Информатика. Кластеризация. DBSCAN.

Содержание

1 Разбор задач с вебинара

1.1 Пример 1

Учёный решил провести кластеризацию некоторого множества звёзд по их расположению на карте звёздного неба. Кластер звёзд – это набор звёзд (точек) на графике, каждая из которых находится от хотя бы одной другой звезды на расстоянии не более R условных единиц. Каждая звезда обязательно принадлежит только одному из кластеров.

Истинный центр кластера, или центроид, – это одна из звёзд на графике, сумма расстояний от которой до всех остальных звёзд кластера минимальна.

Под расстоянием понимается расстояние Евклида между двумя точками $A(x_1, y_1, z_1)$ и $B(x_2, y_2, z_2)$ в трехмерном пространстве, которое вычисляется по формуле:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Аномалиями назовём точки, находящиеся на расстоянии более одной условной единицы от точек кластеров. При расчётах аномалии учитывать не нужно.

В файле А хранятся данные о звёздах **трех** кластеров, где $R = 0.8$ для каждого кластера. В каждой строке записана информация о расположении на карте одной звезды: сначала координата x, затем координата y, затем координата z. Значения даны в условных единицах, которые представлены вещественными числами. Известно, что количество звёзд не превышает 8000.

В файле Б хранятся данные о звёздах **четырех** кластеров, где $R = 0.9$ для каждого кластера. Известно, что количество звёзд не превышает 20000. Структура хранения информации о звездах в файле Б аналогична файлу А.

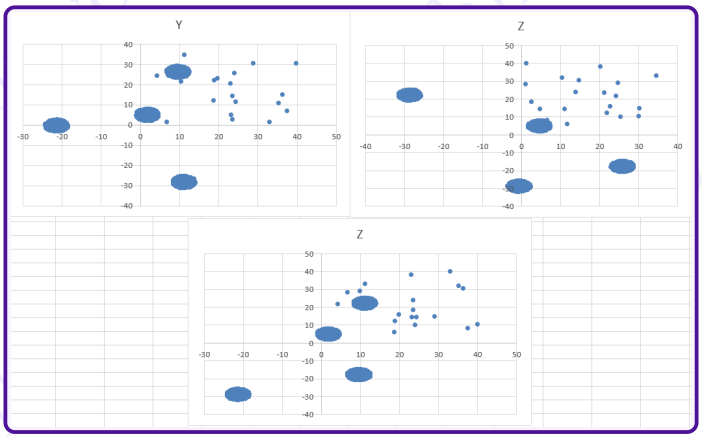
Для каждого файла определите координаты центра каждого кластера, затем вычислите одно число: P_{xyz} — квадрат произведения средних арифметических абсцисс, ординат и аппликат центров кластеров.

В ответе запишите два числа через пробел: сначала целую часть частного $\frac{P_{xyz}}{10}$ для файла А, далее целую часть частного $P_{xyz} \cdot 100$ для файла Б.

Решение (файл Б):

Для начала визуально оценим данные в условии кластеры. Для этого откроем предложенные файлы в *Excel*, перейдем в раздел **«Вставка → Диаграммы → Точечная»**. Построим двухмерные диаграммы для осей XY, YZ, XZ.

Диаграммы для файла Б имеют вид:




```python
from math import dist # Функция для вычисления расстояния Евклида

# Функция для кластеризации
def dbscan(a, r):
    clusters = [] # Создаем массив для кластеров
    while a: # Пока в списке 'a' есть звезды
        clusters.append([a.pop(0)]) # Создаем кластер, добавляем 1ую звезду
        for star in clusters[-1]: # Для каждой звезды в последнем кластере
            for i in a[:]: # Для каждой звезды в списке 'a'
                if dist(star, i) <= r: # Если рядом есть звезда
                    clusters[-1].append(i) # Добавляем звезду в кластер
                    a.remove(i) # Удаляем ее из исходного списка
    return clusters # Возвращаем список кластеров
```

```
f = open('2.txt')# Открываем файл
s = f.readline() # Считываем первую строку с названием столбцов
a = [list(map(float, i.replace(',', '.').split())) for i in f]
r = 0.9 # Радиус кластеризации
clusters = dbscan(a, r) # Кластеризуем звезды

for j in clusters: # Проход по кластерам
    if len(j) > 10: # Если элементов больше, чем 10 (аномалии отбрасываем)
        mn = 1000005000000000 # Минимальное расстояние
        center = [] # Список центров кластеров
        for star in j: # Перебираем каждую звезду как центр
            l = 0 # Суммы расстояний между текущей и другими звездами
            for i in j: # Проходим по всем звездам в кластере
                l += dist(star, i) # Суммируем расстояния до всех других звёзд
            if l < mn: # Если эта сумма меньше минимальной
                mn = l # Обновляем минимальное расстояние
                center = star # Обновляем центр
        print(center)
# Полученные координаты суммируем и ищем среднее арифметическое
s1 = 11.236343608539462 + 1.9595525012326342 + 9.739202985805257 \
     - 21.244428131376644
s2 = -28.522091074401644 + 4.680502331082192 + 25.986734430152122 \
     - 0.6173092699726395
s3 = 21.960963667040744 + 4.6808983632320205 - 18.03892519678243 \
     -29.08523260995357
s1 /= 4
s2 /= 4
s3 /= 4
p = (s1*s2*s3)**2 # Квадрат произведения средних арифметических
print(int(p*100))
```

Проверим разделение на кластеры с помощью модуля turtle.

```
from turtle import *
tracer(0)
m = 20
pu()
for i in clusters:
    if len(i) > 10:
        for j in i:
            x, y = j[:2] # Отсекаем координату z
            goto(x*m, y*m)
            dot(5)
```

Ответ: 68

1.2 Пример 2

Тройная звездная система – это система, в которой три звезды попарно находятся на расстоянии не более t. При этом других звезд на расстоянии менее t у этих трех звезд быть не должно.

Под расстоянием понимается расстояние Евклида между двумя точками $A(x_1, y_1)$ и $B(x_2, y_2)$ на плоскости, которое вычисляется по формуле:

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

Аномалиями назовём точки, находящиеся на расстоянии более одной условной единицы от точек кластеров. При расчётах аномалии учитывать не нужно.

В файле А хранятся данные о звёздах **двух** кластеров, где $R = 0.65$, $t = 0.01$ для каждого кластера. В каждой строке записана информация о расположении на карте одной звезды: сначала координата x, затем координата y. Значения даны в условных единицах, которые представлены вещественными числами. Известно, что количество звёзд не превышает 3000.

В файле Б хранятся данные о звёздах **трех** кластеров, где $R = 0.65$, $t = 0.01$ для каждого кластера. Известно, что количество звёзд не превышает 10 000. Структура хранения информации о звездах в файле Б аналогична файлу А.

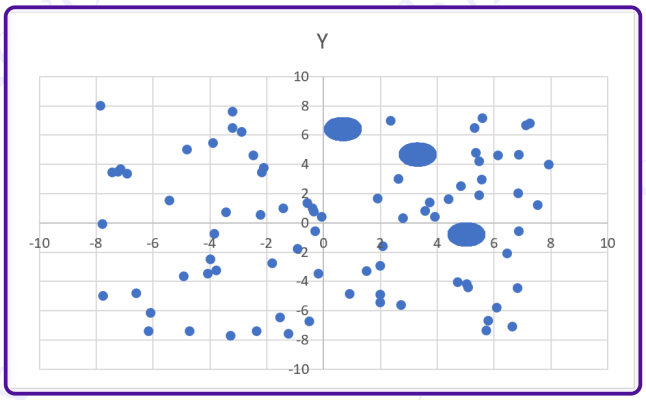
Для каждого файла в каждом кластере найдите тройную звезду, в которой три звезды системы представляют из себя остроугольный треугольник. Если таких звездных систем в кластере несколько, то выбрать стоит систему с наибольшим периметром треугольника. Затем вычислите два числа: P_x — среднее арифметическое абсцисс звезд, и P_y – среднее арифметическое ординат звезд.

В ответе запишите четыре числа через пробел: сначала целую часть произведения $P_x \cdot 10000$ для файла А, затем $P_y \cdot 10000$ для файла А, далее целую часть произведения $P_x \cdot 10000$ для файла Б и $P_y \cdot 10000$ для файла Б.

Решение (файл Б):

Для начала визуально оценим данные в условии кластеры. Для этого откроем предложенные файлы в *Excel*, перейдем в раздел **«Вставка → Диаграммы → Точечная»**.

Диаграмма для файла Б имеет вид:




```python
from math import dist # Функция для вычисления расстояния Евклида

# Функция для кластеризации
def dbscan(a, r):
    clusters = [] # Создаем массив для кластеров
    while a: # Пока в списке 'a' есть звезды
        clusters.append([a.pop(0)]) # Создаем кластер, добавляем 1ую звезду
        for star in clusters[-1]: # Для каждой звезды в последнем кластере
            for i in a[:]: # Для каждой звезды в списке 'a'
                if dist(star, i) <= r: # Если рядом есть звезда
                    clusters[-1].append(i) # Добавляем звезду в кластер
                    a.remove(i) # Удаляем ее из исходного списка
    return clusters # Возвращаем список кластеров
```

```
f = open('1.txt')# Открываем файл
s = f.readline() # Считываем первую строку с названием столбцов
a = [list(map(float, i.replace(',', '.').split())) for i in f]
r = 0.65 # Радиус кластеризации
t = 0.01 # Радиус тройной системы
clusters = dbscan(a, r) # Кластеризуем звезды
cl = [i for i in clusters if len(i) > 10]
sx = sy = 0 # Суммы координат центров кластеров
for i in cl: # Проход по кластерам без аномалий
    st = dbscan(i, t) # Кластеризуем внутри кластера с радиусом t
    mx_final = 0 # Наибольший периметр системы
    triangle = [] # Координаты звезд подходящей системы
    for j in st: # Проходим по каждой подсистеме в кластере
        if len(j) == 3: # Если в системе 3 звезды
            # Находим расстояния между звездами тройной системы
            ab = dist(j[0], j[1])
            ac = dist(j[0], j[2])
            bc = dist(j[1], j[2])
            # Если расстояние между звездами не превышает t
            if ab <= t and ac <= t and bc <= t:
                mx = max(ab, ac, bc) # Максимальная сторона треугольника
                mn = min(ab, ac, bc) # Минимальная сторона
                md = ab+ac+bc - mx - mn # Средняя сторона
                if mn**2 + md**2 > mx**2: # Если треугольник остроугольный
                    if sum([ab, ac, bc]) > mx_final: # Если периметр - макс.
                        mx_final = sum([ab, ac, bc]) # Обновляем периметр
                        triangle = j # Запоминаем звезды системы
    # Суммируем координаты звезд системы с максимальным периметром
    sx += triangle[0][0] + triangle[1][0] + triangle[2][0]
    sy += triangle[0][1] + triangle[1][1] + triangle[2][1]
print(int(sx/9*10000), int(sy/9*10000))
```

Ответ: 28802 36021