ỦY BAN NHÂN DÂN TP HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

Họ và tên sinh viên: Lê Minh Cường

BÁO CÁO

Thuật toán K-Mean cho bài toán phân cụm người dùng truy cập và tương tác trên Instagram

Data Mining

Giảng viên hướng dẫn:Lê Minh Nhựt Triều

TP. Hồ Chí Minh, 23 tháng 3 năm 2021

MỤC LỤC

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN 4

1. CHƯƠNG 1. TỔNG QUAN VỀ BÀI TOÁN 5

1.1. Tóm tắt 5

1.2. Đối tượng 5

1.3. Mục đích 5

2. CHƯƠNG 2. TỔNG KẾT LÝ THUYẾT 6

2.1. Định nghĩa Kmean 6

2.2. Độ phức tạp 7

2.3. Mô tả 7

2.4. Lịch sử 8

2.5. Một số khái niệm liên quan trong Kmean 8

2.5.1. Unsupervised Learning (Học không giám sát) 8

2.5.2. Một số thuật toán liên quan 9

2.6. Công thức tính khoảng cách 9

2.6.1. Minkowski distance 10

2.6.2. Mahattan distance 10

2.6.3. Euclidean distance 10

2.6.4. Cosine distance 11

3. CHƯƠNG 3: TRÌNH BÀI ỨNG DỤNG 13

3.1. Trình bày cách giải Kmean với dữ liệu nhỏ 13

3.1.1. Dữ liệu ban đầu 13

3.1.2. Vòng lặp đầu tiên 14

3.1.3. Vòng lặp 2: 15

3.1.4. Vòng lặp 3: 16

3.2. Mô tả dữ liệu 18

3.2.1. Giới thiệu dữ liệu 18

3.2.2. Giới thiệu thuộc tính của dữ liệu 18

3.2.3. Xử lý dữ liệu đầu vào 18

3.3. Mô tả thuật toán Kmean bằng code 24

3.3.1. Bước 1: Kiểm tra dữ liệu đầu vào 24

3.3.2. Bước 2: Khởi tạo các điểm centroid ban đầu 24

3.3.3. Bước 3: Thực hiện phân cụm dữ liệu 25

3.3.4. Thực hiện hiển thị kết quả 28

3.4. Một số cải tiến và kĩ thuật cho Kmean 29

3.4.1. Cách tìm k 29

CHƯƠNG 4. ĐÁNH GIÁ 37

CHƯƠNG 5: KẾT LUẬN 39

TÀI LIỆU THAM KHẢO 40

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

1. CHƯƠNG 1. TỔNG QUAN VỀ BÀI TOÁN

1.1. Tóm tắt

Mạng xã hội là một trong những nơi chúng ta có thể chia sẻ, tương tác và liên lạc với bạn bè. Từ khi có mạng xã hội, khoảng cách địa lý không còn là vấn đề quan trọng nữa, chúng ta có thể tương tác ở bất kì đâu, bất kì lúc nào không kể thời gian. Sự tiện lợi của mạng xã hội đã khiến nhiều người biết tới, trong đó chúng ta không thể kể đến là: Facebook, Instagram, Zing,...Liệu chúng ta có bao giờ quan tâm đến chỉ số của mỗi người khi truy cập một trang xã hội nào đó chưa? Liệu có sự phân hóa nào giữa mỗi người truy cập và tương tác trên mạng xã hội được ghi nhận? Đó là bài toán mà chúng ta muốn đề cập ở đây.

1.2. Đối tượng

- Số liệu về người dùng truy cập trên Instagram.
- Thuật toán Kmean và các kĩ thuật tối ưu hóa bài toán bằng việc dùng Ellbow và Silhoutte để tìm giá trị k.
- Các lý thuyết và định nghĩa liên quan đến thuật toán Kmean.

1.3. Mục đích

Nhằm tìm được sự phân hóa giữa những người truy cập trên mạng xã hội, về sự tương quan giữa thời gian truy cập và tương tác của mạng xã hội. Nhằm đem lại những cái nhìn cụ thể hơn bằng biểu đồ dễ tương tác và trực quan. Bằng việc sử dụng thuật toán Kmean - một trong những thuật toán rất nổi tiếng trong kĩ thuật Upsupervised, tìm cụm hay nhóm từ những dữ liệu chưa được đánh nhãn sẵn. Từ đó chúng ta sẽ đánh giá kết quả dễ dàng hơn. Chúng ta sẽ bàn về những điểm thuận lợi và không thuận lợi của thuật toán Kmean. Đánh giá kết quả sau khi sử dụng thuật toán Kmean với dữ liệu điểm thực tế.

Cuối cùng chúng ta sẽ bàn thêm về những thông tin hữu ích như việc cải thiện bài toán bằng những hàm những kĩ thuật được sử dụng chung với Kmean, làm cho kết quả bài toán trở nên chính xác và có kết quả hợp lý nhất.

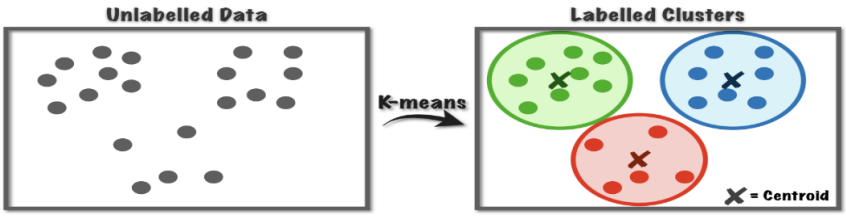
2. CHƯƠNG 2. TỔNG KẾT LÝ THUYẾT

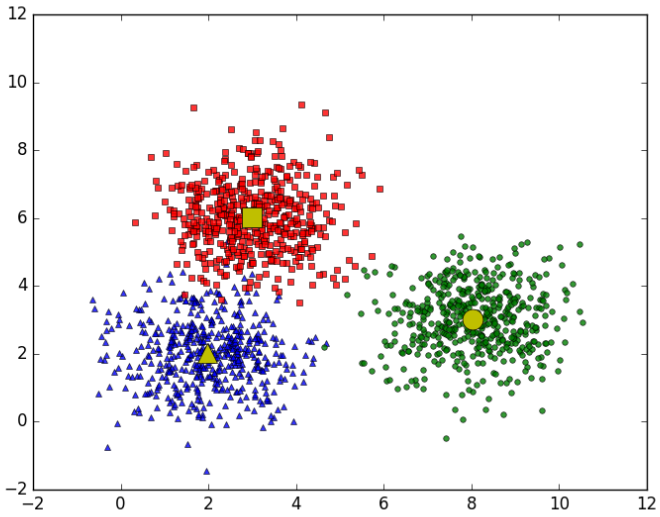
2.1.Định nghĩa Kmean

Là thuật toán lặp đi lặp lại để cố gắng phân chia dữ liệu thành k nhóm nhỏ mà không bị trùng lặp lên nhau, nơi mà mỗi điểm dataset chỉ thuộc về một trong một nhóm nào đó. Thuật toán phân bổ data points vào cluster đã được biết trước đó cụ thể là giá trị k, bằng cách làm giảm trung bình của tổng bình phương khoảng cách giữa tất cả điểm data points với centroid (điểm nằm giữa của cluster) là nhỏ nhất. Ít độ sai lệch nhất có thể.

Thuật toán được mô tả bằng các bước sau đây:

- Bước 1: Chỉ định ra số lượng K clsuters.
- Bước 2: Khởi tạo các điểm centroid bằng việc trộn tất cả dataset và lấy ngẫu nhiên ra K điểm data points làm centroid.
- Bước 3: Tính khoảng cách giữa centroid và datapoint. Sau đó, gán data point với cluster gần nhất (centroid).
- Bước 4: Tính toán lại tất cả centroid bằng việc tính trung bình cộng của tất cả data point mà những điểm thuộc mỗi cluster.
- Bước 5: Lặp cho đến khi không có sự thay đổi giữa các centroid (những điểm data point được gán tới clusters không bị thay đổi).
- Bước 6: Nếu phần tử của bất kì clusters nào đó không bị đổi thì dừng thuật toán, nếu không thì tiếp tục bước 3.
- Bước 7: Tính trung bình của bình phương khoảng cách giữa datapoints và centroid – Distortion (Có nghĩa là trước hết với một cụm, tính tổng bình phương khoảng cách từ centroid với các điểm thuộc về centroid đó, làm tương tự với những cluster khác, cuối cùng thì chúng ta sẽ chia cho tổng số lượng tất cả datapoint. Một cluster sẽ có 1 centroid và nhiều datapoint của cluster đó, chọn 2C1 để tính khoảng cách thì sẽ có n trường hợp. Ví dụ: cluster1 (n1 trường hợp), cluster 2 (n2 trường hợp), cluster 3 (n3 trường hợp) thì cuối cùng cũng bằng với tổng số lượng datapoints).





Thuật toán với k = 3

2.2.Độ phức tạp

Tìm cách tối ưu hóa để giải quyết kmean clustering cho những data point trong n chiều là NP-hard trong không gian Euclidean

2.3.Mô tả

Cho datapoint (x1, x2, x3, ... xn) cái mà mỗi datapoint là d chiều, k mean cố phân n data point thành S={S1, S2, S3, S4,...Sk} cluster để mà cố làm giảm phương sai

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 = \arg\min_{\mathbf{S}} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

Nơi μ_i là trung bình của Si. Nó bằng với việc giảm bình phương độ lệch chuẩn của tất cả các điểm trong cùng một cluster

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \frac{1}{2|S_i|} \sum_{\mathbf{x},\mathbf{y}\in S_i} \|\mathbf{x} - \mathbf{y}\|^2$$

2.4.Lịch sử

Kmean được sử dụng lần đầu tiên bởi James MacQueen vào năm 1967, dựa trên ý tưởng của Hugo Steinhaus trong năm 1956. Chuẩn thuật toán được đề xuất bởi Stauart Lloyd ở phòng thí nghiệm Bell Labs vào năm 1957 như là một kĩ thuật điều chế xung mã. Mặc dù nó không được công khai trên báo cho tới tận 1982. Năm 1965, Edward W. Forgy công khai một phương thức như vậy, cái mà tại sao nó thỉnh thoảng được tham khảo như là Floy-Forgy algorithm.

2.5.Một số khái niệm liên quan trong Kmean

2.5.1. Unsupervised Learning (Học không giám sát)

Trong thuật toán này, chúng ta không biết được kết quả mà chỉ có dữ liệu đầu vào. Thuật toán unsupervised learning sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán.

Một cách toán học, Unsupervised learning là khi chúng ta chỉ có dữ liệu vào XX mà không biết nhãn YY tương ứng.
Những thuật toán loại này được gọi là Unsupervised learning vì không giống như Supervised learning, chúng ta không biết câu trả lời chính xác cho mỗi dữ liệu đầu vào. Giống như khi ta học, không có thầy cô giáo nào chỉ cho ta biết đó là chữ A hay chữ B. Cụm *không giám sát* được đặt tên theo nghĩa này.

Các bài toán Unsupervised learning được tiếp tục chia nhỏ thành hai loại:

2.5.1.1. ***Clustering (phân nhóm)***

Một bài toán phân nhóm toàn bộ dữ liệu XX thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm. Ví dụ: phân nhóm khách hàng dựa trên hành vi mua hàng. Điều này cũng giống như việc ta đưa cho một đứa trẻ rất nhiều mảnh ghép với các hình thù và màu sắc khác nhau, ví dụ tam giác, vuông, tròn với màu xanh và đỏ, sau đó yêu cầu trẻ phân chúng thành từng nhóm. Mặc dù không cho trẻ biết mảnh nào tương ứng với hình nào hoặc màu nào, nhiều khả năng chúng vẫn có thể phân loại các mảnh ghép theo màu hoặc hình dạng.

*2.5.1.2. **Association***

Là bài toán khi chúng ta muốn khám phá ra một quy luật dựa trên nhiều dữ liệu cho trước. Ví dụ: những khách hàng nam mua quần áo thường có xu hướng mua thêm đồng hồ hoặc thắt lưng; những khán giả xem phim Spider Man thường có xu hướng xem thêm phim Bat Man, dựa vào đó tạo ra một hệ thống gợi ý khách hàng (Recommendation System), thúc đẩy nhu cầu mua sắm.

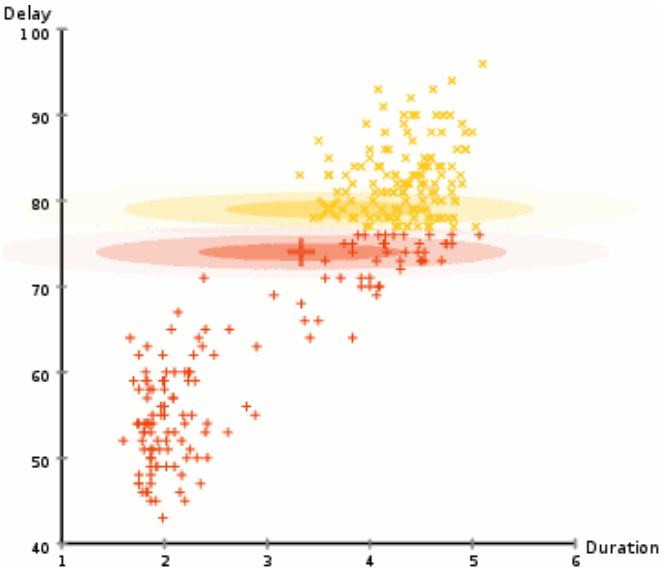
2.5.2. Một số thuật toán liên quan

- K-Medians

Theo định nghĩa, thuật toán K-Median là thuật toán phân tích cụm. Nó là một loại khác của kmean thuật toán này sẽ thay vì tính toán lại trung bình cộng của mỗi cluster để xác định lại centroid, chúng ta sẽ tính toán median của nó.

- Expectation Maximization (EM)

  Trong thống kê, thuật toán kỳ vọng – tối đa hóa (EM) là một phương pháp lặp lại để tìm khả năng tối đa (cục bộ) hoặc tối đa ước tính hậu kỳ (MAP) của các tham số trong mô hình thống kê, trong đó mô hình phụ thuộc vào các biến tiềm ẩn không được quan sát.



2.6.Công thức tính khoảng cách

Chúng ta có thể biết được rất nhiều công thức tính khoảng cách khác nhau, ở đây chúng ta sẽ tìm hiểu:

- Minkowski Distance

- Euclidean Distance
- Mahattan Distance
- Cosine Distance

2.6.1. Minkowski distance

Minkowski Distance là khái quát của công thức tính khoảng cách

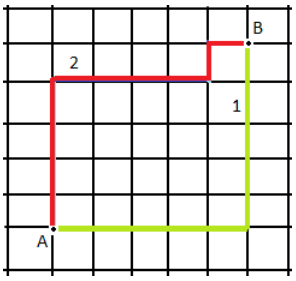
$$\left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p}$$

Trong công thức thì chúng ta có p:

- p = 1 Mahatta distance
- p = 2 Euclidean distance
- p = 3 Chebychev distance

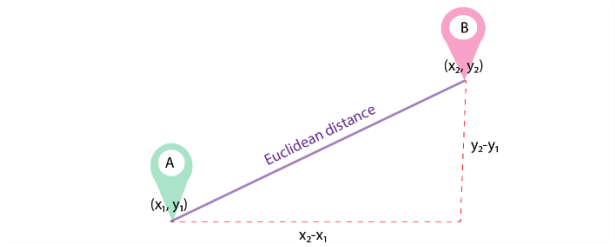
2.6.2. Mahattan distance

Chúng ta sử dụng Mahattan, hay còn được biết tới với tên là "city block distance" hay "taxicab geometry" nếu chúng ta muốn tính khoảng cách giữa 2 điểm trong đường đi grid. Công thức tính khoảng cách Mahattan được hiểu như ví dụ dưới đây



Để tính khoảng cách giữa điểm A, B đã cho trước như đường màu đỏ, thì Mahattan sẽ ra được đường màu vàng chính xác là khoảng cách của 2 điểm A và B. Đường đi sẽ không phải là đường thẳng mà sẽ có khúc rẽ.

2.6.3. Euclidean distance



$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

Công thức tính khoảng cách của Euclidean được dung để tính khoảng cách giữa điểm. Chúng ta có thể thấy nó có liên quan đến công thức của Pythago.

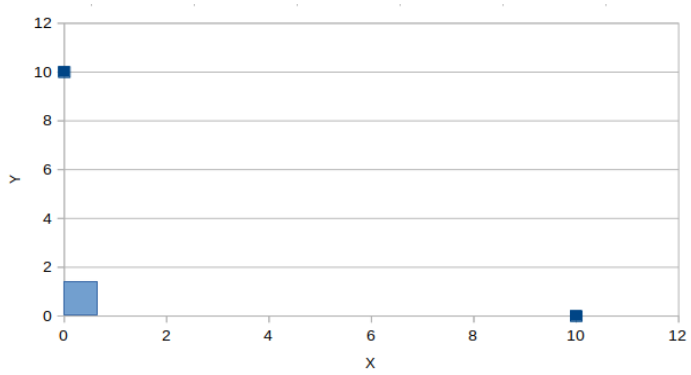
2.6.4. Cosine distance

Cosince distance được sử dụng chính để tìm sự giống nhau giữa hai điểm. Bởi vì khoảng cách giữa điểm tang, độ giống nhau cosine hay là lượng giống nhau sẽ giảm đi.

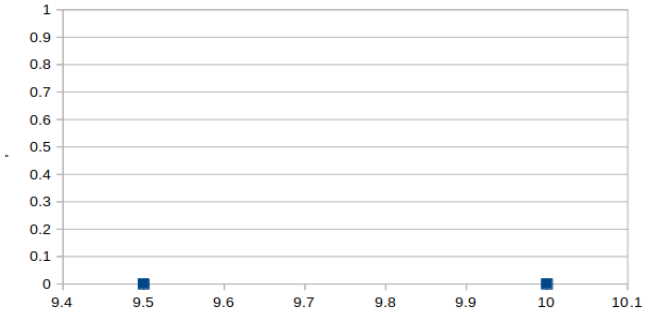
Công thức

Cosine similarity: $\cos\theta$

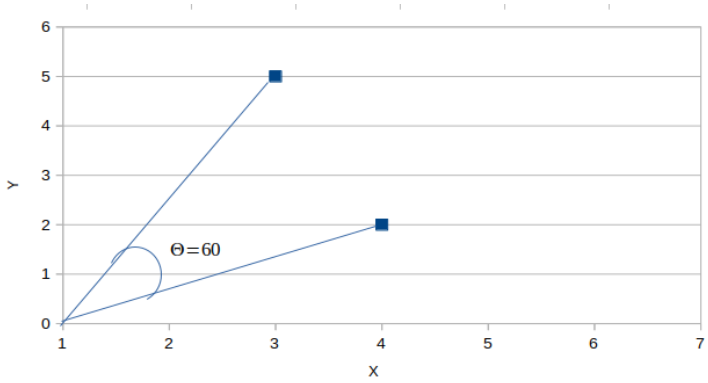
Khoảng cách trong cosine sẽ là:1 - $\cos\theta$



Hiện tại gốc giữa hai điểm là 90 độ. Vậy Cosine similarity sẽ là $\cos 90 = 0$. Vì vậy mà 2 điểm này sẽ không có điểm chung và khoảng cách giữa chúng sẽ là 1 - $\cos\theta = 1$



Nếu gốc giữa 2 điểm là 0 độ thì Cosine similarity sẽ là 1 và khoảng các sẽ là = 0. Vì vậy mà điểm này giống với nhau.



Tương tự với 2 điểm với gốc là 60 độ chúng ta sẽ tì được khoảng cách giữ 2 điểm này là 0.5 và cosine similarity sẽ là 0.5.

Mahattan được sử dụng nhiều hơn Euclidean khi chúng ta có dữ liệu có nhiều chiều hơn 2. Cosine được dùng chính để tính lượng giống tương tự (giống nhau) giữa hai điểm dữ liệu.

Theo như thuật Kmean thì chúng ta cố gắng làm giảm bình phương sự sai lệch (bằng với bình phương khoảng cách của công thức Euclidean). Nếu chúng ta muốn tìm công thức tính khoảng cách khác, chúng ta cần phải thay thế mean với giá trị ước tính một cách thích hợp như là thuật toán K-medoid, nhưng điều này sẽ rất khó khăn trong việc tìm medoids.

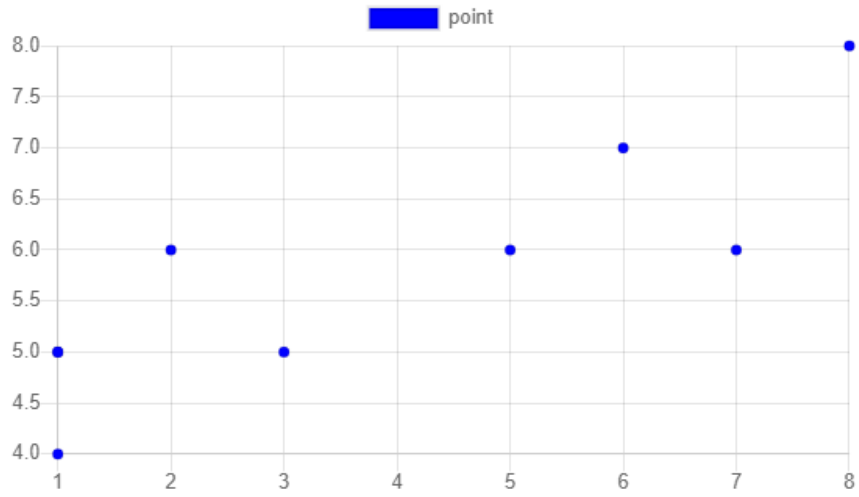
3. CHƯƠNG 3: TRÌNH BÀI ỨNG DỤNG

3.1.Trình bày cách giải Kmean với dữ liệu nhỏ

3.1.1. Dữ liệu ban đầu

Chúng ta sẽ có dữ liệu

index	x	y
0	3	5
1	1	4
2	1	5
3	2	6
4	8	8
5	7	6
6	6	7
7	5	6



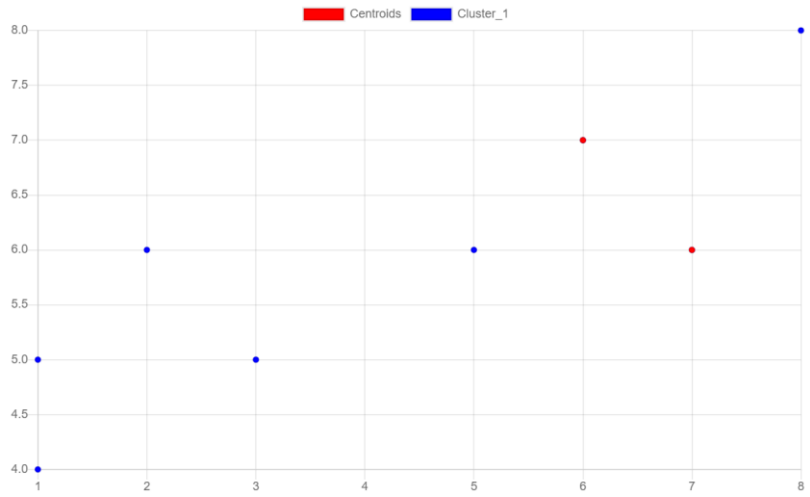
Hiển thị dữ liệu trên đồ thị

Chúng ta sẽ chia thành dữ liệu thành 2 cluster

Bước 1: Chỉ định ra số cluster = 2

Bước 2: Khởi tạo các điểm centroid

```
Centroid_1 = {x: 7, y: 6}
Centroid_2 = {x: 6, y: 7}
```



Bước 3:

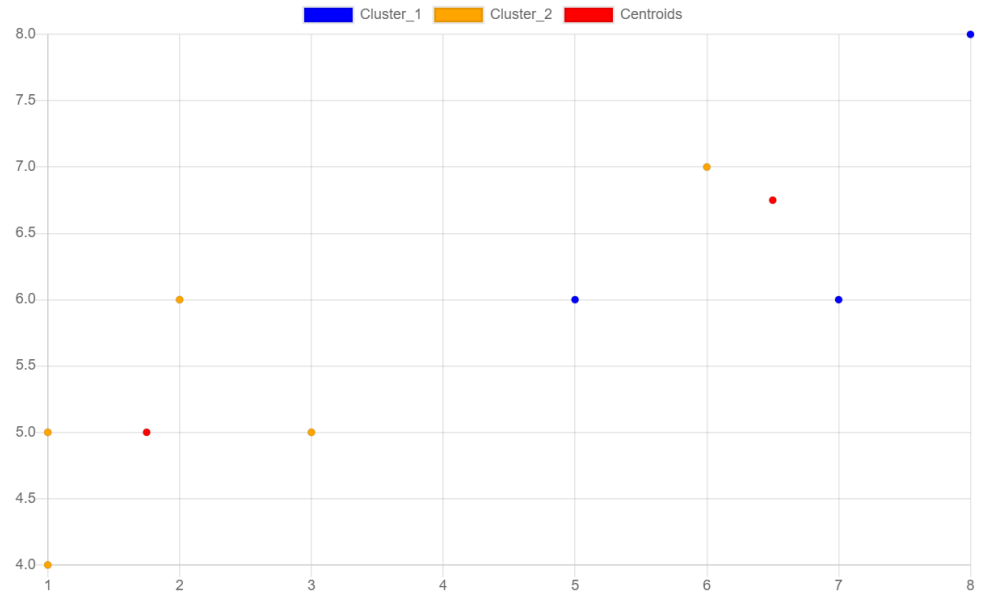
3.1.2. Vòng lặp đầu tiên

Centroid_1 = {x: 7, y: 6}

Centroid_2 = {x: 6, y: 7}

Tính khoảng cách centroid và data point

x	y	d(x, c1)	d(x, c2)	Kết quả
3	5	4.123105625617661	3.605551275463989	2
1	4	6.324555320336759	5.830951894845301	2
1	5	6.082762530298219	5.385164807134504	2
2	6	5	4.123105625617661	2
8	8	2.23606797749979	2.23606797749979	1
7	6	0	1.4142135623730951	1
6	7	1.4142135623730951	0	2
5	6	2	1.4142135623730951	1



Tính toán lại centroid. Do đây là bước đầu tiên.

Centroid_1 = {x: 6.67, y: 6.67}

Centroid_2 = {x: 2.6, y: 5.4}

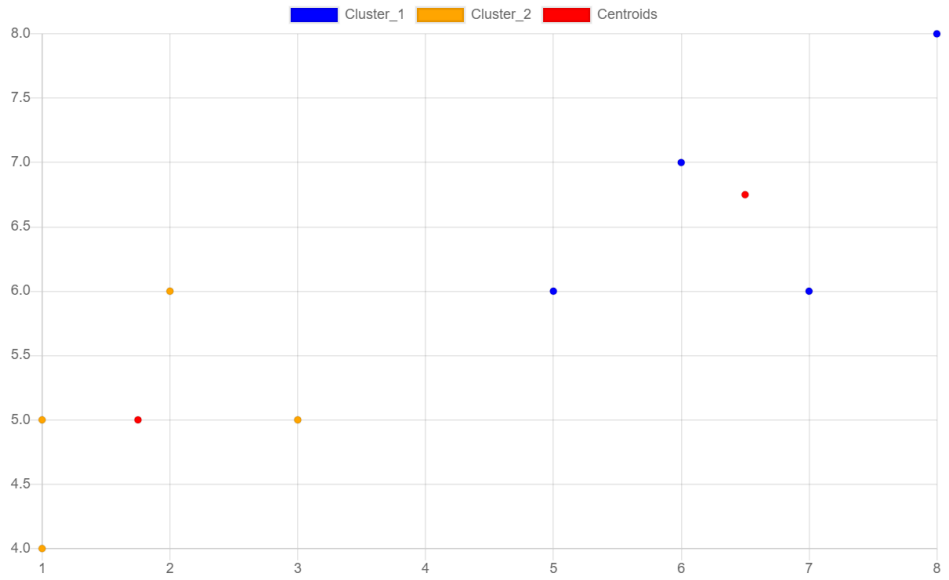
3.1.3. Vòng lặp 2:

Centroid_1 = {x: 6.67, y: 6.67}

Centroid_2 = {x: 2.6, y: 5.4}

Tính khoảng cách centroid và data point

x	y	d(x, c1)	d(x, c1)	Kết quả
3	5	4.032096229010413	0.5656854249492381	2
1	4	6.267200331886639	2.12602916254693	2
1	5	5.910820586010034	1.6492422502470645	2
2	6	4.717817291926426	0.8485281374238569	2
8	8	1.8809040379562165	5.993329625508679	1
7	6	0.7468600939935136	4.440720662234904	1
6	7	0.7468600939935136	3.757658845611187	1
5	6	1.799388785115657	2.473863375370596	1



So sánh với kết quả trước ta nhận thấy rằng khác nhau trong Cluster 1 vì có thêm điểm (6,7) nữa nên chúng ta sẽ tiếp tục tính lại Centroid

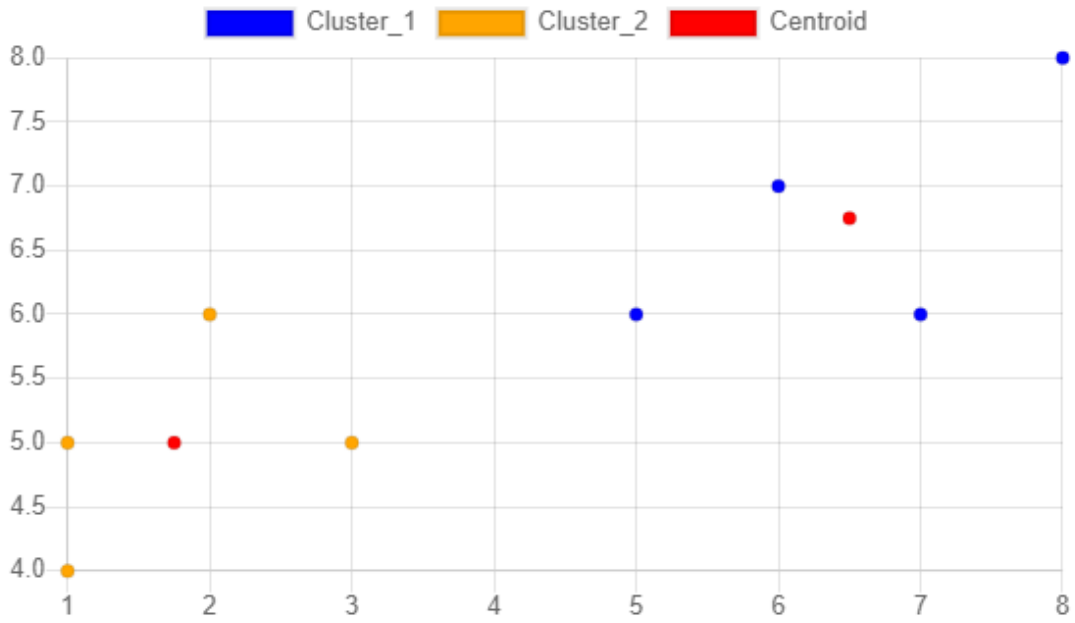
```
Centroid_1 = {x: 1.75, y: 5}
Centroid_2 = {x: 6.5, y: 6.75}
```

3.1.4. Vòng lặp 3:

```
Centroid_1 = {x: 1.75, y: 5}
Centroid_2 = {x: 6.5, y: 6.75}
```

Tính khoảng cách centroid và data point

x	y	d(x, c1)	d(x, c1)	Kết quả
3	5	1.25	3.913118960624632	2
1	4	1.25	6.149186938124422	2
1	5	0.75	5.771698190307598	2
2	6	1.0307764064044151	4.562071897723665	2
8	8	6.932712311931024	1.9525624189766635	1
7	6	5.344389581607987	0.9013878188659973	1
6	7	4.697073557013984	0.5590169943749475	1
5	6	3.400367627183861	1.6770509831248424	1



Như vậy chúng ta thấy rằng tại vòng lặp 3 thì kết quả giống kết quả vòng lặp 2. Vì thế mà chúng ta sẽ dừng vòng lặp tại đây và in kết quả ra màn hình.

Vậy cuối cùng chúng ta sẽ thu được kết quả là

Cluster_1:

x	y
8	8
7	6
6	7
5	6

Cluster_2:

x	y
3	5
1	4
1	5
2	6

Với ví dụ đơn này chúng ta đã hiểu được cách thức để tiến hành một bài toán với dữ liệu thật và to hơn. Chúng ta sẽ bàn thêm về cách chọn k ở bài toán thực tế. Chứ không dừng lại với việc chọn ngẫu nhiên như vậy. Bây giờ chúng ta sẽ bước đến bài toán thực tế của chúng ta.

3.2.Mô tả dữ liệu

3.2.1. Giới thiệu dữ liệu

Bao gồm dữ liệu của những người đã truy cập vào mạng xã hội Instagram, gồm những thuộc tính thể hiện được thời gian họ dành ra và tương tác tới mạng xã hội.

3.2.2. Giới thiệu thuộc tính của dữ liệu

Dữ liệu của chúng ta bao gồm các thuộc tính:

- Id: là mã của người dùng truy cập
- Visit score(1-100): là số điểm mà người đó truy cập trang bao gồm số lượt xem trang nào đó ví dụ như xem thông tin cá nhân của những người khác cũng được tính là truy cập, thời gian dành để truy cập/số lần truy cập, thời gian truy cập 1 trang/số trang,...Cách tính của số liệu này rất phức tạp. Nhưng mà Visit core thể hiện được hành vi của người dùng và cách người đó tương tác với trang mạng xã hội.
- Spending score(1-100): là thang điểm từ 1 đến 100 việc người dùng vào một trang mạng xã hội mất bao lâu để sử dụng.

3.2.3. Xử lý dữ liệu đầu vào

Trước hết ta sẽ xử lý dữ liệu đầu vào với file csv dưới đây.

```
0,63,24.05070845
1,61,25.22329005
2,104,18.52824526
3,82,86.89023241
4,14,31.49239692
5,74,81.19531414
6,83,88.11972945
```

Dữ liệu trong từng dòng được phân cách với nhau bằng dấu (,) lần lượt là user id, visit score và spending score.

```
function readFileCSVAndParseToJSON() {
    let lr = new line('./data/instagram visits clustering.csv',{
        encoding: 'utf8',
        start: 0
    })

    lr.on('error', function (err) {
        console.log(err)
    });

    let result = []
    lr.on('line', function(line) {
        let string = line
        let split = string.split(",")
        let obj = {}
        let userID = split[0]
        let visitScore = split[1]
        let spendingRank = split[2]

        obj = {
            userID,
            visitScore,
            spendingRank
        }
        result.push(obj)
    })

    lr.on('end', function () {
        fs.writeFileSync('./data/finalUser.json', JSON.stringify(result), {flag: 'w'})
    });
```

Chúng ta cần chuyển dữ liệu từ csv sang dạng json để dễ xử lý hơn. Chúng ta sẽ sử dụng thư viên read-by-line của nodejs để thực hiện đọc từng dòng, do là dữ liệu cách

nhau bằng dấu (,) nên chúng ta dễ dàng split ra và lấy theo vị trí index 0, 1, 2. Mỗi lần chúng ta đọc kết quả của 1 dòng, xử lý xong chúng ta sẽ tạo 1 biến obj sau đó đẩy vào mảng kết quả . Sau đó, cuối cùng chúng ta sẽ phải thực hiện dùng thư viên fs để tạo lại file json và lưu vào folder data .

Bây giờ, chúng ta sẽ tới bước mô tả dữ liệu bằng code

```
92    let descriptData = describe(data)
```

Ở line 92 này chúng ta sẽ gọi hàm describe để mô tả dữ liệu từ biến data chúng ta vừa dùng ở trên để ra các giá trị gồm: min, max, mean và các giá trị null.

```
function describe(data) {
    let result = describeOneClass(data, [
        "visitScore",
        "spendingRank"
    ])
    return result
}
```

Trong hàm describe chúng ta sẽ gọi thêm 1 hàm describeOneClass. Hàm describe cơ bản là sẽ xử lý nhiều mảng như data nhưng hiện tại do chỉ có duy nhất có một mảng data nên chúng ta không dùng vòng lặp for. Chúng ta thực hiện chỉnh sửa lại cách trình bày code và gọi đúng 1 hàm describeOneClass, hàm này sẽ truyền vào 2 tham số, tham số 1 là dữ liệu, tham số thứ 2 là những thuộc tính mà chúng ta muốn sử dụng để mô tả. Chúng ta sẽ dùng hết tất cả thuộc tính trừ thuộc tính user_id tức là mã của người dùng mà thôi, vì vậy chúng ta sẽ không thêm thuộc tính này vào.

```
function describeOneClass(arrayItem, attributes) {
    let obj = {};
    let data = []
    for (let i = 0; i < attributes.length; i++) {
        let max = _.maxBy(arrayItem, function (o) {
            return o[attributes[i]];
        });
        let min = _.minBy(arrayItem, function (o) {
            return o[attributes[i]];
        });
        let mean = _.meanBy(arrayItem, function (o) {
            return o[attributes[i]];
        });

        let nullValue = arrayItem.reduce((accumulator, currentValue) => {
            if(currentValue[attributes[i]] === null) {
                accumulator = accumulator + 1
            }
            return accumulator
        }, 0)

        data = arrayItem.map(item => item[attributes[i]])

        obj[attributes[i]] = {
            max: max[attributes[i]],
            min: min[attributes[i]],
            mean: mean,
            null: nullValue,
            data
        };
    }
    return obj;
}
```

Kết quả cuối cùng của hàm trả về min, max, mean, null và data tức là các giá trị của loại thuộc tính đó. Tất cả giá trị này sẽ được lưu lại vào file để hiển thị bằng biểu đồ.

```
    // đếm số lượng dữ liệu ban đầu
    let countItemOriginal = data.length
```

Thực hiện đếm dữ liệu ban đầu

```
// fs.writeFileSync('./data/train.json', JSON.stringify(trainData))
// tính toán tổng số data point trong dữ liệu
let countItemKmeanData = trainData.length
```

Thực hiện đếm dữ liệu train

```
// chúng ta sẽ xử lý dữ liệu đầu vào
trainData = _.uniqWith(data, function(arr, other) {
    return arr.visitScore === other.visitScore && arr.spendingRank === other.spendingRank
})
```

Lọc dữ liệu bị trùng trong mảng trainData.

```
trainData = trainData.filter(item => {
    return item.visitScore >= 0 && item.visitScore <= 100 && item.spendingRank >= 0 && item.spendingRank <= 100
})

```

Do thang điểm của chúng ta là từ 1-100 nên chúng ta cần đảm bảo dữ liệu của chúng ta là hợp lệ.

```
// sử dụng 2 class bất kì để thực hiện tính distortion
// vì chúng ta cần visualize dữ liệu theo x, y nên chúng ta sử dụng 2 class
let distortions = []
let inertias = []
for(let i = 1; i <= 10; i++) {
    let r = kmean(trainData, i, [
                "userID",
                // "visitScore",
                // "spendingRank"
            ]);

    distortions.push({
        k: i,
        distortion: r.distortion
    })
    inertias.push({
        k: i,
        inertia: r.inertia
    })
}
```

Thực hiện dùng thuật toán Ellbow để tìm k. Chúng ta sẽ phải chạy vòng lặp từ 1 đến 10 để có thể tìm distortion, inertia (Chi tiết) và hiển thị lên đồ thị. Trong hàm kmean chúng ta sẽ trả luôn kết quả bao gồm distortion và inertia. Lưu tất cả các giá trị vừa đánh giá ở các bước trên lưu vào file và chuẩn bị cho bước hiện thị lên. Sau khi xong bước này chúng ta sẽ thực hiện hiển thị giá trị để có bước nhìn tổng quan nhất về dữ liệu đầu vào.

Chúng ta dùng thêm 1 thuật toán khác là Silhouette để thực hiện tìm k để đảm bảo kết quả là đúng đắn nhất

```
// tính toán lại silhoutte của giá trị k
let silhoutteResult = silhouette(result.groupArray)
```

```
// lưu vào file
fs.writeFileSync('./data/original.json', JSON.stringify({
    countItemOriginal,
    countItemKmeanData,
    descriptData,
    data
}), { flag: 'w' })

// lưu tất cả kết quả của kmean
fs.writeFileSync('./data/initialKmean.json', JSON.stringify({
    distortions,
    inertias,
    silhoutteResult
}), { flag: 'w' })
```

Thực hiện lưu tất cả dữ liệu trên vào file để hiển thị lên biểu đồ.

```
original = await new Promise((resolve) => {
  $.getJSON('./data/original.json', function(data) {
    resolve(data)
  })
});

initialKmean = await new Promise((resolve) => {
  $.getJSON('./data/initialKmean.json', function(data) {
    resolve(data)
  });
})
```

Đọc dữ liệu file json ra và lưu vào biến

```
// mô tả dữ liệu
moTaDuLieu()
// ======================================================================
// Mô tả Thuộc tính của từng loại dữ liệu
moTaThuocTinhBangTable()
// ======================================================================
// Mô tả Thuộc tính của từng loại dữ liệu bằng chart
moTaThuocTinhBangChart()
// ======================================================================
moTaDuLieuBanDauCacDiemBangChart()
// ======================================================================
moTaSilhoutte()
// ======================================================================
// distorion
moTaDistrotion()
// ======================================================================
// inertia
moTaInertia()
```

Viết hàm để hiển thị kết quả ra màn hình

3.2.3.1. Mô tả tổng dữ liệu đầu vào

Mô tả dữ liệu ban đầu

Dữ liệu ban đầu: 2600 Dòng

Dữ liệu sau khi chúng ta xử lý trùng: 2543 Dòng

Tổng số dữ liệu ban đầu là 2600 dòng, sau bước lọc dữ liệu thì chúng ta còn dữ liệu là 2543 dòng. Chúng ta hãy xem giá trị lúc đầu gồm những gì tiếp theo.

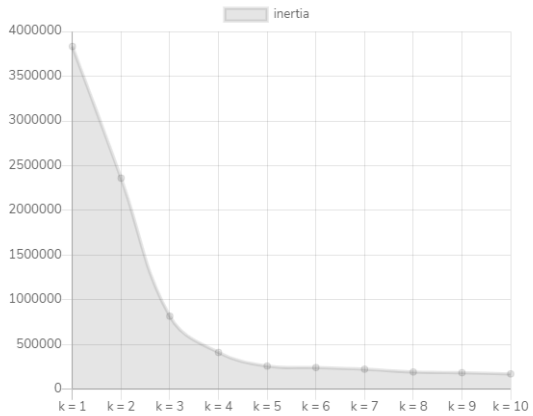
3.2.3.2. Tiếp theo là tất cả thuộc tính bằng bảng:

Mô tả thuộc tính của từng class

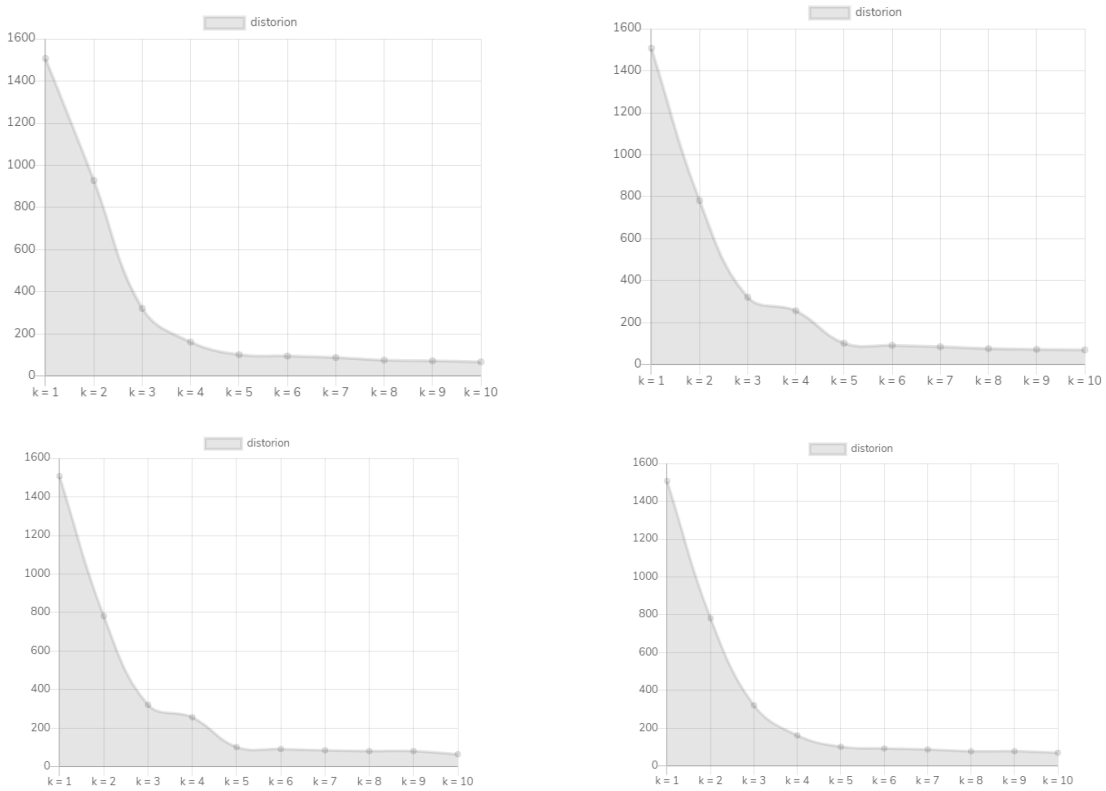
Thuộc tính	max	min	mean	null
visitScore	118	5	63.32346153846154	0
spendingRank	107.3498213	0.940708942	42.84840781350309	0

Ở đây chúng ta thấy rằng không có giá trị null nào cả, vì vậy chỉ có khả năng dữ liệu của chúng ta có một số là không hợp lệ

3.2.3.3. Mô tả Inertia



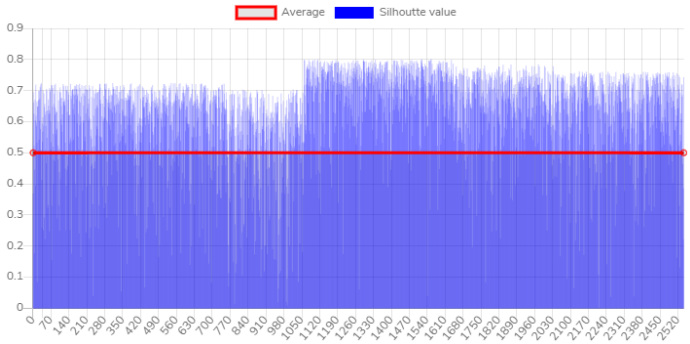
3.2.3.4. Mô tả distortion



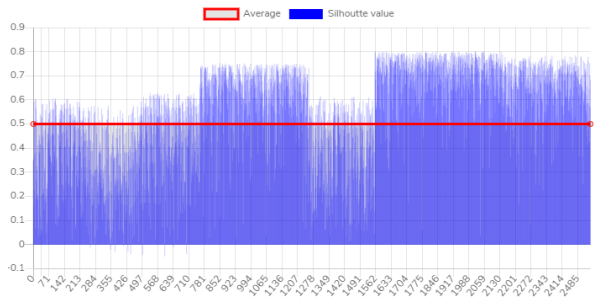
Từ dữ liệu của đồ thị chúng ta có thể thấy, chúng ta sẽ chọn k bằng 5, vì với k bằng 5 thì distortion và inertia giảm dần theo tuyến tính một cách đều đặn nhất. Sau qua 4 lần chúng ta dùng ellbow để thực hiện so sánh kết quả. Chi tiết Ellbow

Một phương pháp thứ 2 chúng ta có thể sử dụng đó là dùng thuật toán Silhoutte để đánh giá mức độ giống nhau giữa 1 điểm thuộc về cluster đó với các điểm cùng nằm chung và mức độ khác nhau giữa 1 điểm thuộc về cluster đó với những điểm khác cluster hiện tại. Chi tiết Silhoutte

Đối với việc chọn k bằng Ellbow hay Silhoutte thì chúng ta cần phải chạy đi kết quả nhiều lần và lấy kết quả tốt nhất của tất cả các trường hợp. Nguyên nhân là việc lấy ngẫu nhiên các điểm centroid ban đầu sẽ dẫn tới việc có các giá trị distortion khác nhau.



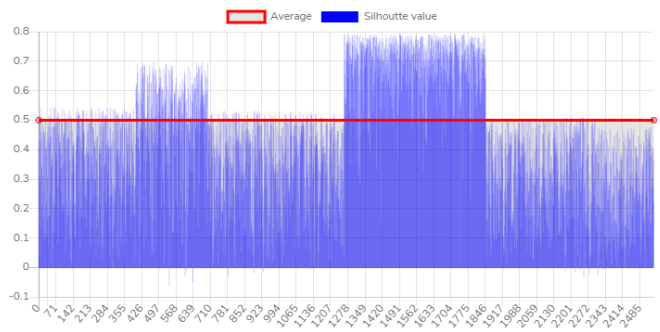
Đường màu đỏ là giá trị trung bình của Silhoutte value, chúng ta nhận thấy rằng giá trị của tất cả các phần tử đều ở trên mức trung bình và gần 0.7 và 0.8. Với mức giá trị như vậy có thể hiểu rằng giá trị mà một điểm datapoint đang nằm trong cluster cao và datapoint đó đang nằm đúng với cluster. Chúng ta có thể thử giá trị Silhoutte với những giá trị k khác:




```
-0.012598816386178768
-0.00045175290275045743
-0.00009897809169912009
-0.0023788481164039865
-0.019936447866042095
-0.0013367761263340938
-0.0017749831567007032
-0.029994919104239237
-0.006754240936925893
-0.028504986481551753
-0.038766879204951366
-0.04308526918170341
-0.0483938685455795 9
-0.009757520799940522
-0.03907050667060952
```

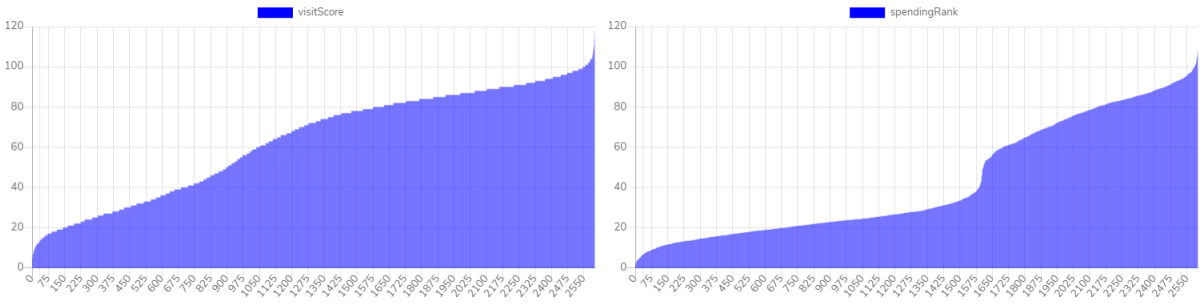
Với giá trị k bằng 7 chúng ta có thể thấy rằng giá trị này làm giảm độ chính xác của giá trị datapoint nằm đúng vị trí cluster mà nó thuộc về. Hình bên phải là giá trị Silhoutte của từng data point, hình bên trái là các giá trị Silhoutte bị âm.

Tương tự k = 8




```
-0.06203699144018082
-0.029852201598379535
-0.005991602510780791
-0.04805826022140147
-0.030138276973402434
-0.0017862899072014216
-0.017868375557761662
-0.01716984510579811
-0.002978217066953115
-0.010614620081270898
-0.016205703618874967
-0.0084556001302015 24
-0.03149879030684022
-0.017924569788998812
-0.0268856544918743
-0.006443698875608805
```

3.2.3.5. Mô tả thuộc tính bằng chart:



Vậy cuối cùng giá trị k chúng ta sẽ chọn là k bằng 5 với 2 thuộc tính score_spending và score_visit.

3.3.Mô tả thuật toán Kmean bằng code

Như vậy sau khi chúng ta đánh giá được giá trị cần lấy và loại bỏ những giá trị null thì chúng ta thực hiện vào

Đầu vào: một mảng data để phân cụm, số k cluster muốn phân cụm.

Đầu ra: dữ liệu đã được phân cụm cùng với một số thuộc tính thêm như, distortion, inertia, mảng centroid.

```
function kmean(data, k, excludeKeys = [])
```

excludeKeys được bổ sung để loại bỏ những attributes không muốn dùng trong thuật toán, thay vì dùng hàm loại bỏ chúng ta ghi thêm để tiện sử dụng.

3.3.1. Bước 1: Kiểm tra dữ liệu đầu vào

```
4    if(data.length <= 0) {
5        throw new Error("missing data")
6    }
7
8    if(!k || isNaN(k)) {
9        throw new Error("missing k")
10   }
11
12   if(k > data.length) {
13       throw new Error("k is greater than data.length")
14   }
```

Chúng ta kiểm tra dữ liệu đầu vào của hàm để tránh trường hợp thiếu hoặc sai dữ liệu đầu vào.

3.3.2. Bước 2: Khởi tạo các điểm centroid ban đầu

```
16   // Initialize centroids by first shuffling the dataset and then randomly selecting K data points for the centroids without replacement.
17   // copy array from source array
18   let arrayCentroid = []
19   let copyArr = [...data]
20   for(let i = 0 ; i < k; i ++) {
21       // random index in copy array
22       let index = Math.floor(Math.random() * copyArr.length);
23       // get item from index in copy array
24       arrayCentroid.push({...copyArr[index]})
25       // delete that item in copy array
26       copyArr = [
27           ...copyArr.slice(0, index),
28           ...copyArr.slice(index + 1)
29       ]
30   }      You, a week ago • [INIT]
```

Ở dòng 18 chúng ta sẽ dùng mảng arrayCentroid để chứa tất cả điểm centroid. Ở dòng 19 chúng ta sẽ dùng mảng này để chọn ra những controid cú pháp 3 chấm (...) được gọi là spread operator trong javascript nhằm tạo một mảng mới y như mảng ban đầu nhưng các giá trị của mảng sẽ không bị tham trị, điều này tránh việc chúng ta xử lý mảng hiện tại làm ảnh hưởng đến mảng gốc.

Dòng lặp for sẽ chạy từ 1 đến k cluster, có bao nhiêu k thì sẽ có bấy nhiêu cluster, tại sao phải random điểm như vậy? Mỗi centroid ngẫu nhiên khác nhau sẽ tạo nên kết quả

khác nhau điều này làm cho bài toán có kết quả đa dạng, chúng ta sẽ phải thực nhiên ngẫu nhiên chọn centroid để bài toán có thể may mắn đạt trường hợp tốt nhất. Dòng 22 là thực hiện việc random một vị trí index trong khoảng từ 0 cho đến chiều dài của mảng copyArr, vì sau khi chúng ta chọn xong chúng ta sẽ phải loại bỏ dữ liệu đó và đưa chúng vào một mảng centroid, điều này sẽ lặp lại cho tới khi chúng ta nhận được k centroid (initial centroid). Dòng 24 là tìm được vị trí và đưa vào mảng centroid. Dòng 27 đến 29 là loại bỏ dữ liệu đó ra khỏi mảng copyArr.

3.3.3. Bước 3: Thực hiện phân cụm dữ liệu

Chúng ta sẽ thực hiện cho đến khi nào kết quả của phân cụm trước bằng với kết quả của phân cụm hiện tại. Đó là lý do chúng ta dùng vòng lặp while true.

```
35    let previousArrayId = []
36    while(true) {
```

```
36    while(true) {
37        // console.log('Time: ', m)
38        // save data into group centroid [[1,2,3], [4], [5,6], [7]]
39        let groupArray = createArrayTwoDemension(k)
40
41        for(let i = 0 ; i < data.length; i++) {
42            // calculate distance between centroid (j) and current point (data[i])
43            let arrayDistance = []
44            for(let j = 0 ; j < k ; j++) {
45                arrayDistance.push(distance(arrayCentroid[j], data[i], excludeKeys))
46            }
47
48            // find min distance between centroid (j) and current point (data[i])
49            let min = arrayDistance[0]
50            let minCentroidIndex = 0  let j: number
51            for(let j = 1 ; j < k ; j++) {
52                if(min > arrayDistance[j]) {
53                    min = arrayDistance[j]
54                    minCentroidIndex = j
55                }
56            }
57            // put data into group minCentroidIndex
58            groupArray[minCentroidIndex].push(data[i])
59        }
```

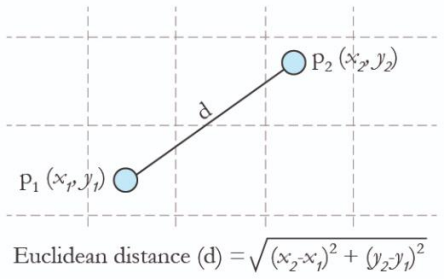
Dòng 39 chúng ta khởi tạo một mảng nhiệm vụ của mảng này là sẽ chứa tất cả các cluster. Ví dụ chúng ta có 3 cluster thì mảng groupArray của chúng ta sẽ có dạng là [[a,c], [b,d,f], [e]]. Điều này sẽ giúp chúng ta quản lý và đưa dữ liệu vào đúng centroid một cách dễ dàng theo index tương ứng. Giả dụ data1 có dữ liệu gần centroid 3 (Vị trí thứ 2 trong arrayCentroid với vị trí index được tính từ 0). Thì chúng ta dễ dàng đẩy dữ liệu vào bằng cú pháp groupArray[vị trí nhóm của centroid 3].push(data1). Vị trí nhóm của centroid 3 chính là 2. Dòng 41 chúng ta sẽ thực hiện lặp theo độ dài của mảng data. Dòng 43 khởi tạo một mảng distance, mục đích là chúng ta muốn biết ở tại data thứ i thì với k điểm centroid, thì độ dài từ centroid nào tới điểm đó là ngắn nhất, nên chúng ta phải thực hiện tính khoảng cách xong rồi mới tìm min. Dòng 49 đến 56 là

thuật toán tìm min trong các khoảng cách từ data[i] tới centroid[j]. Dòng 58 chúng ta thực hiện đưa data[i] vào đúng cluster dựa vào khoảng cách nhỏ nhất của nó với centroid[minCentroidIndex].

Công thức tính khoảng cách ở đây chúng ta sẽ dựa vào công thức Euclidean

Theo Euclidean công thức sẽ có dạng:

- Đối với 2 điểm có 2 chiều



- Đối đối 2 điểm với n chiều:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

p, q = two points in Euclidean n-space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

n = n-space

Chúng ta sẽ dựa vào đó và viết thuật toán trên n chiều

```
39  function distance(x, y, excludeKeys) {
40      // (xv1 - yv1)^2 + (xv2 - yv2)^2 + (xv3 - yv3)^2 = dij^2
41      let sum = 0;
42      Object.keys(x).forEach(key => {
43          if(excludeKeys.indexOf(key) === -1) {
44              sum += Math.pow(Math.abs(x[key] - y[key]), 2)
45          }
46      })
47      // console.log('x', x, 'y', y)
48      // console.log(sum)
49      return Math.sqrt(sum)
50  }       You, a week ago • [INIT]
```

Đầu vào là x,y là 2 điểm có nhiều chiều x và y sẽ có cùng một kiểu lớp dữ liệu, excludeKeys là những thuộc tính trong x và y chúng ta không muốn tính toán trong đó ví dụ như là name, class,...viết thêm để chúng ta tiện sử dụng.

```
42      Object.keys(x).forEach(key => {
```

Ở đây câu lệnh này có nghĩa là chúng ta muốn lặp qua tất cả các attribute của x và y

```
43          if(excludeKeys.indexOf(key) === -1) {
44              sum += Math.pow(Math.abs(x[key] - y[key]), 2)
45          }
```

Nếu như key tức attribute hiện tại không nằm trong mảng excludeKeys thì chúng ta sẽ

thực hiện công thức $(q_i - p_i)^2$. Tổng tất cả lại chúng ta sẽ có dạng $\sum_{i=1}^{n}(q_i - p_i)^2$ Từ đó chúng ta thực hiện căn bậc 2 của tổng đó sẽ ra được khoảng cách từ x tới y

```
return Math.sqrt(sum)
```

```
// [[nhóm của centroid 0], [nhóm của centroid 1], [nhóm của centroid 2]]
// groupArray[indexCentroid].push(data[i])
// [centroid0,            centroid1,            centroid2           ]
```

Sau khi chúng ta thực hiện lặp data.length dữ liệu. Bây giờ chúng ta sẽ phải tính toán lại centroid dựa vào groupArray. Thì hiện tại groupArray[0] sẽ tương ứng với centroid[0], groupArray[1] sẽ tương ứng với centroid[1]. groupArray[i] sẽ là một mảng chứa tất cả phần tử thuộc centroid thứ i đó.

Chúng ta sẽ phải tính toán lại centroid dựa vào trung bình cộng của tất cả data thuộc centroid đó.

```
61      // calculate new centroid
62      for(let i = 0 ; i < k ; i ++) {
63          // loop each k group centroid
64          let tempCentroid = copyObjectWithDefaultValue(data[0]);
65          // sum by key in groupArray
66          for(let j = 0; j < groupArray[i].length; j++) {
67              Object.keys(tempCentroid).forEach(key => {
68                  if(excludeKeys.indexOf(key) === -1) {
69                      tempCentroid[key] += groupArray[i][j][key]
70                  }
71              })
72          }
73          // average by key in groupArray
74          Object.keys(tempCentroid).forEach(key => {
75              if(excludeKeys.indexOf(key) === -1) {
76                  tempCentroid[key] = tempCentroid[key] / groupArray[i].length
77              }
78          })
79
80          // update centroid
81          arrayCentroid[i] = tempCentroid
82      }
```

Dòng 42 chúng ta sẽ tính lại k điểm centroid. Dòng 64 chúng ta sẽ tạo một mảng tạm để chứa tất cả các điểm centroid sau khi chúng ta tính toán lại mới. Dòng 66 mới tất cả data trong groupArray[i] tức là những data thuộc điểm centroid thứ i chúng ta sẽ cộng tất lại theo thuộc tính chung của data.

Cuối cùng chúng ta dùng thêm 1 vòng lặp nữa từ 74 đến 78 để chia trung bình cộng của giá trị centroid. Sau đó cập nhật arrayCentroid với giá trị mới sau khi tính toán xong.

```
// check differ pre and current array group
let count = 0
let currentArrayId = []
for(let i = 0 ; i < k ; i++) {
    let mapId = groupArray[i].map(item => item.id)
    // previousArrayId[i] = undefined
    let p = previousArrayId[i] ? previousArrayId[i].sort().toString() : ""
    if(mapId.sort().toString() === p) {
        count++
    }
    currentArrayId.push(mapId)
}
```

Ở đoạn lệnh này chúng ta sẽ phải kiểm tra rằng kết quả của phân cụm hiện tại có thực sự bằng với kết quả của phân cụm trước đó hay không. Chúng ta sẽ phải so sánh từng cluster. previousArrayId sẽ lưu tất cả những cluster trước đó dạng [[1,2],[3,4], [5]]. Chúng ta sẽ lặp từng cluster của cluster thứ i hiện tại với cluster thứ i trước đó coi có trùng khớp với nhau hay không. Nếu thực sự trùng khớp chúng ta sẽ tăng biến count lên 1. Việc so sánh được tính bằng việc sort lại theo id rồi chuyển thành string

Ví dụ:

- “1,2,3,4,5” với “1,2,3,4,5” => true => count + 1
- “1,2,3” với “1,2” => false

```
if(count === k) {
    let distortion = 0
    for(let m = 0 ; m < k; m++) {
        for(let n = 0 ; n < groupArray[m].length; n ++) {
            // by default distortion is calculated by: total_distortion(sum of distance from cluster to respective cluste
            // distortion = distortion + distance({ x: groupArray[m][n].x, y: groupArray[m][n].y }, arrayCentroid[m])
            distortion = distortion + Math.pow(distance(groupArray[m][n], arrayCentroid[m], excludeKeys), 2)
        }
    }

    distortion = distortion / data.length
ou, a week ago • [INIT]
    return {
        groupArray,
        arrayCentroid,
        distortion: distortion
    }
} else {
    previousArrayId = currentArrayId
}
m++
```

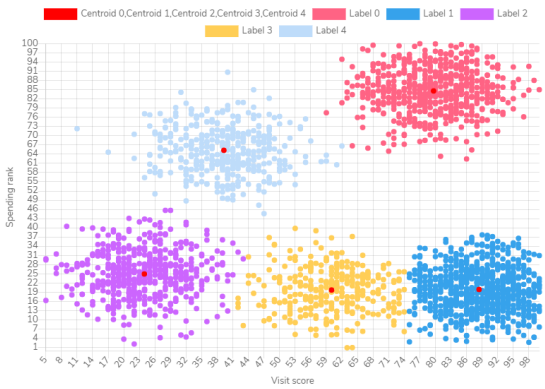
Sau bước tính count chúng ta sẽ coi điều kiện dừng của bài toán. Nếu như mà trùng tất cả k cluster thì dừng bài toán. Ở đây trước khi chúng ta thực hiện dừng bài toán thì chúng ta sẽ tính một số chỉ số như distortion và cuối cùng trả về dữ liệu như là groupArray, arrayCentroid. Nếu không trùng chúng ta sẽ lặp lại cho tới khi tìm kết quả.

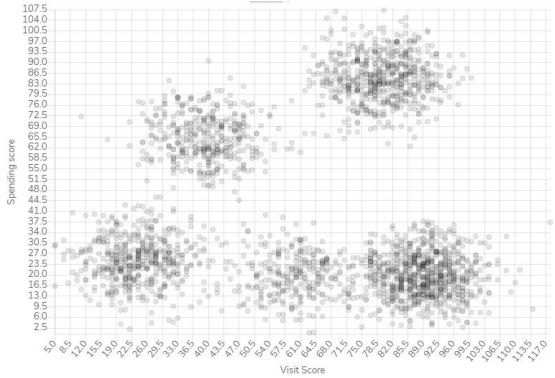
3.3.4. Thực hiện hiển thị kết quả

Chúng ta thực hiện gọi hàm kmean với tham số bằng 5 từ việc xác định distortion ở trên.

```
fs.writeFileSync('./data/kmean.json', JSON.stringify({
    groupArray: result.groupArray,
    arrayCentroid: result.arrayCentroid,
    distortion: result.distortion,
}), { flag: 'w' })
```

Lưu vào file json để hiển thị





Ở đây chúng ta có thể thấy rằng kết quả đã được chia ra rất rõ ràng thành 5 vùng so với hình ảnh của dữ liệu ban đầu.

Phần màu tím, vàng và xanh là những người dành thời gian ít cho việc dùng mạng xã hội nhưng mà ở đây lại phân hóa thành 3 vùng theo sự tương tác của người đó với trang mạng xã hội.

Màu xanh nhạt và màu hồng là những người dành thời gian cho mạng xã hội nhiều và những người này cho thấy mức độ tương tác khá cao. Màu xanh nhạt đã thấy rõ được rõ ràng những người dùng đang có xu hướng tương tác cao hơn, và màu hồng là màu rõ rệt nhất. Có thể nói đây là những người dùng mạng xã hội như là mục tiêu liên lạc, giao tiếp. Nếu như có thêm số liệu về những trang mà người dùng hay tới, những cú nhấp vào thể hiện cảm xúc của những người ở vùng hồng được thể hiện, thì Instagram có thể nắm bắt được xu hướng của nhóm người vùng đó, phát huy thế mạnh của mình. Như vậy mọi người dành nhiều thời gian cho mạng xã hội lâu hơn mục đích là để tương tác nhiều hơn và tương tác rất cao.

3.4.Một số cải tiến và kĩ thuật cho Kmean

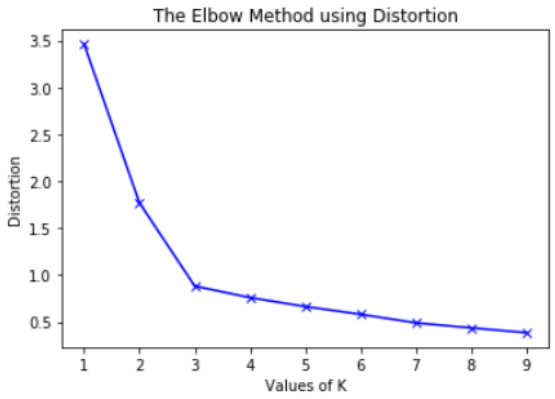
3.4.1. Cách tìm k

Theo như bài toán chúng ta có tất cả là 7 loại hạt, vậy sẽ như thế nào nếu chúng ta chưa hề biết có 7 loại, làm sao để chọn một chỉ số k là tốt nhất. Kmean có 2 phương pháp mà chúng ta có thể dùng để thuận lợi tìm k đó là

- Ellbow Method

- Silhouette Method

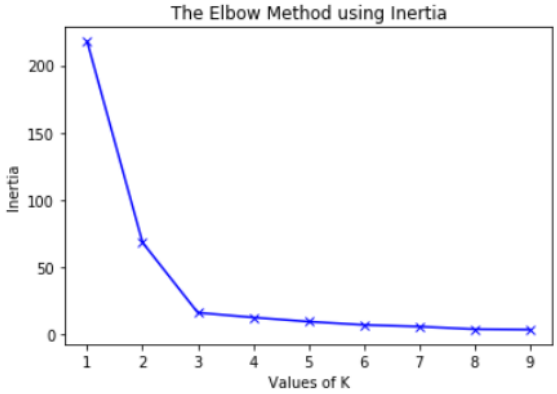
3.4.1.1. Ellbow Method



Ellbow là một phương pháp dùng để tìm k trong Kmean. Trong Ellbow, có một đại lượng tên là distorion với distortion được định nghĩa là trung bình của bình phương khoảng cách từ 1 điểm centroid tới những điểm mà thuộc về centroid đó.

Nguyên văn:

Distortion**:** It is calculated as the average of the squared distances from the cluster centers of the respective clusters. Typically, the Euclidean distance metric is used



Hoặc với Inertia là tổng của bình phương khoảng cách từ 1 điểm giữa (centroid) tới những điểm mà thuộc về centroid đó.

Chúng ta hãy quay về với định nghĩa của Phương sai (variance) vì sao chúng ta lại có công thức distortion, mối quan hệ giữa distortion và var

Công thức:

$$\sigma^2 = \frac{\sum(\chi - \mu)^2}{N}$$

xi là mẫu thứ i

$\bar{x}$ là giá trị trung bình của mẫu

n là tổng số mẫu

Ví dụ ta có: Tuổi của từng người là:

Tuấn 20 tuổi

Bình 25 tuổi

Sơn 19 tuổi

Thì giá trị trung bình $\bar{x}$ được tính bằng

Mean = $\frac{20+25+19}{3}$ = 21.3333

Vậy công thức tính giá trị phương sai sẽ là

σ^2 = $\frac{(21.3333-20)^2+(25-21.3333)^2+(21.3333-19)^2}{3}$ = 6.8888

Trong thực tế người ta thường dùng công thức độ lệch chuẩn vì nó thể hiện được tính chất của đối tượng trong khi phương sai kết quả được bình phương.

Độ lệch chuẩn được coi như là cách đánh giá chuẩn cái gì là bình thường, cái nào là lớn hơn và nhỏ hơn.

S = sqrt(σ^2) = 2.6246

Vậy vùng được gọi là bình thường là 2.6246 - 21.3333 < x < 2.6246 + 21.3333

Phương sai của dãy nào nhỏ hơn thì dãy đó có mức độ phân tán so với trung bình cộng ít hơn.

Theo như công thức phương sai ở trên ta có thể suy ra được distortion của Ellbow.

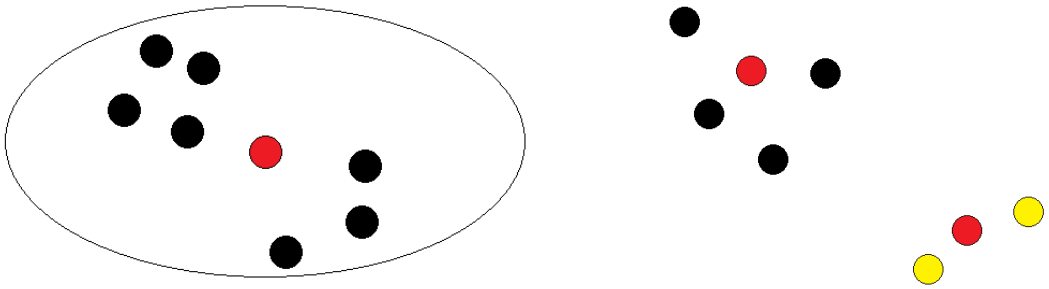
Chúng ta giả sử tất cả các centroid chính là giá trị trung bình đóng vai trò như $\bar{x}$ và tất cả các điểm data point như là x_i.

Thay vì x - $\bar{x}$ mục đích của việc này là tìm độ khác nhau, độ lệch giữa điểm. Áp dụng với một đối tượng có nhiều thuộc tính ta có thể áp dụng công thức tính khoảng cách như là Euclidean, Manhattan,... Sau đó, từ độ khác nhau chúng ta bình phương lên.

Cộng tất cả các trường hợp k của bài toán đó chính là Distortion.

Thường với k càng nhỏ thì độ distortion của chúng sẽ rất cao, vì theo phương sai (cũng có thể hiểu trong Kmean là Distortion) thì phương sai càng lớn thì phân tán càng cao.

Giả dụ với k = 1, chúng ta tìm được centroid, mỗi điểm trong data points đều nằm trong vùng của duy nhất centroid đó, bất kể nó xa hay gần. Vì thế mà làm cho distortion rất cao.



Nếu như k tăng lên thì chắc chắn sẽ làm giảm độ phân tán, lúc này những data point không còn phụ thuộc vào 1 điểm nữa mà có thể phụ thuộc vào centroid nào mà gần nó nhất (tăng sự chọn lựa hơn, 1 centroid thì chỉ chọn có 1, trong khi nếu 2 centroid thì sẽ có thể nằm ở 1 hoặc 2 nếu 1 trong 2 cái nào gần nhất) vì vậy mà khoảng cách giữa centroid và điểm thuộc về centroid sẽ nhỏ, làm cho distortion nhỏ, dẫn tới độ phân tán giảm. Đó là lý do chúng ta random centroid để tìm những centroid tốt hơn vì mỗi lần chúng ta đều khởi đầu với k centroid khác nhau dẫn tới kết quả có thể khác nhau. Vì vậy mà càng tăng k sẽ làm giảm độ distortion, đó là lý do tại sao đồ thị Ellbow có xu hướng giảm dần

Distortion là trung bình cộng của phương sai. Hiểu như là cho một điểm.

Inertia là gần giống với Distortion nhưng mà không có chia trung bình cộng.

Ví dụ về cách tính distortion:

Chúng ta có tập dữ liệu các điểm là:

(index)	x	y
0	3	5
1	1	4
2	1	5
3	2	6
4	1	5
5	6	8
6	6	6
7	6	7
8	5	6
9	6	7
10	7	1
11	8	2
12	9	1
13	8	2
14	9	3
15	9	2
16	8	3

Chúng ta có 2 điểm centroid là: { x: 8.285714285714286, y: 2 }, { x: 3.7, y: 5.9 }

Chúng ta có tập các điểm được phân theo cluster là:

(index)	0	1	2	3	4	5	6	7	8	9
0	{ x: 7, y: 1 }	{ x: 8, y: 2 }	{ x: 9, y: 1 }	{ x: 8, y: 2 }	{ x: 9, y: 3 }	{ x: 9, y: 2 }	{ x: 8, y: 3 }			
1	{ x: 3, y: 5 }	{ x: 1, y: 4 }	{ x: 1, y: 5 }	{ x: 2, y: 6 }	{ x: 1, y: 5 }	{ x: 6, y: 8 }	{ x: 6, y: 6 }	{ x: 6, y: 7 }	{ x: 5, y: 6 }	{ x: 6, y: 7 }

Distortion =

$$\frac{\sqrt{(8.28-7)^2+(2-1)^2}^2+\sqrt{(8.28-8)^2+(2-2)^2}^2+\sqrt{(8.28-9)^2+(2-1)^2}^2+\sqrt{(8.28-8)^2+(2-2)^2}^2+\sqrt{(8.28-9)^2+(2-3)^2}^2}{+\sqrt{(8.28-9)^2+(2-2)^2}^2+\sqrt{(8.28-8)^2+(2-3)^2}^2+\sqrt{(3.7-3)^2+(5.9-5)^2}^2+\sqrt{(3.7-1)^2+(5.9-4)^2}^2+\sqrt{(3.7-1)^2+(5.9-5)^2}^2}{+\sqrt{(3.7-2)^2+(5.9-6)^2}^2+\sqrt{(3.7-1)^2+(5.9-5)^2}^2+\sqrt{(3.7-6)^2+(5.9-8)^2}^2+\sqrt{(3.7-6)^2+(5.9-6)^2}^2+\sqrt{(3.7-6)^2+(5.9-7)^2}^2}{+\sqrt{(3.7-5)^2+(5.9-6)^2}^2+\sqrt{(3.7-6)^2+(5.9-7)^2}^2}{17}$$

$$= \frac{7.428571428571427+61}{17} = 4.0247$$

Trong một số ví dụ khác thì distortion chỉ bằng trung bình cộng của khoảng cách từ centroid tới các điểm thuộc về centroid đó. Nếu như vậy thì ví dụ trên thì sẽ là:

$$\frac{\sqrt{(8.28-7)^2+(2-1)^2}+\sqrt{(8.28-8)^2+(2-2)^2}+\sqrt{(8.28-9)^2+(2-1)^2}+\sqrt{(8.28-8)^2+(2-2)^2}+\sqrt{(8.28-9)^2+(2-3)^2}}{+\sqrt{(8.28-9)^2+(2-2)^2}+\sqrt{(8.28-8)^2+(2-3)^2}+\sqrt{(3.7-3)^2+(5.9-5)^2}+\sqrt{(3.7-1)^2+(5.9-4)^2}+\sqrt{(3.7-1)^2+(5.9-5)^2}}{+\sqrt{(3.7-2)^2+(5.9-6)^2}+\sqrt{(3.7-1)^2+(5.9-5)^2}+\sqrt{(3.7-6)^2+(5.9-8)^2}+\sqrt{(3.7-6)^2+(5.9-6)^2}+\sqrt{(3.7-6)^2+(5.9-7)^2}}{+\sqrt{(3.7-5)^2+(5.9-6)^2}+\sqrt{(3.7-6)^2+(5.9-7)^2}}{17}$$

$$= \frac{6.41+23.65}{17} = 1.7682$$

Cách tính Inertia:

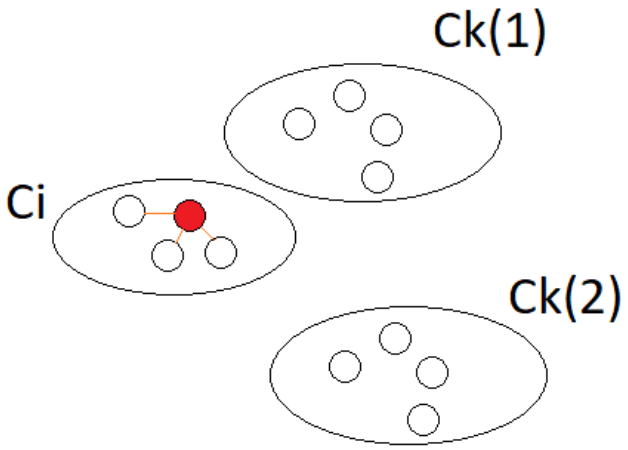
$$\frac{\sqrt{(8.28-7)^2+(2-1)^2}^2+\sqrt{(8.28-8)^2+(2-2)^2}^2+\sqrt{(8.28-9)^2+(2-1)^2}^2+\sqrt{(8.28-8)^2+(2-2)^2}^2+\sqrt{(8.28-9)^2+(2-3)^2}^2}{+\sqrt{(8.28-9)^2+(2-2)^2}^2+\sqrt{(8.28-8)^2+(2-3)^2}^2+\sqrt{(3.7-3)^2+(5.9-5)^2}^2+\sqrt{(3.7-1)^2+(5.9-4)^2}^2+\sqrt{(3.7-1)^2+(5.9-5)^2}^2}{+\sqrt{(3.7-2)^2+(5.9-6)^2}^2+\sqrt{(3.7-1)^2+(5.9-5)^2}^2+\sqrt{(3.7-6)^2+(5.9-8)^2}^2+\sqrt{(3.7-6)^2+(5.9-6)^2}^2+\sqrt{(3.7-6)^2+(5.9-7)^2}^2}{+\sqrt{(3.7-5)^2+(5.9-6)^2}^2+\sqrt{(3.7-6)^2+(5.9-7)^2}^2}$$

= 68.42857142857143

3.4.1.2. Sihouette Method

Silhouette là một công cụ khác để chúng ta có thể xác định được k. Giá trị Silhouette là giá trị để đo lường độ giống nhau giữa một đối tượng của cụm với những đối tượng khác thuộc về cụm đó và với những đối tượng thuộc về cụm khác

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

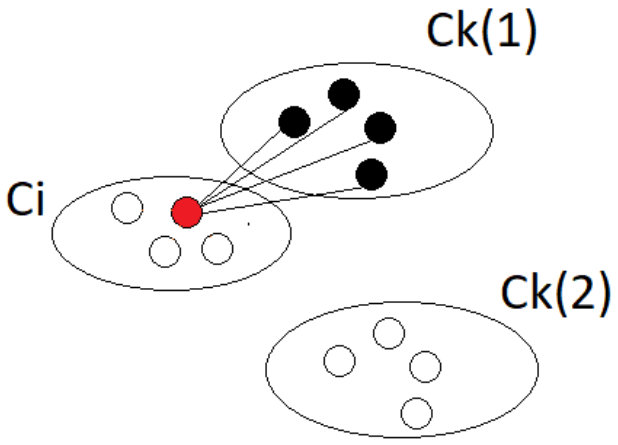


Màu đỏ là i, chúng ta sẽ tính nó với những datapoint Ci nằm trong cluster theo công thức phía trên

Đây là công thức để tính a(i) với i thuộc về Cluster Ci, nó có nghĩa là trung bình khoảng cách giữa i với những điểm khác trong cùng một cluster, d(i, j) là khoảng cách từ i tới j nằm trong cluster Ci. Chúng ta sẽ chia với $|C_i| - 1$ vì chúng ta sẽ chia trung bình trừ đi con i đó.

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i,j)$$

Tiếp theo chúng ta định nghĩa tiếp trung bình độ không khác nhau của điểm i với những cluster khác C_k, với C_k là khoảng cách trung bình từ i tới các điểm trong C_k ($C_k! = C_i$). Chúng ta chỉ chọn b(i) với giá trị b(i) là giá trị nhỏ nhất của trung bình khoảng cách không giống nhau hay còn gọi là "neighboring cluster" của i.



Chúng ta sẽ tính 1 điểm i với các điểm của 1 cluster khác Ci cụ thể trên hình là Ck(1) dùng công thức như trên hình cách tính b(i) Và tìm con Ck nào là nhỏ nhất.

Cuối cùng chúng ta sẽ tính giá trị Silhoutte kí hiệu là s(i):

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \text{ if } |C_i| > 1$$

Đơn giản hóa công thức trên ta được:

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

Giá trị của s(i) nằm trong khoảng từ 1 cho tới -1

b(i) thì thể hiện sự dissimilar.

a(i) thì thể hiện sự similar.

Với giá trị s(i):

- Nếu giá trị của s(i) là gần 1 có nghĩa là a(i) < b(i), tức là khoảng cách từ I đến với những datapoint thuộc cùng một cluster có độ khác biệt với nhau thấp so với I đến với những data point thuộc cluster khác với i. Vậy nên I đang nằm đúng với vị trí của mình.
- Nếu s(i) bằng 0 thì i nằm giữa biên của a(i) và b(i)
- Nếu s(i) < 0, tức là a(i) > b(i). Khoảng cách từ i đến những data point thuộc cùng cluster với nó có sự khác biệt cao làm cho a(i). Trong khi i đến datapoint thuộc cluster khác với nó lại nhỏ b(i). Đáng lẽ i sẽ phải nằm bên vị trí của b(i) đó mới đúng.

```
function silhouette(groupArray) {
    let result = []
    groupArray.forEach((pointsCurrentCluster, indexCurrentCluster) => {
        // let arr = []
        for (let k = 0; k < pointsCurrentCluster.length; k++) {
            // chọn 1 con trong Ai ra
            let i = pointsCurrentCluster[k];
            // tính Ai hiện tại
            let Ai = 0;
            for (let m = 0; m < pointsCurrentCluster.length; m++) {
                if (m === k) {
                    j = pointsCurrentCluster[m];
                    Ai += distance(i, j, ["userID"]);
                }
            }
            Ai = Ai / (pointsCurrentCluster.length - 1);
            let minBi = Infinity;
            for (let m = 0; m < groupArray.length; m++) {
                if (m === indexCurrentCluster) {
                    let pointOtherCluster = groupArray[m];
                    let Bi = 0;
                    for (let n = 0; n < pointOtherCluster.length; n++) {
                        Bi += distance(i, pointOtherCluster[n], ["userID"]);
                    }
                    Bi = Bi / pointOtherCluster.length;
                    if (Bi < minBi) {
                        minBi = Bi;
                    }
                }
            }
            let silhouetteValue = 0
            if(Ai < minBi) {
                silhouetteValue = 1 - Ai/minBi
            } else if(Ai == minBi) {
                silhouetteValue = 0
            } else {
                silhouetteValue = minBi/Ai - 1
            }
            if(silhouetteValue < 0) {
                console.log(silhouetteValue)
            }
            // arr.push(silhouetteValue)
            result.push(silhouetteValue)
        }
        // result.push(arr)
    });
    return result
}
```

Đối với việc tìm k bằng thuật toán Silhoutte sẽ lâu hơn là dùng Ellbow. Tùy vào ngữ cảnh và hiện trạng công cụ thì sử dụng thuật toán thích hợp.
Thuật toán Ellbow thì tính toán nhanh, nhưng độ k có thể không chính xác cao do nó chỉ tính riêng cho 1 cụm cluster. Trong khi Silhoutte thì lại tính 1 điểm với tất cả những điểm còn lại.

CHƯƠNG 4. ĐÁNH GIÁ

Thuận lợi:

Kmean thực sự là một thuật toán hay để có phân cụm tất cả các dữ liệu một cách dễ dàng và với độ chính xác chấp nhận được.

Cấu hình máy hiện tại là:

- Intel Core i7 9th
- Ram 8GB
- SSD Santa 200GB
- HP Pavillion 15

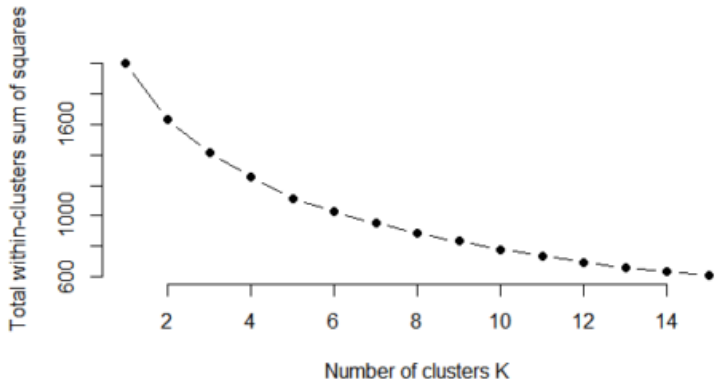
Kmean là nền móng cơ bản cho những bài toán phân cụm khác, ngoài Kmean chúng ta còn rất nhiều thuật toán phân cụm tốt hơn với độ chính xác tốt hơn. Nhưng với người bắt đầu tìm hiểu về machine learning thì nó là điểm sáng duy nhất để khởi động.

Dễ dàng nắm bắt với người bắt đầu tìm hiểu machine learning.

Bất lợi:

Cần biết số lượng cluster:

Khi vừa bắt đầu thuật toán Kmean chúng ta thực sự không biết được lượng cluster để phân ra với dữ liệu chưa được dán nhãn (label). Đối với dữ liệu đã được dán nhãn rồi thì chỉ cần phân trên lượng nhãn đã biết trước đó là hoàn tất.Tuy có áp dụng thuật toán Ellbow để tìm k, nhưng nếu k > 10 thì hoàn toàn khó tìm thấy được k



Như hình chúng ta rất khó xác định được k để đánh giá. Hầu như k tăng là distortion giảm một cách đều đặn. Chúng ta có thể sử dụng một method khác đó Silhouette nhưng sẽ có thời gian ra kết quả khá lâu vì nó tận độ phức tạp là O(n^2).

Các cluster cần có số lượng điểm bằng nhau

Nếu lượng cluster của mỗi điểm chênh lệch quá lớn sẽ ảnh hưởng kết quả ban đầu. Vì một khi dữ liệu của nhóm nào quá lớn nó sẽ làm phân hóa dữ liệu của nhóm nhỏ đi mất làm cho lúc này dữ liệu chúng ta hiển thị đi sẽ còn tính đúng đắn nữa. Vì vậy quá trình chọn lọc dữ liệu cũng là một bước cực kì quan trọng. Chỉ khi giá trị thuộc tính

mà chúng ta chọn cực kì khác biệt với những thuộc tính của nhóm khác thì sẽ không có sự phân hóa đó. Nhưng chung quy cuối cùng thì giá trị của thuộc tính vẫn sẽ thay đổi theo chiều hướng về nhóm có số lượng thuộc tính lớn hơn.

CHƯƠNG 5: KẾT LUẬN

Kmean không chỉ góp phần vào việc sắp xếp được tính chất các người dùng dựa vào đặc tính chúng, kết quả của bài toán tuy nằm ở mức chấp nhận được nhưng nó cũng là nền móng cho các bài toán Unsupervised khác.

Tốc độ của việc xử lý thuật toán trên dữ liệu lớn cũng khá nhanh và trong thời gian hợp lý. Thuật toán kmean tuy không cạnh tranh với các thuật toán khác nhưng đối với bài toán này nó đạt kết quả mong đợi.

Chúng ta có thể áp dụng nhiều thuật toán để tìm độ k cho hợp lý, cải thiện được tốc độ cũng như là độ overlap của các điểm data point như là Ellbow và Silhouette. Trong bài báo cáo này chúng ta nghiên cứu về thuật toán Ellbow và đã thấy được kết quả mà nó mang lại.

Em xin chân thành cảm ơn thầy Lê Minh Nhựt Triều đã giúp em trong quá trình làm đề tài để em hoàn thành xong bài báo cáo thành công

TÀI LIỆU THAM KHẢO

[1]. https://machinelearningcoban.com/2017/01/01/kmeans/

[2]. https://en.wikipedia.org/wiki/K-medians_clustering

[3]. https://www.mathsisfun.com/data/standard-deviation.html

[4]. http://i.vietnamdoc.net/data/file/2015/Thang05/18/bai-giang-phuong-sai-va-do-lech-chuan-Dai-so-10.pdf

[5]. https://nlp.stanford.edu/IR-book/html/htmledition/evaluation-of-clustering-1.html

[6] https://kipalog.com/posts/Thuat-toan-Kmean-va-ung-dung

[7]. https://datascience.stackexchange.com/questions/16700/confused-about-how-to-apply-kmeans-on-my-a-dataset-with-features-extracted

[8]. https://otexts.com/fpp2/useful-predictors.html

[9]. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1