BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC PHENIKAA**

**KHOA CÔNG NGHỆ THÔNG TIN**

**------------oOo-----------**

**Ảnh có chứa Phông chữ, biểu tượng, Đồ họa, văn bản

Mô tả được tạo tự động**

**Đề tài:**

**“Tìm hiểu thuật toán K-means”**

**Giảng viên: TS. Trịnh Thành**

**Sinh viên thực hiên: Nguyễn Thanh Quang**

**Lớp: K15-CNTTVJ1**

**Học phần: Khai phá dữ liệu**

**Hà Nội, 05/2023**

Mục lục

**I. Giới thiệu tổng quan 3**

1. Phân tích cụm 3

2. Thuật toán phân cụm 3

2.1. Prototype-Based Algorithms 3

2.2. Density-Based Algorithms 3

2.3. Graph-Based Algorithms 4

2.4. Hybrid Algorithms 4

2.5.Algorithm-Independent Methods4

**II. Thuật toán** **K-means 4**

1. Khái niệm 4

2. Ý tưởng thuật toán và cách hoạt động 5

3. Hàm tính khoảng cách cho K-means 5

4. Điểm mạnh và điểm yếu 6

4.1. Điểm mạnh 6

4.2. Điểm yếu 7

5. Các ứng dụng 7

**III. Triển khai thực nghiệm7**

1. Triển khai thực nghiệm 7

1.1. Coding7

1.1.1. Ngôn ngữ lập trình7

1.1.2. Thư viện7

1.1.2.1. Numpy7

**1.1.2.2. Pandas**8

**1.1.2.3. Matplotlib**8

1.1.3. Triển khai thuật toán8

1.1.3.1. Lấy dữ liệu đầu vào8

1.1.3.2. Lấy ngẫu nhiên tâm cụm8

1.1.3.3. Hàm tính khoảng cách9

1.1.3.4. Tính khoảng cách của các điểm tới các tâm cụm và phân vùng9

1.1.3.5. Chọn lại tâm cụm11

1.1.3.6. Hoàn thiện code K-means11

1.1.4. Phương pháp Elbow trong việc tối ưu số cụm12

1.2. Thực nghiệm13

1.2.1. Dữ liệu và bài toán thực nghiệm13

1.2.2. Tối ưu hóa số cụm 16

1.2.3. Triển khai thuật toán17

2. Phân tích kết quả thực nghiệm18

**IV. Tài liệu tham khảo22**

**I. Giới thiệu tổng quan**

1. Phân tích cụm

Phân tích cụm là một cách nhìn dữ liệu bằng cách chia các đối tượng có trong dữ liệu vào những cụm mà ở đó những đối tượng có trong một cụm thì sẽ tương đồng với nhau hơn là những đối tượng ở các cụm khác. Bởi vì nó không cần đến những thông tin cố định như là nhãn, phân cụm còn được gọi là học không giám sát trong một số lĩnh vực như là học máy.

Nhìn chung, phân tích cụm được dùng với hai mục đích. Mục đích đầu tiên là để “hiểu biết”, việc này có nghĩa là dùng phân tích cụm để tự động tìm kiếm những nhóm đối tượng có những đặc trưng tương đồng. Việc này có vai trò giúp chúng ta phân tích, mô tả và sử dụng các thông tin chứa các tri thức ẩn có trong những cụm. Mục đích thứ hai là “tiện ích”, phân tích cụm dùng để trừu tượng hóa các nguyên mẫu hoặc các đối tượng đại diện cho từng đối tượng riêng lẻ có trong cùng một cụm. Những nguyên mẫu và những đối tượng này sẽ đóng vai trò làm cơ sở cho một số những kĩ thuật xử lí dữ liệu như summarization, compression và nearest-neighbor finding.

2. Thuật toán phân cụm

Ở đây, chúng ta sẽ chia thuật toán phân cụm thành một số loại sau.

2.1. Prototype-Based Algorithms

Đây là loại thuật toán sẽ tìm ra nguyên mẫu cho từng cụm và định nghĩa kiểu của cụm bằng những dữ liệu các đối tượng xung quanh những nguyên mẫu. Một số thuật toán điển hình cho kiểu này như K-means và Fuzzy c-Means (FCM), nguyên mẫu được gọi là các tâm cụm và các cụm có xu hướng hình cầu.

2.2. Density-Based Algorithms

Thuật toán này coi mỗi cụm là một tập hợp dày đặc các đối tượng được bao quanh bởi một vùng các đối tượng có mật độ thấp. Chúng thường được sử dụng khi các cụm đan xen với nhau hay khi có nhiễu hoặc các ngoại lệ. DBSCAN và DENCLUE là hai đại diện tiêu biểu cho loại này.

2.3. Graph-Based Algorithms

Nếu chúng ta coi một đối tượng là một điểm và khoảng cách giữa hai đối tượng là trọng số của cạnh kết nối hai điểm với nhau, dữ liệu sẽ được biểu diễn như một biểu đồ và một cụm sẽ được định nghĩa như một biểu đồ con được kết nối. Điển hình cho loại này là thuật toán agglomerative hierarchical clustering algorithms (AHC), với việc gộp hai điểm/nhóm gần nhất thành một cho đến khi tất cả các điểm được kết nối.

2.4. Hybrid Algorithms

Kết hợp một hay nhiều thuật toán phân cụm được đề xuất để khắc phục các thiếu sót thuật toán phân cụm đơn lẻ. Chameleon là một thuật toán lai điển hình, nó sẽ dùng thuật toán graph-based algorithm để chia dữ liệu thành nhiều các thành phần nhỏ và sau đó sẽ dùng AHC để tìm ra kết quả là những cụm ở bước cuối cùng của thuật toán.

2.5.Algorithm-Independent Methods

Thay vì chạy trên dữ liệu gốc, chúng ta sẽ chạy trên các kết quả phân cụm của các thuật toán phân cụm. Cho một tập hợp các phân vùng cơ bản của dữ liệu, mục đích tìm một phân vùng duy nhất phù hợp với mọi phân vùng cơ bản nhất có thể.

Trong chủ đề này chúng ta sẽ cùng đi tìm hiểu về thuật toán K-means và triển khai thực tiễn của nó.

**II. Thuật toán** **K-means**

1. Khái niệm

K-means là một thuật thuật toán dùng trong khai phá dữ liệu với mục đích phân cụm, phân nhóm hoặc nhóm N đối tượng dựa trên các đặc trưng hoặc thuộc tính của chúng vào K nhóm (cụm) khác nhau. Nếu chúng ta ứng dụng thuật toán K-means vào một bộ dữ liệu gồm N đối tượng thì kết quả của thuật toán sẽ là K nhóm riêng biệt. Trong đó, mỗi cụm chứa các đối tượng tượng tương đồng với nhau. Lưu ý, K phải là số nguyên dương.

Thuật toán K-means nhận dữ liệu đầu vào gồm bộ dữ liệu gồm các đối tượng có N chiều và K cụm cần phân chia. Vì thế, thuật toán sẽ phân cụm các đối tượng vào khoảng cách của chúng trong không gian. Vậy nên, các đối tượng đầu vào của chúng ta sẽ có dạng:

Ảnh có chứa văn bản, bức thư

Mô tả được tạo tự động

Mục tiêu cuối cùng của thuật toán K-means là phân chia các điểm dữ liệu vào các cụm khác nhau. Vì thế để tìm ra các điểm tương đồng giữa hai điểm với nhau, trong thuật toán K-means, nó sẽ so sánh khoảng cách của điểm cần phân cụm và tâm của mỗi cụm. Theo đó, khoảng cách của điểm cần phân cụm tới tâm nào nhỏ hơn thì nó sẽ thuộc cụm chứa tâm đó.

2. Ý tưởng thuật toán và cách hoạt động

Ý tưởng của thuật toán là việc phân chia dữ liệu thành K nhóm con mà trong mỗi nhóm các điểm dữ liệu có sự tương đồng với nhau. Để có thể thực hiện việc phân cụm cho bộ dữ liệu không có nhãn, Thuật toán K-means sẽ áp dụng cách tính khoảng cách giữa các điểm với tâm của mỗi cụm.

**Bước đầu tiên**, thuật toán K-means sẽ chọn ngẫu nhiên ra K điểm dữ liệu ban đầu làm tâm hay tâm cụm. Việc sử dụng thuật toán K-means để chia bao nhiêu tâm cụm phụ thuộc vào bài toán và mục đích sử dụng riêng biệt. Vậy nên, để sử dụng tối ưu thuật toán K-means thì việc xác định trước số cụm là cần thiết.

**Bước 2**, sau khi đã đẩy được dữ liệu và xác định số cụm cần phân chia, thuật toán sẽ tính toán khoảng cách của những tâm được chọn ngẫu nhiên trước đó với tất cả các điểm trong bộ dữ liệu. Theo đó, những điểm nào ở gần tâm cụm nào nhất sẽ được chia vào cụm chứa tâm đó.

**Bước 3**, sau khi tính toán phân chia xong cụm từ ***Bước 2***, thuật toán K-means sẽ đi tính lại tâm cụm của các cụm tương ứng và đặt lại tâm cụm mới cho cụm tương ứng.

**Bước 4**, nếu như các cụm mới được sinh ra bởi không có thay đổi gì mới so với các cụm trước đó thì dừng thuật toán. Nếu không, thuật toán sẽ quay lại ***Bước 2***. Tóm lại, thuật toán sẽ lập đi lập lại ***Bước 2*** và ***Bước 3*** cho đến khi các cụm đã được phân chia rõ ràng và sẽ không có thay đổi gì mới với các cụm khi thực hiện lại hai bước trên.

Thuật toán sẽ lặp lại cho đến khi các tâm cụm sẽ không còn thay đổi nào hoặc không còn trao đổi những điểm qua lại giữa các cụm qua mỗi lần lặp. Đến lúc này, chúng ta đã thu được K cụm mong muốn nhờ vào thuật toán K-means. Trên thực tế, việc cho lặp đi lặp lại cho đến khi tìm ra các cụm tối ưu là rất mất thời gian và có còn có thể là lặp vô hạn vì không có điểm dừng. Vì vậy, người ta thường sẽ chỉ lặp qua số vòng lặp đã chỉ định hoặc đã đặt được số lượng đối tượng đã đủ cho mỗi cụm.

3. Hàm tính khoảng cách cho K-means

Việc quan trọng nhất của thuật toán K-means là quyết định một điểm dữ liệu bất kì thuộc vào cụm nào trong K cụm mục tiêu. Để tính được điều này, ta phải tính khoảng cách từ điểm bất kì nào đó đến các tâm cụm hay từ các tâm cụm đến các điểm. Như vậy, thuật toán cần có một cách tính toán để tính được khoảng cách giữa chúng. Một số cách tính khoảng cách thường được sử dụng để đo khoảng cách giữa hai điểm (tâm cụm và một điểm bất kì) giả sử được kí hiệu là = (), = ():

* Khoảng cách Euclid:
* Khoảng cách Manhattan:
* Khoảng cách Cosine:

Chúng ta còn có nhiều những phương pháp tính khoảng cách khác và mỗi phương pháp lại có một hiệu quả khác nhau. Tuy nhiên, Mỗi cách tính có công thức riêng và có thể hoạt động tốt với một số loại dữ liệu nhưng không hiệu quả với các loại dữ liệu khác. Như vậy, tùy vào bài toán và mục đích riêng biệt, chúng ta nên thử nhiều cách tính khác nhau để có được hiệu quả tốt nhất. Trong bài tiểu luận này, chúng ta sẽ lấy phương pháp tính khoảng cách Euclid:

Khoảng cách Euclid có thể có một số trường hợp sau đây:

Khi triển khai thuật toán K-means, chúng ta phải đảm bảo thuật toán phải hội tụ về một kết quả sau một số lần lặp và có thể tính toán giá trị trung bình của các cụm và khoảng cách giữa các điểm dữ liệu trong không gian dữ liệu.

4. Điểm mạnh và điểm yếu

4.1. Điểm mạnh

K-means là một thuật đơn giản, dễ hiểu, phân loại tốt trong các bài toán phân cụm, có thể nắm bắt được cấu trúc của dữ liệu khi các cụm có dạng giống hình cầu.

K-means thực hiện tốt việc phân cụm không cần biết nhãn dữ liệu đầu vào.

K-means là nền tảng cho các thuật toán phức tạp sau này.

4.2. Điểm yếu

K-means cần biết trước số cụm cần phân chia. Trên thực tế, việc này có thể ảnh hưởng đến chất lượng của kết quả đầu ra khi nhiều bài toán rất khó để xác định đúng chuẩn xác đầu ra.

Kết quả của K-means còn phụ thuộc nhiều vào các lấy ngẫu nhiên những tâm cụm ban đầu. Thuật toán có thể mắc kẹt ở mức tối ưu cục bộ và có thể không hội tụ tối ưu ở mức tối ưu toàn cục. Do đó, thuật toán không đảm bảo về tối ưu toàn cục (trên toàn bộ dữ liệu).

K-means sẽ không hiệu quả nếu có sự chênh lệch số lượng điểm giữa các cụm, phân bố dữ liệu không có dạng cầu hay bài toán với 1 điểm dữ liệu có thể làm con của hai cụm.

K-means sử dụng phép đo dựa trên khoảng cách để xác định độ giống nhau giữa các điểm dữ liệu.Để phép đo hiệu quả hơn, chúng ta nên chuẩn hóa lại dữ liệu sao cho trung bình bằng không và phương sai bằng 1. Bởi mỗi một đặc trưng có thể có các đơn vị đo khác nhau, ví dụ như tuổi tác so với thu nhập.

5. Các ứng dụng

Phân loại khách hàng dựa trên thói quen mua sắm của họ.

Phân loại các bài viết, bài báo trên không gian mạng.

Mục tiêu của việc nén hình ảnh là giảm kích thước bộ nhớ xuống càng nhỏ càng tốt trong khi vẫn duy trì sự tương đồng với hình ảnh gốc.

**III. Triển khai thực nghiệm**

1. Triển khai thực nghiệm

1.1. Coding

1.1.1. Ngôn ngữ lập trình

Python là [ngôn ngữ lập trình máy tính bậc cao](https://glints.com/vn/blog/ngon-ngu-lap-trinh-bac-cao/) thường được sử dụng để xây dựng trang web và phần mềm, tự động hóa các tác vụ và tiến hành [phân tích dữ liệu](https://glints.com/vn/blog/ky-nang-phan-tich-du-lieu/). Python là ngôn ngữ có mục đích chung, nghĩa là nó có thể được sử dụng để tạo nhiều chương trình khác nhau và không chuyên biệt cho bất kỳ vấn đề cụ thể nào.

1.1.2. Thư viện

1.1.2.1. Numpy

**Numpy (Numeric Python): là một thư viện toán học phổ biến và mạnh mẽ của Python. Cho phép làm việc hiệu quả với ma trận và mảng, đặc biệt là dữ liệu ma trận và mảng lớn với tốc độ xử lý nhanh hơn nhiều lần khi chỉ sử dụng “core Python” đơn thuần.**

**1.1.2.2. Pandas**

**Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu nhanh, mạnh mẽ, linh hoạt và mang hàm ý. Tên thư viện được bắt nguồn từ panel data (bảng dữ liệu). Pandas được thiết kế để làm việc dễ dàng và trực quan với dữ liệu có cấu trúc (dạng bảng, đa chiều, có tiềm năng không đồng nhất) và dữ liệu chuỗi thời gian.**

**1.1.2.3. Matplotlib**

Để thực hiện các suy luận thống kê cần thiết, cần phải trực quan hóa dữ liệu của bạn và Matplotlib là một trong những giải pháp như vậy cho người dùng Python. Nó là một thư viện vẽ đồ thị rất mạnh mẽ hữu ích cho những người làm việc với Python và NumPy. Module được sử dụng nhiều nhất của Matplotib là Pyplot cung cấp giao diện như MATLAB nhưng thay vào đó, nó sử dụng Python và nó là nguồn mở.

1.1.3. Triển khai thuật toán

1.1.3.1. Lấy dữ liệu đầu vào

Bước đầu tiên khi muốn phân tích dữ liệu, ngoài việc thu thập và tiền xử lí dữ liệu, chúng ta cần phải lấy được dữ liệu vào chương trình sau khi đã cấu trúc hóa nó ở bước tiền xử lí (tiền xử lí dữ liệu có thể được thực hiện trong chương trình của bạn). Với bài này, dữ liệu được lưu trữ dạng là một file csv. Vì vậy, chúng ta sẽ sử dụng thư viện pandas để lấy nó vào trong chương trình python của mình. Cụ thể như sau:

df = pd.read\_csv('/content/Mall\_Customers\_withNulls.csv')

Với “df” là tên dataframe khi đưa bảng dữ liệu vào chương trình. Ta dùng phương thức “read\_csv” của pandas để đọc những file có định dạng csv (tham số truyền vào sẽ là đường dẫn đến file dữ liệu). Với “colnames” để lưu tên cột cần quan tâm (tùy từng bái toán có thể khác nhau).

1.1.3.2. Lấy ngẫu nhiên tâm cụm

Bước tiếp theo, chúng ta cần định nghĩa xem bài toán của mình cần có bao nhiêu tâm cụm. Giả sử chúng ta cần có K tâm cụm. Sau đó, ta sẽ lấy ngẫu nhiên K tâm cụm là những điểm dữ liệu ngẫu nhiên trong không gian dữ liệu. Cụ thể như sau:

def initiate\_centroids(k, dset):

    '''

    Select k data points as centroids

    k: number of centroids

    dset: pandas dataframe

    '''

    centroids = dset.sample(k)

    return centroids

Hàm này sẽ nhận đầu vào là “k” và “dset”. Với “k” là số tâm cụm cần lấy ngẫu nhiên (là số nguyên dương lớn hơn 0) và “dset” là tập dữ liệu đang được phân tích. Đầu ra của hàm sẽ là tập các tâm cụm được lấy ngẫu nhiên từ tập dữ liệu. Ở đây, ta định nghĩa ra một hàm để lấy ngẫu nhiên K tâm cụm từ dataframe từ phương thức “sample” của dataframe và trả về các tâm cụm được lấy ngẫu nhiên đó.

1.1.3.3. Hàm tính khoảng cách

Để tính được khoảng cách và so sánh khoảng cách của các điểm với các tâm cụm đã chọn, chúng ta cần một hàm tính khoảng cách. Dựa trên phương pháp tính khoảng cách Euclid:

Hàm tính khoảng cách sẽ được triển khai đơn giản như sau:

def rsserr(a,b):

    '''

    Calculate the root of sum of squared errors.

    a and b are numpy arrays

    '''

    return np.sqrt(np.square(np.sum((a-b)\*\*2)))

Hàm với đầu vào là hai điểm dữ liệu và đầu ra là khoảng cách Euclid giữa chúng.

1.1.3.4. Tính khoảng cách của các điểm tới các tâm cụm và phân vùng

Sau khi đã có hàm tính khoảng cách, công việc tiếp theo chúng ta cần làm là phân chia các điểm dữ liệu về với các cụm có tâm cụm gần nhất. Để làm được điều đó, với mỗi một diểm dữ liệu, ta cần tính khoảng cách giữa chúng và các tâm cụm ngẫu nhiên đã được chọn trước đó. Sau đó, ta sẽ đi so sánh các khoảng cách này và phân chia các điểm dữ liệu về tâm cụm có khoảng cách ngắn nhất thành một cụm. Code triển khai sẽ như sau:

def centroid\_assignation(dset, centroids):

    '''

    Given a dataframe `dset` and a set of `centroids`, we assign each

    data point in `dset` to a centroid.

    - dset - pandas dataframe with observations

    - centroids - pa das dataframe with centroids

    '''

    k = centroids.shape[0]

    n = dset.shape[0]

    assignation = []

    assign\_errors = []

    for obs in range(n):

        # Estimate error

        all\_errors = np.array([])

        for centroid in range(k):

            err = rsserr(centroids.iloc[centroid, :], dset.iloc[obs,:])

            all\_errors = np.append(all\_errors, err)

        # Get the nearest centroid and the error

        nearest\_centroid = np.argmin(all\_errors)

        nearest\_centroid\_error = np.min(all\_errors)

        # Add values to corresponding lists

        assignation.append(nearest\_centroid)

        assign\_errors.append(nearest\_centroid\_error)

    return assignation, assign\_errors

Hàm được định nghĩa bên trên sẽ thực hiện việc nối các điểm dữ liệu về với các tâm cụm gần nhất với chúng. Như theo code, đầu vào của hàm sẽ là tập dữ liệu cần được phân cụm “dset” và tập các cụm ngẫu nhiên “centroids” và trả về hai list, list đầu tiên chứa thông tin tuần tự centroid tương ứng với các điểm dữ liệu trong dataframe và list thứ hai chứa thông tin về khoảng cách giữa điểm dữ liệu đó và centroid gần nhất.

Các công việc thực hiện trong hàm tuần tự từ trên xuống dưới. Đầu tiên, ta cần thông tin về số điểm dữ liệu (centroids.shape[0]) và số centroid đầu vào (dset.shape[0]). Tiếp theo, ta cần có các list để lưu thông tin khi đã được sử lí với list assignation để lưu thông tin về các centroid cho từng điểm và assign\_errors để lưu thông tin về khoảng cách của từng điểm dữ liệu với centroid gần nhất. Sau đó, ta sẽ dùng một vòng lặp để đi qua từng điểm dữ liệu trong mảng. Trong mỗi lần lặp, ta sẽ tính khoảng cách của nó với các tâm đã được chọn trước đó. Khi đã có kết quả về khoảng cách đến các tâm của mỗi diểm nhờ vào việc lưu các giá trị vào mảng all\_errors, ta tiếp tục sẽ tìm tới index có giá trị nhỏ nhất (chính là centroid được biểu diễn dưới dạng category và thêm nó vào list assignation). Nhân tiện đó, ta sẽ thêm luôn khoảng cách giữa nó và centroid gần nhất vào list assign\_errors để tiện cho việc kiểm nghiệm sau này. Cuối cùng, ta chỉ cần trả về các list kết quả.

1.1.3.5. Chọn lại tâm cụm

Như vậy, ta đã có thể tính khoảng cách và phân chia các điểm dữ liệu và các cụm khác nhau với các centroid khác nhau. Công việc tiếp theo chúng ta cần làm là đặt lại tâm cụm để có thể tiếp tục thuật toán. Việc triển khai đặt lại tâm cụm sẽ như sau:

centroids = df.groupby('centroid').agg('mean').loc[:,colnames].reset\_index(drop = True)

Với câu lệnh trên, centroid sẽ được tính lại bằng cách lấy giá trị trung bình của tập dữ liệu đã chia thành các cụm ở bước trên ở từng cột. Đầu tiên, “.groupby” sẽ nhóm các điểm dữ liệu có cùng centroid trong tập dữ liệu lại và sau đó tính trung bình nhờ “.agg” và theo từng cột với “.loc”. Cuối cùng, ta sẽ “reset\_index” để tiện hơn cho việc thống kê.

1.1.3.6. Hoàn thiện code K-means

Như vậy, ta đã có đủ các công cụ cơ bản để xây dựng code của thuật toán K-means. Như đã biết, thuật toán K-means cần tìm ra các cụm mà ở đó qua mỗi lần lặp, các cụm không có quá nhiều khác biệt so với cụm trước đó. Mà thực tế, điều kiện đó khá tốn thời gian và thường thì người ta sẽ đặt số lần lặp cố dịnh hoặc thỏa điều kiện nào đó. Vì vậy, thuật toán của chúng ta cần lặp lại từ bước chọn centroid cho đến chia tập dữ liệu vào các cụm cho đến khi hết số lần lặp và thỏa điều kiện nào đó. Ở bài này, chúng ta sẽ đặt ra một điều kiện dừng là khi hiệu tổng khoảng cách của bước trước đó với tổng khoảng cách ở bước này nhỏ hơn ngưỡng đã định. Triển khai code sẽ như sau:

def kmeans(dset, k=2, tol=1e-4):

    '''

    K-means implementationd for a

    `dset`:  DataFrame with observations

    `k`: number of clusters, default k=2

    `tol`: tolerance=1E-4

    '''

    # Let us work in a copy, so we don't mess the original

    working\_dset = dset.copy()

    # We define some variables to hold the error, the

    # stopping signal and a counter for the iterations

    err = []

    goahead = True

    j = 0

    # Step 2: Initiate clusters by defining centroids

    centroids = initiate\_centroids(k, dset)

    while(goahead):

        # Step 3 and 4 - Assign centroids and calculate error

        working\_dset['centroid'], j\_err = centroid\_assignation(working\_dset, centroids)

        err.append(sum(j\_err))

        # Step 5 - Update centroid position

        centroids = working\_dset.groupby('centroid').agg('mean').reset\_index(drop = True)

        # Step 6 - Restart the iteration

        if j>0:

            # Is the error less than a tolerance (1E-4)

            if err[j-1]-err[j]<=tol:

                goahead = False

        j+=1

    working\_dset['centroid'], j\_err = centroid\_assignation(working\_dset, centroids)

    centroids = working\_dset.groupby('centroid').agg('mean').reset\_index(drop = True)

    return working\_dset['centroid'], j\_err, centroids

Ở hàm kmeans, chúng ta có đầu vào là “dset”, “k” và “tol”. Trong đó, “dset” là tập dữ liệu đầu vào, “k” là số centroid, “tol” là ngưỡng điều kiện kết thúc của thuật toán. Kết quả trả về sẽ là cột thông tin centroid đã được xử lí cho từng điểm dữ liệu, khoảng cách của từng điểm dữ liệu tới centroid gần nhất và tập các centroid (working\_dset['centroid'], j\_err, centroids).

Đầu tiên, khi hàm được gọi, nó sẽ làm việc trên bản sao của tập dữ liệu tránh việc làm thay đổi tập dữ liệu gốc làm cho kết quả đầu ra bị sai khác (working\_dset). Tiếp theo, chúng ta cần có một list để lưu lại tổng khoảng cách của các lần lặp “err” và một biến lưu lại lần lặp “j”. Tiếp theo, theo đúng trình tự, ta cần chọn ngẫu nhiên các centroid và bắt đầu vòng lặp. Trong vòng lặp, chúng ta sẽ lần lượt thực hiện các bước sau theo trình tự là phân chia cụm dữ liệu, cập nhật tổng khoảng cách, cập nhật lại centroid và kiểm tra lại điều kiện lặp. Cuối cùng khi đã thoát khỏi vòng lặp, chúng ta sẽ trả về những kết quả đã qua xử lí.

Dòng code khởi động thuật toán và đưa ra kết quả sau khi xử lí:

df['centroid'], df['error'], centroids =  kmeans(df, 5)

1.1.4. Phương pháp Elbow trong việc tối ưu số cụm

Như đã đề cập từ trước, thuật toán K-means cần phải biết trước số cụm để có thể hoạt động phân chia theo số cụm được định nghĩa. Như vậy, việc chọn số cụm ban đầu cũng ảnh hưởng rất nhiều đến hiệu quả của thuật toán. Vậy nên chúng ta cần có một kĩ thuật nào đó để đánh giá độ tối ưu của kết quả khi định nghĩa số cụm cho dữ liệu. Phương pháp Elbow là một trong số các phương pháp đó. Bằng cách tính tổng các khoảng cách của tất cả các điểm dữ liệu tới tâm cụm gần nhất của chúng qua mỗi lần chọn số tâm cụm tăng dần, chúng ta sẽ có biểu đồ có dạng như sau:

Ảnh có chứa văn bản, hàng, biểu đồ, Sơ đồ

Mô tả được tạo tự động

Dựa vào biểu đồ này, nó sẽ giúp bạn dễ dàng hơn khi đưa ra quyết định tập dữ liệu của bạn sẽ có bao nhiêu cụm. Triển khai trong code ta có:

err\_total = []

n = 10

df\_elbow = df

for i in range(n):

    \_, my\_errs, \_ = kmeans(df\_elbow, i+1)

    err\_total.append(sum(my\_errs))

fig, ax = plt.subplots(figsize=(8, 6))

plt.plot(range(1,n+1), err\_total, linewidth=3, marker='o')

ax.set\_xlabel(r'Number of clusters', fontsize=14)

ax.set\_ylabel(r'Total error', fontsize=14)

plt.xticks(fontsize=12)

plt.yticks(fontsize=12)

plt.show()

1.2. Thực nghiệm

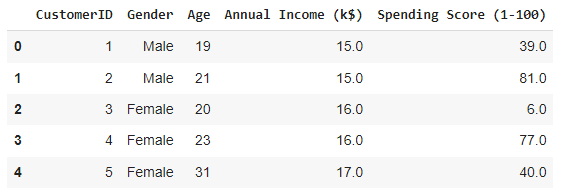
1.2.1. Dữ liệu và bài toán thực nghiệm

Bài toán được đặt ra là phân tích dữ liệu khách hàng và phân loại khách hàng. Phân loại khách hàng là quá trình phân chia khách hàng thành các nhóm khác nhau dựa trên các đặc điểm chung, chẳng hạn như hành vi mua hàng, độ tuổi, giới tính, thu nhập và nhiều hơn nữa. Trong kinh doanh, việc phân loại khách hàng rất quan trọng trong việc đưa ra chiến lược kinh doanh. Dựa vào việc phân loại, doanh nghiệp sẽ hiểu rõ hơn về khách hàng của họ và đáp ứng nhu cầu của họ một cách hiệu quả hơn. Việc phân loại khách hàng có thể giúp các doanh nghiệp xác định được khách hàng ục tiêu, tạo ra các chiến dịch tiếp thị và quảng cáo nhắm mục tiêu, cải thiện mối quan hệ với khách hàng và tăng doanh thu.

Lấy dữ liệu được lưu trữ

df = pd.read\_csv('/content/Mall\_Customers\_withNulls.csv')

Với bộ dữ liệu thu thập thông tin từ doanh nghiệp, chúng ta có thông tin của từng khách hàng như ID, giới tính, tuổi, thu nhập hàng năm và điểm chỉ tiêu. Sau khi thu thập dữ liệu, ta sẽ có một bảng dữ liệu gồm 5 cột và 200 dòng dữ liệu.



Trước khi đưa vào thuật toán, chúng ta cần phải biết thông tin về bảng dữ liệu của mình.

Ảnh có chứa văn bản, biên lai, Phông chữ, đại số

Mô tả được tạo tự động

Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình, biên lai

Mô tả được tạo tự động

Như vậy, bảng dữ liệu của chúng ta đang có những giá trị null trong trường thông tin. Chúng ta cần loại bỏ nó.

df.dropna(inplace=True)

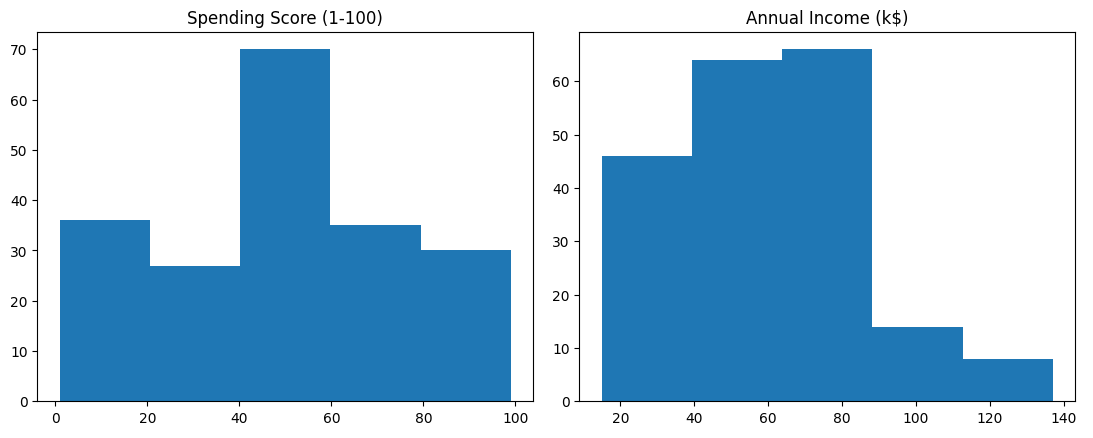
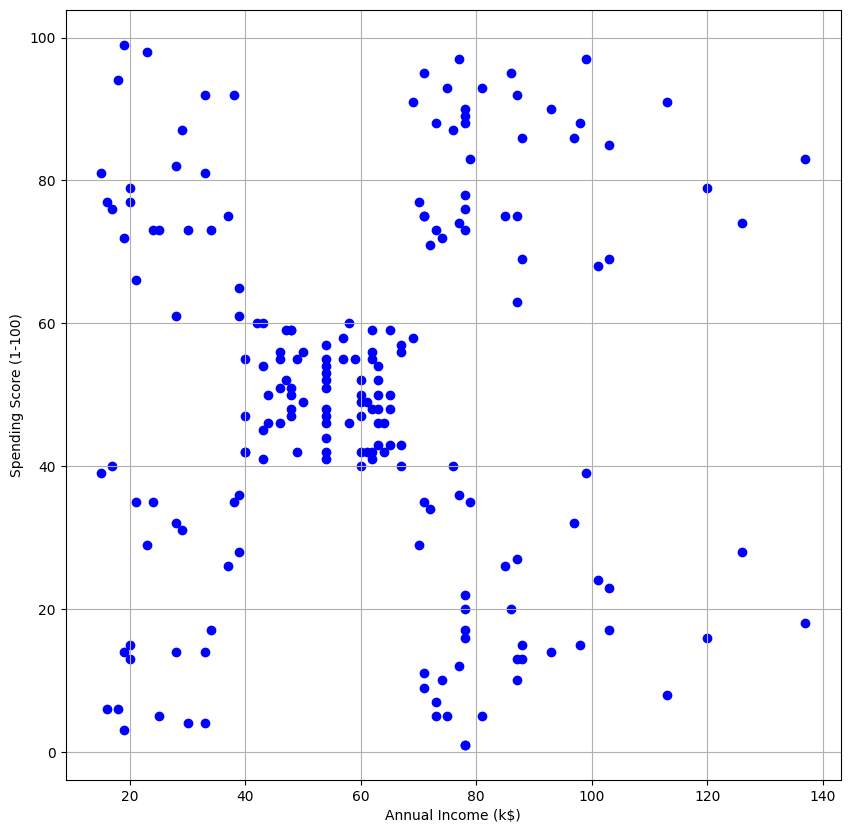
Sau khi loại bỏ được mẫu chứa null, bây giờ chúng ta cần quyết định những trường thông tin nào sẽ được lấy để phân loại khách hàng. Chúng ta cần chọn những trường dữ liệu có ảnh hưởng đến thói quen mua hàng. Vậy nên, các cột “Annual Income (k$)” và “Spending Score (1-100)” sẽ hợp lí nếu dùng để phân loại khách hàng. Vậy nên, chúng ta sẽ bỏ ba cột “CustomerID”, “Gender” và “Age”.

df = df.drop(columns=['CustomerID'])

df = df.drop(columns=['Gender'])

df = df.drop(columns=['Age'])

Trực quan hóa bảng dữ liệu đầu vào:



Nhìn vào các biểu đồ trực quan hóa bên trên, chúng ta thấy, hầu hết phần lớn khách hàng đang tập trung trong khoảng với mức thu nhập hàng năm từ 40 đến 87 k$ và điểm chỉ tiêu trong khoảng từ 40 đến 60. Với thu nhập hàng năm từ 17 đến 40 k$ chiếm số lượng khách hàng nhiều thứ hai và khách hàng có thu nhập hàng năm trên 87k$ ít hơn hẳn. Trong khi đó điểm chỉ tiêu của các khách hàng ngoài khoảng 40 đến 60 ra đang khá đồng đều nhau.

1.2.2. Tối ưu hóa số cụm

Thuật toán K-means là thuật toán dựa trên tính khoảng cách, chúng ta cần phải có bước chuẩn hóa dữ liệu trước khi đưa vào thuật toán. Ở đây tôi sẽ dùng Standard để chuẩn hóa dữ liệu của mình.

from sklearn.preprocessing import StandardScaler

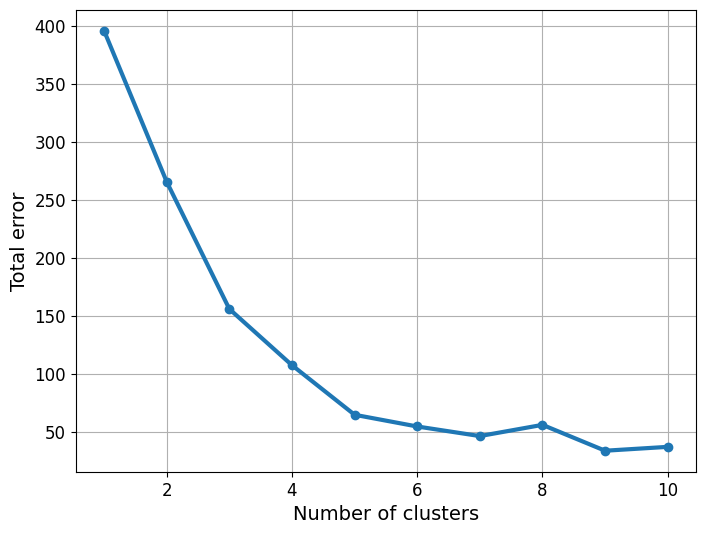
scaler = StandardScaler()

temp = scaler.fit\_transform(df)

df['Annual Income (k$)'] = temp[:,0]

df['Spending Score (1-100)'] = temp[:,1]

Vì dữ liệu không có nhãn nên bước đầu tiên chúng ta cần làm là xác định số loại khách hàng tối ưu hay số cụm tối ưu nhờ phương pháp elbow. Với phương pháp elbow, nó sẽ hỗ trợ rất nhiều cho chúng ta quyết định số cụm sao cho hợp lí nhất.



*Biểu đồ Elbow*

Kết hợp kết quả nhận được từ phương pháp elbow và phần trực quan hóa các điểm dữ liệu. Ta thấy dữ liệu đang có xu hướng phân thành 5 cụm bên trực quan hóa. Bên cạnh đó, biểu đồ elbow cho số phân cụm từ 1 đến 10 cũng chỉ ra rằng, 5 phân cụm đang là điểm tối ưu hóa số cụm của bài toán dựa theo bộ dữ liệu mà chúng a đang có.

Vì kết quả của thuật toán K-means phụ huộc vào cách chọn ngẫu nhiên tâm cụm ban đầu nên có thể sẽ phải thực hiện phương pháp elbow lại nhiều lần để có biểu đồ đẹp nhất. Hơn nữa, phương pháp elbow cung cấp hông tin tham khảo để giúp chúng ta dễ dàng hơn trong việc quyết định số cụm. Vậy nên, tùy từng bài toán và mục đích khác nhau, chúng ta nên cân nhắc thật kĩ lưỡng khi lựa chọn để có được hiệu suất và hiệu quả cao nhất phục vụ cho bài toán của bạn.

1.2.3. Triển khai thuật toán

Sau khi đã xác định được dữ liệu đầu vào và số tâm cụm mục tiêu, chúng ta sẽ tiến hành đưa dữ liệu vào thuật toán để đưa ra kết quá mà chúng ta cần thiết.

df['centroid'], df['error'], centroids =  kmeans(df, 5)

df.head()

Sau khi có kết quả, chủng ta sẽ sắp xếp bảng dữ liệu lại theo các centroid để tiện phân tích kết quả.

df = df.sort\_values(by = 'centroid', ascending=True)

Như vậy, sau bước này, chúng ta đã có kết quả của thuật toán và được sắp xếp của bảng dữ liệu theo centroid. Việc tiếp theo chúng ta cần làm đó là trực quan hóa kết quả mà chúng ta đã thu được. Để làm được điều này, chúng ta cần biểu diễn nó dưới dạng các điểm, mỗi cụm có một màu khác nhau và có các tâm cụm cằm ở trung tâm các cụm. Sau đây là đoạn code sẽ làm cho chúng ta điều này.

plt.figure(figsize=(10,6))

color = ['purple', 'gray', 'cyan', 'green', 'yellow']

# Tạo dữ liệu cho biểu đồ

data = scaler.inverse\_transform(np.array(df.iloc[:,0:2]))

x = data[:,0]

y = data[:,1]

# Vẽ biểu đồ

plt.scatter(x, y, c=df['centroid'], marker='o', alpha = 0.75, cmap='viridis')

for i in range(len(centroids)):

  plt.scatter(centroids['Annual Income (k$)'].iloc[i] \* np.std(data[:,0]) + np.mean(data[:,0]), centroids['Spending Score (1-100)'].iloc[i] \* np.std(data[:,1]) + np.mean(data[:,1]), c=color[i], marker='s', s=100, label=i)

# Thêm nhãn cho các trục

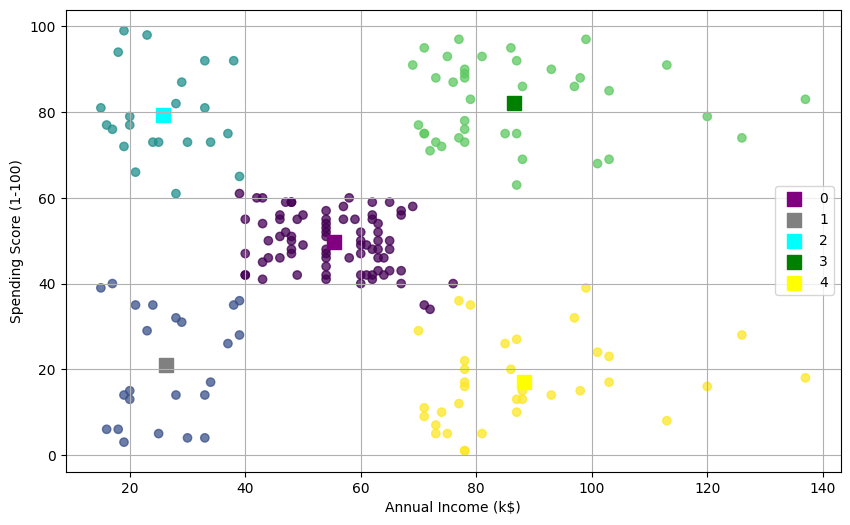
plt.xlabel('Annual Income (k$)')

plt.ylabel('Spending Score (1-100)')

plt.legend()

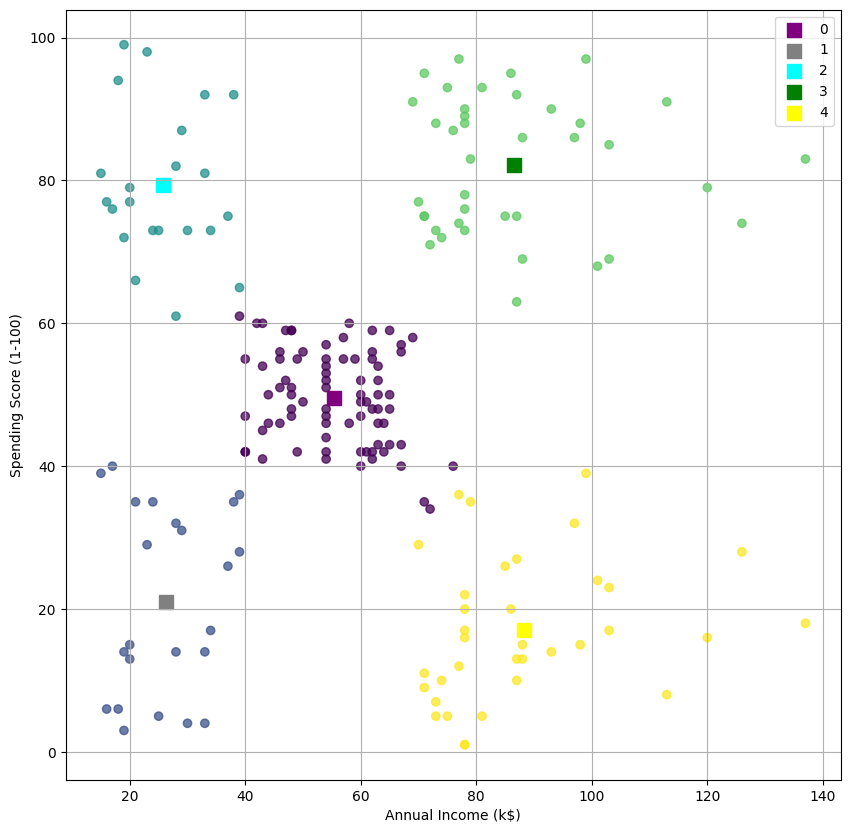
plt.grid()

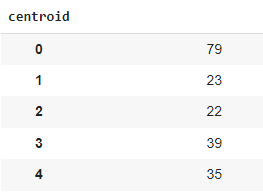
plt.show()



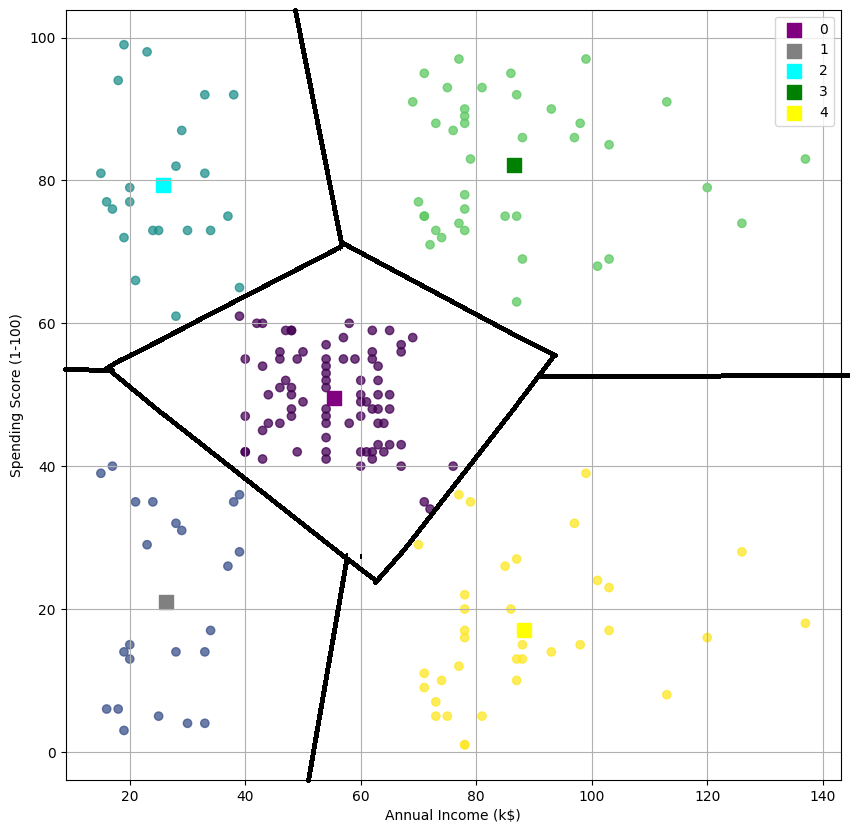
2. Phân tích kết quả thực nghiệm

Như đã đề cập ở trên, kết quả mà chúng ta nhận được





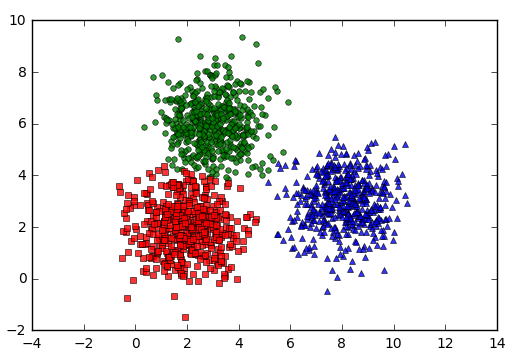
Để dễ dàng quan sát hơn, chúng ta cùng nhìn biểu đồ biểu diễn các điểm dữ liệu trong từng vừng của mỗi cụm:



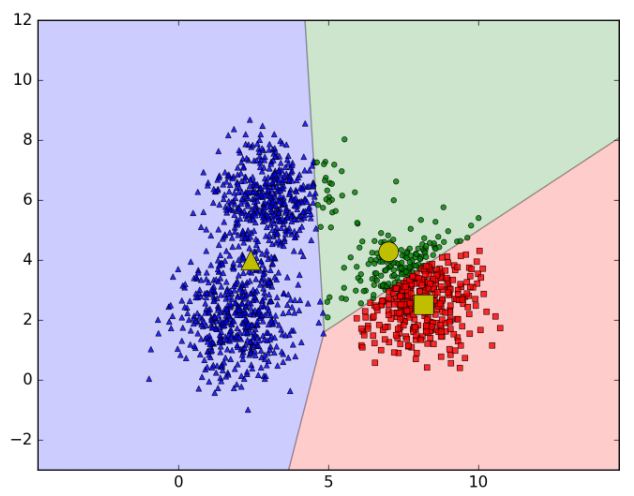
Nhìn vào biểu đồ trực quan, chúng ta thấy rằng kết quả của chúng ta nhận được khá tốt. Thuật toán K-means đang hoạt động và làm rất tốt nhiệm vụ phân vùng dữ liệu cho các điểm dữ liệu trong không gian dữ liệu. Các cụm được tách biệt với nhau khá rõ ràng. Như vậy, thuật toán K-means đã chia thành công khách hàng ra thành 5 loại riêng biệt với độ chính xác khá cao.

Tuy ở tập dữ liệu của chúng ta, các cụm phân chia rõ ràng với nhau. Nhưng, kết quả này chưa là tối ưu nhất, bởi kết quả của K-means phụ thuộc nhiều vào cách chọn centroid khởi đầu và nhiều yếu tố.

Kết quả của K-means phụ thuộc vào các chọn centroid ban đầu. Diều này có thể dẫn tới kết quả đầu ra bị sai và không đúng ý muốn. Dưới đây là biểu đồ về phân cụm dữ liệu mà ta mong muốn khi thuật toán K-means hoạt động:

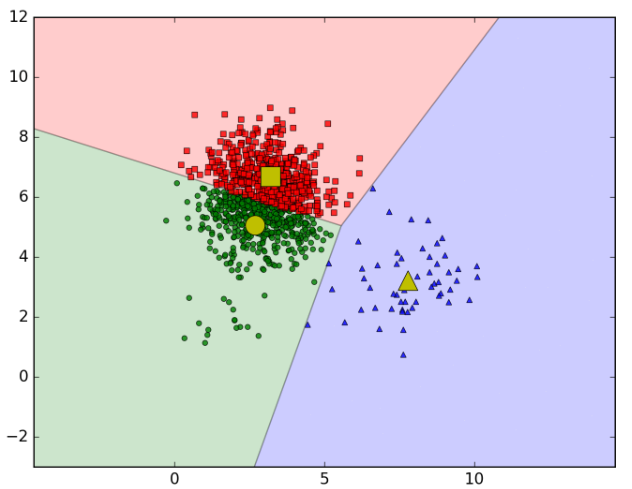


Nhưng nếu như không may mắn khi chọn tâm centroid, kết quả có thể sẽ sai. Hình dưới đây là một trường hợp có thể xảy ra khi các centroid chọn ngẫu nhiên:

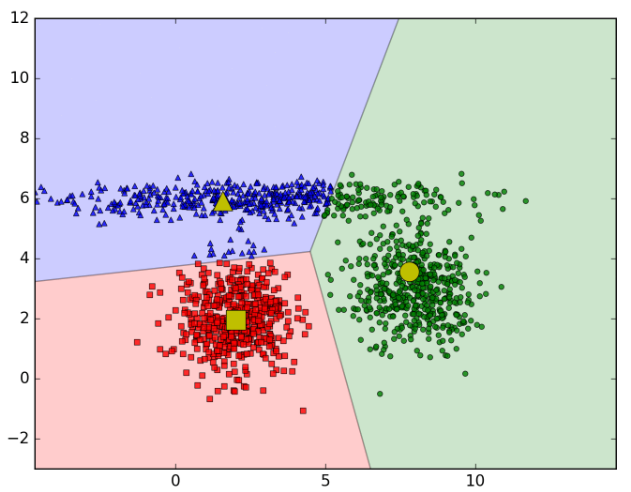


Từ biểu đồ, ta có thể thấy được, thuật toán K-means đã không làm tốt nhiệm vụ của nó là phân chia các cụm. Ta thấy được các cụm lệch rất nhiều so với kì vọng.

Với các cụm trong dữ liệu có số lượng điểm có sự chênh lệch nhau lớn, K-means cũng làm không tốt nhiệm vụ của nó và đưa ra kết quả không chính xác:

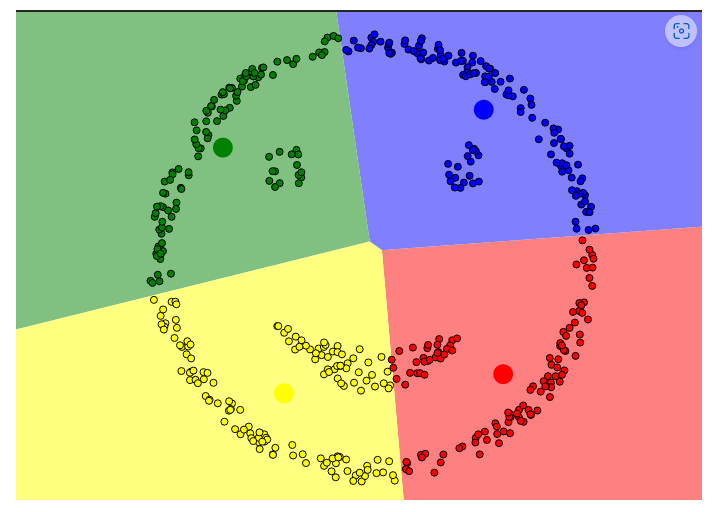


Hơn nữa, K-means cũng chỉ hoạt động tốt nếu như các cụm có dạng hình cầu. Nếu cụm có dạng hình dẹt, thuật toán sẽ hoạt động không hiệu quả:



Ta có thể thấy ở đây, cụm hình dẹt gây khó khăn cho thuật toán K-means để phân cụm nó. Kết quả đưa ra là chưa tối ưu với dữ liệu hiện có và thậm chí nó còn đang phân cụm sai khi nhầm phần tử của các cụm.

Thuật toán Kmeans cũng không thể phân cụm dữ liệu khi có các cụm ở trong cụm khác. Ở đây, ta có thể thấy được có các cụm là mắt phải, mắt trái, miệng và viền bao quanh. Nhưng vì mắt và miệng nằm ở trong khuôn mặt nên K-means không thể phân cụm được:



Mặc dù có những hạn chế, K-means clustering vẫn cực kỳ quan trọng trong Machine Learning và là nền tảng cho nhiều thuật toán phức tạp khác sau này. Vì vậy, khi sử dụng K-means, bạn hãy chú ý đến kết quả đầu ra nếu như muốn một kết quả hiệu quả nhất đối với bài toán của bạn.

**IV. Tài liệu tham khảo**

# Junjie Wu. *Advances in K-means Clustering: A Data Mining Thinking.* Springer Science & Business Media, 2012.

Mahmoud Parsian. *Data Algorithms: Recipes for Scaling Up with Hadoop and Spark.* O'Reilly Media, Inc., 2015.

*Naftali Harris: Visualizing K-Means Clustering.* January 19, 2014. https://www.naftaliharris.com/blog/visualizing-k-means clustering/.

*DOMINO: Getting started with k-means clustering in Python.* April 1, 2022. https://www.dominodatalab.com/blog/getting-started-with-k-means-clustering-in-python