МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №7
по дисциплине «Статистические методы обработки экспериментальных данных»
Тема: Кластерный анализ. Метод поиска сгущений.

Студент гр. 8383 _____ Бабенко Н.С.
Студент гр. 8383 _____ Сахаров В.М.
Преподаватель _____ Середа А.-В.И.

Санкт-Петербург
2022

Цель работы

Освоение основных понятий и некоторых методов кластерного анализа, в частности, метода поиска сгущений.

Основные теоретические положения

Идея метода поиска сгущений заключается в построении гиперсферы заданного радиуса, которая перемещается в пространстве классификационных признаков в поисках локальных сгущений объектов. Метод поиска сгущений требует, прежде всего, вычисления матрицы расстояний (или матрицы мер сходства) между объектами и выбора первоначального центра сферы.

На первом шаге центром сферы служит объект, в ближайшей окрестности которого расположено наибольшее число соседей. На основе заданного радиуса сферы (R) определяется совокупность точек внутри этой сферы, и для них вычисляются координаты центра. Когда очередной пересчет координат центра сферы приводит к такому же результату, как и на предыдущем шаге, перемещение сферы прекращается, а точки, попавшие в нее, образуют кластер, и из дальнейшего процесса кластеризации исключаются.

Перечисленные процедуры повторяются для всех оставшихся точек. Работа алгоритма завершается за конечное число шагов, и все точки оказываются распределенными по кластерам. Число образовавшихся кластеров заранее неизвестно и сильно зависит от заданного радиуса сферы.

Для оценки устойчивости полученного разбиения целесообразно повторить процесс кластеризации несколько раз для различных значений радиуса сферы, изменяя каждый раз радиус на небольшую величину.

Существуют различные способы выбора начального радиуса сферы. В частности, если обозначить через d_{ij} расстояние между i-м и j-м объектами, то в качестве нижней границы значения радиуса сферы можно выбрать минимальное из таких расстояний, а в качестве верхней границы - максимальное:

$$R_{min} = \min_{i,j} d_{ij};\ R_{max} = \max_{i,j} d_{ij}$$

Тогда, если начинать работу алгоритма с

$$R = R_{min} + \delta;\ \delta > 0$$

и при каждом его повторении увеличивать значение δ на некоторую величину, то в конечном итоге можно найти значения радиусов, которые приводят к устойчивому разбиению на кластеры.

После завершения многомерной классификации необходимо оценить полученные результаты. Для этой цели используются специальные характеристики – функционалы качества. Наилучшим разбиением считается такое, при котором достигается экстремальное (минимальное или максимальное) значение выбранного функционала качества.

Постановка задачи

Дано конечное множество из объектов, представленных двумя признаками (в качестве этого множества принимаем исходную двумерную выборку, сформированную ранее в лабораторной работе №4). Выполнить разбиение исходного множества объектов на конечное число подмножеств (кластеров) с использованием метода поиска сгущений. Полученные результаты содержательно проинтерпретировать.

Порядок выполнения работы

a) Нормализовать множество точек, отобразить полученное множество.
b) Реализовать алгоритм поиска сгущений, отобразить полученные кластеры, выделить каждый кластер разным цветом, отметить центроиды.
c) Проверить чувствительность метода к погрешностям. Сделать выводы.
d) Сравнить с методами из лабораторной работы №6. Сделать выводы.

Выполнение работы

- Нормирование

Исходная выборка представлена в таблице 1.

Таблица 1

№	***nu***	***E***	№	***nu***	***E***	№	***nu***	***E***	№	***nu***	***E***	№	***nu***	***E***
1	481	135.2	***21***	418	131.4	***41***	513	159.3	***61***	450	122.3	***81***	475	143.6
2	445	124.7	***22***	378	103.8	***42***	489	149.8	***62***	468	128.9	***82***	518	144.4
3	550	147.9	***23***	521	154.9	***43***	474	132.5	***63***	441	122.8	***83***	566	175.7
4	465	140.9	***24***	394	117.7	***44***	379	94.6	***64***	460	140.7	***84***	464	131.3
5	566	168.5	***25***	504	145.3	***45***	472	135.6	***65***	480	117.7	***85***	394	112.1
6	497	147.3	***26***	440	126.7	***46***	544	169.6	***66***	429	112.9	***86***	480	146.1
7	478	136.6	***27***	465	114.8	***47***	507	142.4	***67***	457	126.4	***87***	321	86.1
8	521	139.6	***28***	418	109.3	***48***	409	116.7	***68***	464	143.2	***88***	502	132.5
9	352	84.9	***29***	418	118.6	***49***	498	164.0	***69***	431	125.0	***89***	460	122.4
10	422	117.9	***30***	465	127.7	***50***	468	142.0	***70***	424	119.0	***90***	458	104.7
11	506	153.5	***31***	447	117.5	***51***	593	187.4	***71***	502	137.2	***91***	362	111.7
12	443	122.9	***32***	433	131.5	***52***	523	152.6	***72***	465	140.7	***92***	503	148.5
13	434	140.4	***33***	460	136.8	***53***	478	126.6	***73***	492	137.5	***93***	446	144.0
14	422	108.6	***34***	382	98.8	***54***	438	122.2	***74***	446	128.4	***94***	421	115.1
15	569	157.4	***35***	532	160.6	***55***	423	115.9	***75***	482	136.4	***95***	407	110.5
16	439	119.2	***36***	482	148.2	***56***	408	110.0	***76***	510	140.6	***96***	448	137.7
17	437	129.4	***37***	472	122.6	***57***	386	105.8	***77***	434	122.3	***97***	490	139.9
18	461	138.6	***38***	532	158.7	***58***	428	130.3	***78***	623	195.7	***98***	482	141.2
19	351	89.0	***39***	473	137.9	***59***	560	169.8	***79***	468	141.2	***99***	463	129.2
20	390	91.4	***40***	525	148.3	***60***	483	130.3	***80***	471	119.7	***100***	459	145.4

Нормализация была выполнена по методу минимакс:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

После нормализации минимальное и максимальное масштабируемые значения равны 0 и 1 соответственно.

Нормализованная выборка представлена в табл. 2 и отображена на рис. 1.

Таблица 2

№	***nu***	***E***	№	***nu***	***E***	№	***nu***	***E***	№	***nu***	***E***	№	***nu***	***E***
1	0.53	0.454	***21***	0.321	0.42	***41***	0.636	0.671	***61***	0.427	0.338	***81***	0.51	0.53
2	0.411	0.359	***22***	0.189	0.171	***42***	0.556	0.586	***62***	0.487	0.397	***82***	0.652	0.537

3	0.758	0.569	***23***	0.662	0.632	***43***	0.507	0.43	***63***	0.397	0.342	***83***	0.811	0.819
4	0.477	0.505	***24***	0.242	0.296	***44***	0.192	0.088	***64***	0.46	0.504	***84***	0.474	0.419
5	0.811	0.755	***25***	0.606	0.545	***45***	0.5	0.458	***65***	0.526	0.296	***85***	0.242	0.245
6	0.583	0.563	***26***	0.394	0.377	***46***	0.738	0.764	***66***	0.358	0.253	***86***	0.526	0.552
7	0.52	0.467	***27***	0.477	0.27	***47***	0.616	0.519	***67***	0.45	0.375	***87***	0.0	0.011
8	0.662	0.494	***28***	0.321	0.22	***48***	0.291	0.287	***68***	0.474	0.526	***88***	0.599	0.43
9	0.103	0.0	***29***	0.321	0.304	***49***	0.586	0.714	***69***	0.364	0.362	***89***	0.46	0.338
10	0.334	0.298	***30***	0.477	0.386	***50***	0.487	0.515	***70***	0.341	0.308	***90***	0.454	0.179
11	0.613	0.619	***31***	0.417	0.294	***51***	0.901	0.925	***71***	0.599	0.472	***91***	0.136	0.242
12	0.404	0.343	***32***	0.371	0.421	***52***	0.669	0.611	***72***	0.477	0.504	***92***	0.603	0.574
13	0.374	0.501	***33***	0.46	0.468	***53***	0.52	0.376	***73***	0.566	0.475	***93***	0.414	0.533
14	0.334	0.214	***34***	0.202	0.125	***54***	0.387	0.337	***74***	0.414	0.393	***94***	0.331	0.273
15	0.821	0.654	***35***	0.699	0.683	***55***	0.338	0.28	***75***	0.533	0.465	***95***	0.285	0.231
16	0.391	0.31	***36***	0.533	0.571	***56***	0.288	0.227	***76***	0.626	0.503	***96***	0.421	0.477
17	0.384	0.402	***37***	0.5	0.34	***57***	0.215	0.189	***77***	0.374	0.338	***97***	0.56	0.496
18	0.464	0.485	***38***	0.699	0.666	***58***	0.354	0.41	***78***	1.0	1.0	***98***	0.533	0.508
19	0.099	0.037	***39***	0.503	0.478	***59***	0.791	0.766	***79***	0.487	0.508	***99***	0.47	0.4
20	0.228	0.059	***40***	0.675	0.572	***60***	0.536	0.41	***80***	0.497	0.314	***100***	0.457	0.546

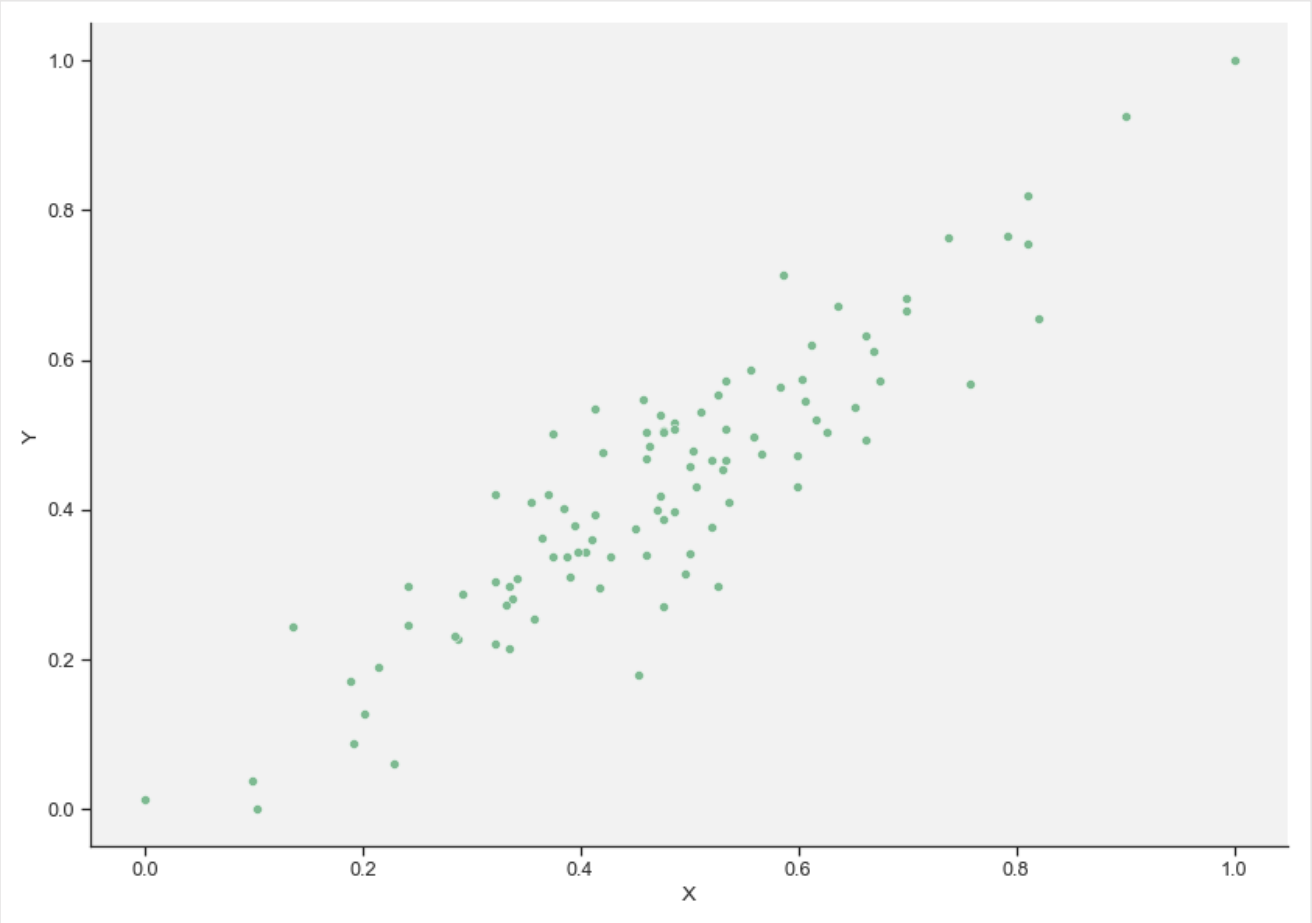

Рисунок 1 – Нормализованная выборка

- Метод поиска сгущений

Реализован метод поиска сгущений. Полученные кластеры были отображены на рисунке, отмечены разными цветами, а также были отмечены их центроиды. Определены нижняя и верхняя границы радиуса сферы:

$$R_{min} = 0.00181;\ R_{max} = 1.40658$$

Значение R было выбрано $R = 0.25$, так как оно позволяет достичь стабильного разбиения на четыре кластера.

Формирование кластеров представлено на рис. 2 – 12. На рисунках текущий кластер выделен фиолетовым, оставшиеся элементы – оранжевым, центроид – красным.

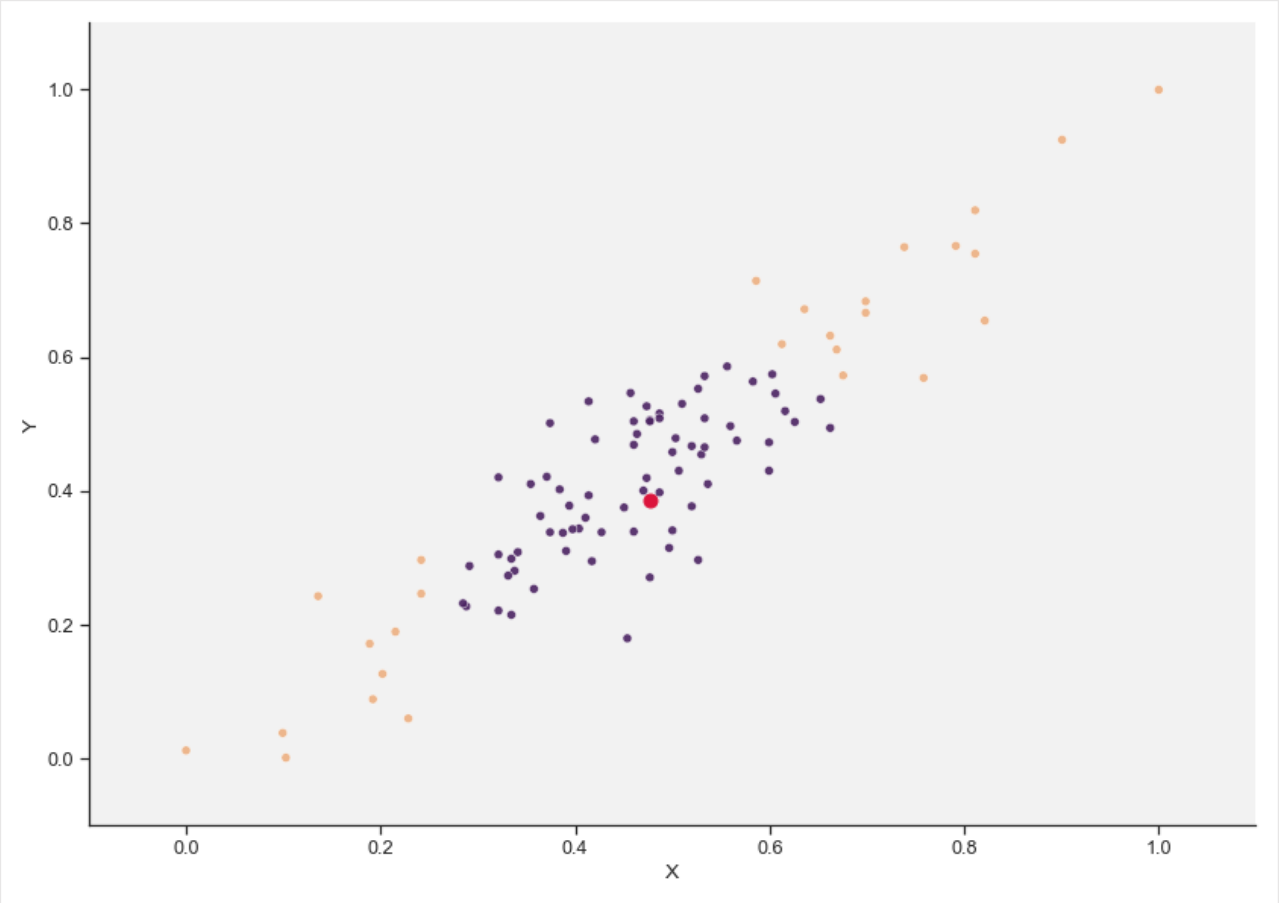

Рисунок 2 – Первый кластер, шаг 1

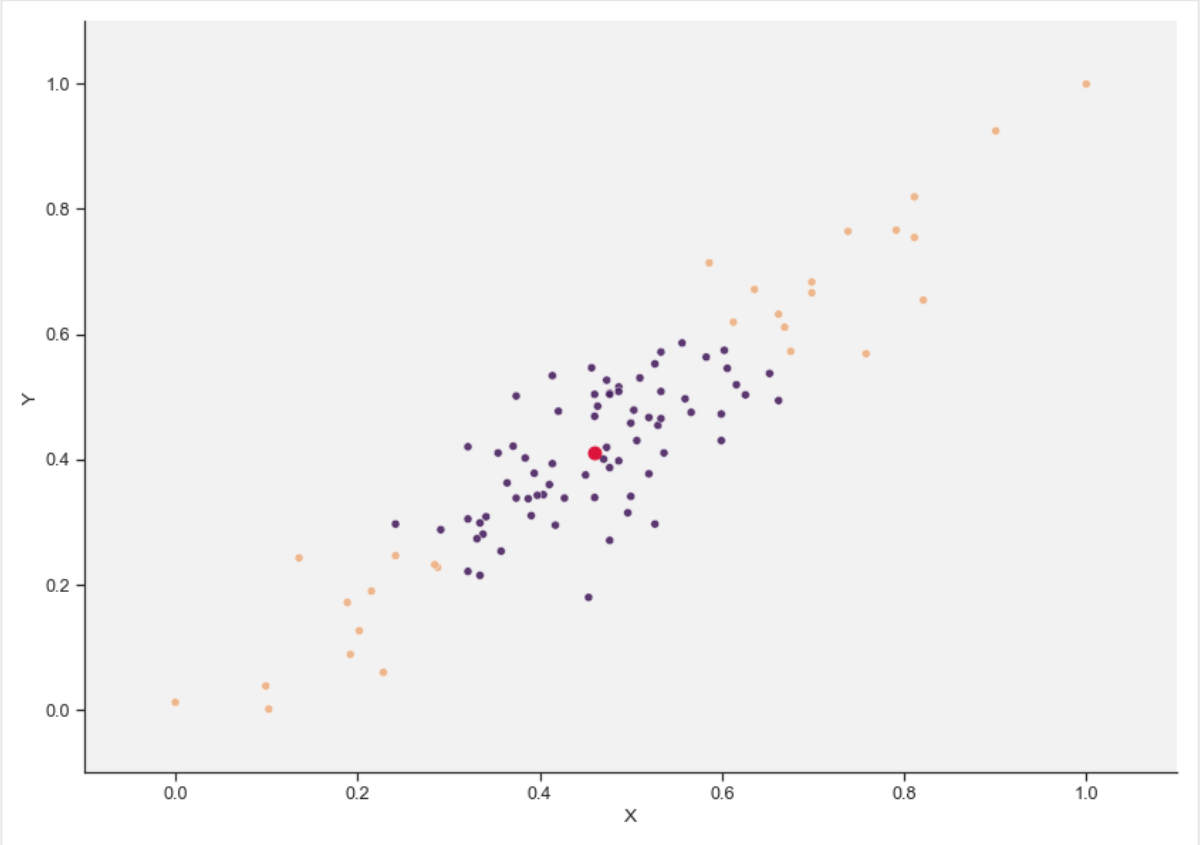

Рисунок 3 – Первый кластер, шаг 2

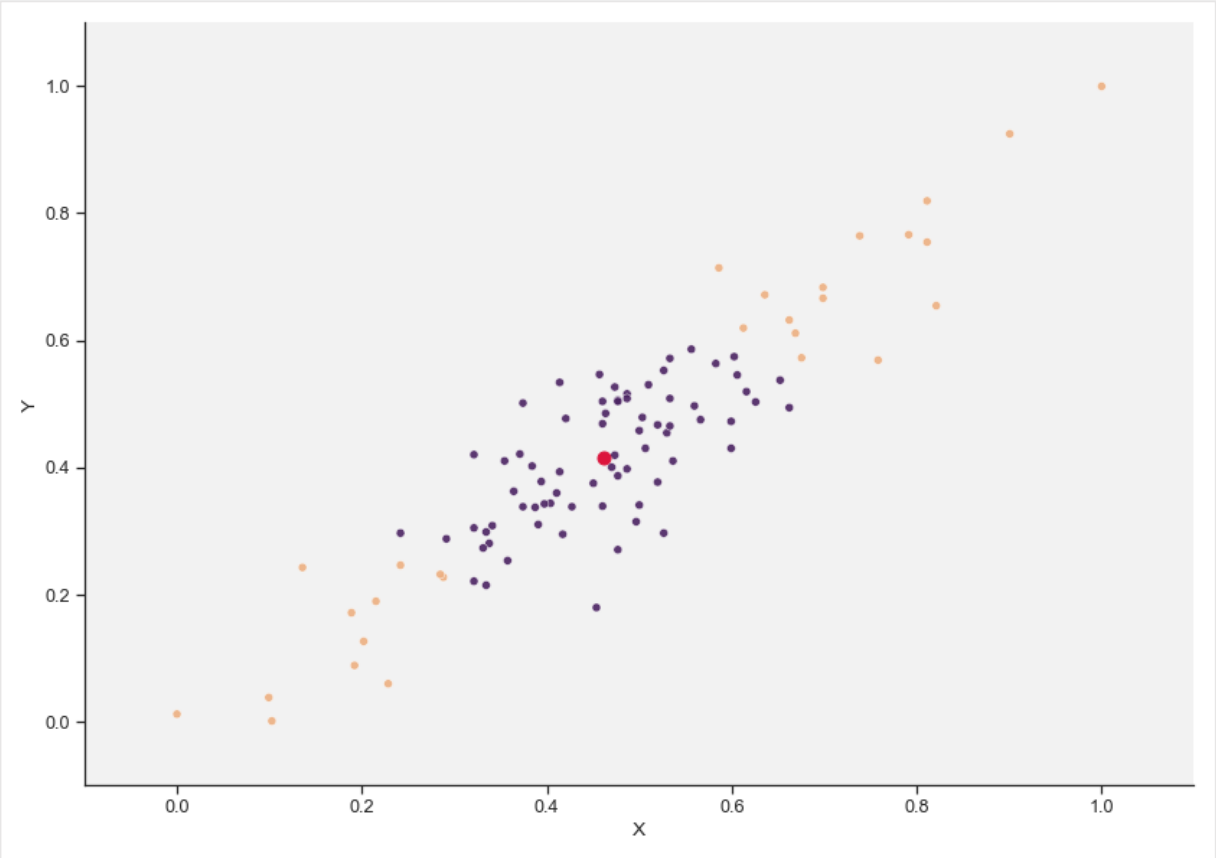

Рисунок 4 – Первый кластер, шаг 3

Таблица 3 – Первый кластер

Шаг	Центр		Количество элементов
1	0.47682	0.38628	73
2	0.45968	0.41116	72
3	0.46146	0.41462	72

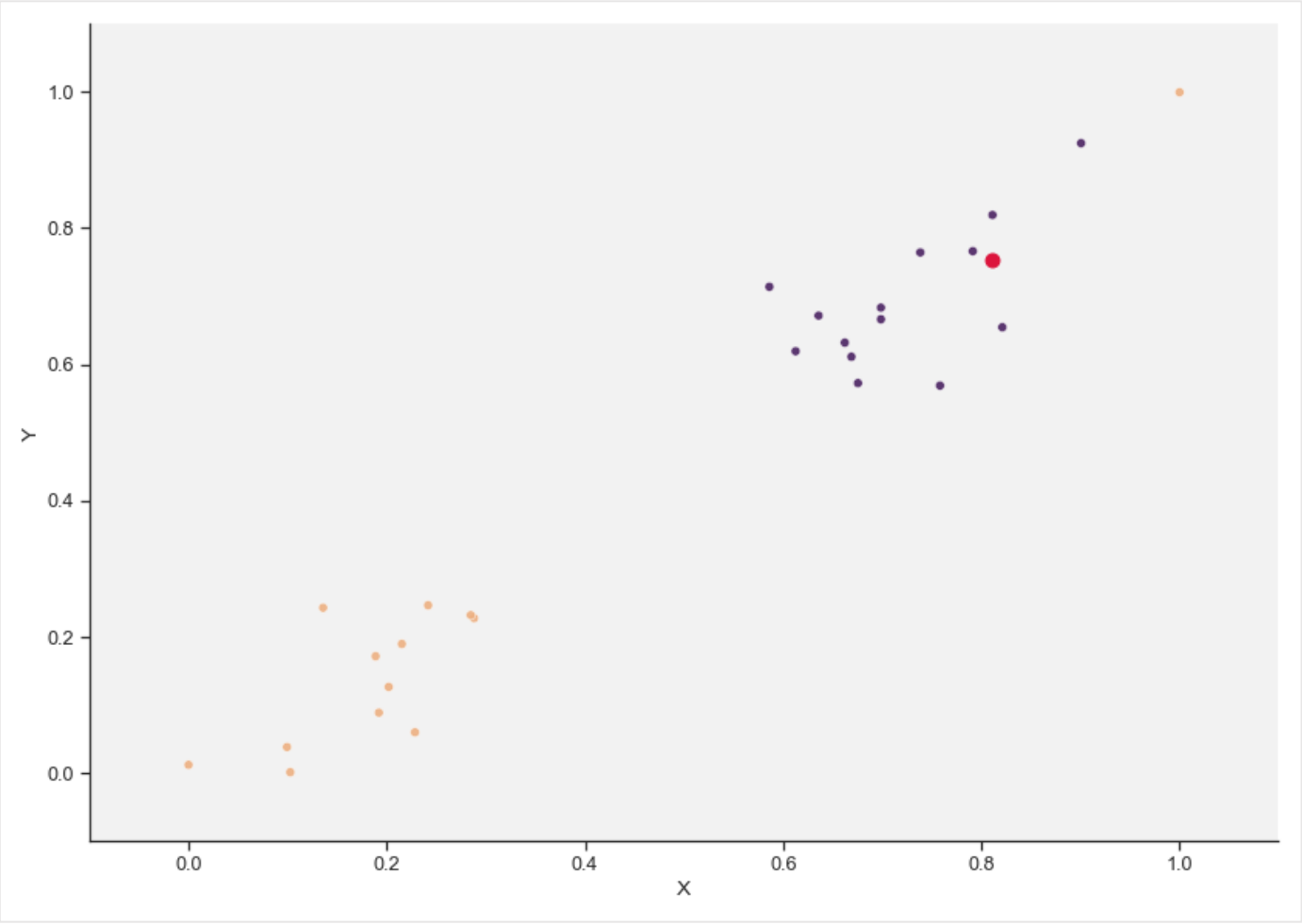

Рисунок 5 – Второй кластер, шаг 1

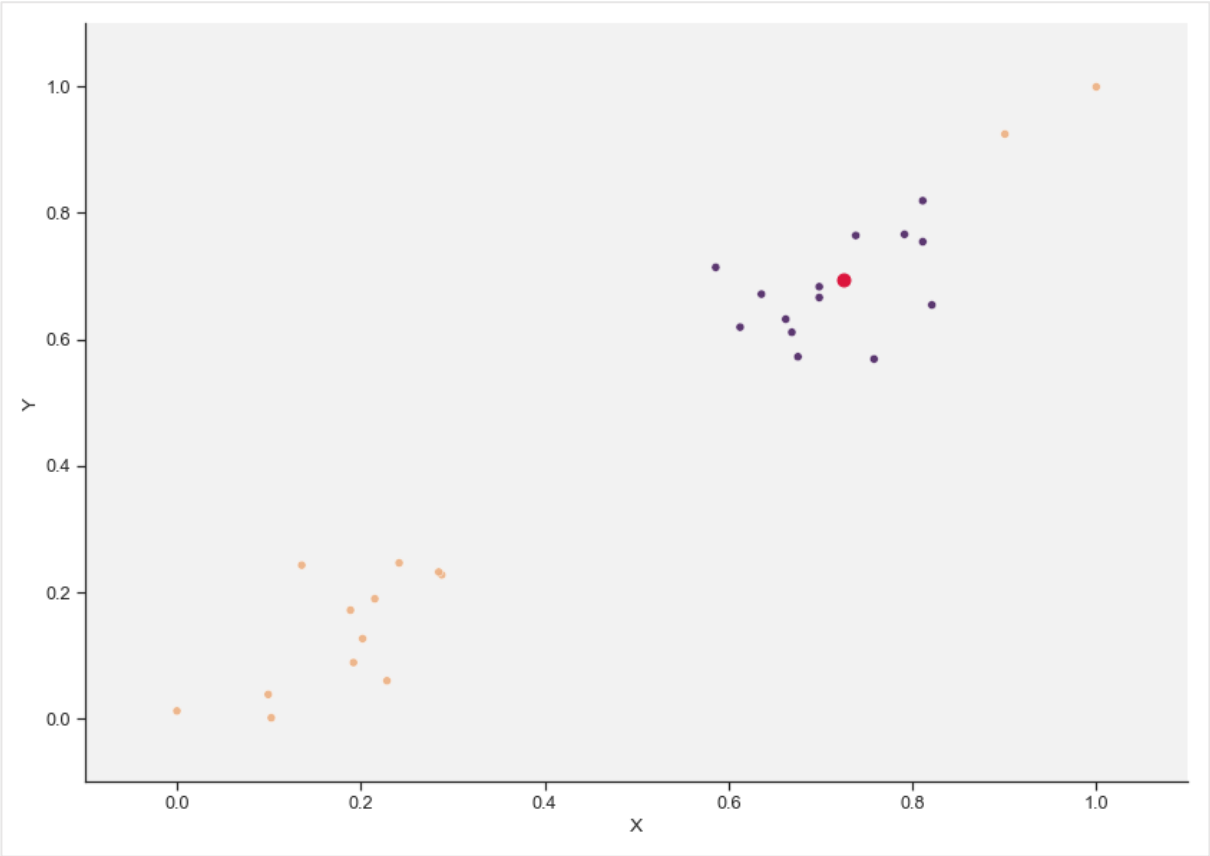

Рисунок 6 – Второй кластер, шаг 2

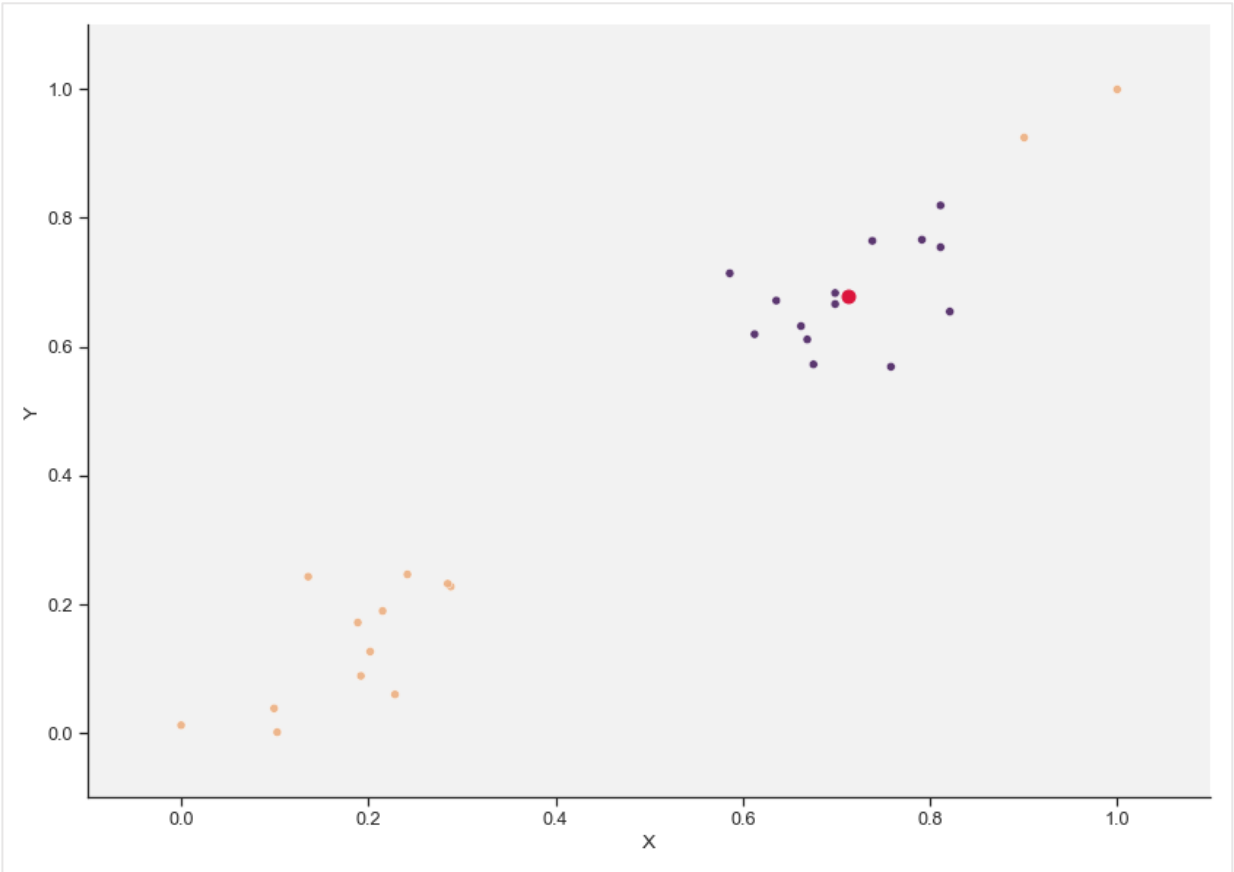

Рисунок 7 – Второй кластер, шаг 3

Таблица 4 – Второй кластер

Шаг	Центр		Количество элементов
1	0.81126	0.75451	15
2	0.72472	0.69477	14
3	0.71216	0.67831	14

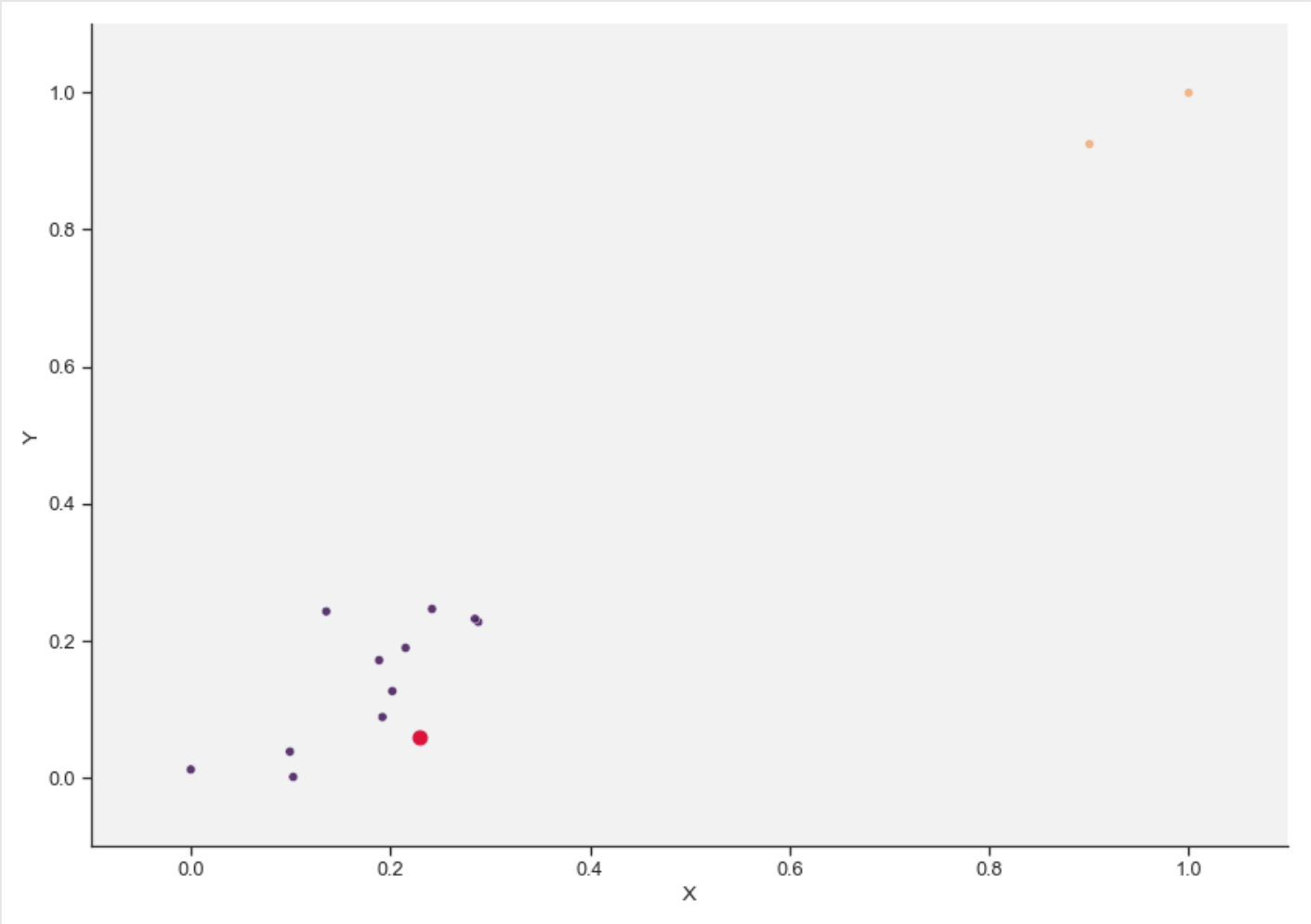

Рисунок 8 – Третий кластер, шаг 1

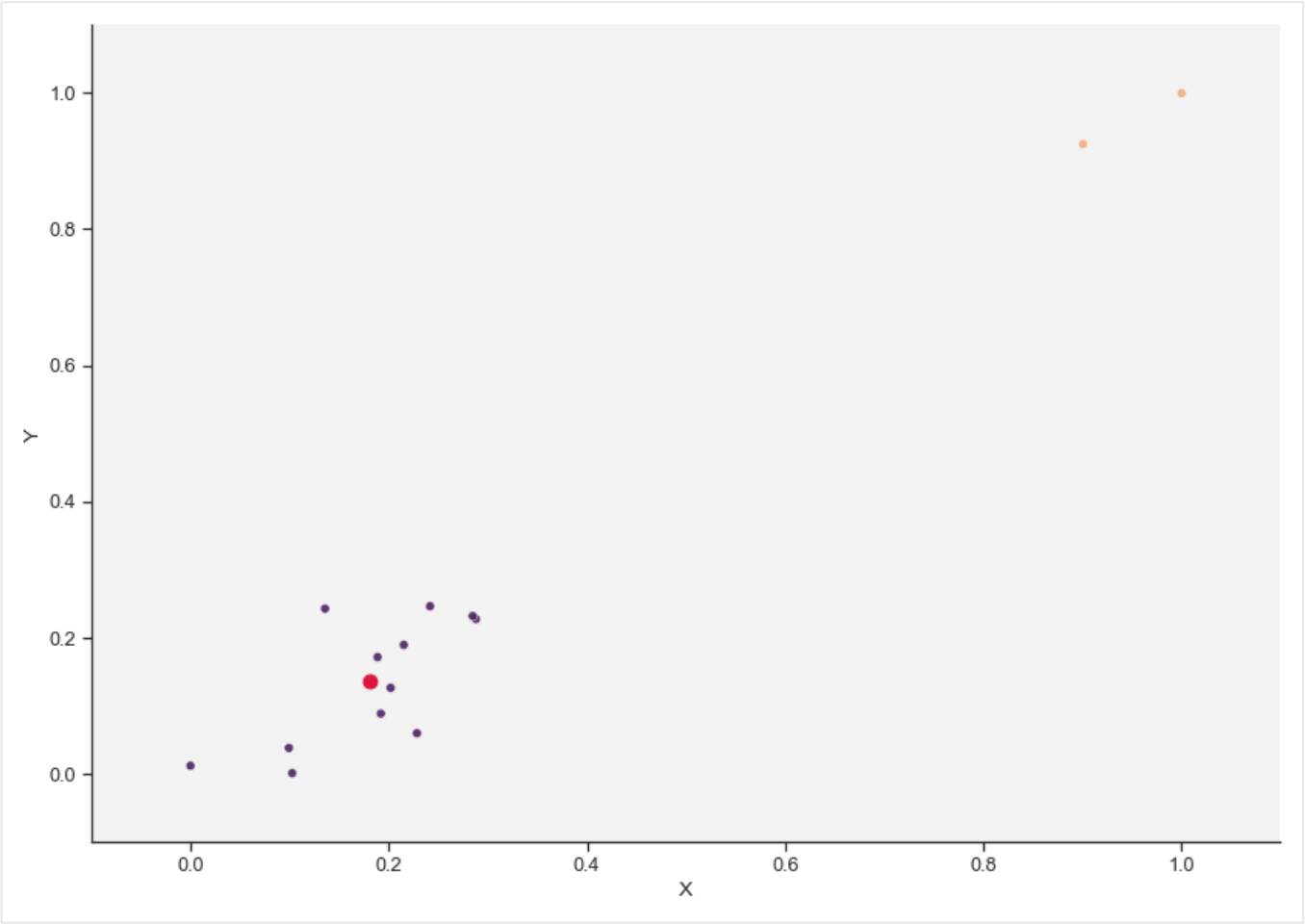

Рисунок 9 – Третий кластер, шаг 2

Таблица 5 – Третий кластер

Шаг	**Центр**		**Количество элементов**
1	0.22848	0.05867	12
2	0.18157	0.1353	12

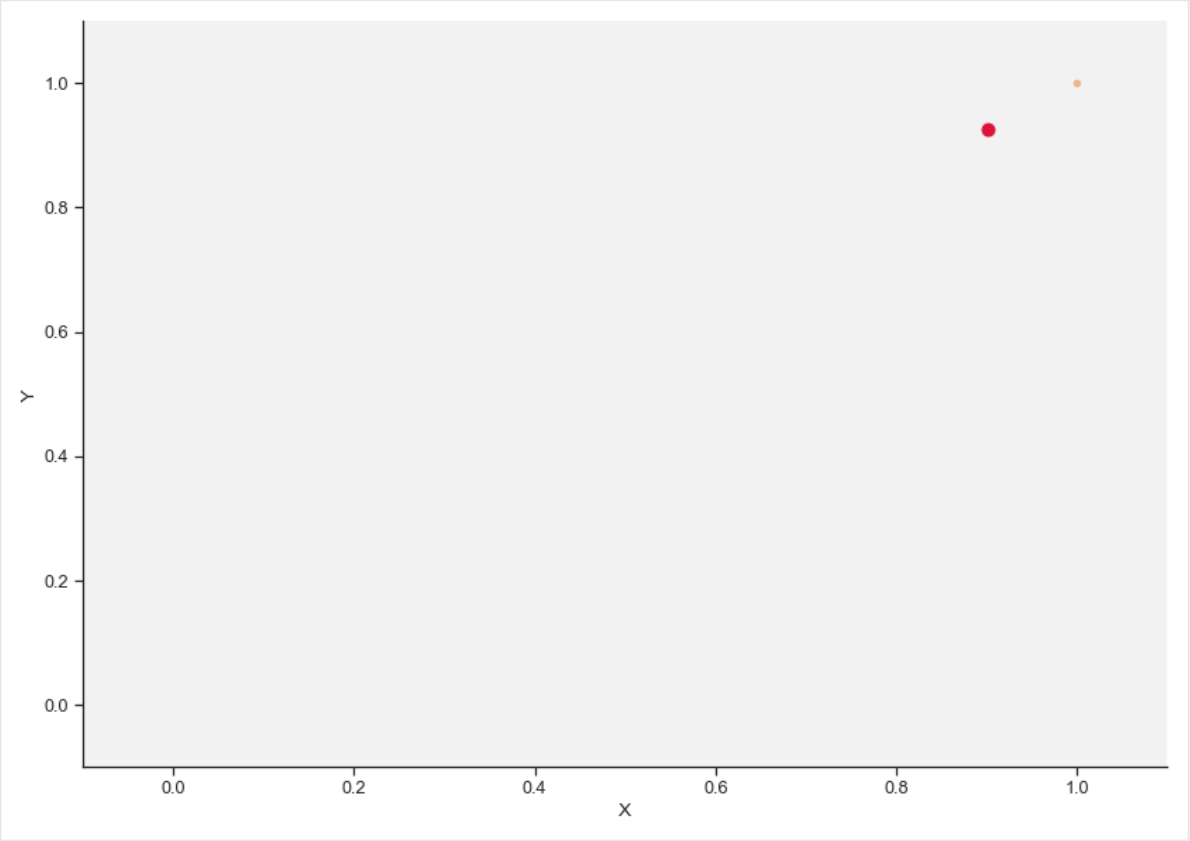

Рисунок 10 – Четвертый кластер, шаг 1

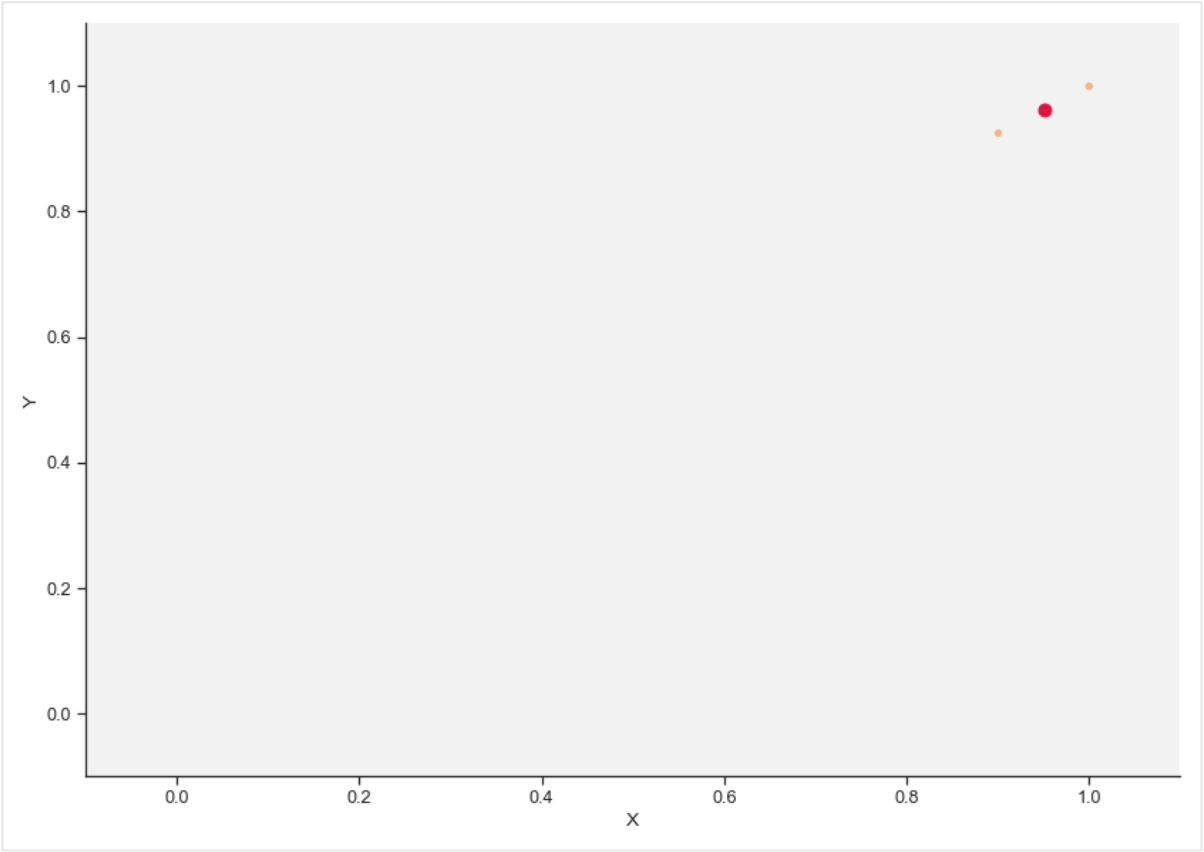

Рисунок 11 – Четвертый кластер, шаг 2

Таблица 6 – Четвертый кластер

Шаг	**Центр**		**Количество элементов**
1	0.90067	0.92509	2
2	0.95033	0.96255	2

Результат кластеризации представлен на рис. 12.

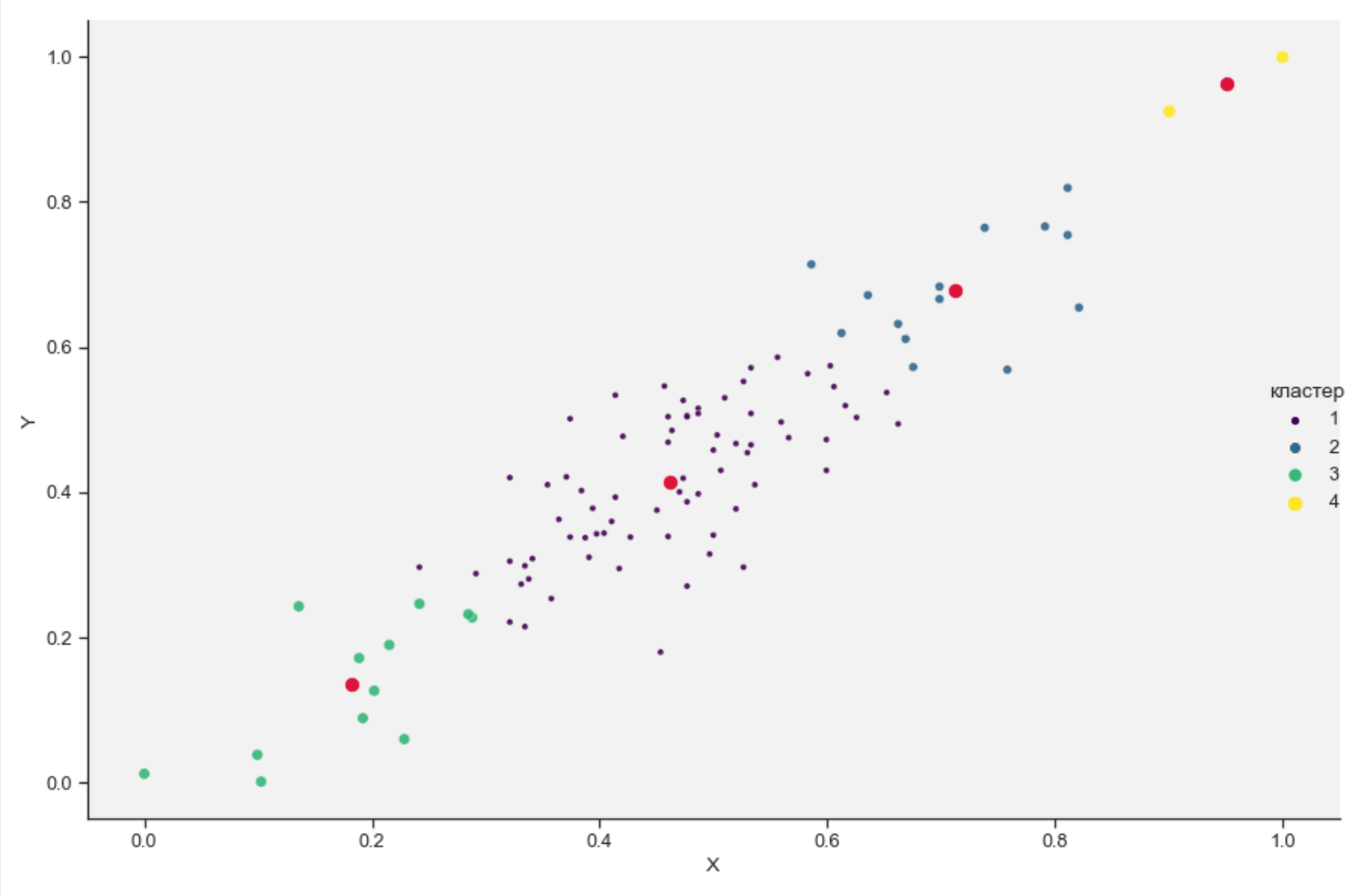

Рисунок 12 – Результат кластеризации

Несмотря на то, что имели место быть пересечения радиусов, спорные объекты не возникали. Это достигается данной реализацией алгоритма, при которой точки, попавшие в кластер, убираются для последующих итераций. А центр первого кластера выбирается как имеющий наибольшее число соседей.

- Чувствительность к погрешностям

Проведена проверка чувствительности метода к погрешностям. Радиус $R = 0.25$ был изменен на небольшое число, после чего сделано сравнение функционалов качества. Результаты представлены в таблице 7.

Таблица 7

Радиус	$\boldsymbol{F_1}$	$\boldsymbol{F_2}$	$\boldsymbol{F_3}$
$\mathbf{0.23}$	1.43236	78.98116	0.04956
$\mathbf{0.24}$	1.48355	79.92584	0.05225
$\boldsymbol{R = 0.25}$	1.67371	100.52548	0.05433
$\mathbf{0.26}$	1.7756	112.27951	0.05507
$\mathbf{0.27}$	1.86215	124.57263	0.05457

Из таблицы видно, что при изменении радиуса на небольшое значение функционалы качества изменяются на существенное значение, поэтому можно сделать вывод, что метод чувствителен к погрешностям.

- Сравнение методов

Сравним методы кластеризации с помощью значений функционалов и графиков конечного разбиения при количестве кластеров равном 4. Значения функционалов представлены в таблице 8. Графики представлены на рис. 13, 14.

Таблица 8

Метод	$\boldsymbol{F_1}$	$\boldsymbol{F_2}$	$\boldsymbol{F_3}$
k-средних	1.171	34.918	0.058
Поиск сгущений	1.67371	100.52548	0.05833

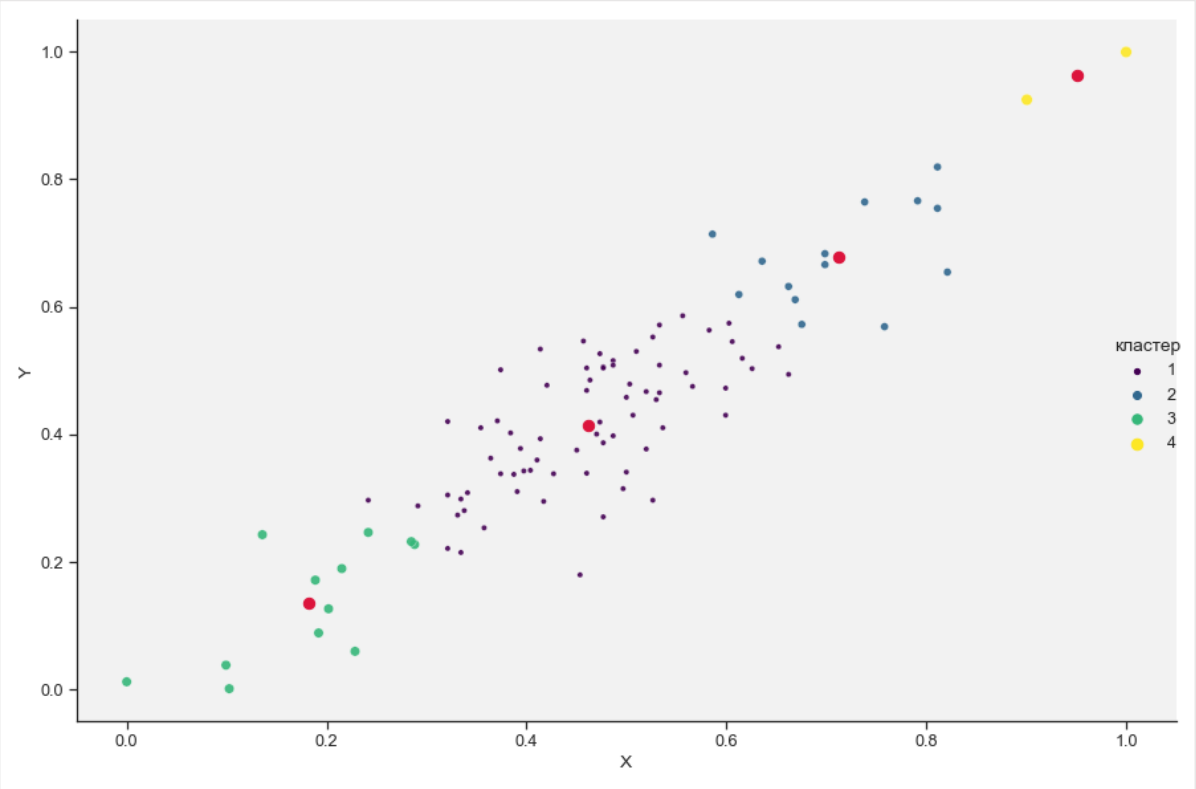

Рисунок 13 – Метод поиска сгущений

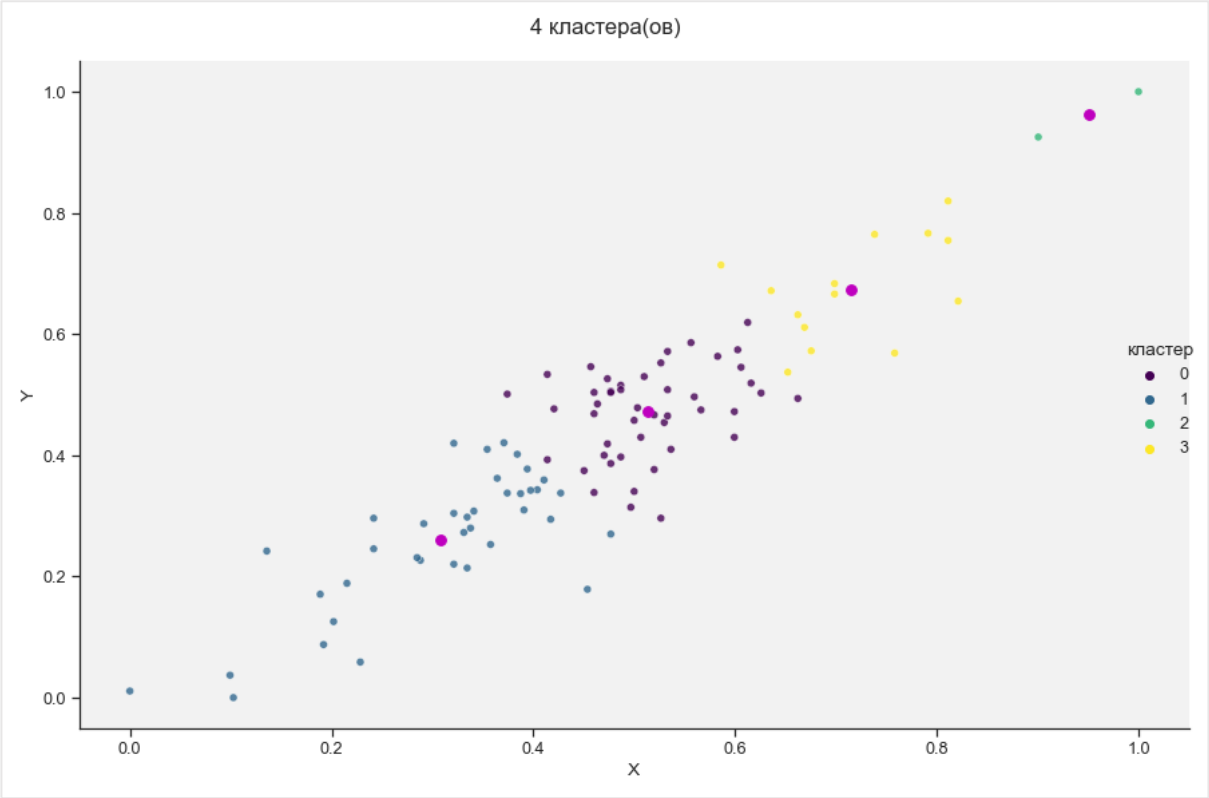

Рисунок 14 – Метод k-средних

Исходя из таблицы можно увидеть, что метод k-средних показал себя намного лучше, чем метод поиска сгущений, так как значения функционалов качества значительно меньше.

На рисунках же можно увидеть, что оба метода примерно одинаково определили два правых верхних кластера, но есть различия в двух других. Кластеры метода k-средних более сбалансированы и расстояние между точками в нем меньше по сравнению с методом поиска сгущений.

Выводы

Освоены основные понятия метода поиска сгущений. Сначала была произведена нормализация множества точек с помощью метода минимакс, так что минимальное значение равно нулю, а максимальное единице. Были найдены границы радиуса сферы:

$$R_{min} = 0.00181; \qquad R_{max} = 1.40658$$

С помощью реализованного метода поиска сгущений для $R = 0.25$ выборка была разбита на четыре кластера. Все шаги алгоритма были отображены, текущий кластер был выделен цветом.

Несмотря на то, что имели место быть пересечения радиусов, спорные объекты не возникали. Это достигается реализацией алгоритма, при которой точки, попавшие в кластер, убираются для последующих итераций. А центр первого кластера выбирается как имеющий наибольшее число соседей.

Проведена проверка чувствительности метода к погрешностям. Из сравнительной таблицы видно, что при изменении радиуса на небольшое значение функционалы качества изменяются на существенное значение, поэтому можно сделать вывод, что метод чувствителен к погрешностям.

Проведено сравнение методов кластеризации. Исходя из сравнительной таблицы можно увидеть, что метод k-средних показал себя намного лучше, чем метод поиска сгущений, так как значения функционалов качества значительно меньше. На рисунках же можно увидеть, что кластеры метода k-средних более сбалансированы и расстояние между точками в нем меньше по сравнению с методом поиска сгущений, что тоже лучше.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler
from scipy.spatial import distance
import functools

df0 = pd.read_csv('data/main_data.csv')
X = df0['nu']
Y = df0['E']

X_norm = MinMaxScaler().fit_transform(df0)
df = pd.DataFrame(data=X_norm, columns=['nu','E'])
df.round(3).to_csv('data/df_norm.csv', index=False)

sns.set_theme(palette='crest', font_scale=1.15)
sns.set_style("ticks", {"axes.facecolor": ".95"})
ax = sns.relplot(data=df, x='nu', y='E', kind='scatter', height=8.27, as-
pect=11.7/8.27)
ax.set_axis_labels('X', 'Y')
plt.tight_layout()
plt.savefig('pics/1.png')

def sc_plots(data, center, R, step, itera):
    ax = sns.relplot(data=data, x='nu', y='E', hue='кластер', kind='scatter',
palette='flare',
                     alpha=0.9, height=8.27, aspect=11.7/8.27, legend=False)
    for j in [center]:
        plt.scatter(j[0],j[1], c='crimson', s=80)

#     df_CN = df_CN.append({'x':center.values[0].round(4),
#                           'y':center.values[1].round(4),
#                           'N':data[data["кластер"]!=-1]["nu"].count()})
    print(center.values[:2])
    print(data[data['кластер']!=-1]['nu'].count())

    circle = np.array([], dtype=np.float64)
    for i in data[data['кластер']!=-1].values:
        circle = np.append(circle, np.linalg.norm(i[:-1]-center.values[:2]))

#     plt.scatter(j[0], j[1], linewidths=1, facecolors='crimson', edgecol-
ors='crimson', s=max(circle)*2*50000, alpha=0.1)

    ax.set_axis_labels('X', 'Y')
```

```
    ax.set(xlim=[-0.1,1.1], ylim=[-0.1,1.1])
    plt.tight_layout()
    plt.savefig(f'pics/{itera}_{step}.png')
    plt.show()

def Fs(data):
    curr_data = data.copy()
    cts = curr_data.groupby('кластер').mean()
    F1,F2,F3 = 0,0,0

    # F1 - сумма кв. расст. точек до центров соотв. кластеров
    for i in range(len(curr_data)):
        dist_F1 = np.linalg.norm(curr_data.iloc[i,:-1].values-cts.val-
ues[curr_data.iloc[i,2]-1])
        F1 += dist_F1**2

    # F2 - сумма кв. расст. до всех точек соотв. кластеров
    for i in range(1,len(cts)+1):
        coords = curr_data[curr_data['кластер']==i].iloc[:,:2].values
        dist_F2 = distance.cdist(coords, coords, 'euclidean')
        F2 += (np.triu(dist_F2,0)**2).sum()

    # F3 - сумма внутрикластерных дисперсий
    F3 = curr_data.groupby('кластер').var().values.sum(where=~np.is-
nan(curr_data.groupby('кластер').var().values), initial=0)

    return F1,F2,F3

def custFE(cur_data, R, itera, plots=1, max_iter=20):
    cur_dist = np.array([], dtype=np.float64)
    data = cur_data.copy()
    coords = data.values

    # расстояние между объектами
    dist = distance.cdist(coords, coords, 'euclidean')
    data['кластер'] = -1

    # сколько объектов с растоянием < R для каждого объекта
    for i in dist:
        cur_dist = np.append(cur_dist, len(i[np.where((i>=0) & (i<=R))]))

    # индекс центра
    center_ind = np.argmax(cur_dist)
    # индексы объектов с расстоянием < R до центра
    cluster_ind = np.where((dist[np.argmax(cur_dist)]>=0) &
                           (dist[np.argmax(cur_dist)]<=R))
    data.iloc[cluster_ind[0],2] = itera
    data.iloc[center_ind,2] = itera
```

```
    if plots == 1:
        sc_plots(data, data.iloc[center_ind], R, 1, itera)
    cur_center = data.iloc[center_ind]

    for it in range(max_iter):
        distl = np.array([], dtype=np.float64)
        # новый центр тяжетси
        center = data[data['кластер']==itera].mean()
        data['кластер'] = -1

        # расстояния до нового центра
        for i in data.iloc[:,:2].values:
            distl = np.append(distl, np.linalg.norm(center[:-1].values-i))
        cluster_ind = np.where((distl>=0) & (distl<=R))

        data.iloc[cluster_ind[0],2] = itera

        if functools.reduce(lambda x, y : x and y, map(lambda p, q: p == q,cen-
ter.values,cur_center.values), True):
            break
        if plots == 1:
            sc_plots(data, center, R, it+2, itera)
        cur_center = center

    # график
    if plots == 0:
        sc_plots(data, center, R, 'последний', itera)

    return data[data['кластер']==-1], data, np.array(center.values[:2])

coords = df.iloc[:,:2].values
dist = np.triu(distance.cdist(coords, coords, 'euclidean'), 0)
rmin = np.amin(dist, where=dist!=0, initial=10)
rmax = np.amax(dist)
rmin.round(5), rmax.round(5)

upd_df = df.copy()
it = 1
radius = 0.37
df['кластер'] = -1
ctrs = np.array([], dtype=np.float64)

while len(upd_df):
    upd_df, main, ctr = custFE(upd_df, radius, it, 0)
    ctrs = np.append(ctrs, [ctr])
    it += 1
    df.loc[main[main['кластер']!=-1].index, :] =
main.loc[main[main['кластер']!=-1].index, :]
```

```
df.to_csv('data/result.csv', index=False)

F1, F2, F3 = Fs(df)
F1.round(5), F2.round(5), F3.round(5)

ax = sns.relplot(data=df, x='nu', y='E', hue='кластер', kind='scatter', pal-
ette='viridis', alpha=0.9,
                 size='кластер', height=8.27, aspect=11.7/8.27)
ctrs = ctrs.reshape((-1,2))
for i in ctrs:
    plt.scatter(i[0], i[1], c='crimson', s=70)
ax.set_axis_labels('X', 'Y')
plt.tight_layout()
plt.savefig('pics/result.png')
plt.show()
```