Bùi Quốc Bảo (1412034)

Hà Gia Phát (1412382)

Vũ Ngọc Minh Hoàng (1412186)

REPORT LAB04

1.2: BÁO CÁO KẾT QUẢ THỰC NGHIỆM THAM SỐ K VÀ RANDOM SEED

	Random seed					
k	10	20	30	40	50	best
2	327.57	327.57	485.54	313.13	313.13	313.13
3	305.03	206.50	261.78	296.73	206.50	206.50
4	192.90	171.96	245.70	188.89	188.89	171.96
5	169.31	161.05	137.75	155.36	171.82	137.75
6	124.83	114.75	106.87	101.42	144.93	101.42

- Qua bảng thống kê trên ta có thể thấy được rằng số lượng cụm tốt nhất K_1 bằng 6

1.3: BÁO CÁO KẾT QUẢ THỰC NGHIỆM ĐỘ ĐO KHOẢNG CÁCH

- **Sử dụng khoảng cách Manhattan:**

K	Best
2	1453.29
3	857.50
4	650.90
5	523.81
6	411.74

Kết luận: Vậy số lượng cụm tốt nhất K_2 là bằng 6

- **Sử dụng khoảng cách Cosine:**

K	Best
2	5.64
3	3.24
4	2.52
5	1.99
6	1.83

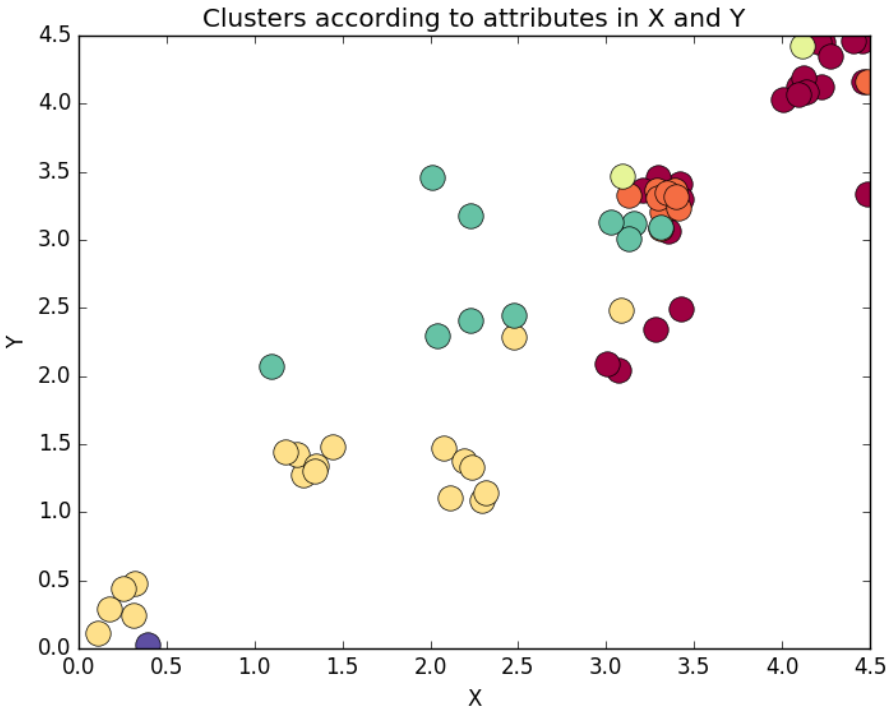
Kết luận: Vậy số lượng cụm tốt nhất K_3 là bằng 6

- **Nhận xét:**

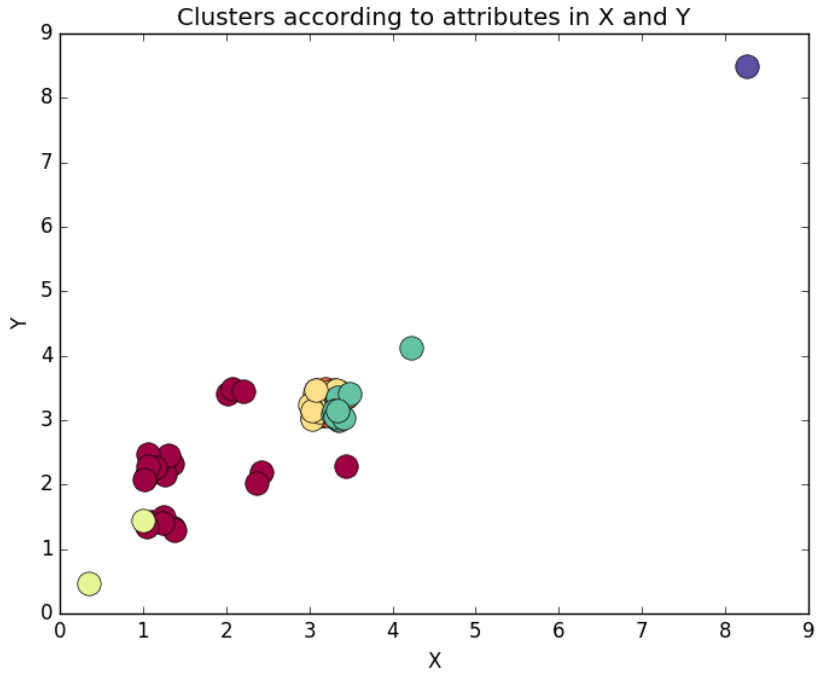
  Ta thấy rằng các giá trị K_1, K_2, K_3 khác nhau. Kết quả của $\boldsymbol{K_1}$ có vẻ đáng tin cậy nhất vì thuật toán K-Means chạy bằng cách lặp lại việc tìm các điểm gần với centroid nhất và cập nhật điểm centroid đó qua từng nhóm gom cụm mới. Thuật toán này ngầm sử dụng các công thức tính khoảng cách giữa các cặp điểm (dữ liệu và centroid) vì ta có thể thấy SSE (sum of errors) của K-Means được tính bằng khoảng cách bình phương giữa hai điểm centroid và dữ liệu. Vì thế công thức tính khoảng cách Euclidean trong gom cụm K_1rất phù hợp cho thuật toán K-Means. Ngoài ra nguồn gốc của từ "centroid" là trong hình học Euclidean, mà không gian Euclidean thì ta nên sử dụng khoảng cách Euclidean, các khoảng cách không thuộc về Euclidean (Non-Euclidean distances) thường sẽ không phù hợp khi sử dụng trong không gian Euclidean.

1.4: PHÂN TÍCH HIỆU QUẢ PHÂN TÁCH CỤM CỦA CÁC CẶP THUỘC TÍNH KHÁC NHAU

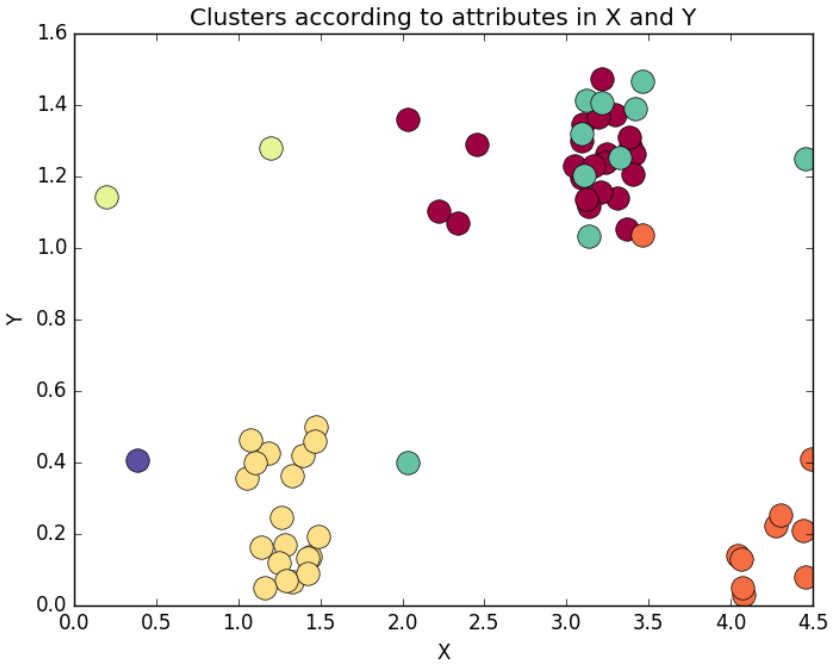
1. Sau khi chọn số lượng cụm tốt nhất $\boldsymbol{K_1}$làm dữ liệu đầu vào, ta sẽ được các hình ảnh gom cụm theo thuộc tính X và Y như sau:



Hình 1.4.1: X = răng tiền hàm trên, Y = răng tiền hàm dưới



Hình 1.4.2: X = răng hàm trên, Y = răng hàm dưới



Hình 1.4.3: X = răng cửa dưới, Y = răng nanh dưới

Hình 1.4.4: X = răng cửa trên, Y = răng nanh trên

Lưu ý: Các hình trên đã được thêm nhiễu vào để cho các điểm không bị trùng lắp nhau.

- Ta sẽ sử dụng chương trình internal_evaluation.py để đánh giá hiệu quả của việc gom cụm. Chương trình sử dụng Davies-Bouldin index để đánh giá, cách chạy chương trình như sau:

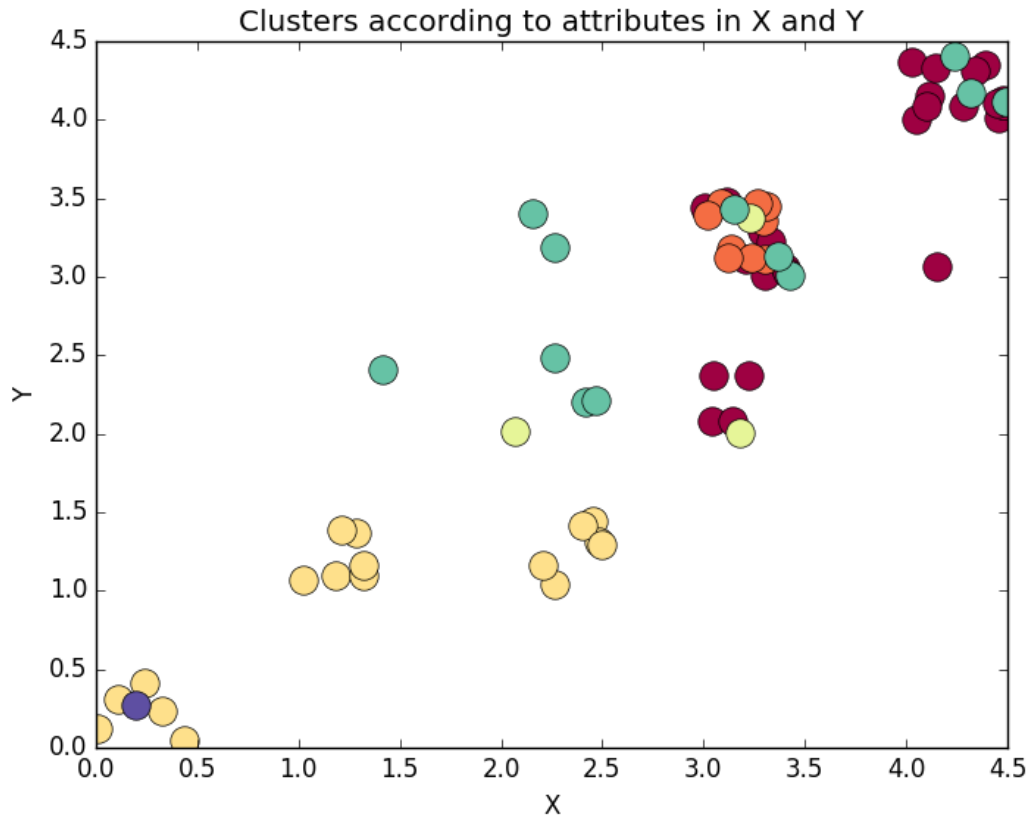
  python internal_evaluation.py <input> <X> <Y>

Trong đó:

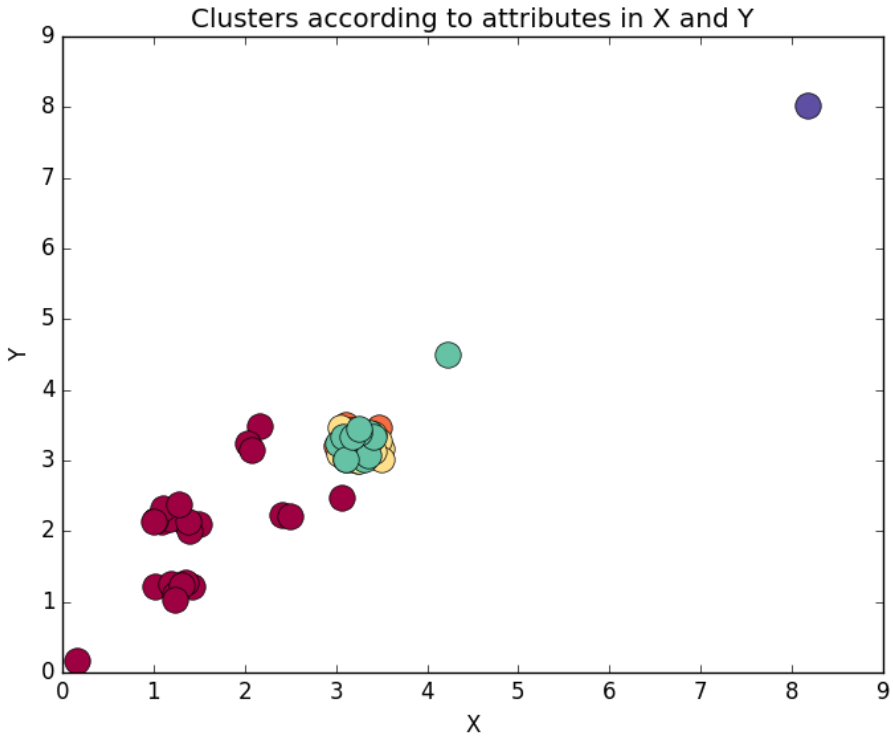
- **input:** Tập tin đầu vào có định dạng .csv, cấu trúc tương tự như tập tin đầu ra của chương trình ở phần 1.1
- **x, y:** Chỉ mục của thuộc tính trục x và trục y (tính từ 0 đến n – 1, với n là số lượng thuộc tính của dữ liệu)

- Sau khi chạy chương trình internal_evaluation.py ta **thấy cặp thuộc tính X = răng cửa dưới, Y = răng nanh dưới** là phân tách cụm tốt nhất cho K_1vì hai thuộc tính ấy có kết quả Davies-Bouldin index bé nhất (0.753005).

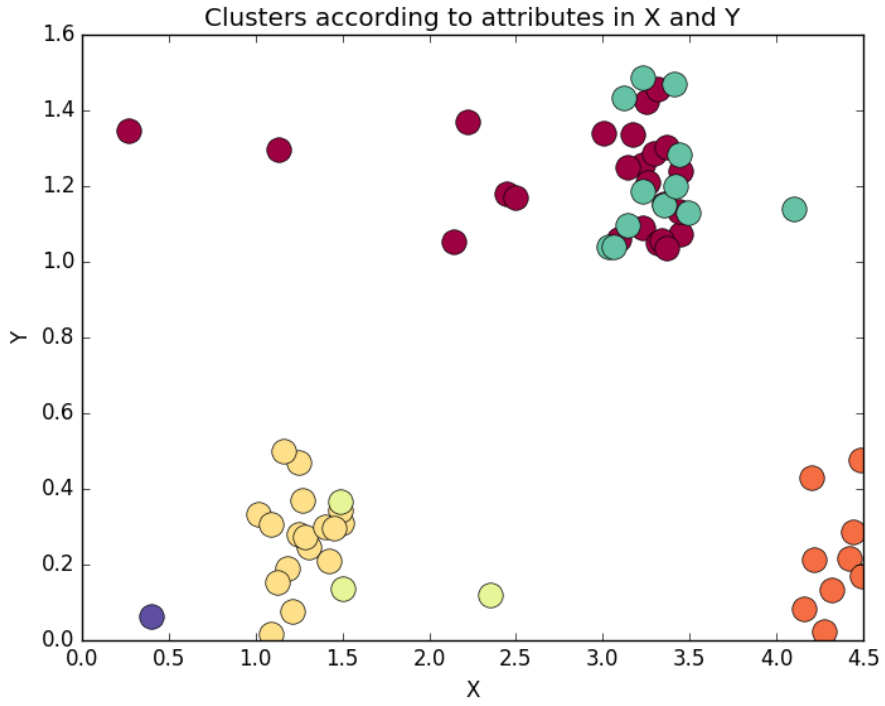
2. Tương tự với kết quả gom cụm khi số lượng cụm tốt nhất là K_2:



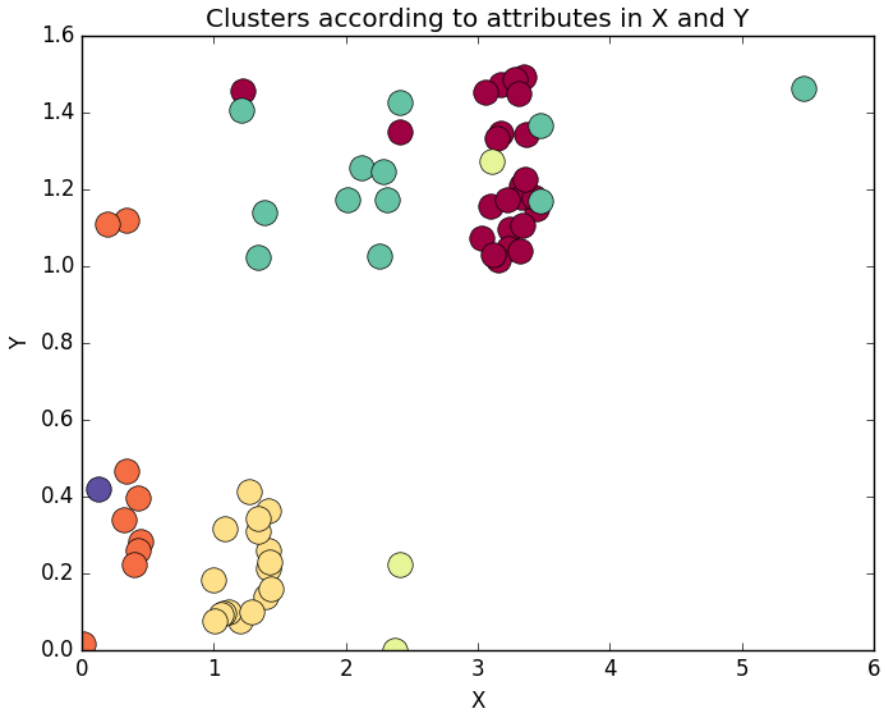
Hình 1.4.5: X = răng tiền hàm trên, Y = răng tiền hàm dưới



Hình 1.4.6: X = răng hàm trên, Y = răng hàm dưới



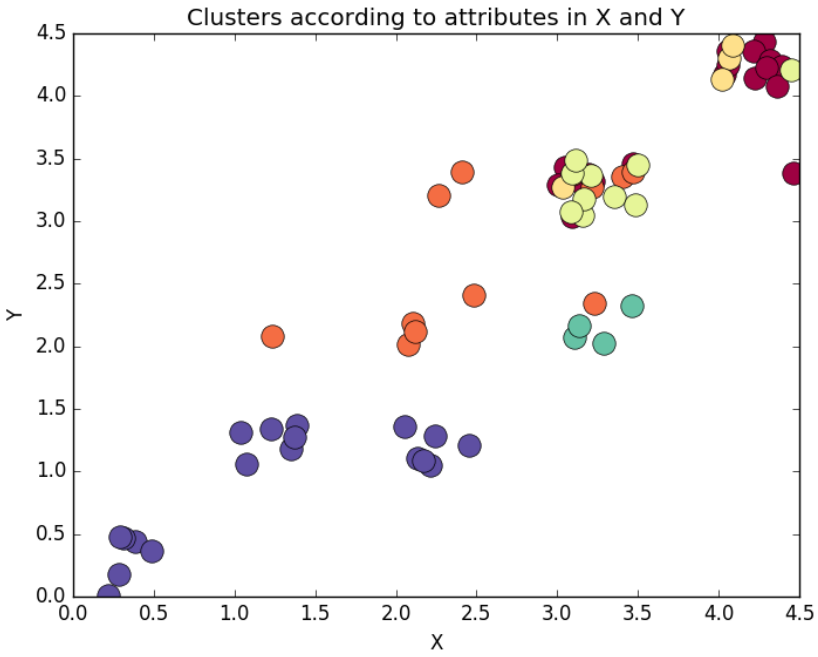
Hình 1.4.7: X = răng cửa dưới, Y = răng nanh dưới



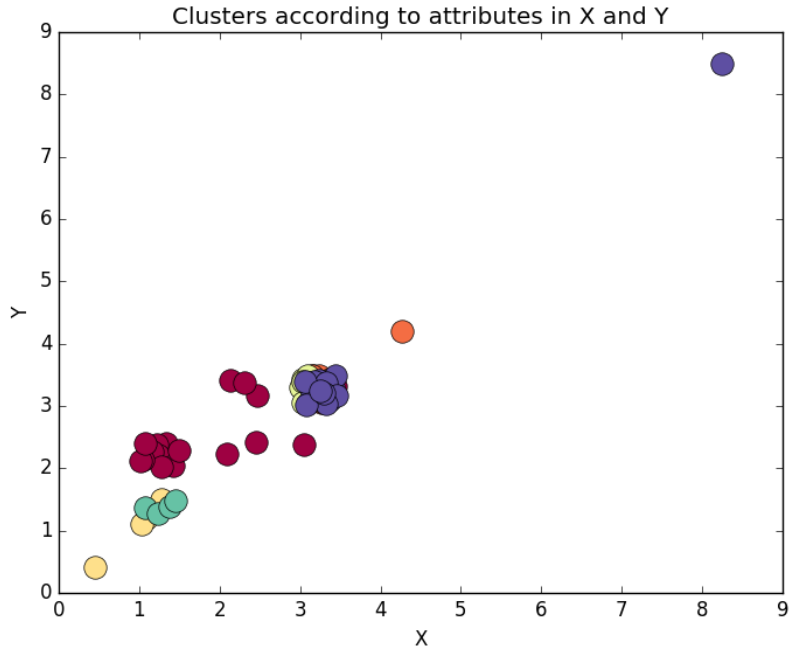
Hình 1.4.8: X = răng cửa trên, Y = răng nanh trên

- Cũng áp dụng chương trình internal_evaluation.py ta thấy cặp **thuộc tính X = răng cửa dưới, Y = răng nanh dưới** cho ta gom cụm tốt nhất với Davies-Bouldin index bé nhất (0.573512).

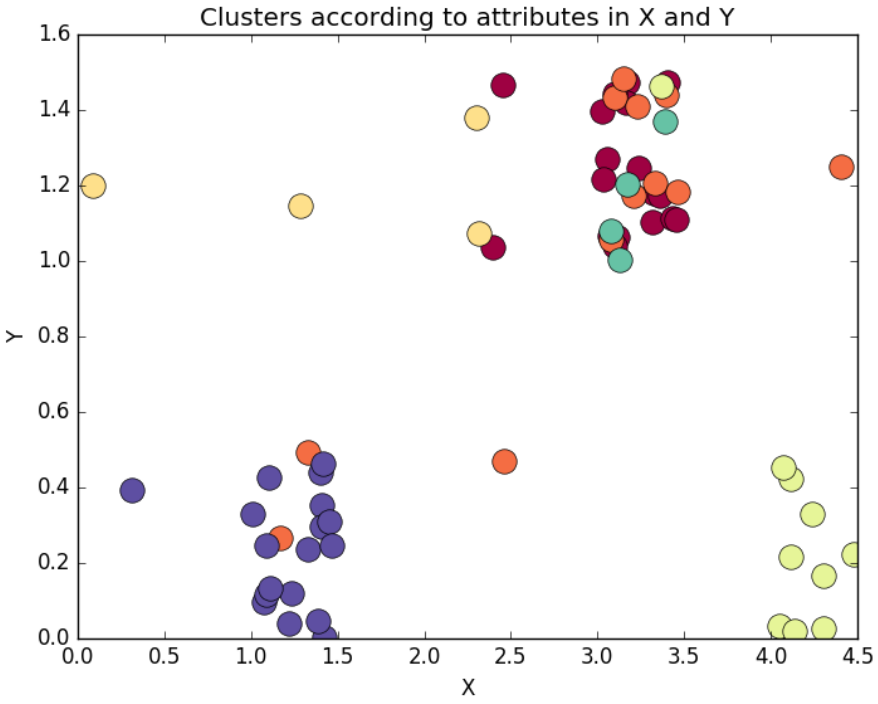
3. Tương tự cho K_3:



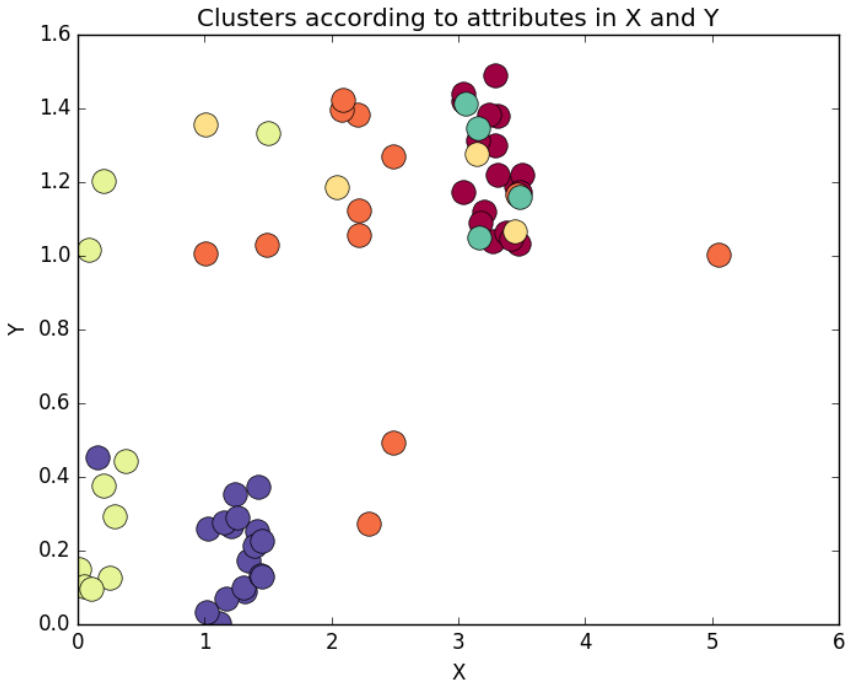
Hình 1.4.9: X = răng tiền hàm trên, Y = răng tiền hàm dưới



Hình 1.4.10: X = răng hàm trên, Y = răng hàm dưới



Hình 1.4.11: X = răng cửa dưới, Y = răng nanh dưới



Hình 1.4.12: X = răng cửa trên, Y = răng nanh trên

- Cặp **thuộc tính X = răng cửa dưới, Y = răng nanh dưới** sẽ phân tách cụm tốt nhất với Davies-Bouldin index = 0.975482

Kết luận: Cả ba số lượng gom cụm tốt nhất K_1, K_2, K_3 đều chọn cặp **thuộc tính X = răng cửa dưới, Y = răng nanh dưới** làm phân tách cụm tốt nhất. Trong đó căp thuộc tính X, Y trên phân tách tốt nhất trong K_1với Davies-Bouldin index = 0.573512 và phân tách tệ nhất trong K_3 với Davies-Bouldin index = 0.974483

1.5: NHẬN XÉT Ý NGHĨA CỦA CỤM DỮ LIỆU

Đối với dữ liệu trên ta thấy rằng với độ đo K_1 và phân chia thành 6 nhóm riêng biệt cho kết quả phân tách cụm tốt nhất. Nhìn vào danh sách các động vật ta có thể nhận thấy có 6 bộ động vật lớn đó là bộ gặm nhấm, bộ dơi, bộ chuột chù, bộ linh trưởng, bộ guốc chẵn, bộ ăn thịt. Điều này rất phù hợp với thực tiễn.

https://vi.wikipedia.org/wiki/L%E1%BB%9Bp_Th%C3%BA

Tuy nhiên, một số động vật không thể chia chính xác vào các bộ mà chỉ phụ thuộc vào đặc tính răng của chúng. Một số động vật thường bị phân cụm sai bao gồm: Sea-lion, Fur-seal (có thể chạy code reg_animal.py để xem kết quả, ý tưởng là chạy phân cụm nhiều lần và đếm số liên kết giữa các động vật, các động vật có liên kết yếu với một số động vật thì có nghĩa nó dễ bị phâm cụm sai)

Lý do phân cụm sai là do phương pháp kmean khá nhạy với các centroid được khởi tạo ban đầu, đồng thời các bộ động vật cần nhiều đặc tính hơn để phân chúng vào đúng các cụm.

HƯỚNG DẪN CÁCH CHẠY CHƯƠNG TRÌNH

1. CHƯƠNG TRÌNH ĐÁNH GIÁ CHẤT LƯỢNG GOM CỤM:

- Ta sẽ sử dụng chương trình **internal_evaluation.py** để đánh giá hiệu quả của việc gom cụm. Chương trình sử dụng **Davies-Bouldin index** để đánh giá, cách chạy chương trình như sau:

  python internal_evaluation.py <input> <X> <Y>

- **Trong đó:**
  - **input:** Tập tin đầu vào có định dạng .csv, cấu trúc tương tự như tập tin đầu ra của chương trình ở phần 1.1

- **x, y:** Chỉ mục của thuộc tính trục x và trục y (tính từ 0 đến n – 1, với n là số lượng thuộc tính của dữ liệu)

2. CHƯƠNG TRÌNH TÌM ĐỘNG VẬT BỊ PHÂN LOẠI SAI:

- Chỉ cần chạy dòng lệnh sau:
  python reg_animal.py