta có thể nhận diện được số K và đưa vào thuật toán ngay. Nhưng đối với các bài toàn phức tạp hơn thì số K này là vấn đề lớn. Việc phân cụm mà số cụm chưa biết rõ thì ta cần phải tính toán để có được số cụm K tối ưu.

Để giải quyết vấn đề này, nhóm sẽ phân tích và xây dựng các phương pháp để tìm được số K tối ưu và áp dùng vào một số bài toán cụ thể. Các phương pháp: elbow method, silthoutte method sẽ giúp tìm được số tâm tối ưu. Tăng hiệu quả bài toán.

* 1. Mục tiêu và phương pháp nghiên cứu
     1. Mục tiêu nghiên cứu

Mục tiêu của đề tài này là nghiên cứu tối ưu thuật toán Kmean và tìm được các phương pháp hiệu quả đề tìm được số k tối ưu. Khi tìm được phương pháp thì nghiên cứu, áp dụng vào thuật toán Kmeans. Phân tích các ưu nhược điểm của các phương pháp và đánh giá nó. Đồng thời sử dụng các phương pháp đó đề tìm số cụm tối ưu cho một số bài toàn cụ thể. Hiểu thêm về các thuật toán máy học, cách sử dụng ngôn ngữ python và các công cụ kèm theo.

* + 1. Phương pháp nghiên cứu.

Nghiên cứu về ý nghĩa thuật toán, công thức toán học của thuật toán phân cụm Kmeans và thực hiện trên ngôn ngữ lập trình python.

Nghiên cứu triển khai các thuật toán về tìm K tối ưu cho thuật toán phân cụm Kmeans. Áp dụng các công cụ, thư viện trong ngôn ngữ lập trình python để thực hiện, mô tả các phương pháp nói trên bằng biểu đồ.

Thử nghiệm trên một số mẫu dữ liệu có sẵn. Lấy giá trị K có được nhờ các phương pháp trên. Áp dụng phân cụm Kmeans và đánh giá kết quả.

* 1. Bố cục báo cáo

Nội dụng báo cáo bao gồm các chương sau:

* Chương I. Giới thiệu tổng quan đề tài: Giới thiệu tổng quan, mục tiêu và phương pháp nghiên cứu đề tài.
* Chương II. Cơ sở lý thuyết. Giới thiệu về machine learning và ngôn ngữ lập trình cùng với các công cụ hỗ trợ.
* Chương III. Thuật toán phân cụm Kmeans. Giới thiệu tổng quan về thuật toán phân cụm Kmeans. Các công thức toán học đằng sau Kmeans và thực thi thuật toán. Đồng bàn về vấn đề khởi tạo tâm ban đầu K-means++ và về chọn số k tâm ban đầu.
* Chương IV. Xây dựng phương pháp tìm K tối ưu: Xậy dựng hàm Kmeans. Giới thiệu, nêu thuật toán của các phương pháp và áp dụng các phương pháp đó vào hàm Kmeans.
* Chương V. Kiểm thử: Sử dụng hàm Kmeans và các phương pháp tìm K tối ưu vào tập dữ liệu thử.
* Chương VI. Kết luận: Đưa ra các đánh giá về các phương pháp. Nêu ra những kết quả đạt được và hạn chế của đề tài.

1. CƠ SỞ LÝ THUYẾT
   1. Giới thiệu lý thuyết về Machine Learning

Cuộc cách mạng công nghiệp 4.0 với bằng chứng là sự nổi lên của AI - Artificial Intelligence (Trí tuệ nhân tạo), cụ thể hơn là Machine learning(Học máy) đã làm thay đổi thể giới(1 – Động cơ hơi nước, 2 – Điện năng, 3 – công nghệ thông tin). Ta có thể thầy rằng là trí tuệ nhân tạo đang len lõi vào mọi mặt của đời sống của chúng ta. Một số ví dụ cho AI trong cuộc sống mà mọi người có thể dễ nhận thấy như là: Hệ thống đề xuất video trên theo sở thích người dùng trên Youtube, xe tự hành của Tesla hay Camera AI trên các mẫu điện thoại thông minh đời mới. Và đặc biệt là trị tuệ nhận tạo AlphaGo, trong một trận đấu cờ vây, đã có thể chiến thắng trước Lee Sedol – Người từng 18 lần vô địch thế giới trong bộ môn này. Hay gần đây hơn là OpenAI có khả năng chơi game Dota2 chiến thắng trước những tuyển thủ chuyên nghiệp hàng đầu thế giới. Từ những vị dụ trên đã cho thấy được sự phát triển không ngừng của AI trên mọi lĩnh vực trong một kĩ nguyên mới.

Machine learning hay học máy(máy học) là một lĩnh vực con thuộc Trị tuệ nhận tạo(AI) nhằm nghiên cứu và sử dụng các thuật toán giúp máy tính hoặc các hệ thống có thể “học” từ dữ liệu nhằm giải quyết những vấn đề cụ thể. Các thuật toán máy học cần được đưa vào một lượng dữ liệu cần thiệt để cho ra một mô hình chính xác. Các tập dữ liệu này sẽ được thu gom, làm sạch và chọn một thuật toán máy học để đưa ra một mô hình mô tả bộ dữ liệu này. Do tập dữ liệu đầu vào là khác nhau nên các thuật toán máy học sẽ có những phương pháp và kỹ thuật “học” khác nhau. Các thuật toán máy học có thể được chia làm thành 3 nhóm:

* Học có giám sát(Supervised Learning): là một phương pháp học mà ở đó các tập dữ liệu đầu vào đã được gán nhãn và nhiệm vụ là dữ báo đầu ra của một bộ dữ liệu mới tương tự. Học có giám sát có thế được phân loại thành hai bài toán nhỏ hơn đó là Phân loại(Classification) và Hồi quy(Regession). Bài toán Phân loài là bài toán mà các nhãn của bộ dữ liệu đầu vào thuộc vào một số nhóm hữu hạn. Bài toán Hồi quy là bài toán mà các nhãn của bộ giữ liệu đầu vào là các giá trị cụ thể nào đó.
* Học không có giám sát(Unsupervised Learning): Là một phương pháp học mà dữ liệu đầu vào khi đó chưa được dán nhãn. Phương pháp này sẽ dựa vào tập dữ liệu đó để có thể thực hiện một công việc nào đó như phân cụm dữ liệu hoặc tìm ra các quy luật hay nguyên tắc cho bộ dữ liệu đó. Do đó sẽ có hai bài toán con trong Học không có giám sát là Phân cụm dữ liệu(Clustering) và tìm Quy tắc kết hợp(association rules). Thuật toán Phân cụm dữ liệu là phân nhóm các điễm dữ liệu dữa trên các tính chất đặc điểm tương tự nhau. Quy tắc kết hợp là dựa vào tập dữ liệu đã cho để khai phá, tìm ra những quy luật, quy tắc nào đó.
* Học tăng cường(Reinforcement Learning): là một phương pháp học tiên tiến nhằm tự xác định các hành động tiếp theo sao cho đạt được hiệu quả hay lợi ích cao nhất. Bài toán này có thể dễ dàng thấy được trong hệ thống xe tự hành.
  1. Giới thiệu về ngôn ngữ lập trình.
     1. Python và phần mềm Jupyter lab

Python là một ngôn ngữ lập trình bậc cao sử dụng cho nhiều mục đích khác nhau do Guido Van Rossum tạo ra và ra mắt lần đầu tiên vào năm 1991. Đặc điểm của Python là một ngôn ngữ mạnh mẽ, dễ học, dễ sử dụng. Nó chứa các cấu trúc câu lệnh tối giản, đơn giản và rõ ràng giúp việc học tập và trao đổi lập trình trở nên dễ dàng.

Python là cộng cụ tuyệt vời cho việc lập trình các thuật toán máy học. Ngôn ngữ này cho phép việc lập trình trở nên ngắn ngọn và dễ hiểu. Đồng thời có chứa nhiều các hàm và thư viện được xây dựng sẵn trong Python. Trong khi đó, máy học và trí tuệ nhân tạo chứa nhiều các thuật toán và các công thức toán học phức tạp cùng với lượng dữ liệu lớn nhằm phục vụ nó. Sự đơn giản của Python giúp lập trình viên tập trung vào các vấn đề của máy học thay vì là các kĩ thuật sử dụng ngôn ngữ lập trình.

Python chứa nhiều các thư viện hỗ trợ việc tính toán và nghiên cứu máy học. Việc thực thi các thuật toán máy học và sử dụng dữ liệu lớn có thể sẽ phức tạm và tốn nhiều thời gian. Trong khi đó, Python cung cấp môi trường làm việc, thư viện hỗ trợ đã được lập trình và có hiệu năng mạnh mẽ. Việc này sẽ giúp tiết kiệm được nhiều vấn đề khác nhau và giúp quá trình lập trình diễn ra được tốt nhất.

Project Jupyter là một dữ án nhằm để phát triển phần mềm mã nguồn mở, các tiêu chuẩn mở và các dịch vụ cho tính toán tương tác trên nhiều ngôn ngữ lập trình khác nhau. Jupyter Notebook là một ứng dụng trên web mã nguồn mở cho phép tạo và chia sẽ các tài liệu có chưa mã lập trình trực tiếp, các phương trình, hình ảnh và văn bản tường thuật. Jupyter lab là một môi trường phát triển có tính tương tác cao trên web dành cho Jypyter Notebook, lập trình và dữ liệu. JupyterLab có tính linh hoạt cao, giao điện người dùng đã được tinh chỉnh và sắp xếp lại để hỗ trợ cho nhiều quy trình làm việc khác nhau trong khoa học dữ liệu, khoa học máy tính và máy học. JupyterLab có thể được mở rộng và tích hợp nhiều các tính năng phụ trợ khác nhau tuy theo nhu cầu của người dùng.

Anaconda là một phần mềm miễn phí, giúp dễ dàng tải về và quản lý các gói tài nguyên, môi trường làm việc và các phiên bản Python. Anaconda cung cấp JupyterLab miễn phí và người dùng có thể dễ dàng tải vệ những thư viện mà mình cần.

Đề tài này sẽ sử dụng Anaconda JupyterLab với Python phiên bản 3.8.5 cùng với các thư viện dưới đây để chạy hầu hết các phần lập trình.

* + 1. Thư viện numpy

Numpy là một thư viện mã nguồn mở thuộc Python dùng để làm việc với các mảng và ma trận lớn, nhiều chiều, cùng với số lượng lớn các hàm toán học cấp cao để hoạt động trên các mảng này. Nó hỗ trợ nhiều hàm phục vụ cho việc thực hiện các phép biến đổi, tinh toán trong đại số tuyến tính và ma trận. Numpy được xây dựng bởi Tavis Oliphant vào năm 2005. NumPy là viết tắt của Numerical Python.

Một đặc điểm của Numpy là các mảng Numpy được lưu trữ liên tục lại một nơi trong bộ nhớ, làm cho tốc độ truy cập và xữ lý trở nên hiệu quả. Tóc độ sử lý này nhanh hơn nhiều so với kiểu dữ liệu List được viết sẳn của python, kiểu dữ liệu List lưu trữ dữ liệu rời rạc trên bộ nhớ. Việc này rất hữu ích cho tiệc tính toán các phép tính có khối lượng dữ liệu lớn và nhiều chiều.

* + 1. Thư viện pandas

Pandas là một thư viện mã nguồn mở của Python. Đây là một cộng cụ mạnh mẽ, nhanh nhẹn, linh hoạt và dễ dàng sử dụng để thực hiện phân tích và xữ lý dữ liệu. Pandas hỗ trợ nhiều hàm giúp phân tích, làm sạch, đọc và xữ lý dữ liệu. Pandas được tạo ra bởi Wes McKinney vào năm 2008. Pandas là công cụ tuyệt vời trong ngành khoa học dữ liệu.

Pandas là một cộng cụ tuyệt vời cho tiền xữ lý dữ liệu. Việc đọc file dữ liệu lớn với python trở nên rất dễ dàng với hàm pandas.read\_csv(). Đây là một ví dụ cho thậy sự tối ưu của thư viện pandas nói riêng và Python nói chung cho sự tiện lời và mạnh mẽ của nó. Đồng thời nó còn hỗ trợ các hàm xữ lý dữ liệu khác như pandas.drop(): bỏ một cột hay dòng nào đó, pandas.isnull(): Kiểm tra xem trong dữ liệu có phần từ nào bị trống không,…

* + 1. Thư viện matplotlib

Matplotlib là một thư viện mã nguồn mở Python giúp hỗ trợ vẽ đồ thị. Đây là một cộng cụ rất hữu ích khi làm việc với Python và numpy. Module được sử dụng nhiều nhất của Matplotib là Pyplot cung cấp giao diện như MATLAB.

* + 1. Lưu ý

Để tiện cho viêc phần thực thi thuật toán của các chương sau. Đoạn code dưới dây sẽ gọi các thư viện đã được đề cập ở trên và sẽ được sử dụng cho suột bài báo cáo này:

1. import numpy as np
2. import pandas as pd
3. import random as rd
4. import matplotlib.pyplot as plt
5. import matplotlib.cm as cm

Đồng thời các phần thực thi sẽ viết theo dạng lập trình hướng đối tượng. Thế nên phân lớp chính sẽ được biết trong lớp KMean, lớp này được khởi tạo trong Kmean như sau:

1. class KMean():
2. def \_\_init\_\_(self, X, max\_iters=50, plot\_steps=False, init='kmeans++'):
3. self.X=X
4. self.n,self.d = X.shape
5. self.max\_iters=max\_iters
6. self.plot\_steps = plot\_steps
7. self.initial = init
8. # X là dữ liệu đầu vào
9. # max\_iters là số lần lặp tối đa trong hàm Kmean, mặc định là 50
10. # plot\_steps dùng để xác định xem có in ra các lần chạy của hàm Kmean hay không.
11. # init phương thức khởi tao tâm ban đầu, mặc định là kmeans++.
    1. Một số vấn đề về dữ liệu
       1. Tập dữ liệu kiểm thử

Vì dữ liệu rất quan trọng trọng việc đánh giá xem việc một thuật toán máy học có chính xác hay không. Để đảm bảo chính xác nhất thì dữ liệu phải được thu gom lại và làm sách, đồng thời tập dữ liệu đó phải thích hợp với thuật toán mình đang triển khai.

Trong bài báo cáo này sẽ sử dụng dữ liệu sinh ra thử thư viện. Do các điểm dữ liệu trong tập dữ liệu được sinh ra bằng thư viện mang sự nhất quan với nhau rất cao, cho nên kết qua cho ra thường chính xác theo kì vòng. Đồng thời số liệu của mỗi điễm dữ liệu là đủ nhỏ để cho tốc độ tính toán cao. Thông qua các thông số cụ thể thì ta có thể điều chỉnh đườc các thông số của bộ dữ liệu thử như: số cụm, phương sai, số lượng dữ liệu,…. Cho nên việc đánh giá sẽ trở nên đúng đắn và dễ dàng hơn.

Tuy vậy thì số lượng điễm dữ liệu cho mỗi cụm tạo ra bằng cách trên là tương tự nhau. Cho nên khi xuất lên biều đồ thì sẽ cân bằng hơn và dễ hiệu hơn. Những khi sử dụng các tập dữ liệu khác với số lượng điểm mỗi cụm không cân bằng thì khi xuất lên biều đồ có thể sẽ khác những nhìn chung thì kết qua vẫn sẽ có thể chính xác.

* + 1. Tối ưu hóa dữ liệu

Đối với thuật toán Kmeans thì bộ dữ liệu đầu vào thì kiểu dữ liệu phải là các số thuộc tập số thực hoặc dữ liệu ở dạng liên tục. Các số này không được quá lớn vì các số này sẽ khiên máy tính toán nặng hơn khi áp dụng các công thức toán học, gây lãng phí tài nguyên máy tính. Trong bộ dữ liệu không có các điểm dữ liệu trống sai định dạng.

Khi áp dụng những bộ dữ liệu thức tế thì dữ liệu cần phải được làm sạch. Bộ dữ liệu đầu vào có thể sẽ có những cột mà ở đó các dữ liệu có thể là các giá liên tục hoặc rời rác, dữ liệu mang số quá lớn hay sai định dạng. Khi đó để thuật toán có thể sử dụng được thì dữ liệu cần phải được làm sạch. Sau đây là một số phương pháp để làm sạch dữ liệu.

Đối với dữ liệu bị sai định dạng: Ta cần chuyên định sang sai đó thành định dạng đúng hoặc xóa điểm dữ liệu đó.

* Ví dụ: Môt điểm dữ liệu có dạng x=(2,’415’) thì ‘415’ không phải là một số nguyên dương mà là kiểu dữ liệu dạng chuỗi(string). Khi đó ta phải chuyển về thành x =(2,415).

Đối với dữ liệu dạng rời rạc: Ta cần chuyển định dạng dữ liệu đó về dạng liên tục.

* Vị dụ: Nhãn của cột giới tính có hai giá trị là ‘Nam’ và ‘Nữ’. Với kiểu dữ liệu này thì thuật toán khó có thể hiểu được. Khi đó ta có thể sử dụng phương pháp LabelEncoder để chuyên các nhãn này thành các kí tự số. Theo đó ta có thể chuyển ‘Nam’🡪1 và ‘Nữ’ 🡪 0.

Đối với dữ liệu là các số thực những các giá trị của nó là quá lớn hay một cột dữ liệu có khoảng giá trị quá lớn 1-10000 và cột khác có khoảng giá trị quá nhỏ 0-10. Lúc này ta cần phải chuẩn hóa dữ liệu. Có 2 phương pháp chính để tối chuẩn hóa dữ liệu là:

* Chuyển khoảng giá trị: là phương pháp rất đơn giản để đưa các giá trị của các cột về cùng một khoảng. Để có thể đưa điểm dữ liệu thứ thuộc cột dữ liệu **.** Ta có thể sử dụng công thức sau:

Với và lần lượt là giá trị thứ của cột dữ liệu trước và sau khi được chuẩn hóa. và lần lượt là giá trị lớn nhất và giá trị thấp nhất của cột dữ liệu **.**

Code thực thi:

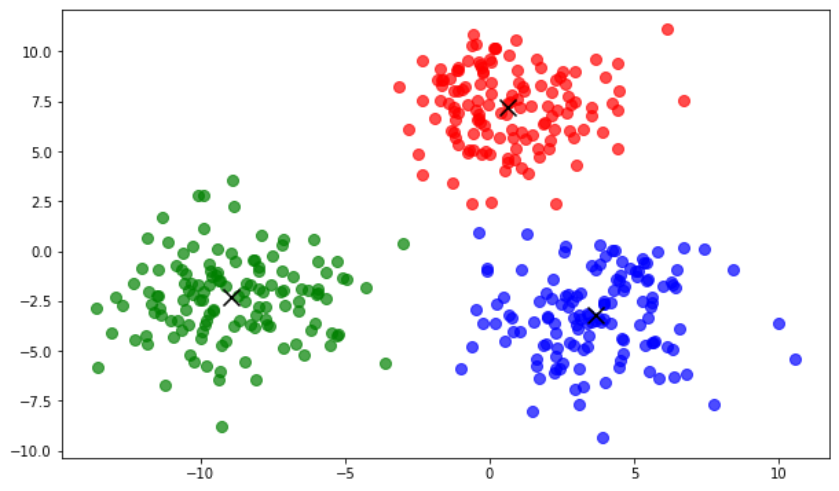
1. def normalize(X):
2. n,d=X.shape
3. temp=np.zeros((n,d))
4. for i in range(d):
5. temp[:,i]=(X[:,i]-np.min(X[:,i]))/(np.max(X[:,i])-np.min(X[:,i]))
6. return temp

* Chuẩn hóa theo phân phối chuẩn:

Code thực thi:

1. def normalize\_std(X):
2. n,d = X.shape
3. temp = np.zeros(shape=(n,d))
4. for i in range(d):
5. mean = np.mean(X[:,i])
6. std = np.std(X[:,i])
7. temp[:,i] = (X[:,i]-mean)/std
8. return temp
9. THUẬT TOÁN PHÂN CỤM KMEANS
   1. Thuật toán phân cụm K-means
      1. Giới thiệu thuật toán phân cụm Kmeans

Phân cụm là nhiệm vụ phân chia dân số hoặc các điểm dữ liệu thành một số nhóm sao cho các điểm dữ liệu trong cùng một nhóm giống với các điểm dữ liệu khác trong cùng một nhóm hơn các điểm dữ liệu trong các nhóm khác. Nói cách đơn giản, mục đích là để tách các nhóm có đặc điểm giống nhau và gán chúng thành các cụm. Mục tiêu của thuật toán Kmean là tìm các nhóm trong dữ liệu, với số lượng nhóm được đại diện bởi biến K. thuật toán hoạt động lặp đi lặp lại để gán mỗi điểm dữ liệu cho một trong k nhóm dựa trên các tính năng được cung cấp. Trong hình ảnh tham khảo bên dưới, k = 3, và có ba cụm được xác định từ tập dữ liệu nguồn.



Hình 1. Bài toán phân cụm với số K=3

* + 1. Phân tích toán học:

Cho tập dữ liệu với d là số chiều của vector , Kmean mục tiêu là chia N số lượng phần từ trong thành KN cụm và các tâm cụm sao cho cho giá trị hạm mất mát. Hàm mất mát được tính như sau:

Xét mỗi điểm dữ liệu được phân vào cụm thì sẽ sai số là:

Khi đó ta sẽ tính sai số cho tất cả các điểm trong mỗi cụm. Tương tự với số cụm K= 1 có thì tổng sai số của cụm này đươc tính như sau:

Tương tự khi có K cụm và dữ liệu được đã được gán nhãn và phân vào mỗi cụm ta có:

Vậy khi đó ta sẽ có được ma trận nhãn và tâm cụm

* + 1. Tính khoảng cách Euclid:

Cho a, b là 2 điểm dữ liệu, khoảng cách Euclid từ điểm a đến điểm b:

Với a, b có số chiều lớn hơn 1 ta có:

Với số chiều là 2:

Với số chiều là n:

* + 1. Giải thích thuật toán

Thuật toán Kmeans có để được thực hiện như sau.

* Đầu vào: Ma trận dữ liệu và số lượng tâm KN cần tìm
* Đầu ra: Ma trận tâm cụm , và ma trận nhãn .

1. Khởi tạo K điểm bất kì làm các tâm khởi tạo ban đầu của **M**.
2. Tính khoảng cách Euclid từ các điểm đến tâm **M**.
3. Gán nhãn cho mỗi điểm dữ liệu vào cụm có tâm gần nó nhất.
4. Nếu việc phân cụm dữ liệu ở Bước 3. Không thay đổi so với vòng lặp trước thì kết thúc thuật toán
5. Cập nhật các tâm cụm bằng cách lấy trung bình cộng các điểm dữ liệu đã được gán nhãn.
6. Quay lại bước 2.
   * 1. Thực thi:
7. def kmean\_random\_centroids(self,k):
8. return self.X[np.random.choice(self.n,k,replace=False)]
10. # Tinh khoang cach Euclid tu cac diem den cac tam
11. def kmean\_euclid\_dis(self,k):
12. distances=np.zeros((self.n,k))
13. for i in range(k):
14. for j in range(self.d):
15. distances[:,i]+=(self.X[:,j]-self.centroids[i,j])\*\*2
16. distances=np.sqrt(distances)
17. return distances
19. # lay nhan cua cac diem gan voi cac tam nhat
20. def kmean\_assign\_labels(self,k):
21. distances = self.kmean\_euclid\_dis(k)
22. return np.argmin(distances,axis=1)
24. # tinh lai vi tri cac tam
25. def kmean\_update\_centroids(self,k):
26. return np.array([np.mean(self.X[self.labels == i,:],axis=0) for i in range(k)])
28. # Kiem tra xem 2 tam co trung nhau khong
29. def kmean\_check\_centroids(self, new\_centroids):
30. return (set([tuple(i) for i in self.centroids])==set([tuple(i) for i in new\_centroids])) # Vu Huu Tiep book
32. # Tao cac Cluster
33. def kmean\_create\_clusters(self):
34. self.clusters = [self.X[self.labels == i,:] for i in range(len(self.centroids))]
35. return self.clusters
37. # Plot kmean
38. def kmean\_plot(self):
39. fig, ax = plt.subplots(figsize=(10, 6))
40. defi\_color=['b','g','r','c','m','y','k','b','g','r','c','m','k','w','b','g','r']
41. for i in range(len(self.centroids)):
42. ax.scatter(self.clusters[i][:,0],self.clusters[i][:,1],marker='o',color=defi\_color[i],alpha=0.7,s=8\*\*2)
43. ax.scatter(self.centroids[i,0],self.centroids[i,1],marker='x',color='k',s=12\*\*2)
44. plt.show()
45. # Chay thuat toan Kmean
46. def fit(self,k,plot\_steps=True):
47. if self.initial.lower() in ['random','rd']:
48. self.centroids=self.kmean\_random\_centroids(k) # Thiet lap tam ngau nhien k tam nhau nhien ban dau
49. elif self.initial == 'kmeans++':
50. self.centroids=self.kmean\_plus\_plus(k)
51. for \_ in range(self.max\_iters):
53. self.labels=self.kmean\_assign\_labels(k)
55. self.clusters=self.kmean\_create\_clusters()
56. if plot\_steps and self.plot\_steps:
57. self.kmean\_plot()
59. new\_centroids = self.kmean\_update\_centroids(k)
60. if self.kmean\_check\_centroids(new\_centroids):
61. break
62. self.centroids=new\_centroids
64. return self.centroids, self.labels
    * 1. Nhược điểm của Kmean:

Kmean không phải là một thuật toán máy học hoàn hảo để phân cụm dữ liệu. Dễ thấy rằng Kmean tính khoảng cách Euclid để tìm ra nhãn của tập dữ liệu và các tâm cụm. Đồng thời dựa vào thuật toán trên thì ta có thấy được một số nhược điểm đối với Kmean như sau:

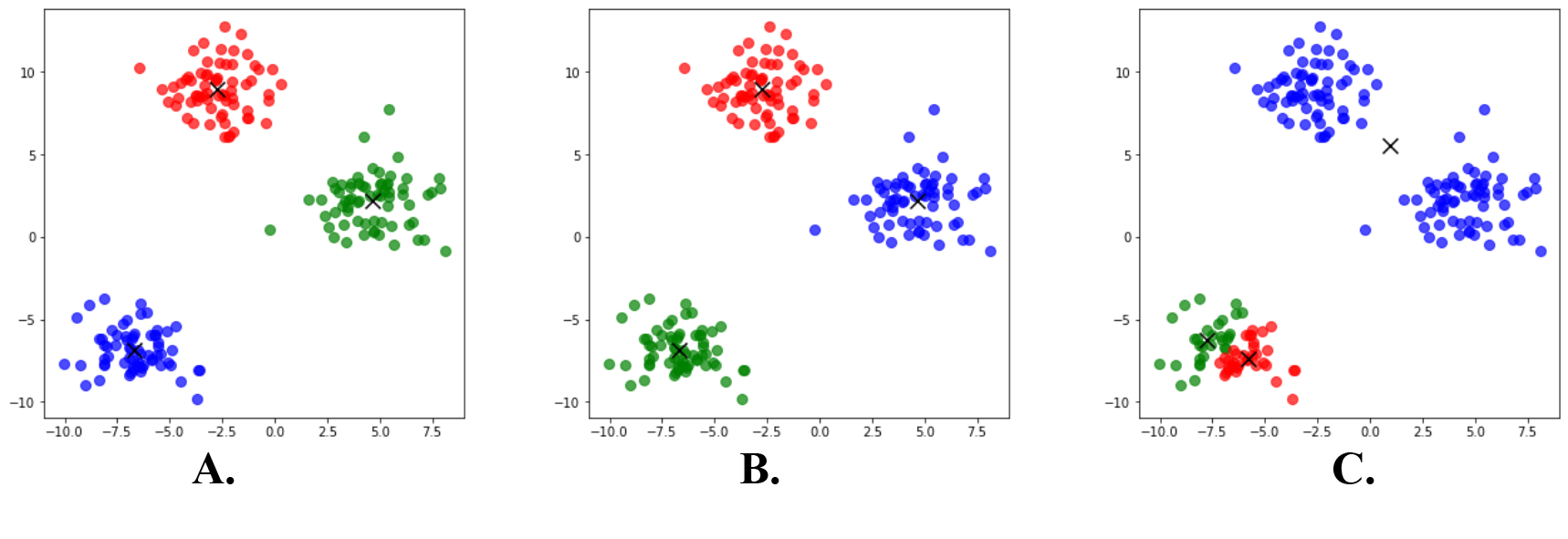
Thứ nhất là vấn đề chọn số lượng tâm cụm(K). Phân này sẽ được giải thích cụ thể hơn ở phần III.3 và đây chính là vấn đề chính trong bài báo cáo này.

Thứ hai là thuật toán Kmean không thế xữ lý được được những dữ liệu bị nhiễu và các ngoại lệ. Đây chính là vấn đề chính khiến độ chính xác của thuật toán này thấp. Do kết quả tính khoảng cách là chính xác tuyết đối và Kmean cần tính cho toàn bộ các điểm nên rất khó để có thể xử lý các điểm này. Việc dữ liệu có nhiễu và các ngoại lệ là không thể tránh khỏi trọng việc thu thập dữ liệu trong thực tế. Cho nên muốn cho thuật toán này mang lại độ chính xác cao thì dữ liệu, một là được thu thập cẩn thận sao cho không hoặc ít xuất hiện các điểm nhiễu hay ngoại lệ. Hai là sử dụng các phương pháp đặc biệt để có thể loại bỏ nó.

Cuối cùng là Kmean chỉ có thể phân cụm được những cụm dữ liệu dạng lồi. Vì Kmeans sử dụng phương pháp tính khoảng các giữa các điễm và lấy những điễm gần nhất với các tâm. Cho nên, hình dạng của càng cụm thường có dạng lồi hay các cụm có dạng tròn gần nhau. Cho nên nếu hình dạng của cụm khác đi hay tuân theo một hình dạng khác thì Kmean khó có thể xữ lý được. Khi đó ta có thể sử dụng các phương pháp phân cụm khác như DBCAN,…

* 1. Kmeans++

Trong bài toán Kmeans, ta phải lựa chọn K tâm ban đâu để thuật toán có thể điều tâm và phân cụm. Lựa chọn tâm theo cách chọn nhẫu nhiễn K điểm dữ liệu đã cho ban đầu là một cách đơn giản và nhanh chống. Tuy nhiên phương pháp này nay sinh ra một số vấn đề có thể gây ảnh hưởng đến kết quả cuối cùng cho bài toán phân cụm. Các tâm được sinh ra khác nhau khiến cho một nghiệm của bài toán là khác nhau. Trong trường hợp tệ nhất là khiên cho bài toán phân cụm bị lệch.



Hình 2. Đầu ra của bài toán phân cụm với mỗi sự lựa chọn tâm khác nhau

Trong hình trên ta có thể thấy: Hình A. B. là có sự khác nhau về màu trong 2 cụm. Riêng hình C. thì các cụm được phân bố khác đi so với hình A. và B.. Dễ thấy rằng, sự phân cụm như hình C. là có sự sai lệch. Sự lựa chọn tâm ngẫu nhiên quá gần nhau đã dẫn đến điều này.

Đề làm giảm đi sự sai lệch do khởi tao tâm ban đầu, ta có thể khởi tạo các tâm ban đầu sử dụng phương pháp Kmeans++. Đây là phương pháp được David Arthur và Sergei Vassilvitskii viết trong một bài báo nói về ưu điểm của việc gieo hạt cẩn thận(“k-means++: The Advantages of Careful Seeding”). Đây là phương pháp giúp việc lựa chọn tâm ban đầu được giàn trải hơn so với phương pháp lựa chọn tâm ngẫu nhiên cho toàn bộ các tâm.

Thuật toán được mô tả như sau:

1. Lựa chọn một tâm ngẫu nhiên là một điểm thuộc tập dữ liệu đầu vào
2. Với mỗi điểm **x**, xác định D(**x,c**) với D(**x,c**) là khoảng cách tối thiểu từ **x** đến tâm **c**.
3. Chọn tâm **c** tiếp theo thuộc **X** sao cho tỉ lệ giữa khoảng cách d(x,ci)2 với tổng khoảng cách đối với tâm gần nhất là cao nhất
4. Lặp lại bước 2, 3 cho đến khi đủ số lượng tâm cần tìm.

Áp dụng thuật toàn này vào nghiên cứu sẽ giúp hạn chế dược sai số trong quá trình thử nghiệm các thuật toán. Tuy vậy Kmeans++ sẽ có tốc độ chậm hơn so với việc lựa chọn ngẫu nhiên cho toàn bộ các tâm do phải thực hiện tính toán lại khoảng cách các điểm với các tâm. Điều này thậm chí còn tệ hơn nếu số K lớn.

Trong suốt bài báo cáo này, các tâm ban đầu sẽ được khởi tạo ngẫu nhiên bằng thuật toán Kmeans++ để tăng độ chính xác và hiệu suật bài toán mặc dù tốc độ thực thi có thể sẽ chậm hơn.

Code thực thi cho Kmeans++:

1. # lay k tam theo thuat toan Kmeans++
2. def kmean\_plus\_plus(self,k):
3. centroids\_temp = []
4. centroids\_temp.append(self.X[np.random.choice(self.n)])
5. for \_ in range(k-1):
6. dist= []
7. for i in range(self.n):
8. data = self.X[i,:]
9. temp\_dist=[]
10. for centroid in centroids\_temp:
11. temp\_dist.append(np.sum((data - centroid)\*\*2))
12. dist.append(np.min(temp\_dist))
13. centroids\_temp.append(self.X[np.argmax(dist),:])
14. centroids\_temp = np.array(centroids\_temp)
15. return centroids\_temp
    1. Vấn đề chọn số lượng tâm với Kmeans.

Muốn cho thuật toán phân cụm Kmeans phân chia được thì ta cần phải biết được số cụm cần thiết để đưa vào. Câu hỏi đặt ra là: với một tập dữ liệu đã có sẵn thì phân bao nhiêu cụm là hợp lý, tối ưu? Đây là câu hỏi phổ biến nhất trong thuật toán phân cụm Kmeans.

K tối ưu là số lượng cụm K sao cho sự khác biết giữa cụm này và cụm khác là thấp nhất so với tổng thể bài toán.

Đối với một bộ dữ liệu đẹp, tức là tập dữ liệu có số chiều nhỏ và số lượng dữ liệu ít. Thì ta có thể dễ dàng biết được số cụm K cần phân chia nhờ vào cảm quan. Tuy nhiên số k này có thể được chia khác nhau cho một tập dữ liệu. Vì các cụm trong Kmeans được chia sao cho các điểm trong mỗi cụm là giống nhau nhất có thể. Sự giống nhau này càng tăng thì số cụm càng tăng. Cho nên, số lượng cụm tối đa trọng một bài toán có thể bằng số lượng điểm dữ liệu trong tập dữ liệu. Cho nên ta cần phải đi tìm K tối ưu cho tập dữ liệu đó.

Trong khi đó với bộ dữ liệu có số chiều nhiều và lượng dữ liệu lớn thì việc tìm k bằng cảm quan sẽ không còn hiệu quả nữa. Khi đó ta chỉ có thế biết được k bằng hai cách. Thứ nhất là số k này đã được biết trước cho tập dữ liệu đã cho. Khi đó ta chỉ cần áp dụng thuật toán phân cụm để lẫy nhãn của từng điểm dữ liệu. Thứ hai là khi ta chưa biết được số tâm k cho sẵn, khi đó ta phải đi tìm kiếm số K tối ưu. Chương sau sẽ trình bày 3 phương pháp để tìm k tối ưu cho thuật toán Kmeans.

Trong thực tế việc chọn số tâm K sẽ còn dựa vào nhiều yếu tô khác. Đối với dữ liệu dạng 2-d hay 3-d thì ta có thể dễ hàng hình dung được hình dạng của tâm. Tuy nhiên khi số chiều dữ liệu d từ 4 trở lên thì ta chỉ có thể tự hình dung. Khi đó thì cần thiệt phải có sự phân tích dữ liệu trong tập dữ liệu đã có được. Phân tích ở đầy có thể là phân tích dưa trên mỗi tương quan của các thuộc tính với nhau. Một thuộc tình này có thể tương tác mạnh với thuộc tính kia. Đối với một số trường hợp ta có thể gán thêm trọng số đối với một số thuộc tính quan trọng dẫn đến phân cụm. Ngoài ra ta có thể sử dụng phương pháp như PCA để có thể giảm được chiều dữ liệu, nếu có thể giảm số chiều về 2,3 thì có thể biểu thị ra và phân tích.

Việc chọn K còn dựa trên mục đích của phân cụm. Với mỗi bộ dữ liệu mà số chiều trong nó là lớn thì việc phân tích K có thể gặp nhiều khó khăn hơn. Khi đó ta có thể dựa vào các phương pháp và phân tích dữ liệu để có thể chọn ra những đặc tính nổi bật trong dữ liệu, theo đó có cái hình tông quát hơn về dữ liệu. Theo đó ta có thể chọn được số K phù hợp để phân cụm.

1. XÂY DỰNG CÁC PHƯƠNG PHÁP TÌM K TỐI ƯU
   1. Phương pháp Elbow

Phương pháp Elbow(Củi chỏ) là một trong những phương pháp nổi tiếng nhất và đơn giản mà ta có thể dùng để chọn giá trị K phù hợp và tăng độ chính xác của mô hình.

Ý tưởng chính của thuật toán Kmean như sau. Với một tập dữ liệu ban đầu, phân tập dữ liệu đó thành K cụm khác nhau. Với mối cụm sẽ có một tâm gọi là Centroid, ta tính tổng khoảng cách từ tâm này đến toàn bộ các điểm dữ liệu thuộc cụm của tâm đó. Tham số này gọi là WSS (Within-cluster Sum of Square). Sau đó ta tính tiếp tục tìm WSS cho các tâm còn lại là lấy tổng của nó. Tổng này gọi là TWSS(Total Within-cluster Sum of Square). Giá trị này có thể được tính như sau:

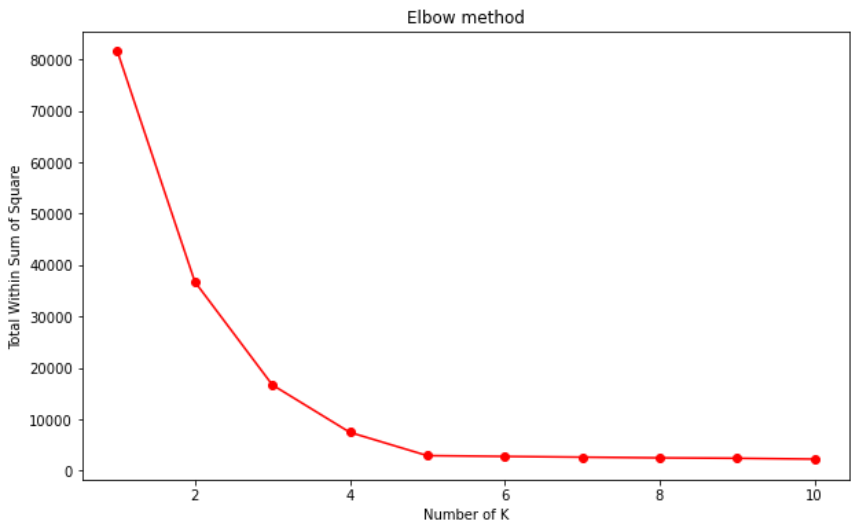
Vì mục tiêu của thuật toán phân cụm là phân chia toàn bộ các điểm của tập liệu ban đầu thành K cụm khác nhau sao cho các điểm dữ liệu thuộc cùng một cụm là giống với nhau nhất có thể, hay TWSS nhỏ nhất có thể, nên việc lựa chọn phương thức tính khoảng cách là rất quan trọng vì khoảng cách được sử dụng như là một thước đo mức độ tương đồng của các quan sát. Tuy vậy với số lượng K càng lớn thì các giá trị TWSS càng nhỏ dần và không có sự thay đổi lớn và dẫn vô nghĩa với số K rất lớn. Do đó ta nên chọn giá trị K cho TWSS tối ưu nhất.

Theo đó ta có quy trình triển khai Elbow method như sau:

* Đầu vào: Ma trận dữ liệu và khoảng số lượng NumK<N cần tìm.
* Đầu ra: TWSS của từng giá trị K trong khoảng NumK

1. Triển khai thuật toán phân cụm với số cụm K trong khoảng NumK bắt đầu từ 1.
2. Với mỗi giá trị K, tính giá trị TWSS tương ứng.
3. Biểu thị biểu đồ Elbow Method theo các giá trị TWSS của K đã tính ở bước 2.
4. Dựa vào biểu đồ trên, lấy điểm K sao cho giá trị TWSS của K đó không thay đổi quá nhiều so với trước.

Ta lựa cọn điểm K trong biểu đồ sao cho giá trị của nó có sự giảm không đáng kể với các điểm tiếp theo nó.



Hình 3. Ví dụ một mô hình Elbow method với tâm thích hợp là 5

Tuy vậy như đã nói ở các chương trước thì phân bố các điểm dữ liệu có thể là không được chuẩn hoàn theo như dữ liệu mô hình ở trên. Cho nên biểu đồ sẽ cho ra kết quả không rõ ràng. Do đó việc chọn tâm sẽ trở nên khó khăn hơn. Khi đó ta sẽ dựa vào cảm quan để có thể lựa chọn k tốt nhất trong mô hình.

Code thực thi:

1. # Nhom Elbow
2. # Ham elbow
3. def elbow\_method(self,num):
4. elbow\_values = np.zeros(num+1)
5. elbow\_values[0]=None
6. for i in range(1,num+1):
7. Centroids,labels=self.fit(i,plot\_steps=False)
8. for j in range(i):
9. elbow\_values[i]+=np.sum((self.X[labels==j,:]-Centroids[j])\*\*2)
10. self.elbow\_values=elbow\_values
11. return elbow\_values
13. # Hien thi cac gia tri cua elbow method
14. def Print\_elbow\_method(self):
15. try:
16. for i in range(1,len(self.elbow\_values)):
17. print('k={}:{}'.format(i,self.elbow\_values[i]))
18. except :
19. print("No values in elbow method yet!")
21. # Bieu do hoa cac gia tri cua elbow method
22. def Show\_elbow\_method(self):
23. try:
24. plt.plot(self.elbow\_values,'or-')
25. plt.show()
26. except:
27. print("No values in elbow method yet!")
    1. Phương pháp Silhouette

Phương pháp Silhouette(Hình bóng) là một phương pháp dùng để tính toán, đánh giá mức độ thích hợp của kĩ thuật phân cụm. Giá trị của có được gọi là hệ số Silhouette(Silhouette Coefficient) hay điểm Silhouette(ilhouette score). Giá trị này giao động từ trong khoảng [-1;1].

* Các giá trị càng gần 1 chứng tỏ cụm đó đã tách xa và phân biệt với các điểm thuộc cụm khác.
* Giá trị 0 mang ý nghĩa là cụm đó không có sự khác biệt giữa cụm này và cụm kia. Hoặc khoảng cách giữa các cụm là không khác biệt
* Giá trị càng về -1 thì có nghĩa là cụm đó đã bị phân cụm sai.

Chung quy lại nếu giá trị Silhouette càng lớn thì cụm đó đã được phận cụm chính xác

Giá trị Silhouette cho mỗi điểm có thể được tính như sau:

Với mỗi điểm dữ liệu (điểm dữ liệu i thuộc cụm đặt:

Với a(i) là tổng khoảng cách từ điểm dữ liệu i đến tất cả các điểm dữ liệu j thuộc sau cho i khác j chia cho số điểm dữ liệu thuộc trừ 1(không tính i vào số lượng). Ta có thể hiểu a(i) như là cách để đánh giá điểm dữ liệu i có thích hợp với cụm hay không. Giá trị a(i) càng nhỏ thì điểm dữ liệu i càng hợp với cụm và ngược lại.

Tiếp theo ta xác định sự khác biệt trung bình của điểm dữ liệu i đến các tâm tính trung bình khoảng cách giữa điểm dữ liệu i đến tất cả các điểm dữ liệu của các cụm không phải là cụm :

Giá trị nhỏ nhất ở đây tức là ta lấy trung bình tổng khoảng cách đến gần cụm nhất. Muốn làm vậy ta buộc phải tính trung bình khoảng cách của tất cả các cụm và chọn giá trị nhỏ nhất.

Sau đó ta có thể tính giá trị Silhouette bằng công thức sau:

Và gắn cho nếu số lượng của cụm, có rằng buộc này để ngắn việc số lượng cụm tăng lên một cách đáng kể và tránh gặp lỗi khi số lượng nhỏ bằng 0 khiến máy tính chia cho 0:

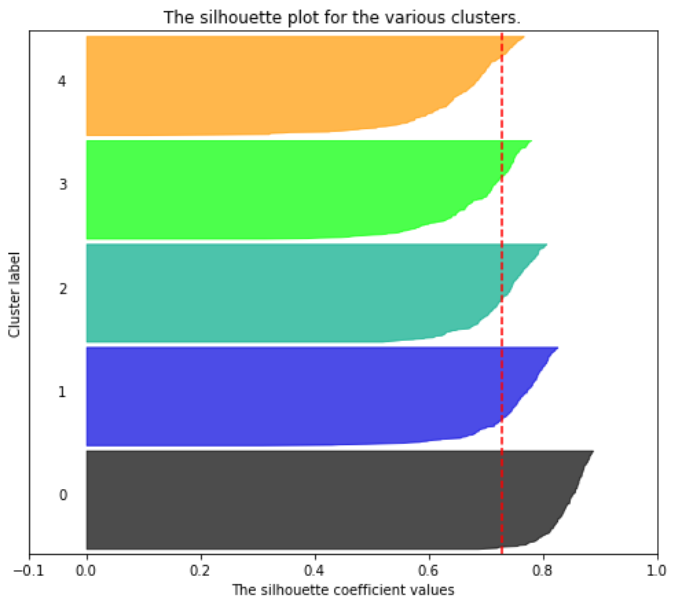
Ta có thể viết lại rằng

Do đó cho thấy:

Muốn cho giá trị về gần 1 thì cần phải có . Do là giá trị dùng để đánh giá sự không giống nhau của một phần tử trong cụm, a(i) càng nhỏ thì càng thích hợp. Và thêm nữa, là giá trị chỉ độ không phù hợp với cụm lân cận của nó, b(i) càng lớn thì càng không thích hợp. Nếu s(i) càng về -1 thì ngược lại. Tức giá trị s(i) càng lớn thì độ thích hợp của điểm đó với cụm là càng cao.

Để có thể biểu diễn ra biểu đồ thì ta tính giá trị Silhouette cho toàn bộ các điểm và phân cụm mỗi điểm ra. Đồng thời tính giá trị Silhouette trung bình cho cụm đó và biệu thị ra biểu đồ. Thông thường thì ta sẽ chọn K nào có biểu đồ Silhouette có giá trị trung bình cao và phân bố các giá trị s(i) đồng đều, ít/không có giá trị s(i) nào bé hơn 0 ở mỗi cụm.

Tương tự như Phương Pháp Elbow, việc mô hình sẽ không được đẹp và rõ ràng như dữ liệu kiểm thử. Do đó ta phải dựa vào cảm quan và các yêu tố liên quan khác để chọn được số K phù hợp.



Hình 4. Biêu đồ biểu thị các giá trị Silhouette.

Code thực thi:

1. # Nhom Silhouette
2. # Ham silhoutte
3. def silhouette\_method(self,num,Test=True):
4. self.sild\_num = num
5. Silhoue\_mean=np.zeros(num+1)
6. Silhoue\_mean[0]=Silhoue\_mean[1]=None
7. Silh\_k=[]
8. silh\_kmean\_values=[]
9. for k in range(2,num+1):
10. X\_centroids\_s, labels\_s=self.fit(k, plot\_steps=False)
11. silh\_kmean\_values.append([X\_centroids\_s, labels\_s])
12. Xlabels=np.concatenate((self.X,labels\_s.reshape(-1,1)),axis=1)
13. silh=np.zeros((self.n,2))
14. for index in range(self.n):
15. Si=self.X[index]
16. #Tách các cụm.
17. Atemp = Xlabels[Xlabels[:,self.d]==Xlabels[index,self.d]]
18. Btemp = Xlabels[Xlabels[:,self.d]!=Xlabels[index,self.d]]
19. #Chọn một giá trị ngẫu nhiên trong cụm và loại bỏ giá trị đã chọn trong cụm đó.
20. if Si in Atemp:
21. Atemp=np.delete(Atemp,np.where(Atemp==Si),axis=0)
22. if Si in Btemp:
23. Btemp=np.delete(Btemp,np.where(Btemp==Si),axis=0)
24. # Nếu số lượng phần tử trong cụm = 1 thì trả về 0 để tránh trường hợp chia cho 0.
25. if (Atemp.shape[0]-1)==0:
26. silh[Si]=0
27. continue
28. # Tổng khoảng cách từ i đến các phần tử j trong cụm
29. SumSa=0
30. for j in range(Atemp.shape[1]-1):
31. SumSa+=(Si[j]-Atemp[:,j])\*\*2
32. SumSa=np.sum(np.sqrt(SumSa))
33. SA = (1/(Atemp.shape[0]-1))\*SumSa
34. # Tổng khoảng cách từ i đến các phần tử không thuộc cụm i
35. othercl=np.delete(np.array(range(k)),int(Xlabels[index,self.d]))
36. # Biến lưu các khoảng cách để chọn giá trị thấp nhất.
37. TempBtemp =np.zeros(k-1)
38. for l in range(k-1):
39. SumSb = 0
40. Temp1=Btemp[Btemp[:,self.d]==othercl[l]] # lấy các phần tử thuộc các cụm khác cụm của i.
41. for j in range(Temp1.shape[1]-1):
42. SumSb+=(Si[j]-Temp1[:,j])\*\*2
43. SumSb=np.sum(np.sqrt(SumSb))
44. TempBtemp[l]=(1/Temp1.shape[0])\*SumSb
45. SB = np.amin(TempBtemp,axis=0)
46. silh[index]=[(SB-SA)/(np.max([SB,SA])),Xlabels[index,self.d]]
47. Silh\_k.append(silh)
48. Silhoue\_mean[k]=np.mean(silh[:,0])
49. self.Silhoue\_mean=Silhoue\_mean
50. self.Silh\_k\_values=Silh\_k
51. self.Silh\_kmean\_values= silh\_kmean\_values
52. return Silh\_k
53. # Hien thi cac gia tri cua silhoutte method
54. def Print\_silhoutte\_method(self):
55. try:
56. for i in range(2,len(self.silhouette\_values)):
57. print('k={}:{}'.format(i,self.silhouette\_values[i]))
58. except :
59. print("No values in silhoutte method yet!")

Hàm Plot:

1. def Show\_silhoutte\_method(self,s=True):
2. if s is False:
3. plt.plot(self.silhouette\_values,'or-')
4. plt.show()
5. if s is True:
6. for k in range(2,self.sild\_num+1):
7. fig, (ax1, ax2) = plt.subplots(1, 2)
8. fig.set\_size\_inches(18, 7)
9. ax1.set\_xlim([-0.1, 1])
10. ax1.set\_ylim([0, self.n + (k + 1) \* 10])
11. clusterer, cluster\_labels = self.Silh\_kmean\_values[k-2] #Km3.fit(k)
12. Texx=self.Silh\_k\_values[k-2]
13. y\_lower = 10
14. for i in range(k):
15. ith\_cluster\_silhouette\_values = Texx[Texx[:,1]==i]
16. ith\_cluster\_silhouette\_values= ith\_cluster\_silhouette\_values[:,0]
17. ith\_cluster\_silhouette\_values.sort()
18. size\_cluster\_i = ith\_cluster\_silhouette\_values.shape[0]
19. y\_upper = y\_lower + size\_cluster\_i
20. color = cm.nipy\_spectral(float(i) / k)
21. ax1.fill\_betweenx(np.arange(y\_lower, y\_upper),0, ith\_cluster\_silhouette\_values,facecolor=color, edgecolor=color, alpha=0.7)
22. # Label the silhouette plots with their cluster numbers at the middle
23. ax1.text(-0.05, y\_lower + 0.5 \* size\_cluster\_i, str(i))
24. # Compute the new y\_lower for next plot
25. y\_lower = y\_upper + 10 # 10 for the 0 samples
26. ax1.set\_title("The silhouette plot for the various clusters.")
27. ax1.set\_xlabel("The silhouette coefficient values")
28. ax1.set\_ylabel("Cluster label")
29. # The vertical line for average silhouette score of all the values
30. ax1.axvline(x=self.Silhoue\_mean[k], color="red", linestyle="--")
31. ax1.set\_yticks([]) # Clear the yaxis labels / ticks
32. ax1.set\_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])
33. # 2nd Plot showing the actual clusters formed
34. colors = cm.nipy\_spectral(cluster\_labels.astype(float) / k)
35. ax2.scatter(self.X[:, 0], self.X[:, 1], marker='o', s=30, lw=0, alpha=0.7,c=colors, edgecolor='k')
36. Kiểm Thử
    1. Thư viện sklearn và hàm make\_blops.

Để có thể chọn kiểm tra được tính chính xác và hiệu quả của bài toán, ta cần có một tập dữ liệu kiểm thử. Điều kiện để có một bộ dữ liệu kiểm thử đẹp là các cụm được phân tích rõ ràng, sữ phân tán không quá lớn, độ lớn của giá trị không quá cao và có sự nhất quán nhật định. Tuy nhiên các tập dữ liệu thực tế hiệm khi có được những bộ dữ liệu như vậy dể đánh giá. Do đó ta cần một phương án khác để thay thế.

Phương án thay thế đó là hàm make\_blops trong thư viện sklearn. Thư viện sklearn là một thư viện nổi tiếng trong Machine Learning và lập trình python. Nó chưa nhiều các hàm khác nhau để hỗ trợ việc tính toán, phân tích dữ liệu, các thuật toán máy học, tạo và xữ lý dữ liệu,...Trong đó có hàm make\_blops.

Hàm make\_blops là một hàm giúp tạo ra số lượng tùy ý các cụm theo hàm phân phối chuẩn Gaussian giúp hỗ trợ kiểm tra các bài toán phân lớp. Trong hàm này có các đại lượng có thể tùy chỉnh như:

* n\_sample: số lượng điểm dữ liệu đưa vào
* n\_features: Số lượng thuộc tính hay số nhiều đưa vào.
* centers: số lượng tâm hay cụm tạo ra
* cluster\_std: độ lệch chuẩn theo phân phối Gaussian, kiểm soát được phân bộ hay độ rộng của mỗi cụm.
* random\_state: Xác định việc tạo ngẫu nhiên của tập dữ liệu, nếu đưa vào cùng một số và các tham số khác dữ nguyên thì đâu ra của hàm luôn giống nhau.

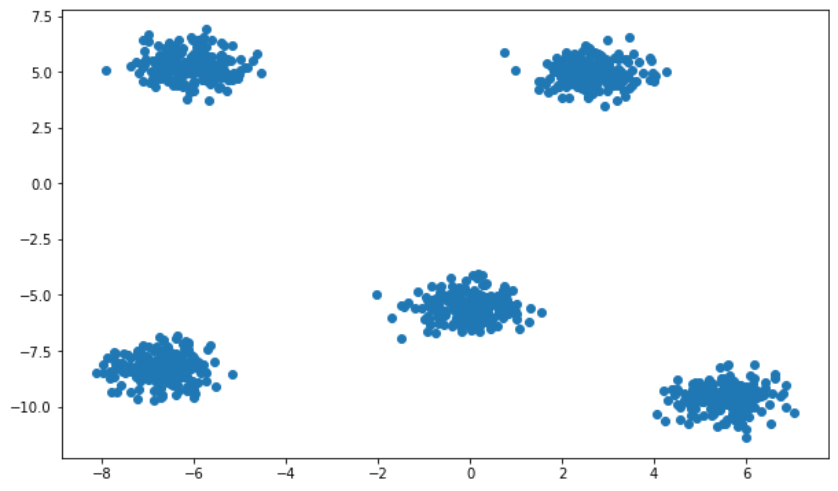
Ta thêm thư viện và hàm.

1. from sklearn.datasets import make\_blobs

Sau đó ta có thể tạo ra một tập dữ liệu để tính toán và kiểm thử như sau, các tham số đưa vào lần lượt là: số điểm = 1000, số chiều = 2, số tâm = 5, std = 0.5, tham số ngẫu nhiên = 10.

1. features, true\_labels = make\_blobs(n\_samples=1000,n\_features=2,centers=5,cluster\_std=0.6,random\_state=10)
2. fig = plt.figure(figsize=(10,6))
3. ax = fig.add\_subplot()
4. ax.scatter(features[:,0],features[:,1])
5. plt.show()

Theo đó ta có một tập dữ liệu và tập dữ liệu đó có dạng:



Hình 5. Tạo dữ liệu

Theo như dữ liệu tạo ra ta có số tâm đúng K=5.

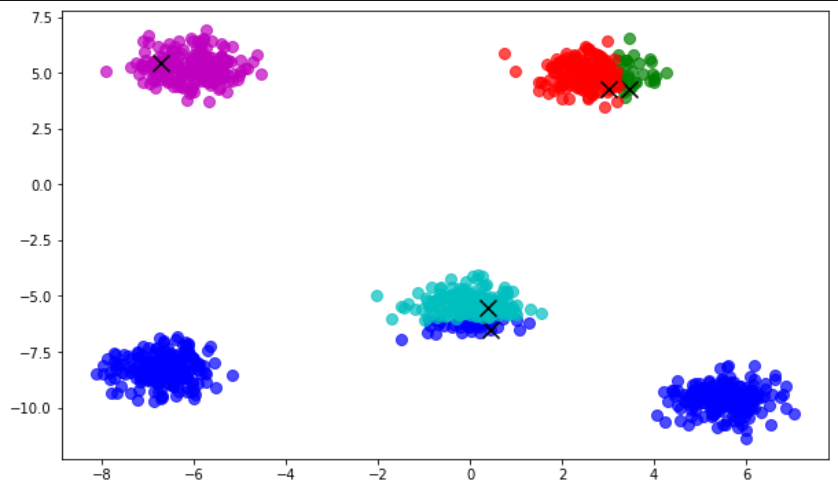
Tiếp theo ta có thể sử dụng hàm KMean với dữ liệu đã tạo ở trên.

1. Km3 = KMean(features)
   1. Thực thi phân cụm bằng Kmean.

Để kiểm tra xem hàm Kmean đã được xây dựng chính xác chưa. Ta sẽ chạy thử trên bô dữ liệu vừa được tạo ở trên. Ta sẽ xuất ra các bước điều chỉnh tâm. Tuy nhiên trong bài báo cáo sẽ không đưa ra hết toán bộ các bước mà chỉ lấy bước khởi tạo đầu tiên và bước cuối.

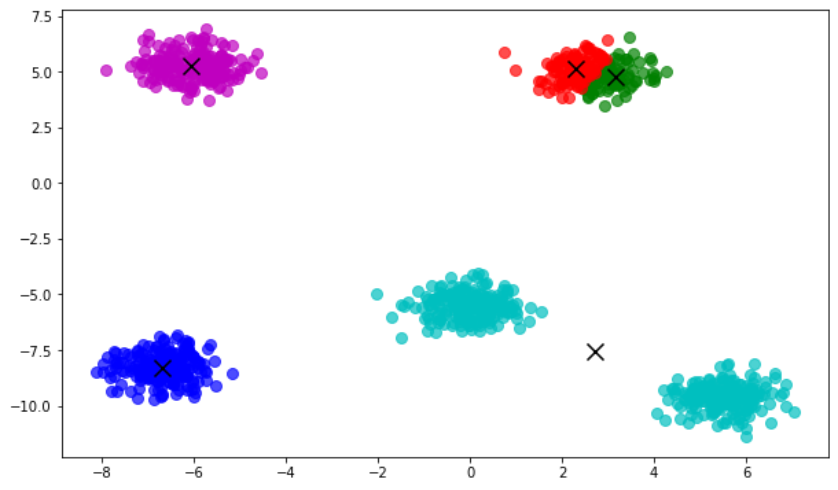
Ta sẽ chạy thử với các tâm đầu vào được khởi tạo ngẫu nhiên. Với cách khới tạo ngẫu nhiên thì khả năng rất cao là các tâm khởi tạo ban đầu sẽ sai dẫn đến kết quả cuối cùng không chính xác.

1. Km1=KMean(features,init='rd',plot\_steps=True)
2. values1=Km1.fit(5)



Hình 6. Kmean init='rd' – 1

*Ta có thể thấy ngay ở bước đầu tiên thì có 2 cặp tâm được khởi tạo quá gần nhau. Điều này có thể dận đến số chạy lớn để có thể điều chỉnh được tâm và khả năng lớn dẫn đến kết quả cuối cùng sai.*



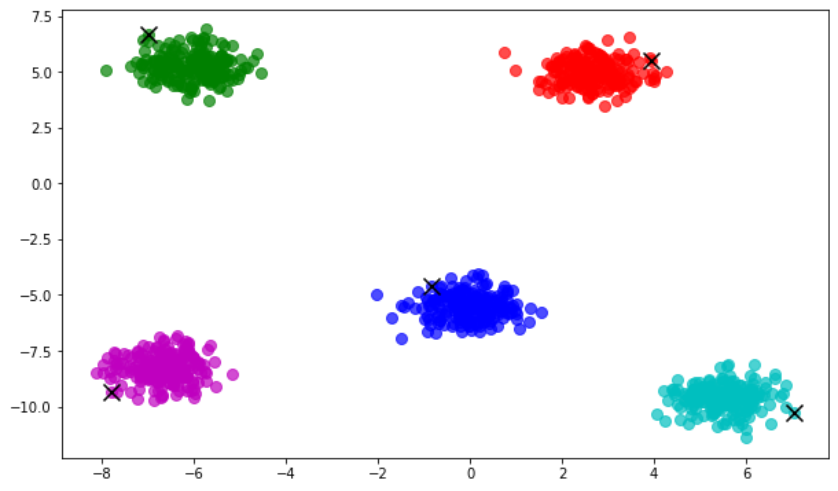
Hình 7. Kmean init='rd' – 2

*Như đã nói ở trên thì sau 11 lần cập nhật thì cho ra kết quả như trê. Dễ thấy rằng các tâm đã bị điều chỉnh sai.*

*Lưu ý: Với việc khởi tạo tâm ngẫu nhiên thì kết qua cho cuối cụng cho ra vẫn sẽ chính xác tuy nhiên với tỉ lệ thấp. Khi sử dụng cho các bài toán lớn hơn thì có thể dẫ đến kết quả sai hoặc tốc độ tính toán chậm do số lần cập nhật tâm lớn. Và mô hình trên chỉ chứng tỏ việc tâm nhẫu nhiên dẫn đến kết quả cuồi cùng có thể bị sai.*

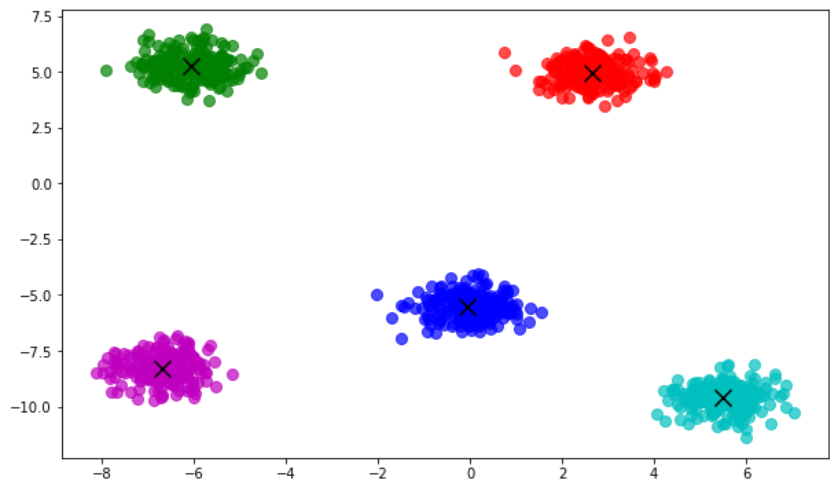
Ta sẽ tiếp tục chạy thuật toán Kmeans với các tâm ban đầu được khởi tạo ban đầu sử dùng thuật toán Kmeans++. Với điều này, ta kĩ vọng sẽ cho kết quả chính xác và sẽ dựa trên kết quả này để đánh giá xem thuật toán đã chạy đụng chưa.

1. Km2=KMean(features,plot\_steps=True)
2. Values2=Km2.fit(5)



Hình 8. Kmeans init = 'Kmeans++' – 1

*Dễ thấy rằng, các tâm lần này đã được điều chỉnh theo thuật toán Kmeans++ đã được phân bố rộng hơn. Điều này cho phép việc cập nhật tâm trở nên nhanh hơn và chính xác hơn.*



Hình 9. Kmeans init = 'Kmeans++' – 2

*Chỉ cần 1 lần cập nhật tâm thì các tâm và các cụm đã được cập nhật chính xác.*

**Kết luận:** Từ các kết quả trên, cho thấy các hàm và thuật toán đã được xây dựng chính xác.

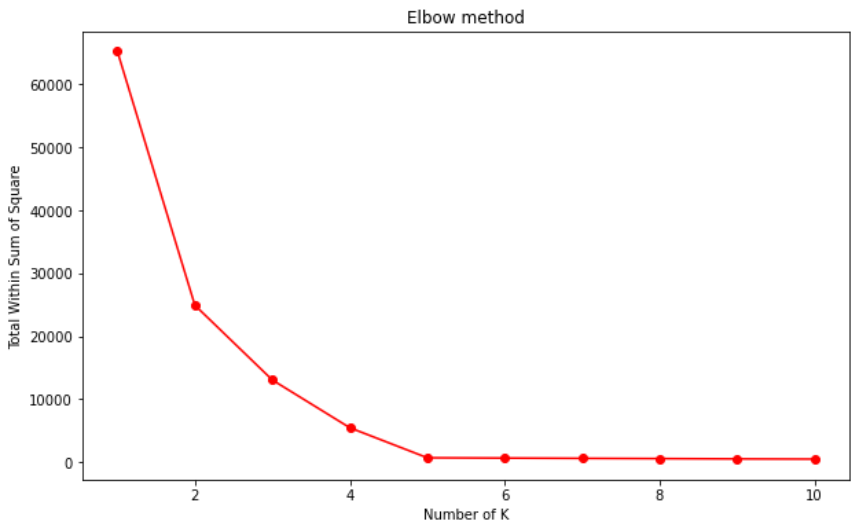
* 1. Lựa chọn K trên phương pháp Elbow.

Ta sẽ kiểm tra và tìm số k tối ưu dưa trên phương pháp Elbow với hàm đã được xậy dụng. Ta sẽ chọn số lượng dữ đoán tối đa là 10 khi đó ta có thể bằng lệnh như sau.

1. Km3.elbow\_method(10)
2. Km3.Show\_elbow\_method()

Kết quả:

1. k=1:65250.6871957869
2. k=2:24908.134768466385
3. k=3:13030.531842396324
4. k=4:5831.7188469759885
5. k=5:669.4928101529986
6. k=6:627.512762141783
7. k=7:576.1830116287447
8. k=8:529.647378440143
9. k=9:486.1602391797609
10. k=10:440.3949439862806



Như ta thấy ở trên thì K=1 có giá trị lớn nhất và luôn sẽ là lớn nhất. Sau đó các giá trị bắt đầu giảm giần và dừng lại ở K=5. Sau K=5 thì giá trị không có sự biến động cho thấy K=5 là tối ưu nhất.

Điều này cho thấy là kết quả chạy hàm theo phương pháp Elbow đã chính xác.

* 1. Lựa chọn K trên phương pháp Silhouette.

Tương tự với phương pháp Elbow. Ta chạy hàm Silhouette như sau với K tối đa là 7.

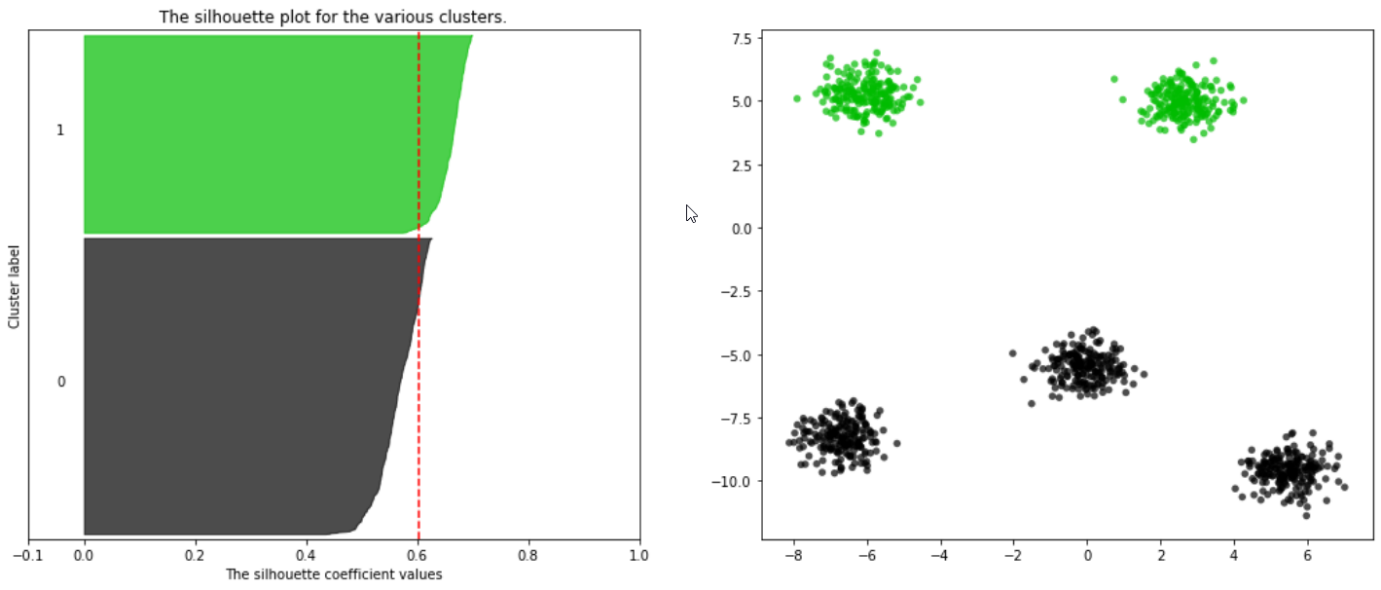
1. Silhouette\_values1=Km3.silhouette\_method(7)
2. Km3.Print\_silhoutte\_method()
3. Km3.Show\_silhoutte\_method()

Kết quả:

1. k=2:0.6036157364018033
2. k=3:0.6530334786547721
3. k=4:0.7543039824604906
4. k=5:0.8629392768705967
5. k=6:0.7545617804247856
6. k=7:0.6362134294159024

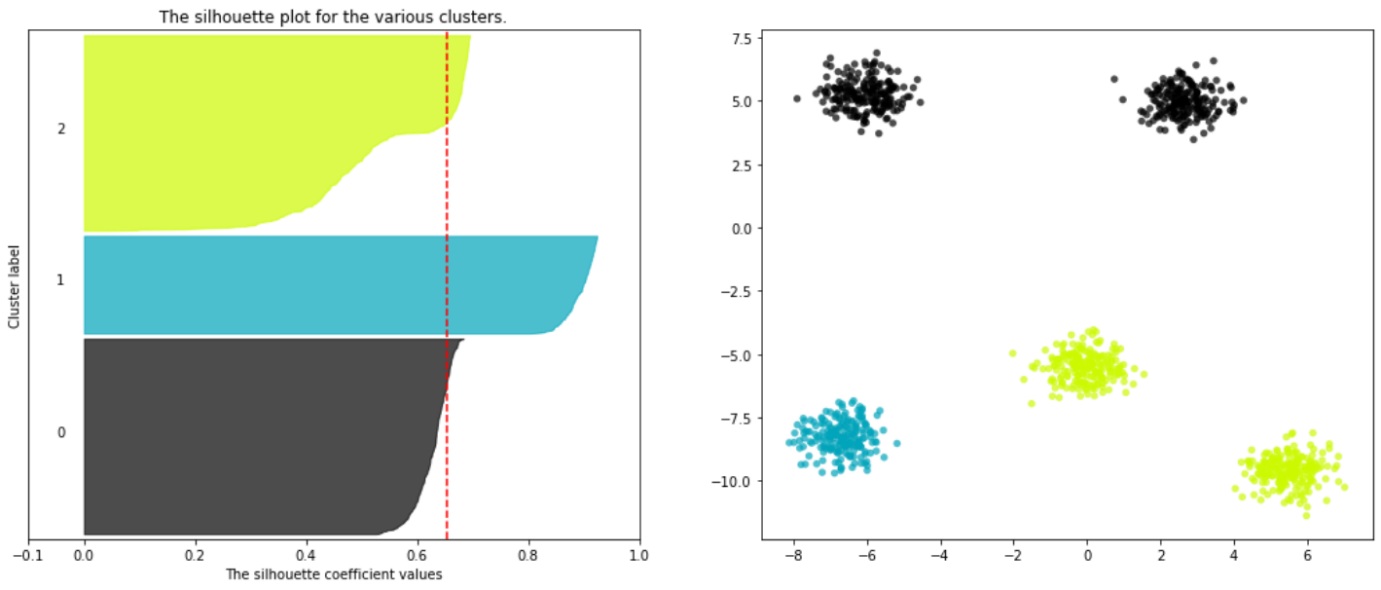
Như ta có thể thấy kết quả ở trên. Với K = 5 cho ra kết quả cao nhất so với các K còn lại. Điều có thế cho ta biết rằng, với K = 5 thì độ tạch biệt giữa cụm này và các cụm khác của giữ liệu là cao nhất. Cụ thể hơn ta có thể xem mô hình phân tích điểm Silhouette cho từng điểm và mô hình chia cụm.

Chú ý: Mô hình phía bên phải của mỗi hình chỉ có thể biểu diễn được dưới dạng 2 chiều và không thể biễn diễn cho số chiều khác lơn hơn. Mỗi cụm được gãn nhãu và có màu riêng biệt. Mỗi điễm dữ liệu được gán nhãn và phân mỗi cụm, được tính điểm silhouette và sắp xếp lại và được biểu diễn ở mô hình bên trái. Đồng thời đã được căn chỉnh tỷ lệ sao cho thích hợp với mô hình. Đường gạch đỏ là điểm Silhouette trung bình cho mỗi cụm được có được ở kết quả.



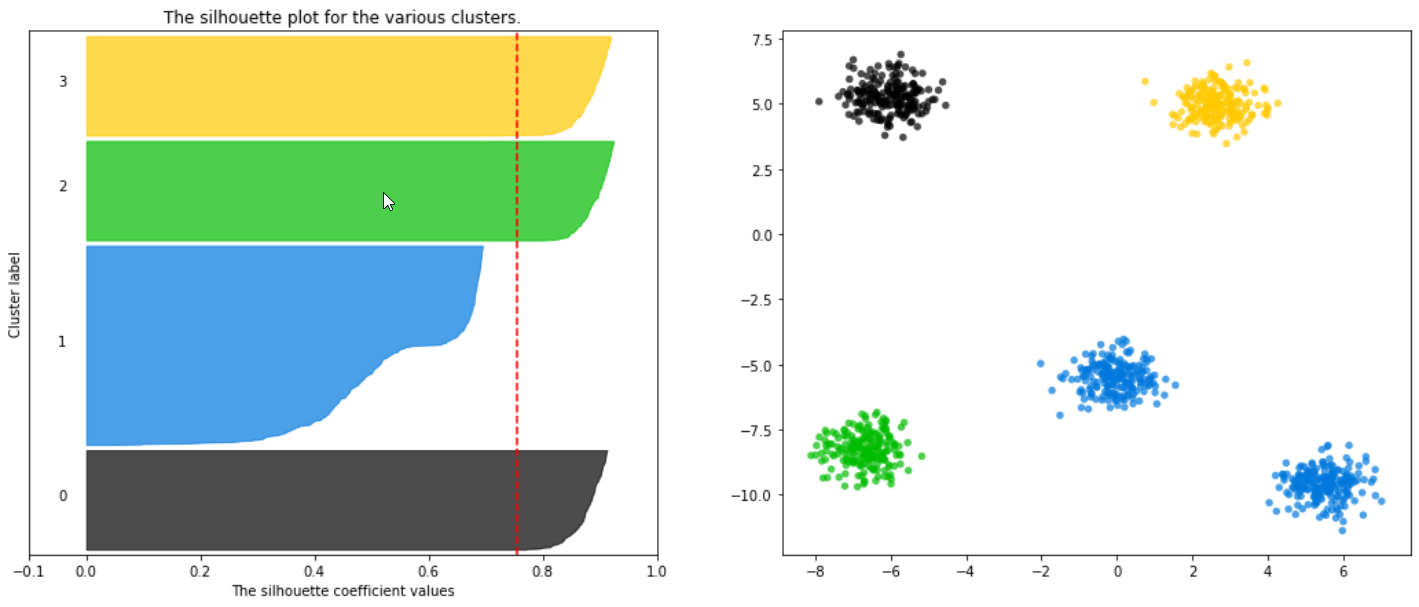
Hình 10. Mô hình Silhouette K=2

*Với K = 2 thì ta có thể coi là các cụm đã được phân tách với nhau và số điễm mỗi cụm là khá đồng đều. Tuy vậy hơn 2/3 số điểm ở cụm 0 là dưới trung bình và mô hình bên phải cho thấy k=2 vẫn chưa thích hợp.*



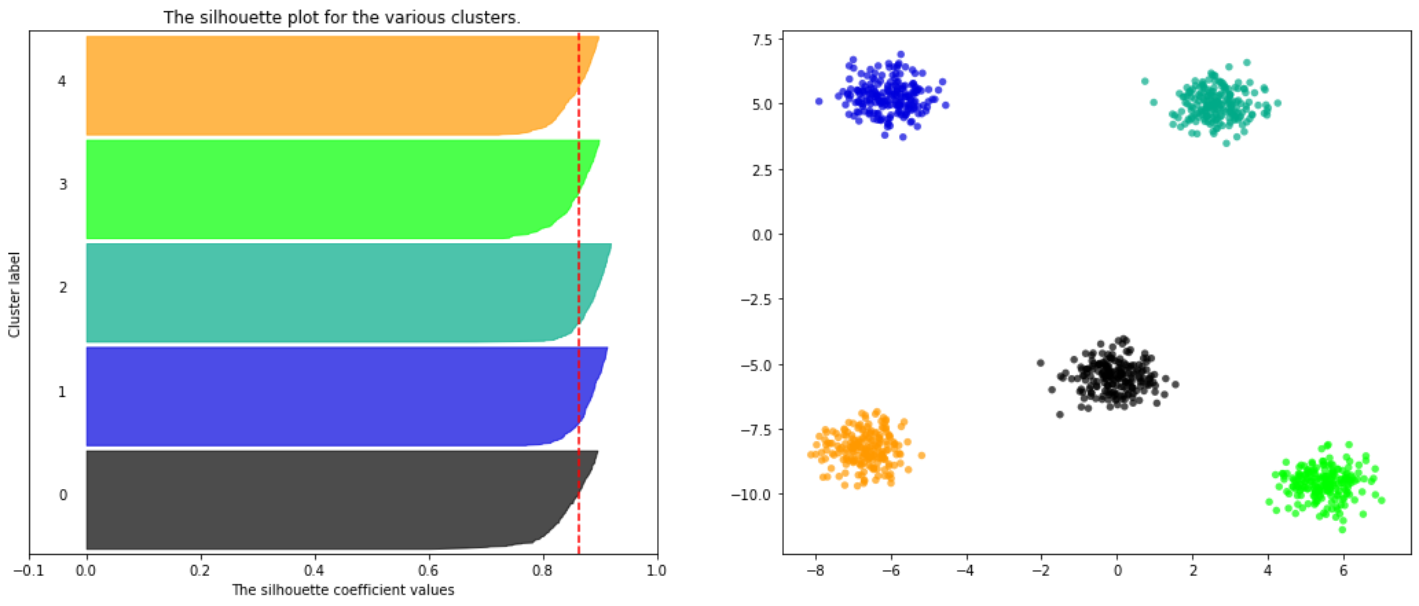
Hình 11. Mô hình Silhouette K=3

*Với K=3 ta có thể nhận thấy một số điểm bất thường trong phân bổ điểm silhouette ở các cụm. Cụm nhãn 1 có điểm rất cao chứng cho cụm này đã được phân chia rõ rệt với các cụm khác trong khi các cụm còn lại có số điểm thấm hơn trung bình và cụm nhãn 2 phần nữa có giá trị thấp.*



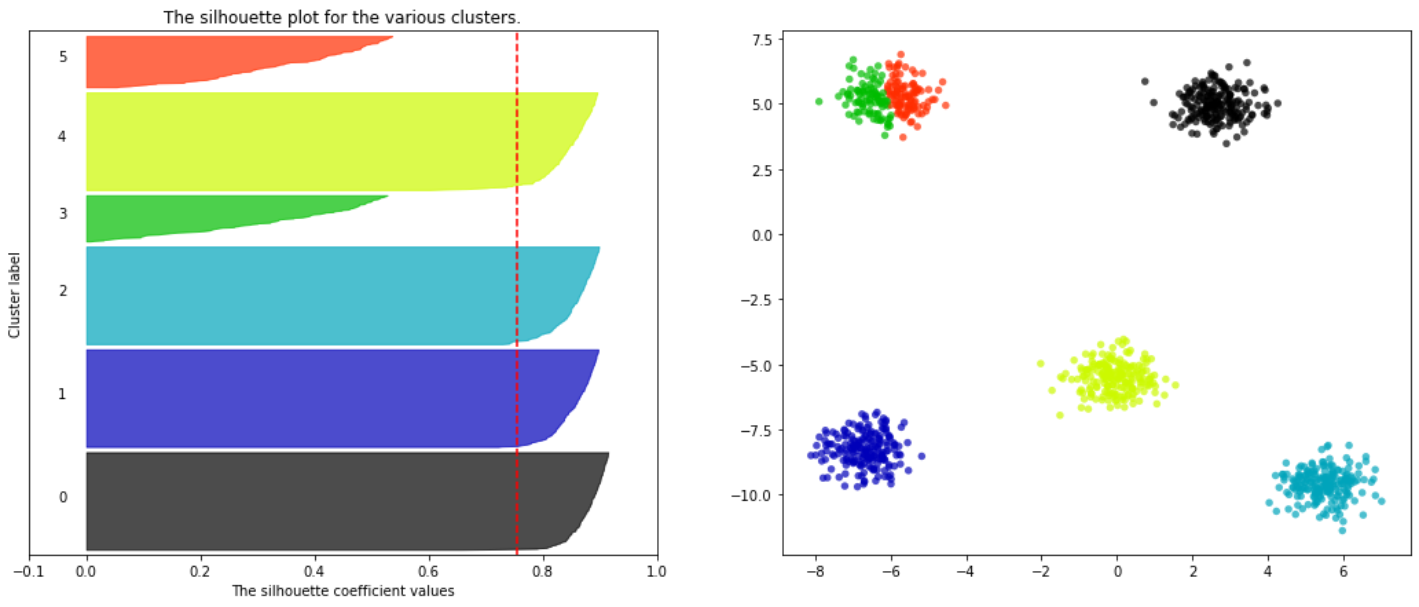
Hình 12. Mô hình Silhouette K=4

*Với K=4 ta dễ dàng nhận thấy được các cụm có nhãn 0,2,3 có số điểm cao và đồng đều. Chững cho các cụm này đã được phân chia tốt. Con Cụm nhãn 1 không đạt giá trị trung bình.*

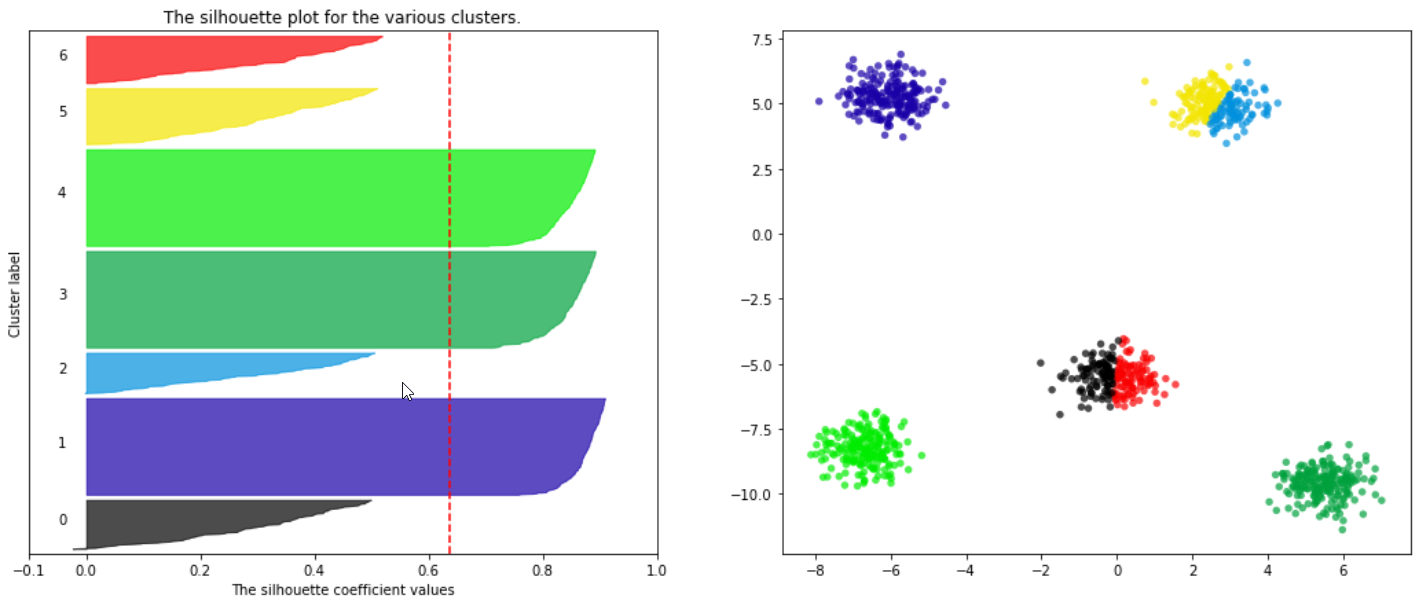


Hình 13. Mô hình Silhouette K=5

*Với số K=5 ta có thấy số điểm của mỗi cụm là rất cao và không có cụm nào là khác biệt. Cho thấy đây là mô hình đẹp nhất và ta nên chọn số K=5. Qua mô hình này ta có thể có cái hình tổng quan nhất về thế nào khi các cụm được chia tách rõ ràng.*



Hình 14. Mô hình Silhouette K=6



Hình 15. Mô hình Silhouette K=7

*Đối với số cụm = 6 và 7 ta dễ dàng nhận thấy có một số cụm không đạt ngưỡng trung mình và điểm rất thấp so với các cụm còn lại nên ta không chọn số k tượng tự*

**Kết luận:** Với mô hình K=5 như trên thì đã chứng tỏ rằng được rằng hàm Silhouette đã được xây dựng chính xác và cho ra được số k đúng.

1. KẾT LUẬN
   1. So sánh các thuật toán.

Ưu điểm chung: các thuật toán đã có thể chạy thành công và cho ra được số K đúng.

Nhược điểm chung: Các thuật toán sẽ chạy chậm hơn đáng kể với số K đầu vào lớn.

Phương pháp Elbow: Nhìn chung thì đơn giản hơn so với phương pháp Silhouette. Có tốc độ thực thi cao hơn. Tuy vậy mực chi tiết thì sẽ kém hơn.

Phương pháp Silhouette: Phức tạp và có tốc độ thực thi chậm hơn phương pháp Elbow. Tuy nhiên phương pháp này cho cái nhìn cụ thể hơn đối với mỗi cụm trong mỗi K ta lựa chọn, cho phép ta có thể lựa chọn chính xác hơn.

* 1. Các kết quả đạt được

Qua quá trình nghiên cứu thì bài báo cáo này đã đạt được môt số kết quả nhất định.

Thứ nhất, đã làm rõ được các vấn đề liên quan tới thuật toán Kmeans. Bài báo cáo đã trình bày những nét nổi bật, công dụng của thuật toán phân cụm Kmeans. Đồng thời nêu lên được thuật toán và các thực thi trên ngôn ngữ Python. Kết hợp với một số cải thiện trọng việc khởi tạo tâm ban đầu của thuật toán. Làm cho thuật toán này có phần tinh cậy hơn.

Thứ hai, bài báo cáo này đã trình bày được 2 phương pháp phổ biến và thông dụng để tìm K tối ưu đó là Elbow method và Silhouette method. Đã giải thích được thuật toán cũng như là cách sử dụng. 2 phương pháp này đã được thực thi thành công trên ngôn ngữ lập trình Python. Đồng thời đã cho ra được một số kết quả như ý muốn.

Thứ ba, với các phương pháp tìm k có sẳn. Đã có thể áp dụng được để tìm K tối ưu cho một số tập dữ liệu mẫu. Tuy rằng kết quả phân cụm có thể khác với thực tế. Những đã cho thấy rằng việc thực thi các phương pháp trên là thành công.

* 1. Các hạn chế

Kmeans là một thuật toán đơn giản và phổ biển dùng để phân cụm dữ liệu. Tuy vậy Kmeans gặp phải nhiều hạn chế khi chỉ có thể phân cụm dữ liệu mạng một số thuộc tính nhất định. Khi gặp một số cụm giữ liệu phức tạp, có kiểu hình khác nhau khí khó có thể áp dụng đúng được. Đồng thời với tập dữ liệu có số chiều D lớn thì tý lệ chính xác phần nào cũng giảm. Điều này là cho trong bào báo cáo này Kmeans vẫn chưa được điều chính về phương pháp tính khoảng cách. Chú yếu vẫn dùng phương pháp tính khoảng cách cố điển là Euclid.

Qua trình vẫn dụng dữ liệu thực tế cho thấy k có thể bị lệch do dữ liệu. Trong thực tế thì các tập dữ liệu thì các cụm có thể được phân bố ngẫu nhiên và không đồng đều. Đẫn đến số K tối ưu có thể phân bố trong một khoảng nào đó mà không phải là một số K cụ thể. Điều này dẫn đến việc phân tích kĩ càng hơn để chọn số K cụ thể hơn mà bài báo cáo này vẫn chưa đề cập tới được.

* 1. Hướng phát triển:

Bài toán này có thể được phát triển trên bằng cách tiếp tục tối ưu một số khía cạnh khác của Kmean như cách tính khoảng cách.

Dữ liệu là một vấn đề lớn đối với mọi bài toán Machine Learning. Nếu có thể tối ưu dữ liệu bằng nhiều cách khác như giảm chiều, giảm độ nhiễu,… thì bài toán sẽ cho kết quả chính xác hơn và nhanh hơn.

Vận dụng vào một số lĩnh vực mà Kmeans có thể áp dụng tốt như phân loại hình ảnh, nhận diện chữ biết, phân bố dân cư….

Các phần code thực thì có thể tối ưu thêm bằng cách giảm vòng lặp và sử dụng các hàm tối ưu hơn.

LIÊN KẾT VÀ TÀI LIỆU THAM KHẢO

Toàn bộ các tệp và dữ liệu được lưu trữ ở Github: [K\_Optimal\_Kmeans](https://github.com/superhighlevel/K_Optimal_Kmeans).

Tài liệu tham khảo:

* [book\_ML.pdf](https://github.com/tiepvupsu/ebookMLCB)
* Kmeans++: <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>
* Silhuette method: [Silhouette (clustering)](https://en.wikipedia.org/wiki/Silhouette_(clustering))
* Elbow method: [en.wikipedia.org/wiki/Elbow\_method\_(clustering)](https://en.wikipedia.org/wiki/Elbow_method_(clustering))
* Gap Statistic: [gap.pdf](https://web.stanford.edu/~hastie/Papers/gap.pdf)