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 _____ Киреев К.А.

Студент гр. 8383 _____ Муковский Д.В.

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

Метод поиска сгущений является еще одним итеративным методом кластерного анализа.

Основная идея метода 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

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

Выполнение работы

Нормировка данных

Первоначальная выборка представлена на рис. 1.

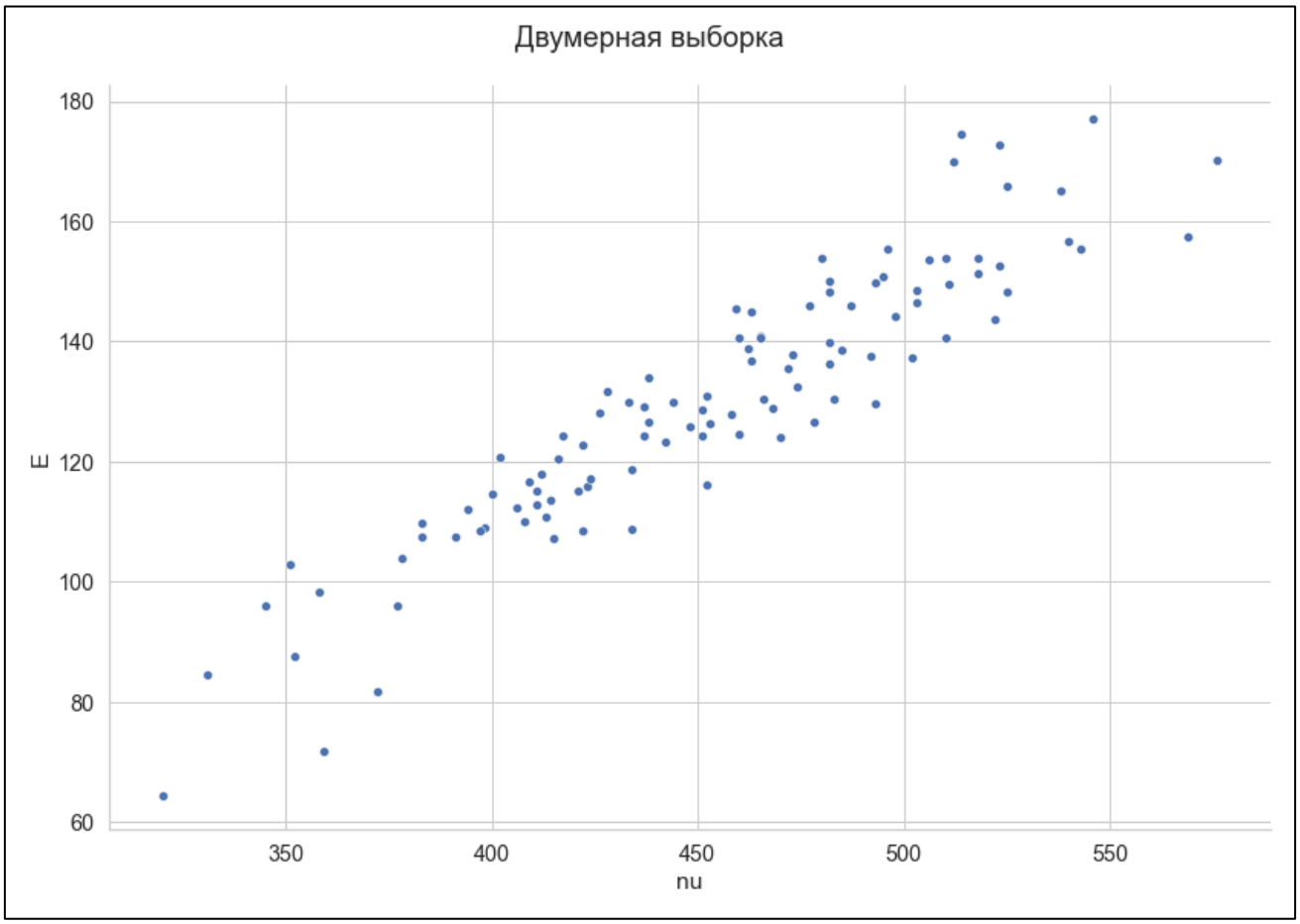

Рисунок 1 – Первоначальная выборка

Нормализуем множество точек как:

$$z_x = \frac{x - \bar{x_в}}{s_x}; z_y = \frac{y - \bar{y_в}}{s_y}$$

Тогда среднее значение будет равно нулю, а стандартное отклонение единице. Нормализованная выборка представлена на рис. 2.

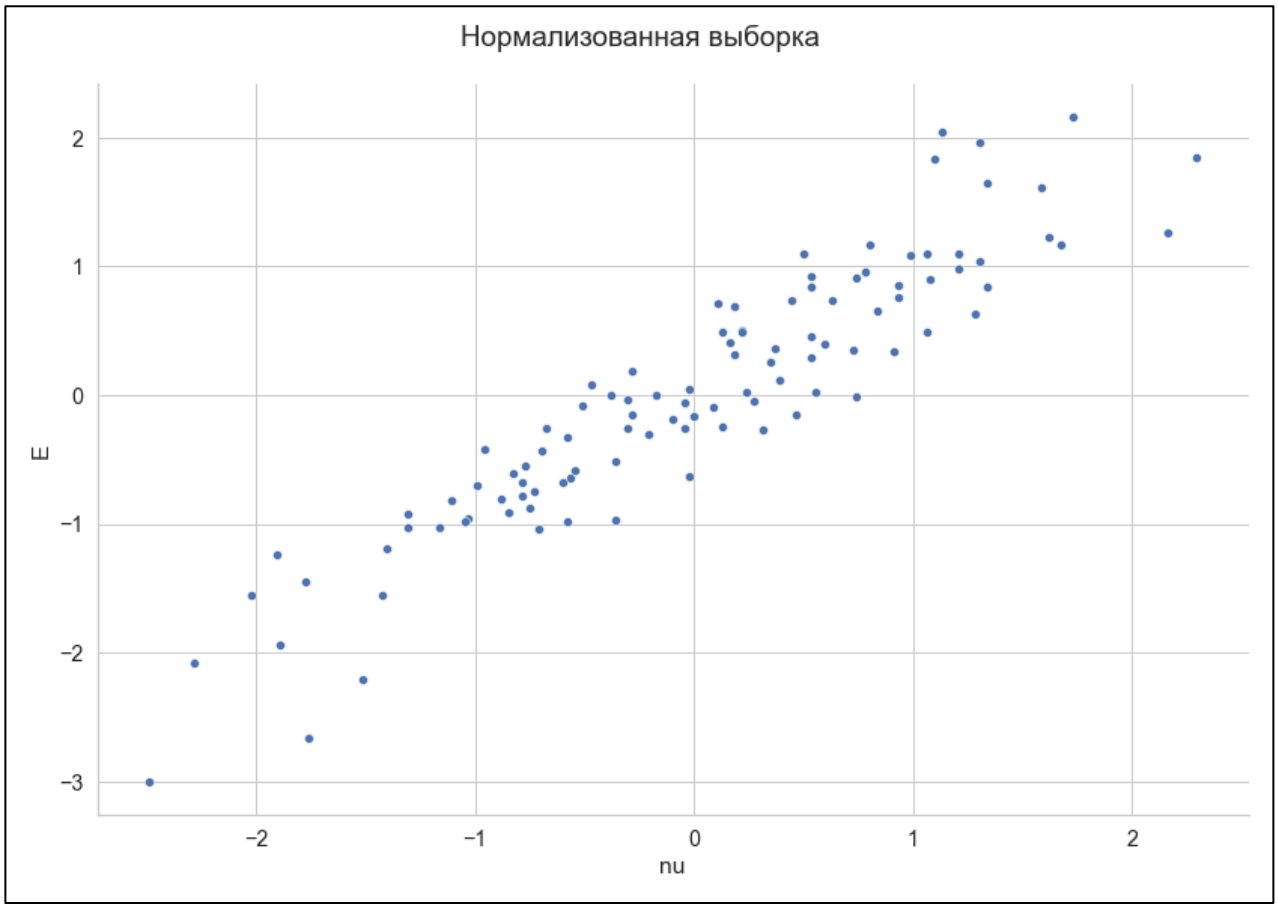

Рисунок 2 – Нормализованная выборка

Алгоритм поиска сгущений

Был реализован алгоритм поиска сгущений. Полученные кластеры были отображены разными цветами.

Вычислены нижняя и верхняя границы радиуса сферы:

$$R_{min} = mind_{ij} = 0.0092$$

$$R_{max} = maxd_{ij} = 6.8015$$

Начиная с $R_{min} = 0.1$ и изменяя радиус на $\delta = 0.05$ было найдено значение радиуса, которое приводит к устойчивому разбиению.

$$R = 1.15$$

В качестве центра сферы на первом шаге был выбран объект, в ближайшей окрестности которого было расположено максимальное число соседей – $(x_{58}; y_{58}) = (-0.2091; -0.2994)$.

В заголовках рисунков можно увидеть изменение центров кластеров и количества элементов.

Формирование кластеров представлено на рис. 3 - 16:

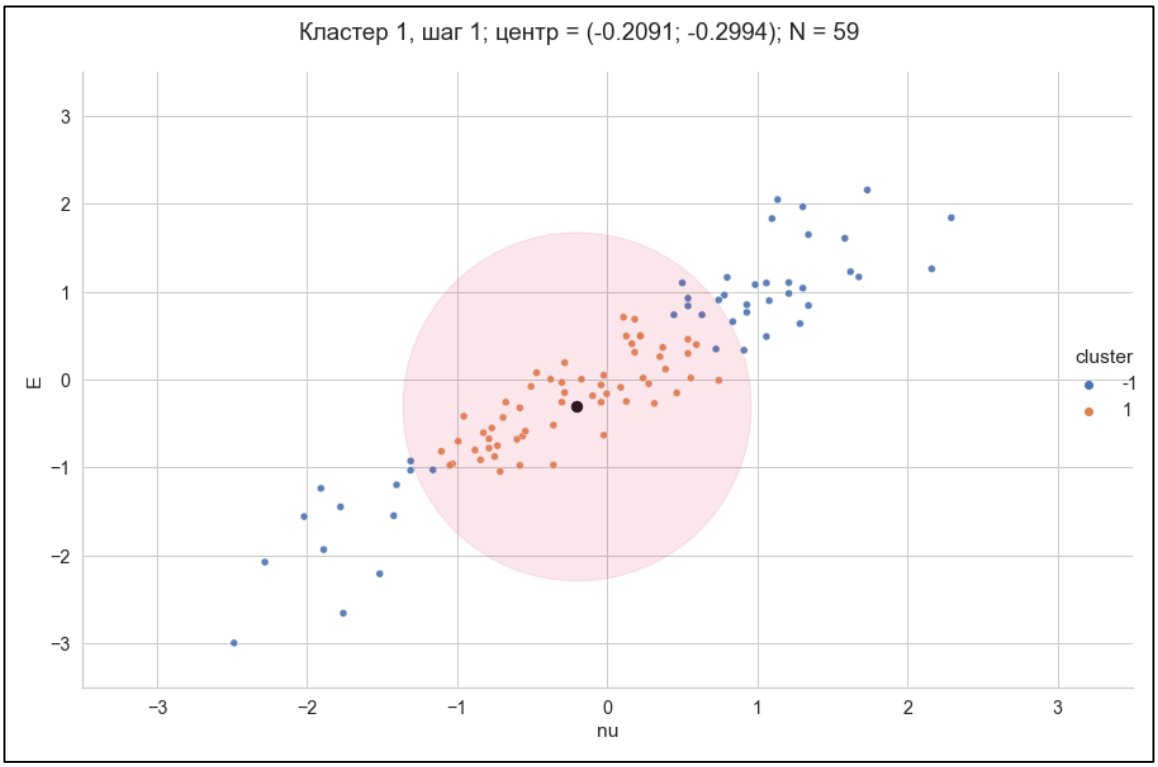

Рисунок 3

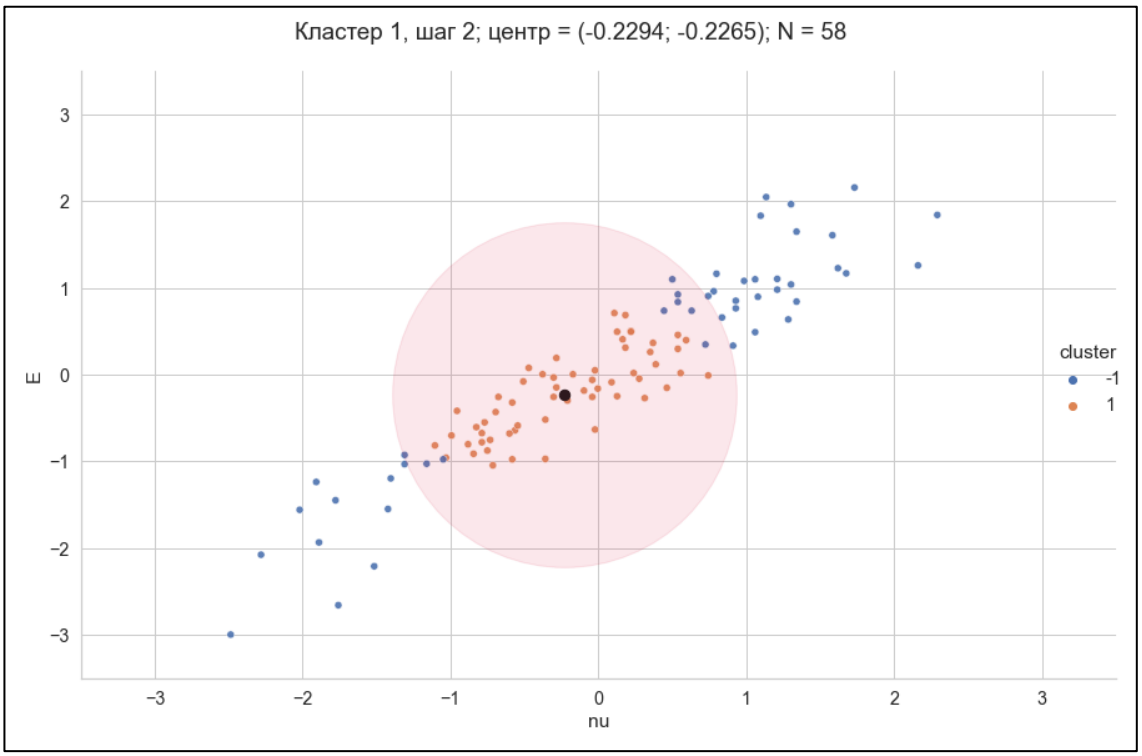

Рисунок 4

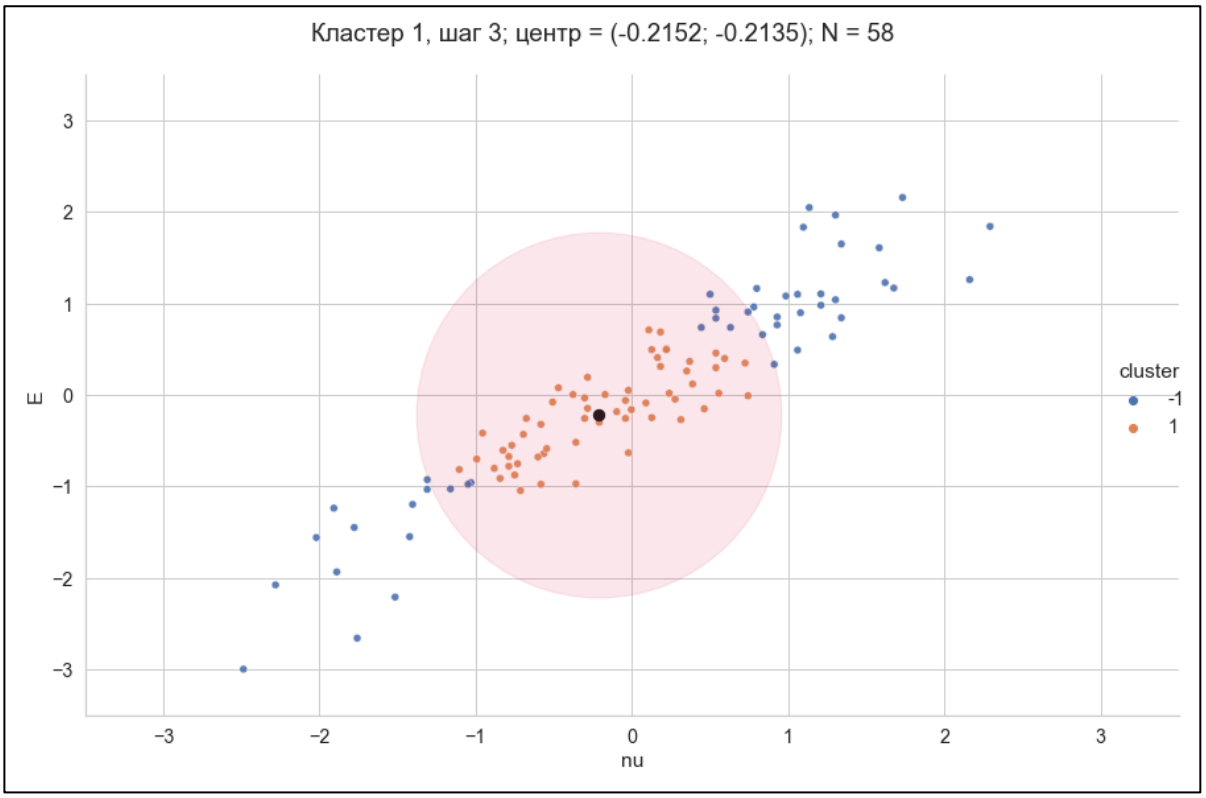

Рисунок 5

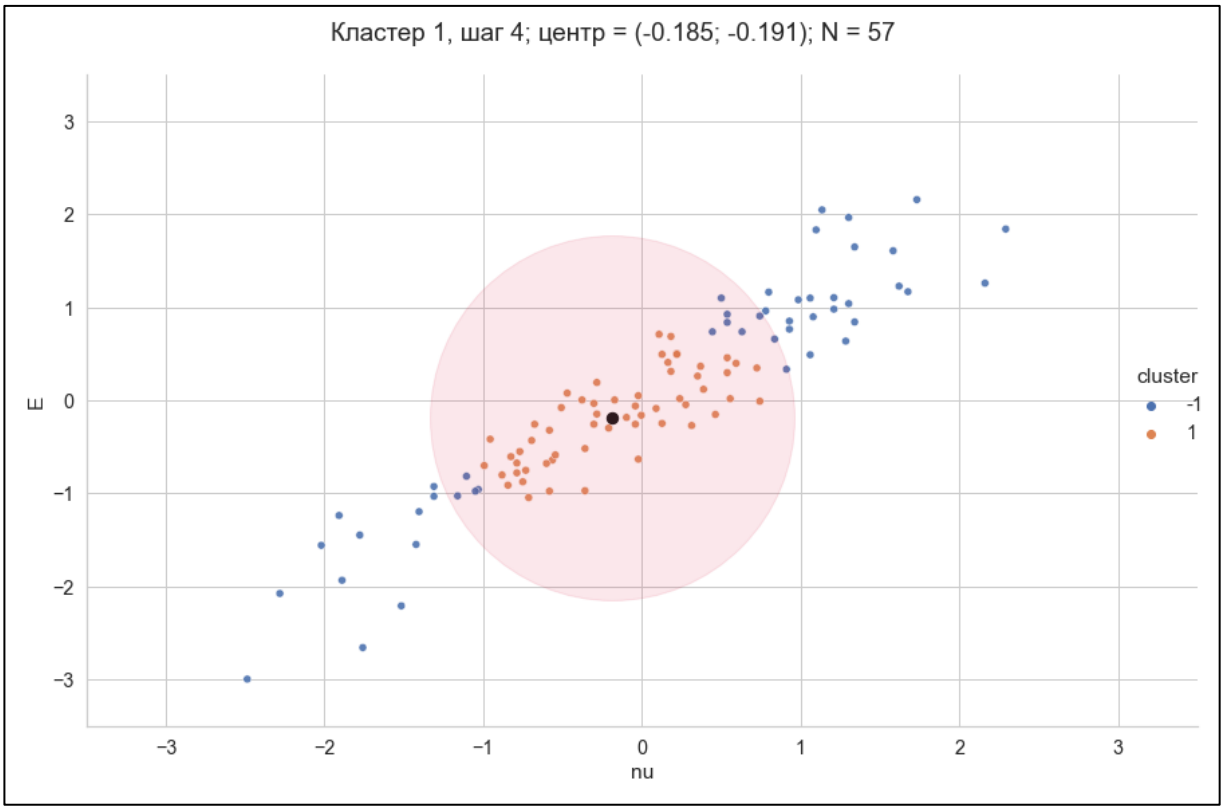

Рисунок 6

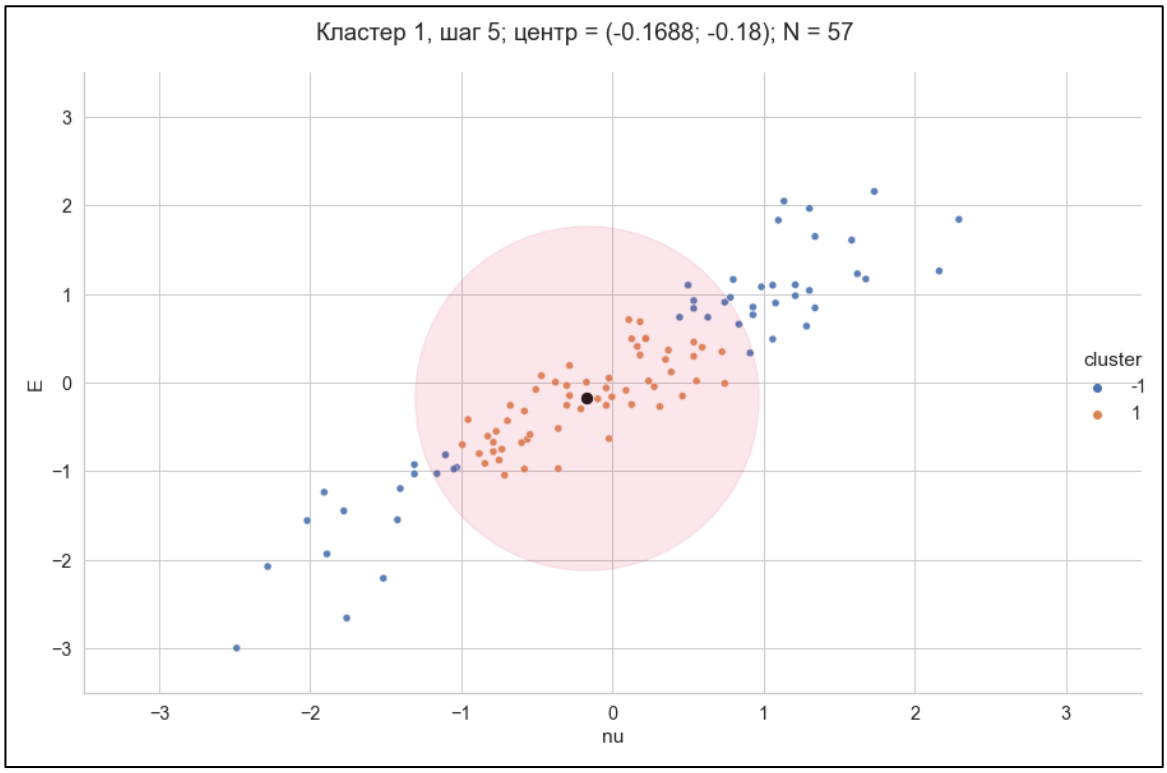

Рисунок 7

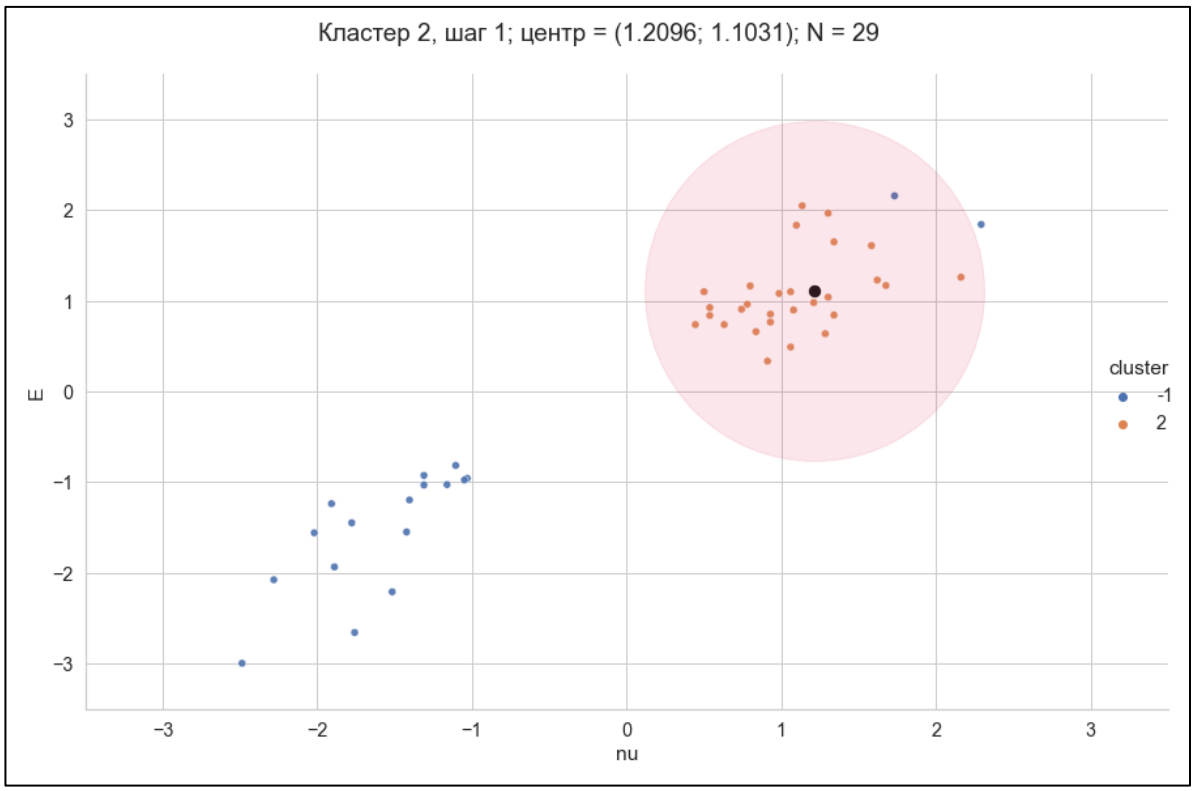

Рисунок 8

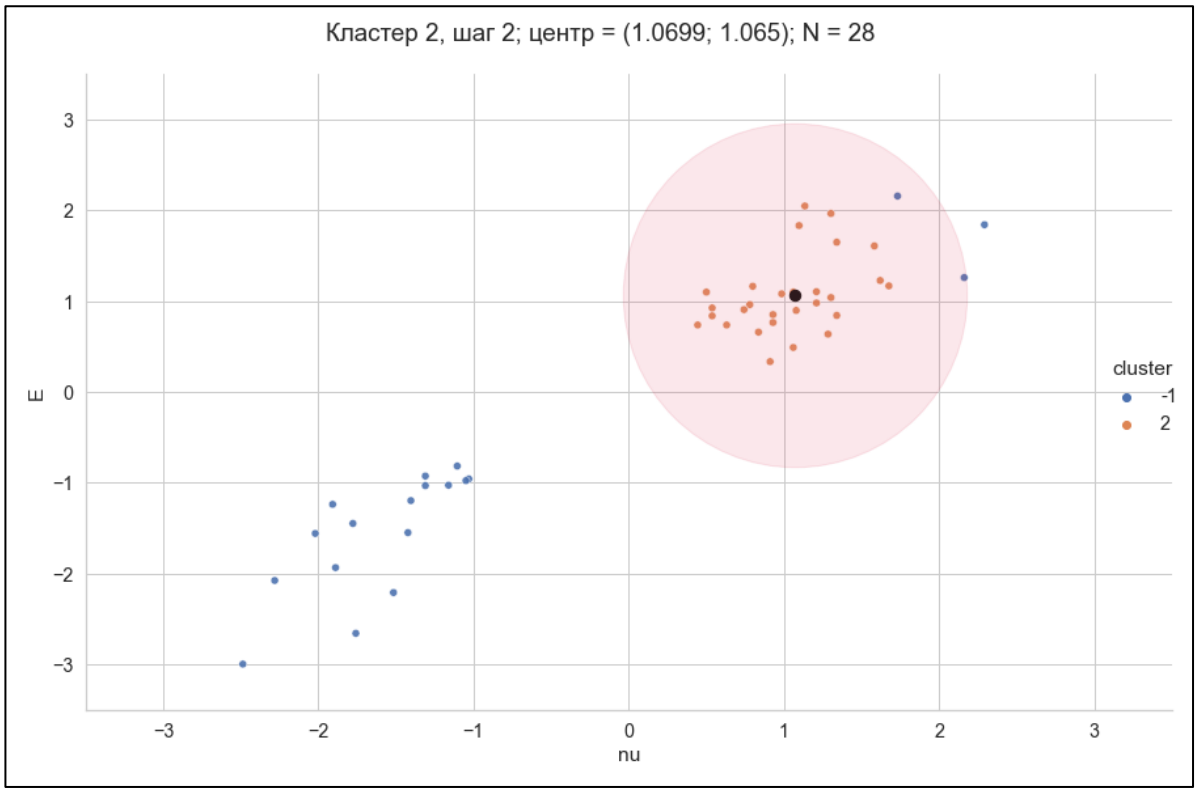

Рисунок 9

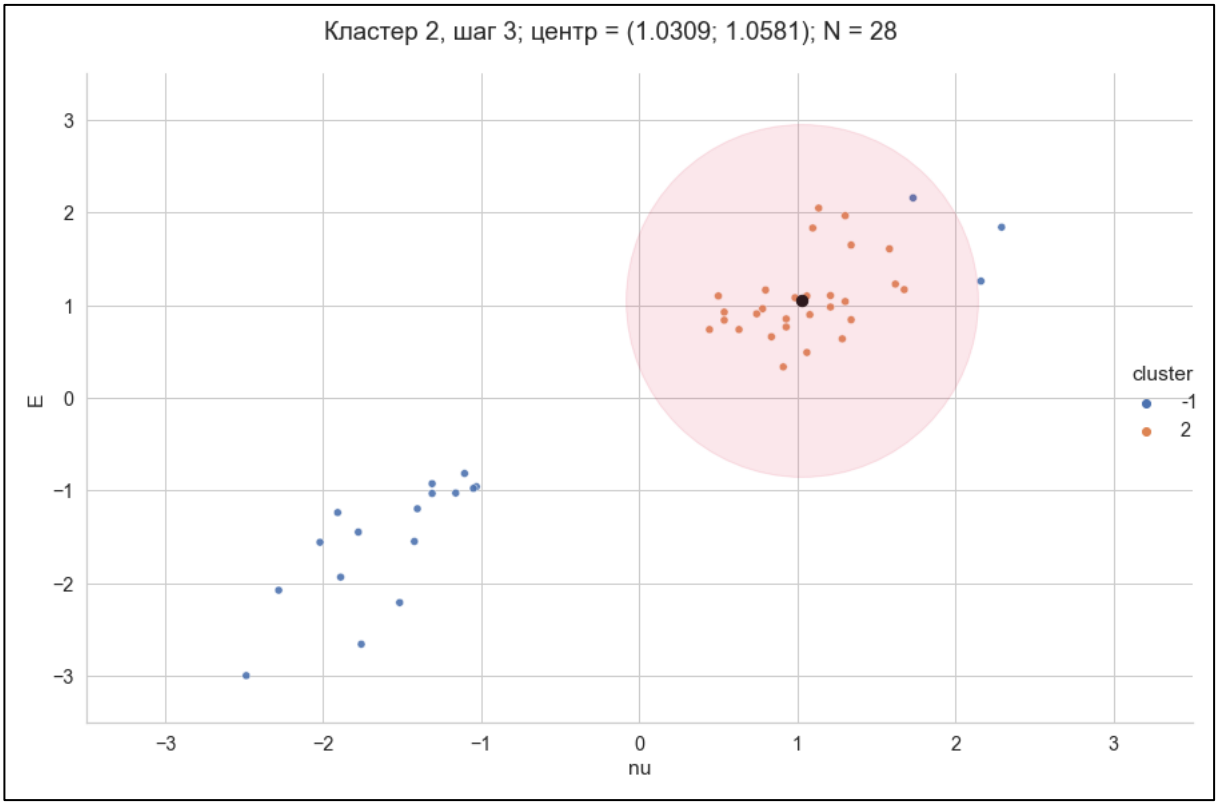

Рисунок 10

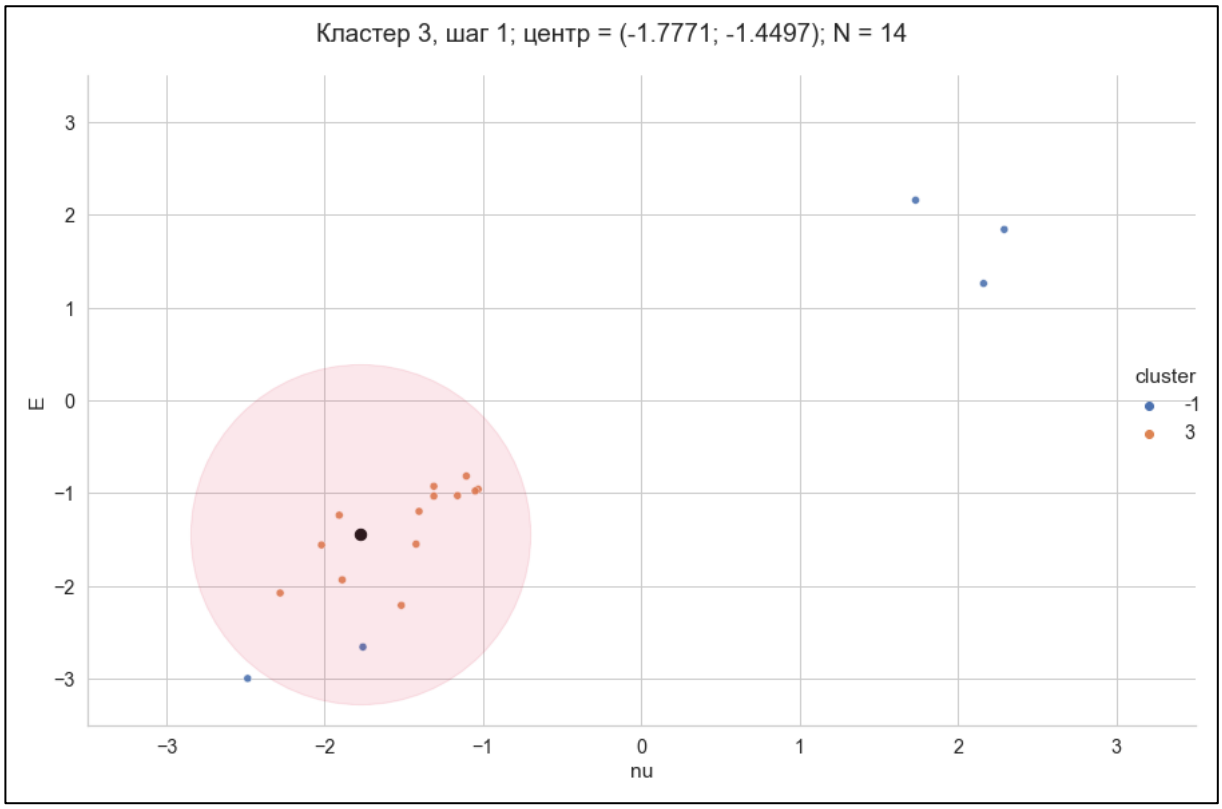

Рисунок 11

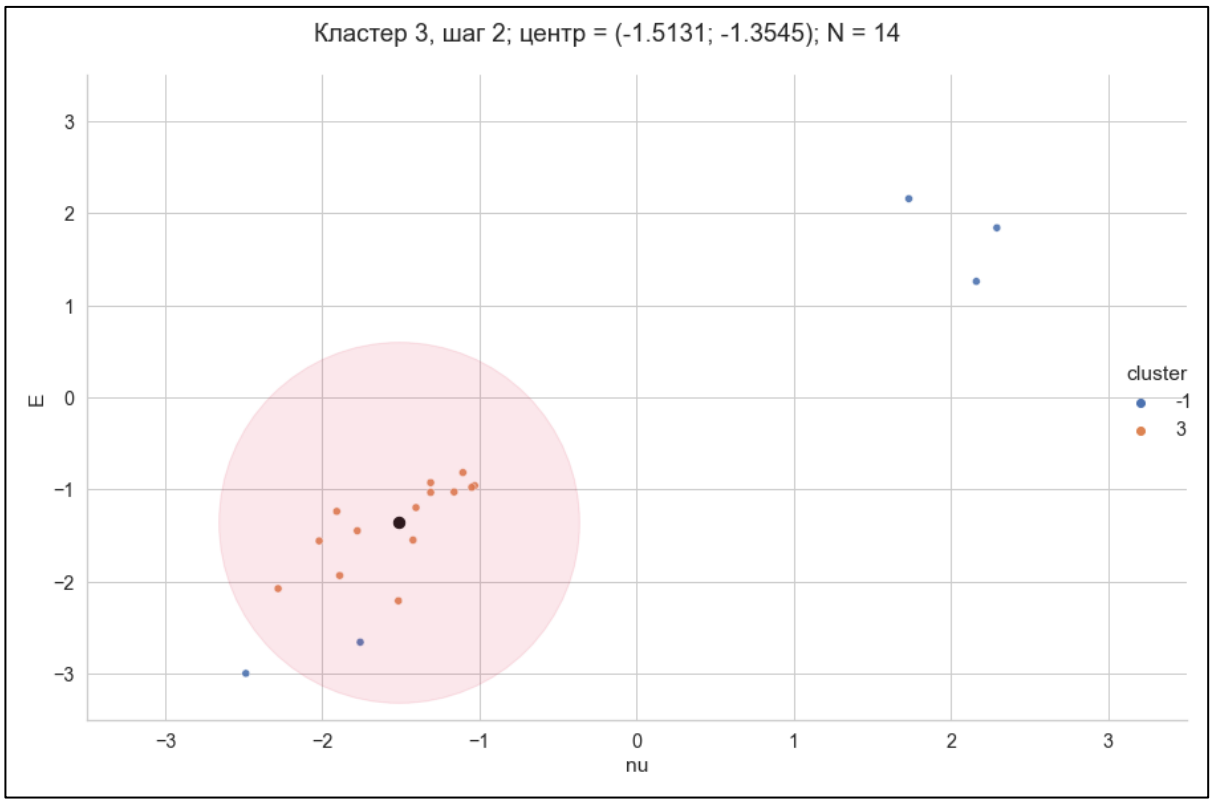

Рисунок 12

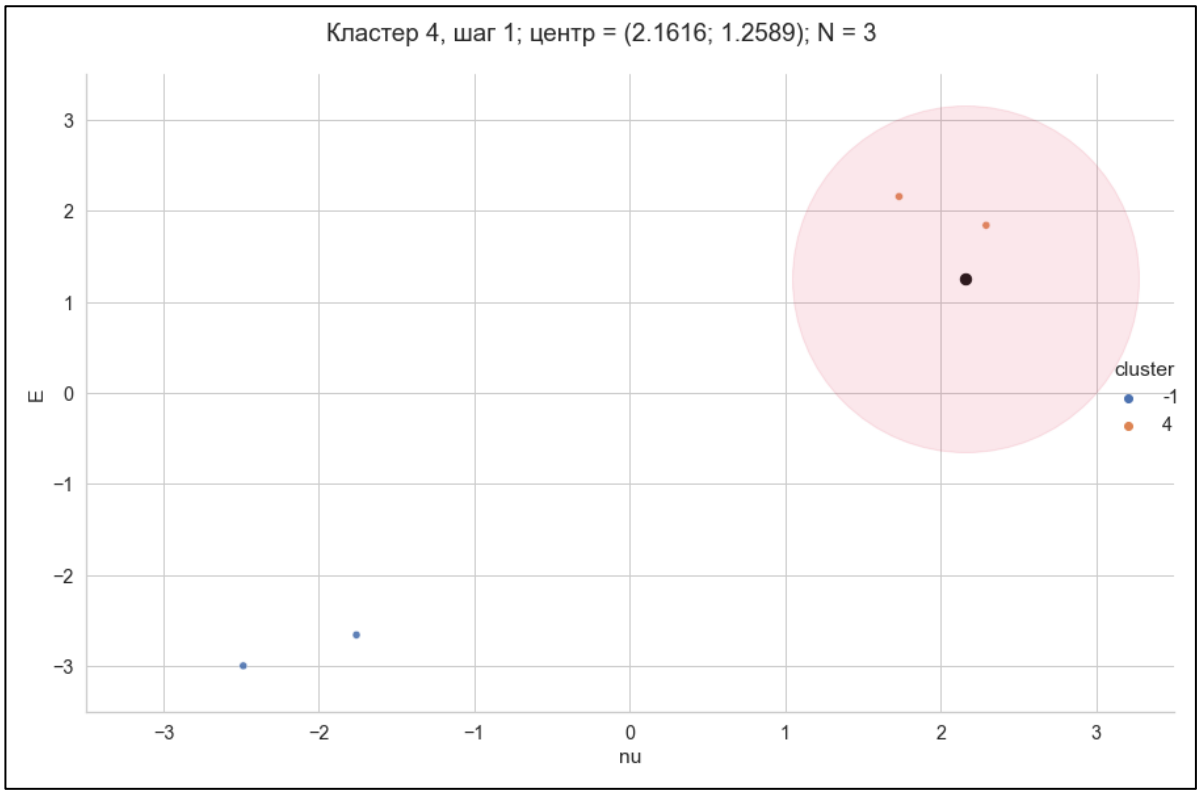

Рисунок 13

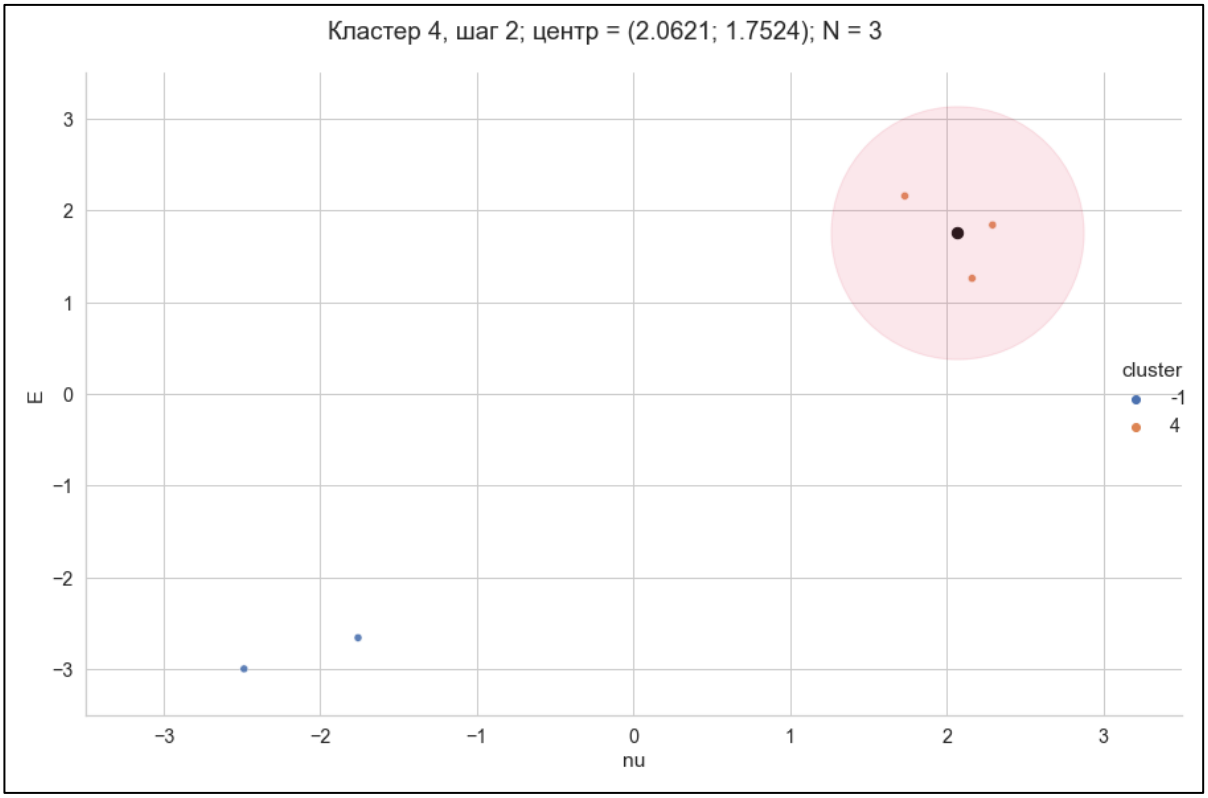

Рисунок 14

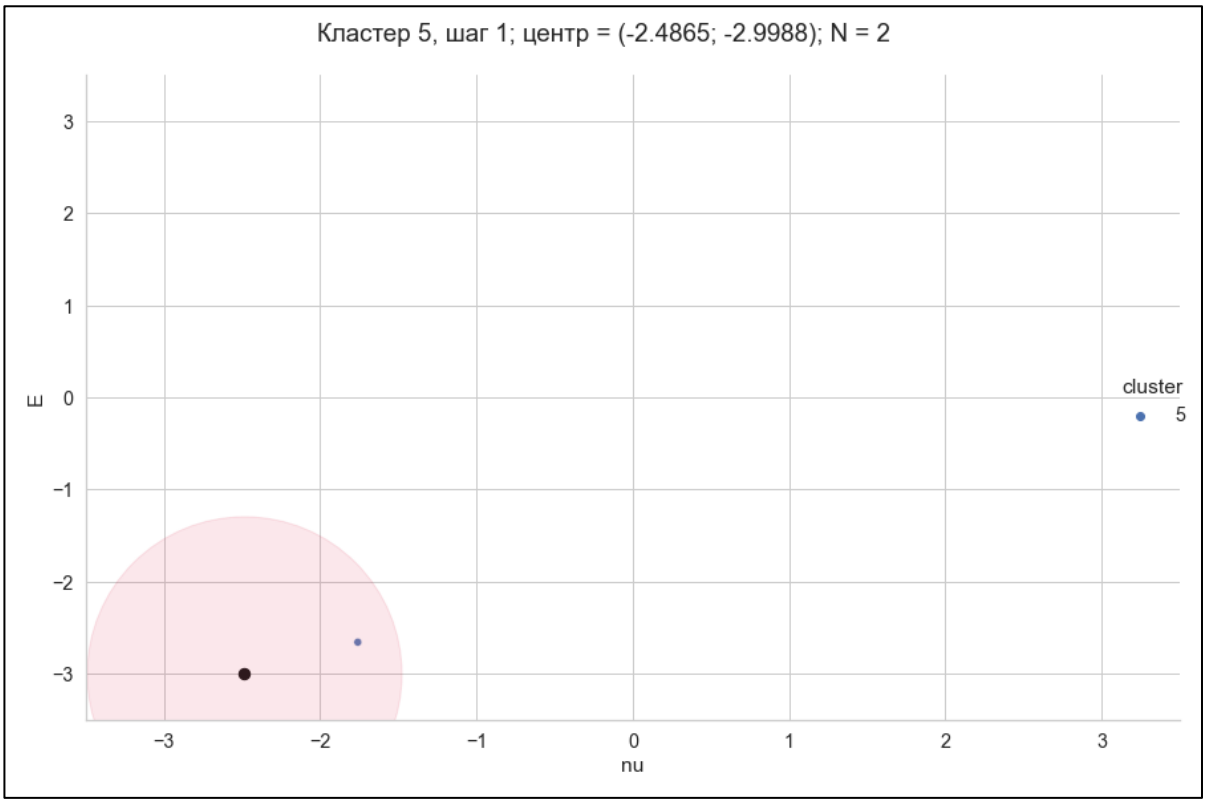

Рисунок 15

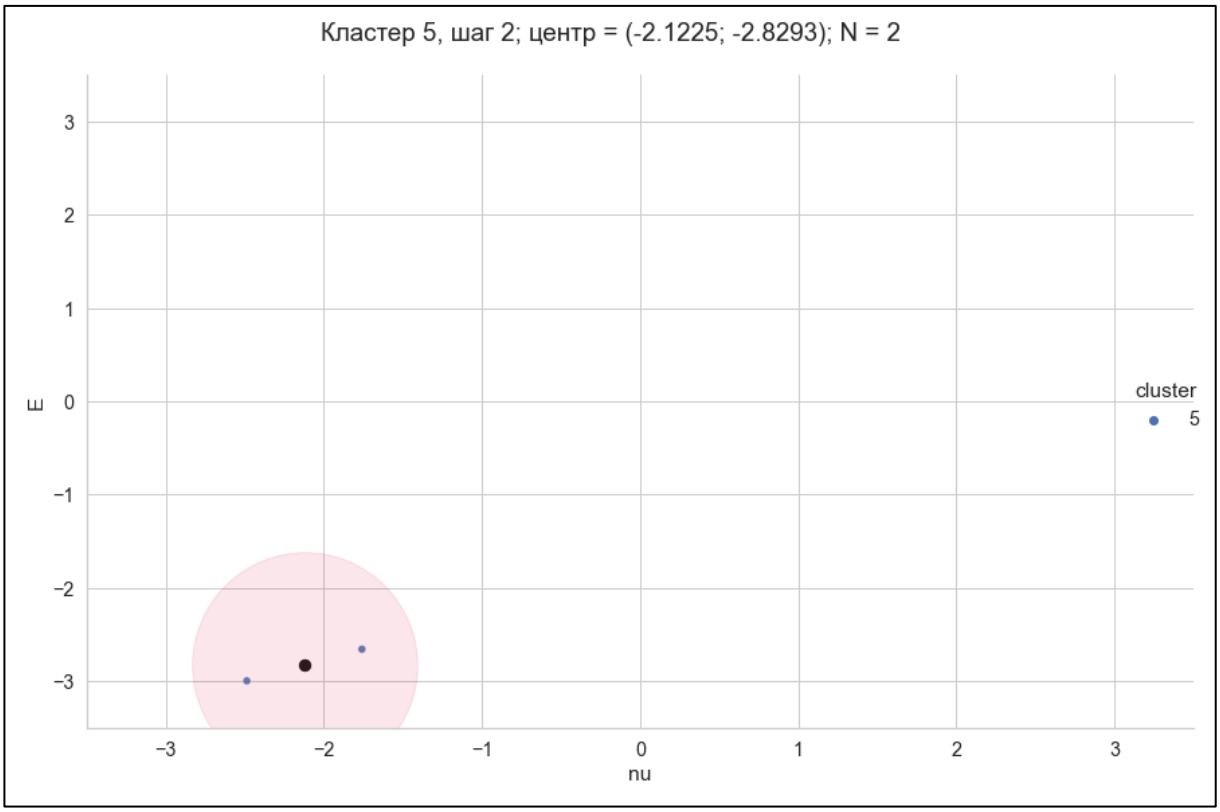

Рисунок 16

Результат кластеризации представлен на рис. 17.

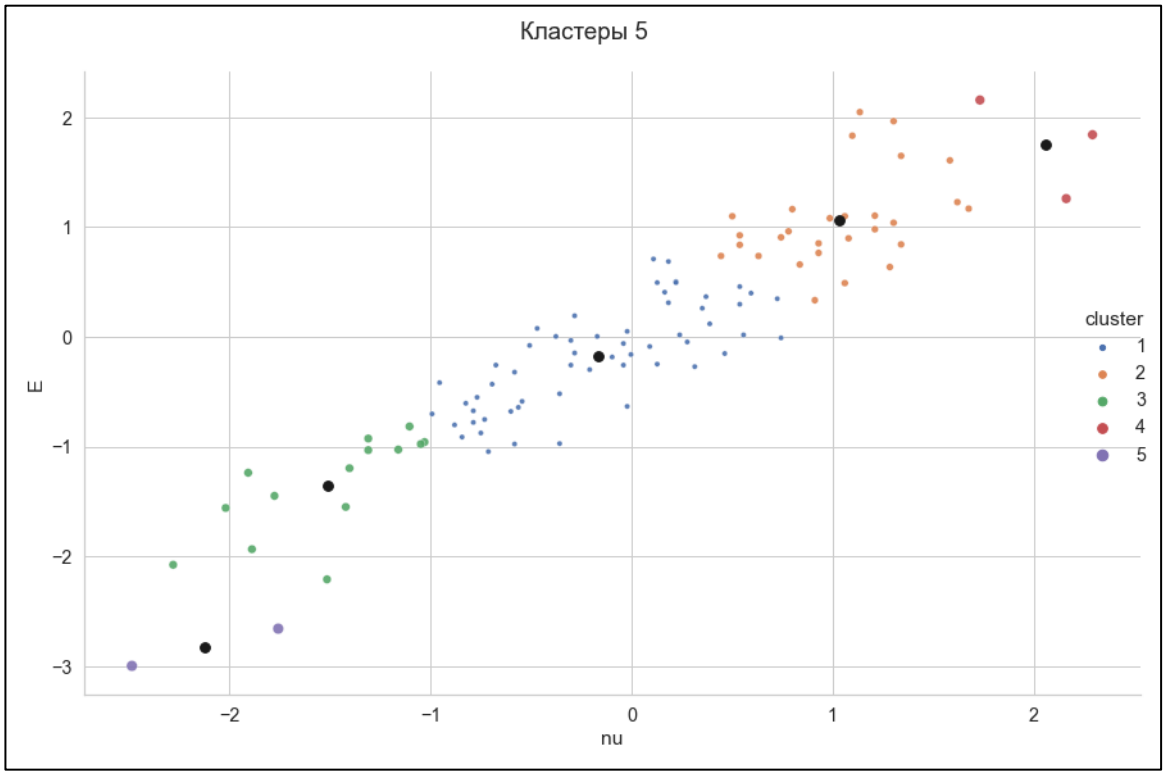

Рисунок 17

Чувствительность к погрешностям

Для проверки чувствительности метода к погрешностям было произведено сравнение значений функционалов качества для первоначального значения R=1.15 и для его изменений на $\delta = 0.02$. Функционалы F_1, F_2, F_3 определены из прошлой лабораторной работы. Значения представлены в таблице 1.

Таблица 1

Радиус	F_1	F_2	F_3
R	41.0773	1928.5299	1.6819
$R - \delta$	41.1502	1944.8672	1.6772
$R + \delta$	43.6907	2144.739	1.8447

На основании данных таблицы можно увидеть, что значения функционалов качества растут (хоть и немного) при изменении радиуса на небольшую дельту. Можно сделать вывод, что метод чувствителен к погрешностям.

Сравнение методов

Сравним метод k-means с методом поиска сгущений. Количество кластеров равно 5. Визуальное сравнение представлено на рис. 18 и 19.

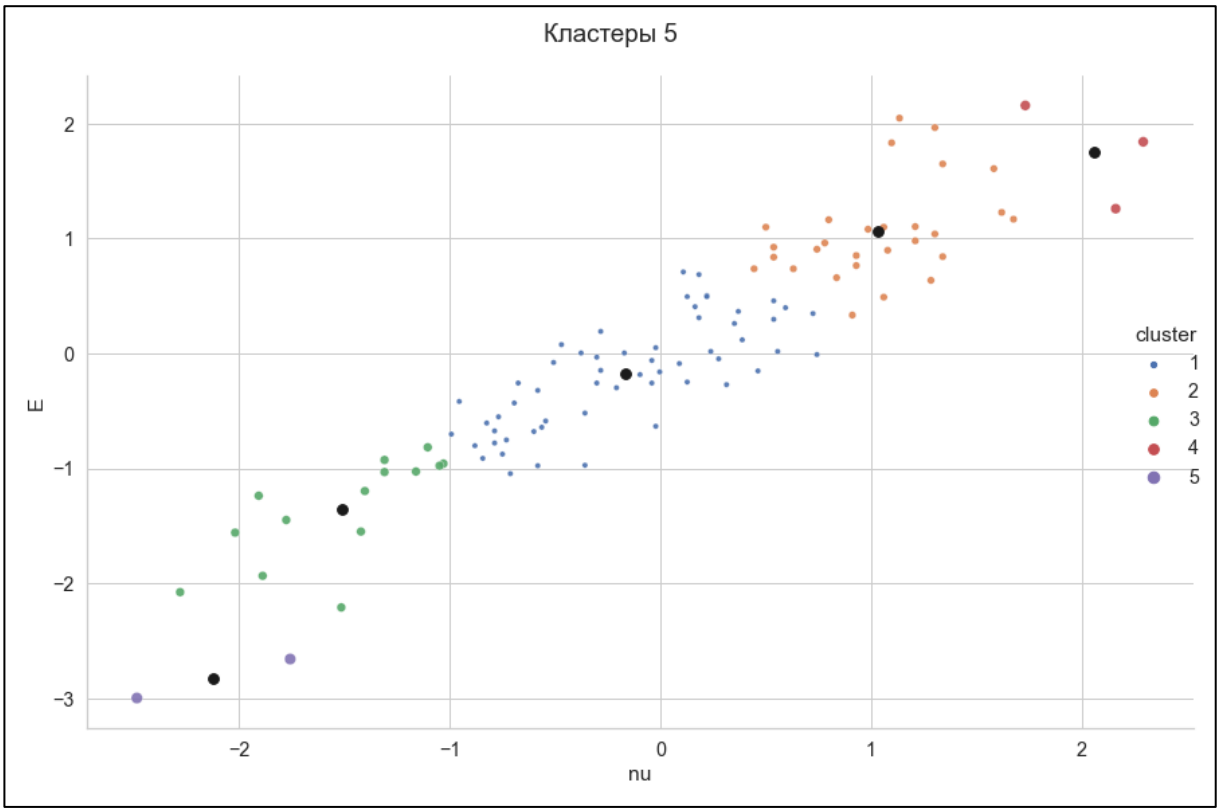

Рисунок 18 – Метод поиска сгущений

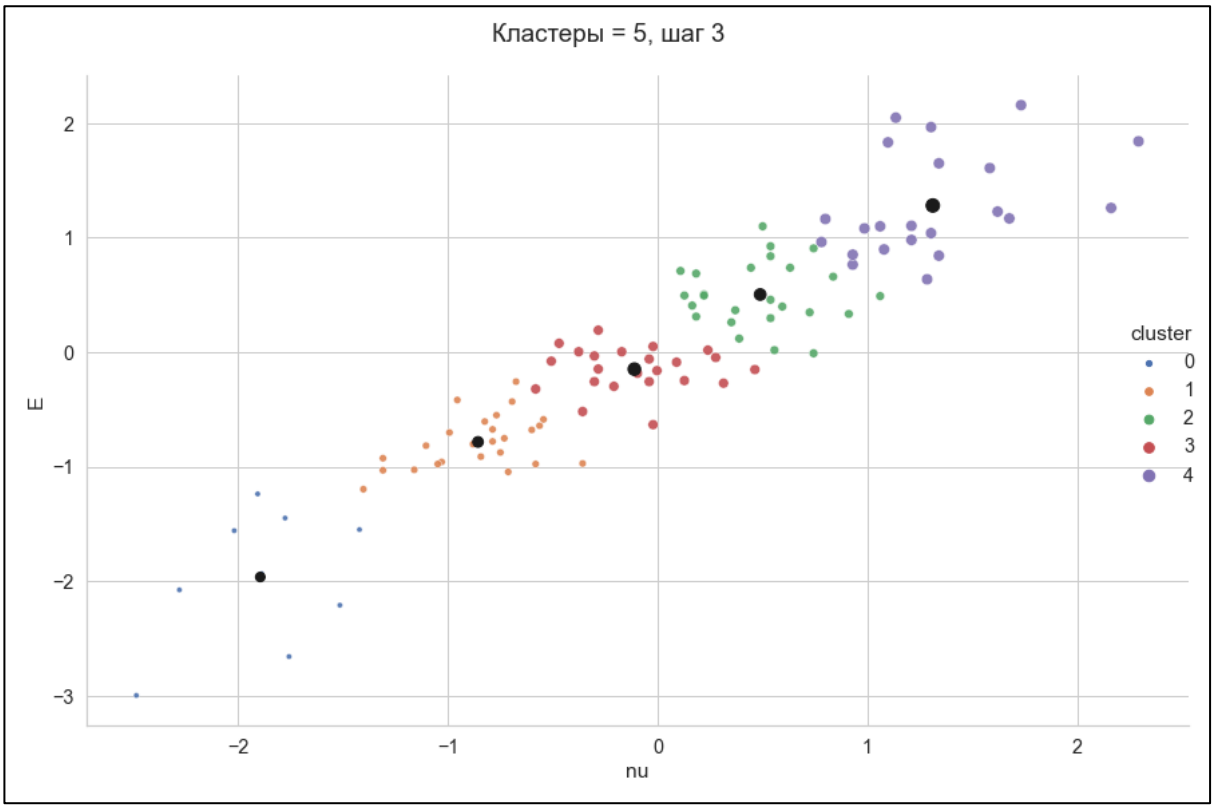

Рисунок 19 – Метод k-средних

Визуально можно увидеть, что в методе k-средних количество элементов в кластерах примерно одинаковое в отличие от метода поиска сгущений.

В таблице 2 приведены значения функционалов качества для методов.

Таблица 2

Метод	F_1	F_2	F_3
k-means	24.084	654.669	1.259
Поиск сгущений	41.0773	1928.5299	1.6819

Видно, что значения функционалов качества метода k-средних намного меньше, можно сделать вывод, что использование метода k-средних предпочтительнее.

Выводы

Освоены основные понятия кластерного анализа и метода поиска сгущений, в частности. Было нормализовано множество точек.

Были найдены границы радиуса сферы.

$$R_{min} = mind_{ij} = 0.0092$$

$$R_{max} = maxd_{ij} = 6.8015$$

Был реализован алгоритм поиска сгущений, с помощью которого выборка была разбита на 5 кластеров для $R = 1.15$. Кластеры были отображены, выделены цветом, отмечены центроиды.

Была проведена проверка чувствительности метода к погрешностям. На основании данных можно увидеть, что значения функционалов качества растут при изменении радиуса на дельту. Можно сделать вывод, что метод чувствителен к погрешностям.

Было проведено сравнение метода k-means с методом поиска сгущений. Визуально можно увидеть, что в методе k-средних количество элементов в кластерах примерно одинаковое в отличие от метода поиска сгущений. А также значения функционалов качества метода k-средних намного меньше, следовательно можно сделать вывод, что использование метода k-means предпочтительнее.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
from scipy.spatial import distance
import functools


# In[2]:


df0 = pd.read_csv('c:/Users/gandh/dev/unv/smoed/me/data/main_data.csv')
X = df0['nu']
Y = df0['E']


# In[3]:


sns.set_theme(style="whitegrid", palette='deep', context='notebook',
font_scale=1.3)
ax = sns.relplot(data=df0, x='nu', y='E', kind='scatter', height=8.27, as-
pect=11.7/8.27)
ax.set_axis_labels('nu', 'E')
ax.fig.suptitle('Двумерная выборка')
plt.tight_layout()
plt.savefig('pics/0.png')


# In[4]:


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


# In[5]:


sns.set_theme(style="whitegrid", palette='deep', context='notebook',
font_scale=1.3)
```

```
ax = sns.relplot(data=df, x='nu', y='E', kind='scatter', height=8.27, as-
pect=11.7/8.27)
ax.set_axis_labels('nu', 'E')
ax.fig.suptitle('Нормализованная выборка')
plt.tight_layout()
plt.savefig('pics/1.png')


# ## Алгоритм

# In[6]:


def sc_plots(data, center, R, step, itera):
    ax = sns.relplot(data=data, x='nu', y='E', hue='cluster', kind='scatter',
palette='deep',
                     alpha=0.9, height=8.27, aspect=11.7/8.27)
    for j in [center]:
        plt.scatter(j[0],j[1], c='k', s=70)
#     print(center.values[:2])
#     print(data[data['cluster']!=1]['nu'].count())

    circle = np.array([], dtype=np.float64)
    for i in data[data['cluster']!=-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*35000, alpha=0.1)

    ax.set_axis_labels('nu', 'E')
    ax.fig.suptitle(f'Кластер {itera}, шаг {step}; центр = ({center.val-
ues[0].round(4)}; {center.values[1].round(4)}); N = {data[data["cluster"]!=-
1]["nu"].count()}')
    ax.set(xlim=[-3.5,3.5], ylim=[-3.5,3.5])
    plt.tight_layout()
    plt.savefig(f'pics/{itera}_{step}.png')
    plt.show()


# In[7]:


def Fs(data):
    curr_data = data.copy()
    cts = curr_data.groupby('cluster').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['cluster']==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('cluster').var().values.sum(where=~np.is-
nan(curr_data.groupby('cluster').var().values), initial=0)

    return F1,F2,F3


# In[8]:


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['cluster'] = -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['cluster']==itera].mean()
        data['cluster'] = -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['cluster']==-1], data, np.array(center.values[:2])


# ## Основа

# In[9]:


coords = df.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(4), rmax.round(4)


# In[22]:


upd_df = df.copy()
it = 1
radius = 1.13
df['cluster'] = -1
ctrs = np.array([], dtype=np.float64)


# In[23]:


while len(upd_df):
    upd_df, main, ctr = custFE(upd_df, radius, it, 2)
    ctrs = np.append(ctrs, [ctr])
    it += 1
    df.loc[main[main['cluster']!=-1].index, :] = main.loc[main[main['clus-
ter']!=-1].index, :]
df.to_csv('data/result.csv', index=False)


# ### Финальное разбиение

# In[24]:


F1, F2, F3 = Fs(df)
F1, F2, F3


# In[18]:


F1, F2, F3 = Fs(df)
F1, F2, F3
```

```
# In[21]:


F1, F2, F3 = Fs(df)
F1, F2, F3


# In[15]:


ax = sns.relplot(data=df, x='nu', y='E', hue='cluster', kind='scatter', pal-
ette='deep', alpha=0.9,
                 size='cluster', height=8.27, aspect=11.7/8.27)
ctrs = ctrs.reshape((-1,2))
for i in ctrs:
    plt.scatter(i[0], i[1], c='k', s=60)
ax.set_axis_labels('nu', 'E')
ax.fig.suptitle(f'Кластеры {len(df["cluster"].unique())}')
plt.tight_layout()
plt.savefig('pics/result.png')
plt.show()
```