TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC THÀNH PHỐ HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

Tìm hiểu giải thuật gom cụm k-Means của Scikit-Learn để gom cụm bộ dữ liệu mẫu Iris

Môn học: Trí Tuệ Nhân Tạo

GV hướng dẫn: PGS.TS Dương Tuấn Anh

Sinh viên thực hiện: Uông Thành Trung-22DH113985
Đỗ Văn Dũng- 22DH110615
Lê Phạm Hoàng Vũ- 22DH114826

Thành phố Hồ Chí Minh, tháng 7/2024

MỤC LỤC

Danh mục hình

Danh mục bảng

1. Giới thiệu thuật toán k-Means

1.1. Thuật toán k-Means là gì

K-mean clustering là một phương pháp để tìm các cụm và hạt nhân - trung tâm của cụm trong một tập hợp dữ liệu không được gắn nhãn. Người ta chọn số lượng hạt nhân cụm mong muốn phân chẳng hạn như k cụm. Thuật toán K-mean di chuyển lặp đi lặp lại các hạt nhân để giảm thiểu tổng số trong phương sai cụm. Với một tập hợp các hạt nhân ban đầu, thuật toán Kmeans lặp lại hai bước:

- Đối với mỗi hạt nhân, tính toán khoảng cách giữa các training ponit với nó và nếu gần nó hơn -> sẽ gán là cụm của hạt nhân đấy.
- Sau khi phân được cụm như ở bước trên, thì tiếp theo các training point của các cụm tính toán vector trung bình để được vị trí của hạt nhân mới và lặp lại bước trên đến khi không thể thay đổi được vị trí hạt nhân nữa.

Phân cụm có nhiều hữu ích đặc biệt và cực kỳ phổ biến trong ngành khoa học dữ liệu. Trong đó như:
Phân tích cụm được sử dụng rộng rãi trong nghiên cứu thị trường, nhận dạng mẫu, phân tích dữ liệu và xử lý ảnh. Phân tích cụm cũng có thể giúp các nhà khoa học dữ liệu khám phá ra các nhóm khác hàng của họ. Và họ có thể mô tả đặc điểm nhóm khách hàng của mình dựa trên lịch sử mua hàng. Trong lĩnh vực sinh học, nó có thể được sử dụng để xác định phân loại thực vật và động vật, phân loại các gen có chức năng tương tự và hiểu sâu hơn về các cấu trúc vốn có của quần thể.

Vậy để giải các bài toán về phân cụm cần có công cụ/phương pháp, K-means là một trong những thuật toán được sử dụng phổ biến nhất. Trong hướng dẫn này mình sẽ bắt đầu với cơ sở lý thuyết của thuật toán K-means, sau đó sẽ ứng dụng vào 1 ví dụ đơn giản với Python và thư viện Sklearn.

1.2. Mô tả thuật toán k-Means

Các giải thuật gom cụm phân hoạch thường sinh ra một phân hoạch của tập dữ liệu. Một giải thuật gom cụm nổi tiếng nhất trong loại giải thuật gom cụm này là giải thuật *k-means*.

Giải thuật *k-means* được mô tả gồm các bước sau đây:

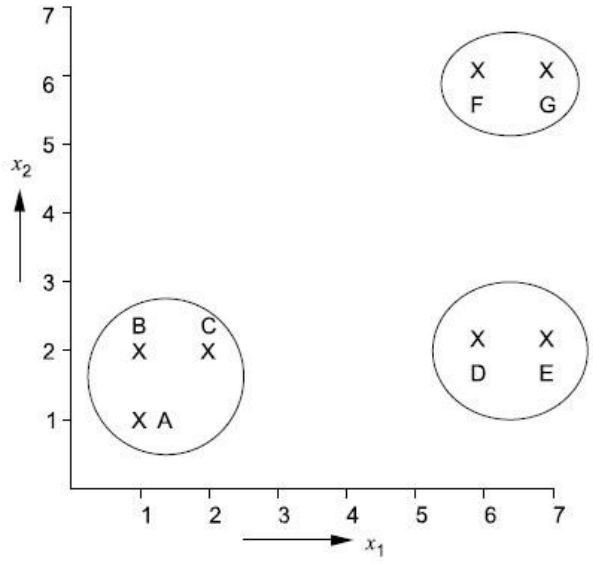
- Bước 1: Chọn ngẫu nhiên *k* mẫu trong số n mẫu để làm các trung tâm cụm đầu tiên. Gán mỗi mẫu trong số *n* - *k* mẫu đến *k* cụm theo nguyên tắc mỗi mẫu được gán vào cụm có trung tâm cụm gần với nó nhất.
- Bước 2: Tính các trung tâm cụm theo các cụm mới hình thành sau bước gán các mẫu nêu trên.
- Bước 3: Gán mỗi mẫu trong tập *n* mẫu đến cụm mà có trung tâm cụm gần với nó nhất.
- Bước 4: Nếu không có sự thay đổi khi gán mẫu vào các cụm giữa hai lượt lặp kế tiếp nhau thì ta dừng giải thuật; ngược lại thì quay lại bước 2.

Việc chọn *các trung tâm cụm ban đầu* (initial cluster centers) là một vấn đề rất quan trọng.

Để hiểu rõ hơn về thuật toán k-Means chúng ta xem ví dụ sau:

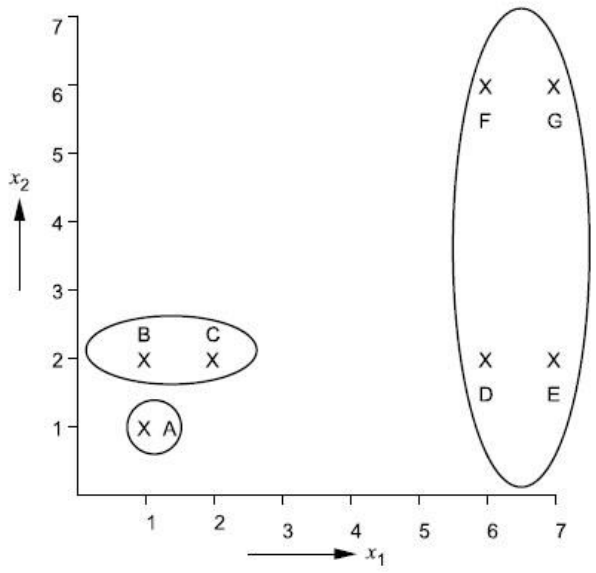
- Giải thuật *k*-Means với tập dữ liệu gồm 7 điểm hai chiều được minh họa trong hình 7.4 và 7.5.
- Các mẫu được cho là các điểm có các tọa độ như sau: A = (1, 1), B = (1, 2), C = (2, 2), D = (6, 2), E = (7, 2), F = (6, 6), G = (7, 6). Ta muốn gom cụm tập điểm này thành 3 cụm (*k* = 3).
- Nếu **A, D và F** được chọn làm các trung tâm cụm ban đầu. Cụm 1 sẽ có A= (1, 1) là trung tâm cụm của cụm 1. Cụm 2 có D= (6, 2) là trung tâm cụm của nó và cụm 3 có F= (6, 6) là trung tâm cụm của nó. Rồi thì, B, C $\Rightarrow$ Cụm 1; E $\Rightarrow$ Cụm 2; và G $\Rightarrow$ Cụm 3.

- Trung tâm mới của cụm 1 sẽ là điểm trung bình của các mẫu trong cụm 1, tức là (**1.33, 1.66**). Trung tâm mới của cụm 2 sẽ là (**6.5, 2**) và trung tâm mới của cụm 3 sẽ là (**6.5, 6**). Sau đó, A, B, C ⇒ Cụm 1, D, E ⇒ Cụm 2 và F, G ⇒ Cụm 3. Vì không có sự thay đổi trong các cụm vừa hình thành nên giải thuật kết thúc và các cụm đó là kết quả của giải thuật gom cụm k-means.



Hình 1.2. 1 Sự phân hoạch tốt nhất khi chọn A, D và F làm các trung tâm cụm ban đầu.

Điều này đem lại một phân hoạch tốt gồm 3 cụm {A, B, C}, {D, E} và {F, G}.



Hình 1.2. 2 Một sự phân hoạch không được tốt khi chọn A, B và C làm các trung tâm cụm ban đầu.

Nếu bắt đầu với A, B và C làm các trung tâm cụm ban đầu thì ta sẽ đạt được một sự gom cụm như trong hình 1.2.2.

Sự gom cụm này đem lại hai cụm có sự sai biệt nhỏ và cụm thứ ba có sai biệt lớn.

2. Độ đo khoảng cách

2.1. Euclidean

Khoảng cách Euclidean của một điểm P trong không gian n chiều đến gốc tọa độ được tính bằng căn bậc hai của tổng bình phương các tọa độ thành phần:

$$d(O, P) = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2}$$

Đây cũng chính là độ dài của vector nối từ gốc tọa độ đến điểm P. Với hai điểm P và Q, ta có khoảng cách Euclidean giữa chúng như sau:

$$d(P, Q) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}$$

Khoảng cách Euclidean đơn giản chỉ tính độ dài hình học giữa các điểm khác nhau trong không gian, coi mọi khoảng cách của các điểm có vai trò như nhau.

2.2. Manhattan

Khoảng cách Manhattan còn được gọi là khoảng cách L1, là 1 dạng khoảng cách giữa 2 điểm trong không gian Euclid và hệ tọa độ Descartes và được tính bằng tổng chiều dài của hình chiếu của đường thẳng nối 2 điểm này trong hệ trục tọa độ Descartes

Tọa độ 2 điểm A (X_A, Y_A) và B (X_B, Y_B) khi đó khoảng cách Manhattan giữa 2 điểm như sau:

$$d\ (A, B) = |X_A - X_B + Y_A - Y_B|$$

3. Chọn K làm trung tâm cụm ban đầu (initial cluster centers)

3.1. Chọn ngẫu nhiên

Mô tả: Chọn ngẫu nhiên kkk điểm dữ liệu từ tập dữ liệu để làm trung tâm cụm đầu tiên.

Ưu điểm: Đơn giản và dễ triển khai.

Nhược điểm: Có thể dẫn đến các kết quả kém nếu các điểm được chọn không đại diện tốt cho các cụm trong dữ liệu. Kết quả có thể không ổn định và thuật toán có thể hội tụ đến các cực trị địa phương.

3.2. Phương Pháp Trung Bình (Forgy Method)

Mô tả: Chọn ngẫu nhiên các điểm dữ liệu làm trung tâm cụm, nhưng sau đó tính lại các trung tâm cụm bằng cách tính trung bình cộng của tất cả các điểm trong mỗi cụm.

Ưu điểm: Đơn giản, nhưng có thể không hiệu quả trong việc chọn các trung tâm cụm tốt.

Nhược điểm: Có thể dẫn đến các trung tâm cụm kém đại diện cho các cụm thực tế trong dữ liệu.

4. TÍNH TOÁN ĐỘ CHÍNH XÁC

Đo lường độ chính xác của thuật toán K-Means có thể không trực tiếp như trong các bài toán phân loại có nhãn. Tuy nhiên, có thể sử dụng một số chỉ số và phương pháp để đánh giá chất lượng của kết quả phân cụm của K-Means. Dưới đây là một số cách phổ biến để tính toán và đánh giá độ chính xác hoặc chất lượng của phân cụm trong K-Means:

- Tổng Bình Phương Khoảng Cách (Within-Cluster Sum of Squares - WCSS)
- Chỉ Số Silhouette
- Chỉ Số Davies-Bouldin
- Chỉ Số Calinski-Harabasz
- Visual Inspection (Nhìn bằng mắt)

Đánh giá chất lượng của phân cụm K-Means có thể được thực hiện bằng nhiều phương pháp khác nhau, tùy thuộc vào loại dữ liệu và mục tiêu phân tích. WCSS và các chỉ số như Silhouette, Davies-Bouldin, và Calinski-Harabasz cung cấp các phương pháp định lượng để đánh giá phân cụm, trong khi trực quan hóa có thể giúp hiểu rõ hơn về phân cụm bằng mắt.

5. Ưu điểm của K-Means

Một đặc điểm quan trọng của giải thuật k-means là nó *cực tiểu hóa* tổng các độ sai biệt của các mẫu trong một cụm đến trung tâm cụm của mẫu.

Nếu C_i là cụm thứ i và μ_i là trung tâm cụm của nó, thì *hàm đánh giá* phải cực tiểu hóa bởi giải thuật là hàm

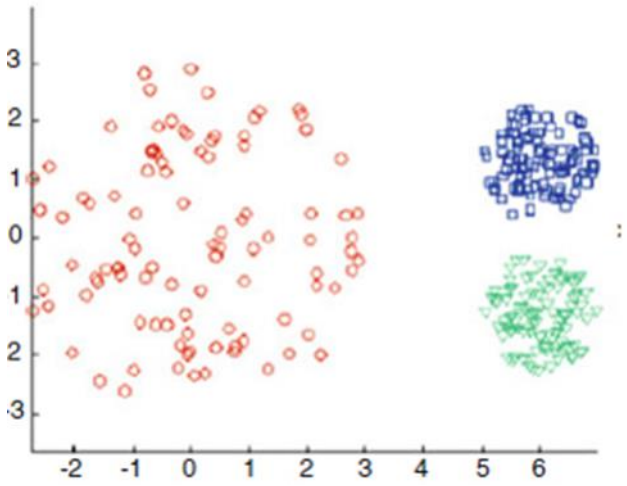
$$\sum_{i=1}^{k} \sum_{x \in C_i} || x - \mu_i ||$$

Độ phức tạp về thời gian tính toán của giải thuật k-means là $O(nkdl)$, với l là số lượt lặp của giải thuật, d là số chiều của dữ liệu, k là số cụm và n là số mẫu của tập mẫu.
Giải thuật k-Means là một trong số những giải thuật được ưa chuộng nhất. K-Means là 1 trong 10 giải thuật đứng đầu trong lãnh vực khai phá dữ liệu.

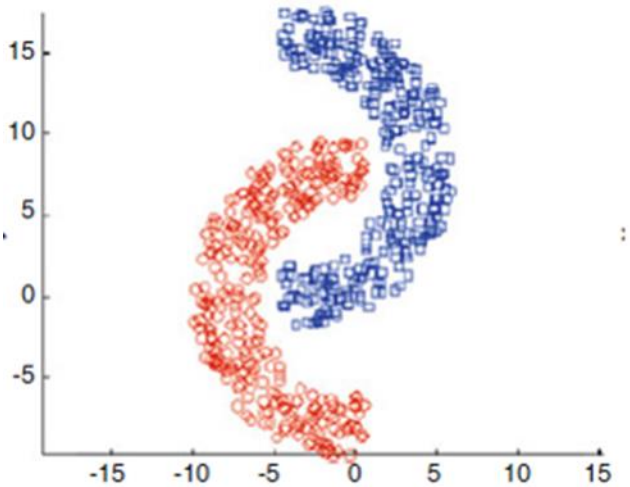
6. Khuyết điểm của KMeans

- Người dùng phải xác định thông số *k*, số cụm và phải xác định những trung tâm cụm ban đầu.
- K-Means không thể xử lý trường hợp các cụm có dạng *không phải là hình cầu* (non-spherical clusters).
- K-Means rất nhạy cảm khi dữ liệu có chứa các điểm ngoại biên.
- K-Means chỉ có thể áp dụng được với loại dữ liệu phù hợp với khái niệm centroid.

Hình dạng của cụm ban đầu



Hình 6. 1 Hình dạng của cụm

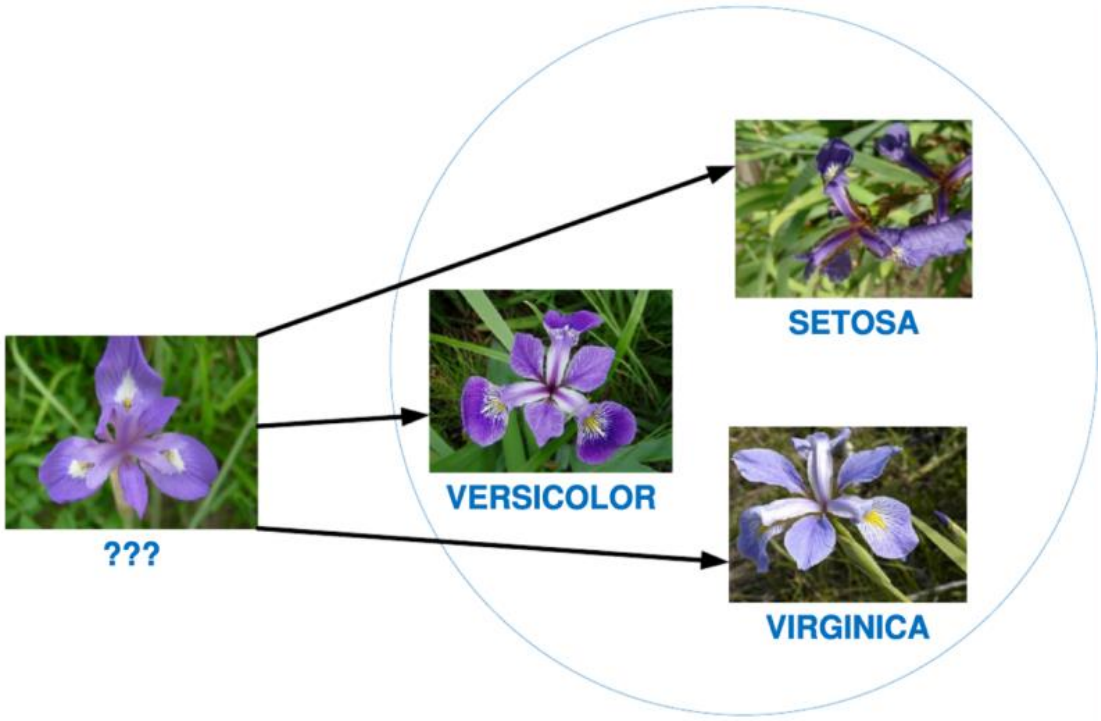


Hình 6. 2 trong hình 6.1 trên gồm 3 cụm có hình cầu nhưng ba cụm có mật độ khác nhau

7. ÁP DỤNG k-Means VÀO PHÂN LỚP HOA IRIS

Hình 7. 1 Ba Loài Hoa IRIS (Nguồn WIKI)

7.1. Định nghĩa bài toán



Hình 7. 2 Bài toán phân lớp hoa Iris

Với một bộ dữ liệu về hoa Iris đã có về các cá thể hoa thuộc các loài Iris cụ thể, ta không cần phải phân tích về gen hay phân tích về sinh học phức tạp để nhận biết ra một cá thể hoa đang xét thuộc loài Iris nào, bằng các dữ liệu đã có sẵn và dựa vào các thuộc tính của cá thể, ta có thể đưa ra nhận biết nhanh chóng cá thể đó thuộc thoài hoa Iris nào bằng cách tính toán số học trên máy tính.

7.2. Thu thập và tiền xử lý dữ liệu

Bộ dữ liệu về hoa Iris được thu thập bởi Edgar Anderson – nhà thực vật học, sau đó được thống kê và rút gọn lai bởi Ronald Aylmer Fisher với các thuộc tính như sau: - Bốn thuộc tính kiểu số: § Chiều dài đài hoa § Chiều rộng đài hoa § Chiều dài cánh hoa § Chiều rộng cánh hoa - Một thuộc tính còn lại là tên của loài Iris § Iris Setosa § Iris Versicolor § Iris Virginica Trong tập dữ liệu Iris, các giá trị nhỏ nhất, trung bình và lớ nhất ứng với các thuộc tính lần lượt như sau:

Bảng 7.2. 1: Gía trị trong tập dữ liệu Iris

	Gía trị nhỏ nhất	**Gía trị lớn nhất**	**Gía trị trung bình**
Chiều dài đài hoa	4.3	**7.9**	**5.84**
Chiều rộng đài hoa	**2.0**	**4.4**	**3.05**
Chiều dài cánh hoa	**1.0**	**6.9**	**3.76**
Chiều rộng cánh hoa	**0.1**	**2.5**	**1.20**

Tỷ lệ phân chia cho mỗi loài trong ba loài Iris là 33.33%

7.2.1. Làm sạch dữ liệu (data cleaning)

Thiếu giá trị: khi xảy ra sự thiếu thông tin ở một thuộc tính nào đó trong một bộ dòng của bộ dữ liệu thu thập, khi mà tính đảm bảo số bộ dòng chia đều cho ba loài hoa trên đã có (33.33%) và số lượng bộ dòng thiếu là ít ta có thể áp dụng phương pháp loại bỏ. Trường hợp cần điền giá trị thiếu, ta có thể áp dụng các giá trị trung bình bên trên

Dữ liệu nhiễu: khi xuất hiện một giá trị bất ngờ nào đó, đột nhiên vượt qua các giá trị biên đã được thống kê, ta có thể sửa lại giá trị đó thành các giá trị ở vùng biên.

7.2.2. Chọn lọc dữ liệu (data selection)

Tích hợp và dư thừa dữ liệu: do dữ liệu của Iris được thu thập từ một nguồn duy nhất nên việc tích hợp là không cần thiết trong trường hợp này, các thuộc tính của dữ liệu độc lập nhau, không có mối quan hệ tương quan nào, các thuộc tính đã được rút gọn chọn lọc nên không cần việc phân tích dư thừa dữ liệu đối với tập hoa Iris này

7.2.3. Chuyển đổi dữ liệu (data transformation)

Với tập hoa Iris, chuẩn hoá dữ liệu về khoảng giá trị [0, 1] được sử dụng để đảm bảo việc cân bằng dữ liệu, không có thuộc tính mang giá trị lớn sẽ ảnh hưởng lớn đến đến các thuộc tính còn lại.

7.2.4. Chuẩn hoá dữ liệu (data normalization)

Khi có một thuộc tính trong dữ liệu (hay phần tử trong vector) lớn hơn các thuộc tính khác rất nhiều (nguyên nhân thường dẫn đến việc này là đơn vị đo các thuộc tính không giống nhau, ví dụ thay vì đo bằng cm thì một kết quả lại tính bằng mm), khoảng cách giữa các điểm sẽ phụ thuộc vào thuộc tính này rất nhiều. Để có được kết quả chính xác hơn, một kỹ thuật thường được dùng là Data Normalization (chuẩn hóa dữ liệu) để đưa các thuộc tính có đơn vị đo khác nhau về cùng một khoảng giá trị, thường là từ 0 đến 1, trước khi thực hiện KNN. Có nhiều kỹ thuật chuẩn hóa khác nhau và các kỹ thuật chuẩn hóa được áp dụng với không chỉ KNN mà còn với hầu hết các thuật toán khác. Ta cùng theo dõi bảng giá trị bên dưới để thấy rõ điều này

Trích từ bộ dữ liệu hoa Iris, giá trị các thuộc tính trước và sau khi chuẩn hoá dữ liệu

Bảng 7.2.4. 1: Dữ liệu minh họa trước khi chuẩn hóa

Chiều dài đài hoa	Chiều rộng đài hoa	Chiều dài cánh hoa	Chiều rộng cánh hoa	Tên loài
5.1	3.5	1.4	0.2	Setosa
4.7	3.2	1.3	0.2	Setosa
7.0	3.2	4.7	1.4	Versicolor
6.3	3.3	6.0	2.5	Virginica
5.9	3.0	4.2	1.5	Versicolor
5.1	3.8	1.6	0.2	??? (Điểm cần phân loại)

Bảng 7.2.4. 2:Khoảng cách minh họa trước khi chuẩn hóa

Tên loài	Khoảng cách
Setosa	$\sqrt{((5.1-5.1)^2+(3.8-3.5)^2+(1.6-1.4)^2+(0.2-0.2)^2)}$ **= 0.361**
Setosa	$\sqrt{(5.1-4.7)^2+(3.8-3.2)^2+(1.6-1.3)^2+(0.2-0.2)^2}$ **= 0.781**
Versicolor	$\sqrt{(5.1-7.0)^2+(3.8-3.2)^2+(1.6-4.7)^2+(0.2-1.4)^2}$ **= 3.876**
Virginica	$\sqrt{(5.1-6.3)^2+(3.8-3.3)^2+(1.6-6.0)^2+(0.2-2.5)^2}$ **= 5.132**
Versicolor	$\sqrt{(5.1-5.9)^2+(3.8-3.3)^2+(1.6-4.2)^2+(0.2=1.5)^2}$ **= 3.120**

Với dữ liệu chưa được chuẩn hoá, dễ thấy khoảng cách sẽ phụ thuộc vào các điểm mang giá trị số lớn rất nhiều, điều này làm cho kết quả phân lớp có thể không chính xác

Bảng 7.2.4. 3:Dữ liệu minh hoạ sau khi chuẩn hoá

Chiều dài đài hoa	Chiều rộng đài hoa	Chiều dài cánh hoa	Chiều rộng cánh hoa	Tên loài
5.1/7.0 = 0.73	3.5/3.8 = 0.92	1.4/6.0 = 0.23	0.2/2.5 = 0.08	Setosa
4.7/7.0 = 0 .67	3.2/3.8 = 0.84	1.3/6.0 = 0.22	0.2/2.5 = 0.08	Setosa
7.0/7.0 = 1	3.2/3.8 = 0.84	4.7/6.0 = 0.78	1.4/2.5 = 0.56	Versicolor
6.3/7.0 = 0.9	6.3/7.0 = 0.9	6.0/6.0 = 1	2.5/2.5 = 1	Virginica
5.9/7.0 = 0.84	3.0/3.8 = 0.79	4.2/6.0 = 0.7	1.5/2.5 = 0.6	Versicolor
5.1/7.0 = 0.73	3.8/3.8 = 1	1.6/6.0 = 0.27	0.2/2.5 = 0.08	???

Bảng 7.2.4. 4:Khoảng cách minh hoạ sau khi chuẩn hoá

Tên loài	Khoảng cách
Setosa	$\sqrt{(0.73-0.73)^2+(0.92-1)^2+(0.23-0.27)^2+(0.08-0.08)^2}$ **= 0.088**
Setosa	$\sqrt{(0.67-0.73)^2+(0.92-1)^2+(0.23-0.27)^2+(0.08-0.08)^2}$ **=0.178**
Versicolor	$\sqrt{(1-0.73)^2+(0.84-1)^2+(0.78-0.27)^2+(0.56-0.08)^2}$ **= 0.767**
Virginica	$\sqrt{(0.9-0.73)^2+(0.87-1)^2+(1-0.27)^2+(1-0.08)^2}$ **= 1.194**
Versicolor	$\sqrt{(0.84-0.73)^2+(0.79-1)^2+(0.7-0.27)^2+(0.6-0.08)^2}$ **= 0.715**

Chọn k = 3 thì khoảng cách nhỏ nhất lần lượt là 0.008, 0.178 và 0.715 tương ứng với tên loài là Setosa, Setosa và Versicolor. Trong đó có 2 lân cận gần nhất là Setosa và 1 thuộc về Versicolor nên ta kết luận loài cần phân lớp thuộc Setosa.

Chọn k = 4 tương tự như trên lần lượt ta có 2 lân cận gần nhất là Setosa và 2 lân cận gần nhất thuộc về Versicolor nên ta không thể kết luận loài cần phân lớp sẽ thuộc về lớp nào

Với dữ liệu sau khi được chuẩn hoá, việc tính toán khoảng cách sẽ cho kết quả chính xác hơn khi không có sự phụ thuộc vào đơn vị hay giá trị thuộc tính

7.2.5. Cách thức đo độ chính xác của mô hình

Silhouette Score là một phương pháp dùng để đo lường chất lượng của cụm (cluster) trong giải thuật K-means (và các giải thuật clustering khác). Nó được tính dựa trên hai yếu tố:

- a: Khoảng cách trung bình từ một điểm đến tất cả các điểm khác trong cùng một cụm.

- b: Khoảng cách trung bình từ một điểm đến tất cả các điểm trong cụm gần nhất mà điểm đó không thuộc về.
- Silhouette Score cho một điểm dữ liệu được tính bằng công thức:

$$s = \frac{b - a}{\max(a, b)}$$

Silhouette Score tổng thể là trung bình cộng của tất cả các Silhouette Scores của từng điểm dữ liệu.

7.3. Triển khai phân lớp hoa Iris

Iris flower dataset là một bộ dữ liệu nhỏ. Bộ dữ liệu này bao gồm thông tin của ba loại hoa Iris (một loài hoa lan) khác nhau: Iris setosa, Iris virginica và Iris versicolor. Mỗi loại có 50 bông hoa được đo với dữ liệu là 4 thông tin: chiều dài (length), chiều rộng (width) đài hoa (sepal), và chiều dài, chiều rộng cánh hoa (petal).

```
data shape: (150, 5)
```

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Hình 7.3. 1 Giá trị mẫu 3 dòng dữ liệu Iris

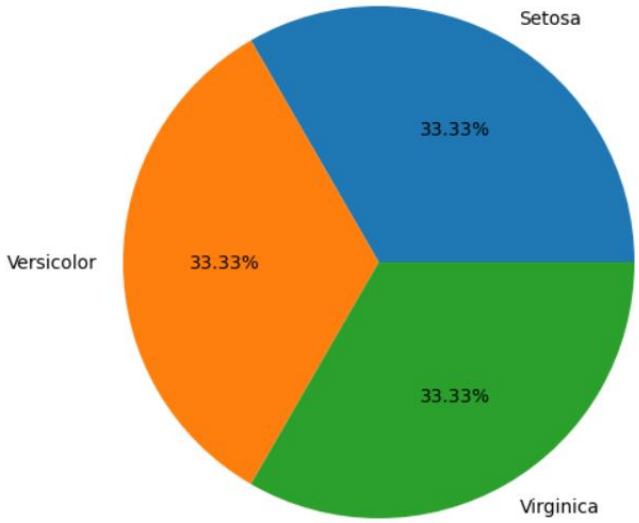
Dataset không có các dữ liệu missing hay null. Có 150 mẫu dữ liệu trong bộ dataset. Phân bố dữ liệu trong dataset là đều không có hiện tượng mất cân bằng dữ liệu.

```
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Id             150 non-null    int64
 1   SepalLengthCm  150 non-null    float64
 2   SepalWidthCm   150 non-null    float64
 3   PetalLengthCm  150 non-null    float64
 4   PetalWidthCm   150 non-null    float64
 5   Species        150 non-null    object
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

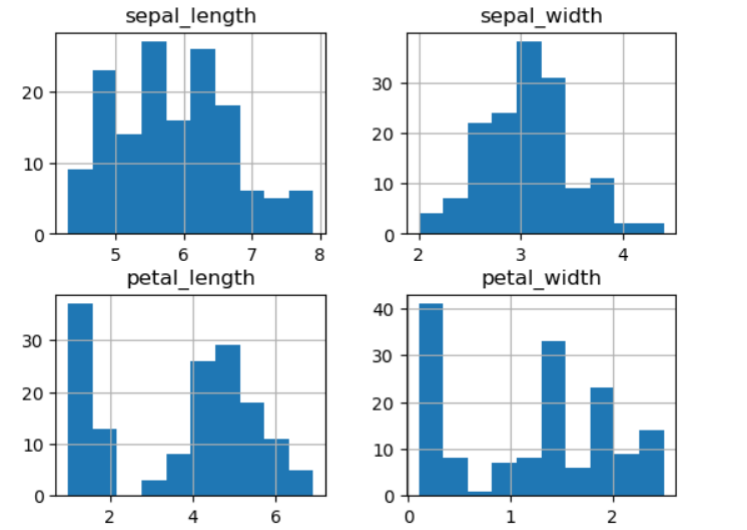
Giá trị của các đặc trưng khá chênh lệch. Ví dụ như 'SepalLength' có giá trị max = 150, trong khi đó PetalWidth có giá trị max = 2.5. Nên chuẩn hoá (normalize) dữ liệu về tầm giá trị [0, 1]. Nhóm sẽ so sánh kết quả độ sự ảnh hưởng độ chính xác khi không chuẩn hoá và sau chuẩn hoá dữ liệu đầu vào.

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

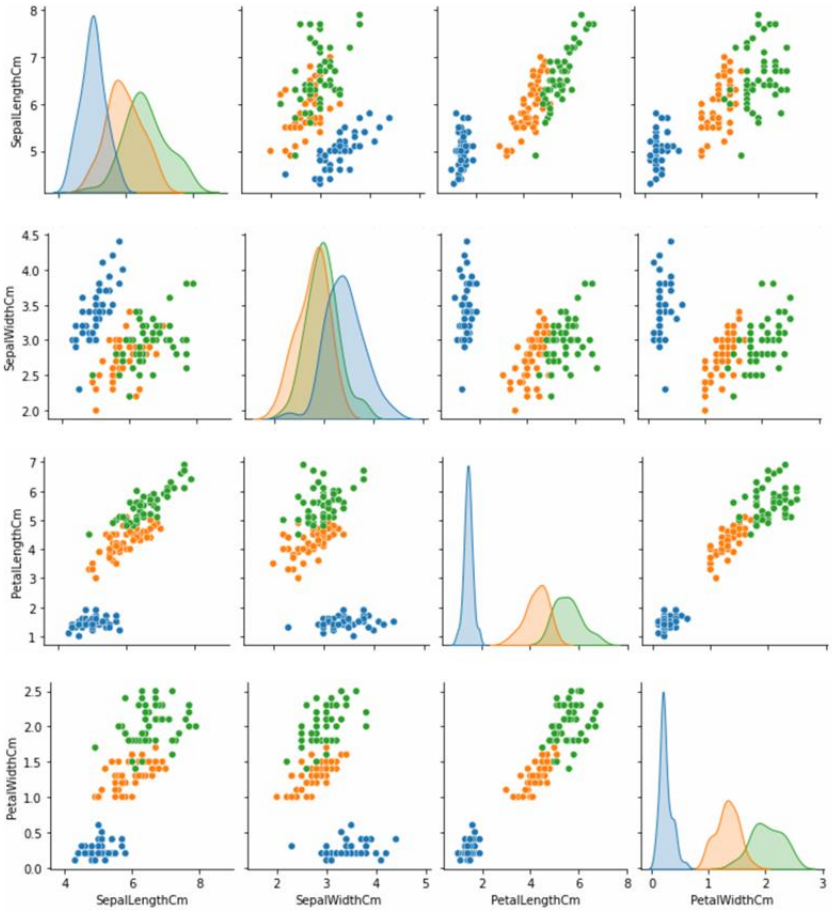
Hình 7.3. 2 Bảng giá trị đặc trưng chênh lệch



Hình 7.3. 3 Biểu đồ tròn

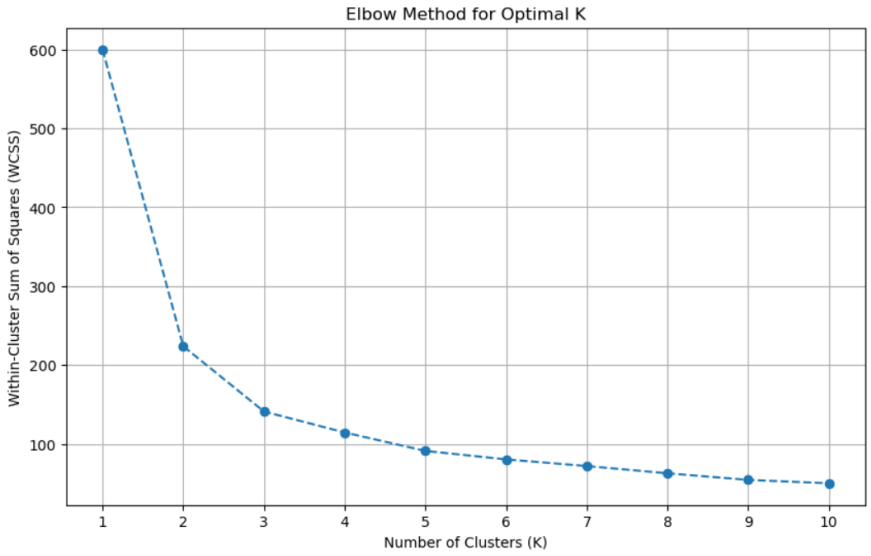


Hình 7.3. 4 Biểu đồ cột

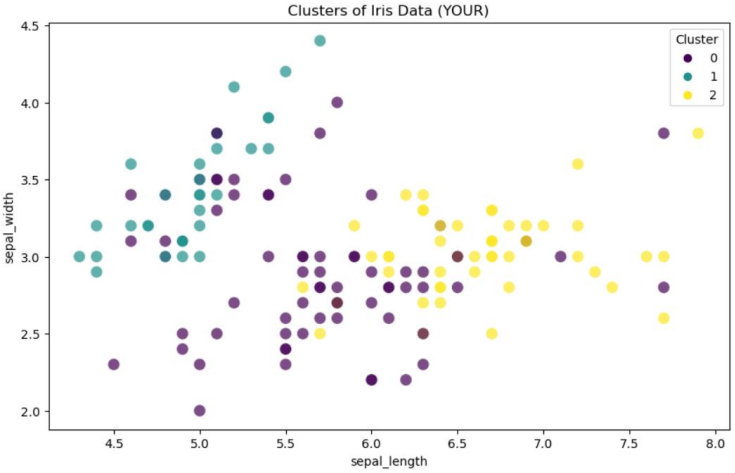


Hình 7.3. 5 Sự tương quan phân bố dữ liệu theo thuộc tính

7.4. Phương pháp chọn số k thích hợp



Hình 7.4. 1Chọn số K thích hợp



Hình 7.4. 2 Sau khi gom cụm

8. Kết luận

K-means là một phương pháp hiệu quả để gom cụm dữ liệu Iris:

Với dữ liệu không trọng số, K-means có thể tạo ra các cụm rõ ràng và chặt chẽ, phù hợp với phân loại thực tế của dữ liệu Iris.

Ứng dụng thực tế:

K-means là một công cụ mạnh mẽ và đơn giản cho việc phân cụm dữ liệu. Tuy nhiên, cần cẩn trọng khi áp dụng trọng số và nên kiểm tra kỹ lưỡng các kết quả để đảm bảo tính chính xác và ý nghĩa của clustering.

->Dựa trên Silhouette Scores và phân tích kết quả, ta có thể kết luận rằng K-means là một phương pháp hiệu quả cho việc phân cụm dữ liệu Iris. Tuy nhiên, việc sử dụng trọng số cần được thực hiện cẩn trọng và dựa trên hiểu biết rõ ràng về dữ liệu và mục tiêu của phân tích.

9. MÃ NGUỒN

Giải thuật được thực hiện trên môi trường Jupyter Notebook
Link tham khảo code: https://drive.google.com/drive/folders/14h6lXi7-fUEpZhhvPgKBRJIODQKpruFD?usp=sharing

9.1. Cài đặt các thư viện cần thiết

Một số thư viện cần thiết được dùng trong bài gồm:

- **import numpy as np**: Thư viện NumPy cung cấp hỗ trợ cho các mảng và toán học số học.
- **import matplotlib. pyplot as plt**: Thư viện matplotlib để tạo các biểu đồ và đồ thị.
- **import seaborn as sns**: Thư viện Seaborn mở rộng matplotlib với các biểu đồ đẹp hơn và dễ sử dụng hơn.
- **import pandas as pd**: Thư viện Pandas để xử lý dữ liệu dạng bảng (DataFrame).
- **from sklearn. cluster import KMeans**: Cung cấp lớp KMeans từ scikit-learn để thực hiện phân cụm K-Means.
- **from sklearn. preprocessing import StandardScaler**: Cung cấp lớp StandardScaler để chuẩn hóa dữ liệu.
- **%matplotlib inline**: Magic command của IPython/Jupyter để hiển thị biểu đồ trong notebook.
- **from warnings import filterwarnings**: Cung cấp hàm filterwarnings để quản lý các cảnh báo.
- **filterwarnings(action='ignore')**: Tắt các cảnh báo để làm sạch đầu ra.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
%matplotlib inline
from warnings import filterwarnings
filterwarnings(action='ignore')
```

9.2. Tải dữ liệu và kiểm tra dữ liệu

- columns = **['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']**: Xác định tên cột cho DataFrame.
- df = **pd. read_csv ('iris.data', names=columns)**: Đọc dữ liệu từ tệp CSV vào DataFrame với các cột đã đặt tên.
- print **('Data shape:', df. shape)**: In kích thước của DataFrame (số hàng và số cột).
- print **(df. head ())**: Hiển thị 5 hàng đầu tiên của DataFrame.
- print **(df. describe ())**: Cung cấp các thống kê mô tả cơ bản (như trung bình, độ lệch chuẩn) cho các cột số.

```
# Load the dataset
columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'class']
df = pd.read_csv('iris.data', names=columns)
print('Data shape:', df.shape)

# Display the first few rows of the dataset
print(df.head())

# Display descriptive statistics
print(df.describe())
```

9.3. Tạo biểu đồ pie

- **fig = plt.figure()**: Tạo một đối tượng figure mới.
- **ax = fig.add_axes([0, 0, 1, 1])**: Thêm trục vào figure.
- **ax.axis('equal')**: Đảm bảo rằng biểu đồ pie có hình tròn.
- **labels = ['Setosa', 'Versicolor', 'Virginica']**: Các nhãn cho từng phân đoạn trong biểu đồ pie.
- **sizes = [50, 50, 50]**: Kích thước tương ứng cho mỗi phân đoạn (giả định rằng có 50 mẫu cho mỗi lớp).
- **ax.pie(sizes, labels=labels, autopct='%1.2f%%')**: Tạo biểu đồ pie với tỷ lệ phần trăm hiển thị.
- **plt.show()**: Hiển thị biểu đồ.

```
# Create a pie chart
fig = plt.figure()
ax = fig.add_axes([0, 0, 1, 1])
ax.axis('equal')
labels = ['Setosa', 'Versicolor', 'Virginica']
sizes = [50, 50, 50]
ax.pie(sizes, labels=labels, autopct='%1.2f%%')
plt.show()
```

9.4. Tạo biểu đồ histogram

- **df.hist()**: Tạo biểu đồ histogram cho tất cả các cột số trong DataFrame.
- **plt.show()**: Hiển thị biểu đồ histogram.

```
# Create histograms for each feature
df.hist()
plt.show()
```

9.5. Tạo biểu đồ pairplot

- **sns.pairplot(df, hue='class')**: Tạo biểu đồ pairplot để hiển thị mối quan hệ giữa các đặc trưng và phân loại dữ liệu theo lớp.
- **plt.show()**: Hiển thị biểu đồ pairplot.

```
# Create a pairplot
sns.pairplot(df, hue='class')
plt.show()
```

9.6. Chuẩn bị dữ liệu cho phân cụm

- **X = df.drop('class', axis=1)**: Loại bỏ cột 'class' để chỉ giữ lại các đặc trưng cho phân cụm.

- **scaler = StandardScaler()**: Tạo đối tượng StandardScaler để chuẩn hóa dữ liệu.
- **X_scaled = scaler.fit_transform(X)**: Chuẩn hóa dữ liệu để có trung bình bằng 0 và độ lệch chuẩn bằng 1.

```
# Dropping the class column for clustering purposes
X = df.drop('class', axis=1)

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

9.7. Tính toán WCSS cho các giá trị khác nhau của K

- **wcss = []**: Tạo danh sách để lưu trữ giá trị WCSS cho từng số lượng cụm.
- **for i in range(1, 11):**: Vòng lặp từ 1 đến 10 để kiểm tra các số lượng cụm khác nhau.
- **kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)**: Tạo đối tượng KMeans với số lượng cụm i.
- **kmeans.fit(X_scaled)**: Thực hiện phân cụm trên dữ liệu đã chuẩn hóa.
- **wcss.append(kmeans.inertia_)**: Thêm giá trị WCSS vào danh sách.
- **plt.figure(figsize=(10, 6))**: Tạo một figure mới với kích thước xác định.
- **plt.plot(range(1, 11), wcss, marker='o', linestyle='--')**: Vẽ đồ thị Elbow Method với số lượng cụm và giá trị WCSS.
- **plt.title('Elbow Method for Optimal K')**: Đặt tiêu đề cho biểu đồ.
- **plt.xlabel('Number of Clusters (K)')**: Đặt nhãn cho trục x.
- **plt.ylabel('Within-Cluster Sum of Squares (WCSS)')**: Đặt nhãn cho trục y.
- **plt.xticks(range(1, 11))**: Đặt các tick cho trục x.
- **plt.grid(True)**: Bật lưới cho biểu đồ.
- **plt.show()**: Hiển thị biểu đồ.

```
# Calculate WCSS for different values of K
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Plot the Elbow Method
plt.figure(figsize=(10, 6))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Within-Cluster Sum of Squares (WCSS)')
plt.xticks(range(1, 11))
plt.grid(True)
plt.show()
```

9.8. Thực hiện phân cụm K-Means với số lượng cụm tối ưu

- **optimal_k = 3**: Chọn số lượng cụm tối ưu dựa trên biểu đồ Elbow Method.

- **kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300, n_init=10, random_state=42)**: Tạo đối tượng KMeans với số lượng cụm tối ưu.
- **df['Cluster'] = kmeans.fit_predict(X_scaled)**: Thực hiện phân cụm và gán nhãn cụm cho DataFrame.

```
# Optionally, perform K-means with the optimal K
optimal_k = 3  # This value should be chosen based on the Elbow plot
kmeans = KMeans(n_clusters=optimal_k, init='k-means++', max_iter=300, n_init=10, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
```

9.9. Vẽ các cụm

- **plt.figure(figsize=(10, 6))**: Tạo một figure mới với kích thước xác định.
- **sns.scatterplot(data=df, x='sepal_length', y='sepal_width', hue='Cluster', palette='viridis', s=100, alpha=0.7)**: Tạo biểu đồ phân tán với các điểm dữ liệu được phân loại theo cụm.
- **plt.title('Clusters of Iris Data')**: Đặt tiêu đề cho biểu đồ.
- **plt.show()**: Hiển thị biểu đồ.

```
# Plot the clusters
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='sepal_length', y='sepal_width', hue='Cluster', palette='viridis', s=100, alpha=0.7)
plt.title('Clusters of Iris Data')
plt.show()
```

10. Tài liệu tham khảo

[1] PGS.TS Dương Tuấn Anh, Chương 7 –Gom cụm, Trí Tuệ Nhân Tạo, Khoa Công Nghệ Thông Tin, Đại Học Ngoại Ngữ Tin Học TPHCM.
[2] Data về IRIS http://archive.ics.uci.edu/dataset/53/iris
[3] https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/
[4] https://realpython.com/k-means-clustering-python/
[5] MẪU BÁO CÁO TRIỂN KHAI GIẢI THUẬT K-NN VÀ ÁP DỤNG VÀO PHÂN LỚP TẬP DỮ LIỆU HOA IRIS Môn học: Học Máy và Ứng Dụng TRƯỜNG ĐẠI HỌC BÁCH KHOA TPHCM KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH