Информатика. Задания на кластеризацию. Метод dbscan (дополнительный вебинар).

Фрагмент звёздного неба спроецирован на плоскость с декартовой системой координат. Учёный решил провести кластеризацию полученных точек, являющихся изображениями звёзд, то есть разбить их множество на N непересекающихся непустых подмножеств (кластеров), таких что точки каждого подмножества лежат внутри прямоугольника со сторонами длиной H и W, причём эти прямоугольники между собой не пересекаются. Стороны прямоугольников не обязательно параллельны координатным осям. Гарантируется, что такое разбиение существует и единственно для заданных размеров прямоугольников.

Будем называть центром кластера точку этого кластера, сумма расстояний от которой до всех остальных точек кластера минимальна. Для каждого кластера гарантируется единственность его центра. Расстояние между двумя точками на плоскости $A(x_1, y_1)$ и $B(x_2, y_2)$ вычисляется по формуле:

$$d(A, B) = \sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$$

В файле *А* хранятся координаты точек двух кластеров, где $H = 3$, $W = 3$ для каждого кластера. В каждой строке записана информация о расположении на карте одной точки: сначала координата x, затем координата y. Известно, что количество точек не превышает 1000.

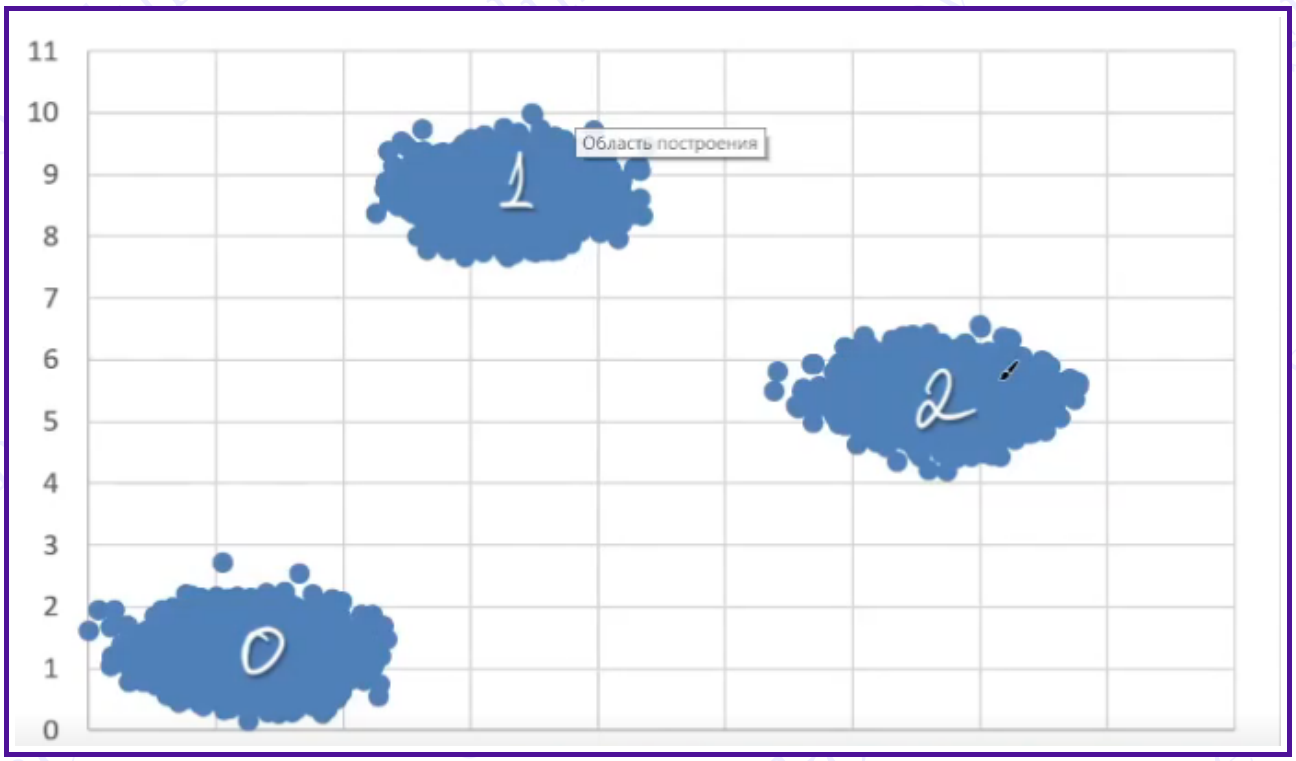
В файле *Б* хранятся координаты точек трёх кластеров, где $H = 3$, $W = 3$ для каждого кластера. Известно, что количество точек не превышает 10 000.

Структура хранения информации в файле *Б* аналогична файлу *А*. Для каждого файла определите координаты центра каждого кластера, затем вычислите два числа: P_x – среднее арифметическое абсцисс центров кластеров, и P_y – среднее арифметическое ординат центров кластеров.

В ответе запишите четыре числа: в первой строке сначала целую часть произведения $P_x \times 10000$, затем целую часть произведения $P_x \times 10000$ для файла *А*, во второй строке – аналогичные данные для файла *Б*.

1 способ решения (базовый)

Для начала обязательно открываем файл в *Excel* и строим заданные нам кластеры:



После анализа диаграммы становится ясно, что нулевой клестер имеет ординаты, меньшие трех.

Второй кластер – ординаты, большие семи.

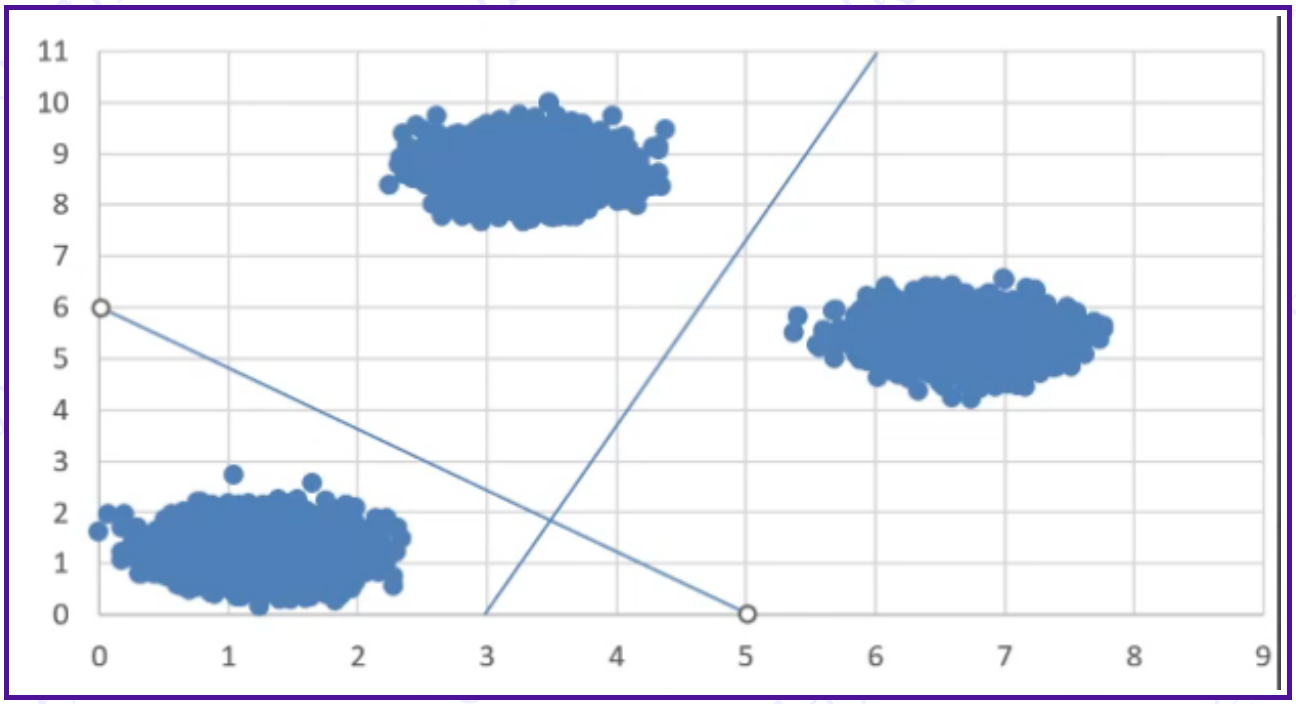
А третий кластер – все остальные.

```python
from math import dist

f = open("27B.txt")
a = [list(map(float, i.repalce(',', '.').split())) for i in f]
cl = [[], [], []]
for i in a: # Разделяем звезды по трем кластерам
    x, y = i
    if y < 3:
        cl[0].append(i)
    elif y > 7:
        cl[1].append(i)
    else:
        cl[2].append(i)
sx = sy = 0 # Переменные для сумм координат центров кластеров
for k in range(3):
    min_len = 1000000500000 # Начальное значение минимальной длины
    for j in cl[k]:  # Проходим по каждой точке в текущем кластере
        star = j  # Текущая звезда
        s_len = 0  # Сумма расстояний для текущей звезды
        for i in cl[k]:  # Считаем расстояние от текущей звезды до всех точек
            s_len += dist(star, i)  # Суммируем расстояния
# Если сумма расстояний меньше минимальной, обновляем минимальную длину
# и запоминаем текущую звезду
        if s_len < min_len:
            min_len = s_len
            min_star = star
# Добавляем координаты центра текущего кластера к общим координатам
    sx += min_star[0]
    sy += min_star[1]
print(int(sx / 3 * 10000), int(sy / 3 * 10000))
```

2 способ решения (с помощью прямых)

Аналогично сначала обязательно открываем файл в *Excel* и строим заданные нам кластеры. Затем строим прямые по двум целым точкам, разграничивающие кластеры:



Найдем уравнение прямой, отделяющей левый нижний кластер. Она проходит через две точки: (5, 0) и (0, 6). Составим уравнение прямой по двух точкам:

$$\frac{x-5}{0-5}=\frac{y}{6-0}$$

$$\frac{x-5}{-5}=\frac{y}{6}$$

Получаем, что точки левого нижнего кластера находятся ниже этой прямой, то есть принадлежать неравенству:

$$\frac{x-5}{-5}>\frac{y}{6}$$

Точки кластера посередине имеют координату абсциссы, большую 7. Все остальные точки принадлежат третьему кластеру.

Реализуем программное решение:

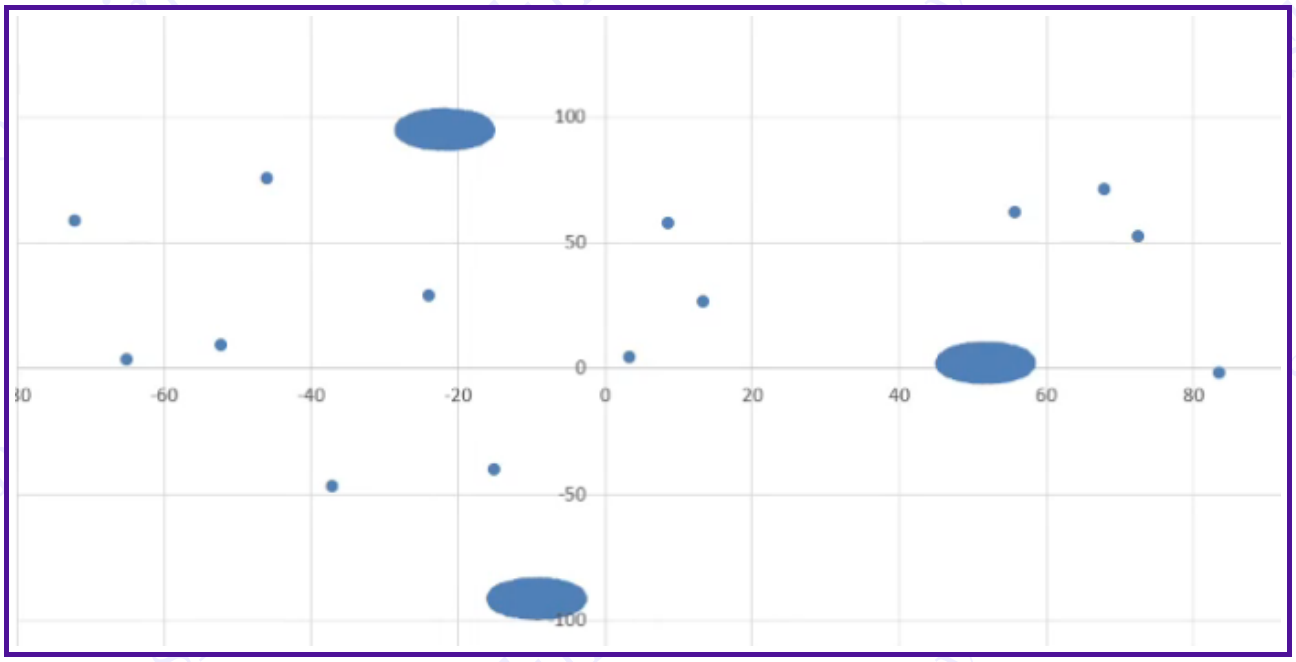
```
from math import dist

f = open("27B.txt")
a = [list(map(float, i.replace(',', '.').split())) for i in f]
cl = [[], [], []]
for i in a: # Разделяем звезды по кластерам
    x, y = i
    if y / 6 < (x - 5) / -5:
        cl[0].append(i)
    elif y > 7:
        cl[1].append(i)
    else:
        cl[2].append(i)
sx = sy = 0 # Переменные для сумм координат центров кластеров
for k in range(3):
    min_len = 1000000500000 # Начальное значение минимальной длины
    for j in cl[k]:  # Проходим по каждой точке в текущем кластере
        star = j  # Текущая звезда
        s_len = 0  # Сумма расстояний для текущей звезды
        for i in cl[k]:  # Считаем расстояние от текущей звезды до всех точек
            s_len += dist(star, i)  # Суммируем расстояния
# Если сумма расстояний меньше минимальной, обновляем минимальную длину
# и запоминаем текущую звезду
        if s_len < min_len:
            min_len = s_len
            min_star = star
# Добавляем координаты центра текущего кластера к общим координатам
    sx += min_star[0]
    sy += min_star[1]
print(int(sx / 3 * 10000), int(sy / 3 * 10000))
```

Ответ: 37522 51277

3 способ решения (метод DBSCAN)

Для реализации метода **DBSCAN** возьмем другой файл, содержащий мусорные точки.



Напомним себе принцип работы алгоритма:

Алгоритм ***DBSCAN*** – это метод кластеризации, который позволяет выделять кластеры различной формы и размера. Принцип работы алгоритма следующий:

- Перед началом работы алгоритма необходимо определить радиус R, в пределах которого будут рассматриваться соседние точки.
- Алгоритм начинается с выбора произвольной точки из набора данных, от которой будут искаться все остальные точки кластера.
- Для выбранной точки ищутся все точки, которые находятся на расстоянии R.
- Процесс продолжается до тех пор, пока все точки на заданном расстоянии не будут обработаны. Как только все возможные точки добавлены в кластер, алгоритм переходит к следующей непроверенной точке в наборе данных (то есть начинает формировать следующий кластер).
- Алгоритм повторяет шаги для всех непроверенных точек в наборе данных, пока не будут обработаны все точки.

Реализуем программное решение:

```
from math import dist
f = open("27.txt")
s = f.readline()
a = [list(map(float, i.replace(',', '.').split())) for i in f]
cl = []
while a: # Пока есть точки в списке a, продолжаем кластеризацию
    cl.append([a.pop(0)])  # Создаем кластер и добавляем в него первую точку
    for j in cl[-1]: # Проходим по всем точкам в текущем кластере
        for i in a: # Сравниваем каждую точку в кластере с другими точками
            if dist(j, i) < 0.5:  # Если расстояние между точками меньше 0.5,
                cl[-1].append(i) # добавляем точку i в текущий кластер
                a.remove(i)  # Удаляем точку i из списка a
sx = sy = 0 # Переменные для сумм координат центров кластеров
for k in range(len(cl)): # Проходим по каждому кластеру
    if len(cl[k]) > 1: # Проверяем, что в кластере больше одной точки
        min_len = 1000000500000 # Начальное значение минимальной длины
        for j in cl[k]:  # Проходим по каждой точке в текущем кластере
            star = j  # Текущая звезда
            s_len = 0  # Сумма расстояний для текущей звезды
            for i in cl[k]:  # Расстояние от текущей звезды до всех точек
                s_len += dist(star, i)  # Суммируем расстояния
# Если сумма расстояний меньше минимальной, обновляем минимальную длину
# и запоминаем текущую звезду
            if s_len < min_len:
                min_len = s_len
                min_star = star
# Добавляем координаты центра текущего кластера к общим координатам
        sx += min_star[0]
        sy += min_star[1]
print(int(sx / 3 * 10000), int(sy / 3 * 10000))
```

Ответ: 69272 18677