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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Статистические методы обработки экспериментальных данных»

Тема: Кластерный анализ. Метод k-средних.

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 k-means.

Основные теоретические положения

Задача кластерного анализа заключается в том, чтобы на основании данных, характеризующих исследуемые объекты, разбить множество объектов G на m кластеров (подмножеств G) $G_1, G_2, \dots, G_m$ таких, что:

$$G_1 \subset G; G_2 \subset G; \dots; G_m \subset G$$

$$G_1 \cup G_2 \cup \dots \cup G_m = G$$

$$G_i \cap G_j = \emptyset \; \forall i \neq j$$

К характеристикам кластера относятся:

- Центр кластера – это среднее геометрическое место точек, принадлежащих кластеру, в пространстве данных.
- Радиус кластера – максимальное расстояние точек, принадлежащих кластеру, от центра кластера.
- Кластеры могут быть перекрывающимися. В этом случае невозможно при помощи используемых процедур однозначно отнести объект к одному из двух или более кластеров. Такие объекты называют спорными.
- Спорный объект – это объект, который по мере сходства может быть отнесен к более, чем одному кластеру.
- Размер кластера может быть определен либо по радиусу кластера, либо по среднеквадратичному отклонению объектов для этого кластера. Объект относится к кластеру, если расстояние от объекта до центра кластера меньше радиуса кластера. Если это условие выполняется для двух и более кластеров, объект является спорным.

Большое значение в кластерном анализе имеет выбор масштаба. Обычно требуется нормировка переменных. Существуют различные способы нормировки данных:

$$z = \frac{(x - \bar{x})}{\sigma}; z = \frac{x}{\bar{x}}; z = \frac{x}{x_{max}}; z = \frac{(x - \bar{x})}{x_{max} - x_{min}}$$

Расстоянием (метрикой) между объектами a и b пространстве параметров называется такая величина d_{ab}, которая удовлетворяет аксиомам:

1. $d_{ab} > 0$, *если* $a \neq b$, 2. $d_{ab} = 0$, *если* $a = b$;
3. $d_{ab} = d_{ba}$; 4. $d_{ab} + d_{bc} \geq d_{ac}$.

Мерой близости (сходства) называется величина μ_{ab}, имеющая предел и возрастающая с возрастанием близости объектов и удовлетворяющая условиям:

$$\mu_{ab} \text{ непрерывна; } \mu_{ab} = \mu_{ba}; 0 \leq \mu_{ab} \leq 1.$$

Существует возможность простого перехода от расстояния к мерам близости:

$$\mu = \frac{1}{1 + d}.$$

Суть метода k-средних заключается в том, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} \left(x_j - \bar{x}\right)^2$$

Центроиды выбираются в тех местах, где визуально скопление точек выше. Алгоритм разбивает множество элементов векторного пространства на заранее известное число кластеров k. Основная идея заключается в том, что на каждой итерации пересчитывается центр масс для каждого кластера, полученного на предыдущем шаге, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на какой-то итерации не происходит изменения центра масс кластеров.

Возможны две разновидности метода k -средних.

Первая предполагает пересчет центра кластера после каждого изменения его состава, а вторая – лишь после завершения цикла.

Перед началом работы метода целесообразно нормировать характеристики объектов:

$$\widehat{X} = \frac{x - \bar{x}_в}{S_x}; \widehat{Y} = \frac{y - \bar{y}_в}{S_y}$$

Задание количества кластеров является сложным вопросом. Если нет разумных соображений на этот счет, рекомендуется первоначально создать 2 кластера, затем 3, 4, 5 и так далее, сравнивая полученные результаты.

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.

В качестве таких функционалов могут быть использованы:

1. Сумма квадратов расстояний до центров кластеров

$$F_1 = \sum_{k=1}^{K} \sum_{i=1}^{N_k} d^2(X_i^{(k)}, X^{(k)}) \Rightarrow \min$$

2. Сумма внутрикластерных расстояний между объектами

$$F_2 = \sum_{k=1}^{K} \sum_{X_i, X_j \in S_k} d^2(X_i, X_j) \Rightarrow \min$$

3. Сумма внутрикластерных дисперсий

$$F_3 = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \sigma_{ij}^2 \Rightarrow \min$$

Здесь σ - дисперсия *j*-й переменной в *k*-м кластере.

Оптимальным следует считать разбиение, при котором сумма внутрикластерных (внутригрупповых) дисперсий будет минимальной.

Судить о качестве разбиения позволяют и некоторые простейшие приемы. Например, можно сравнивать средние значения признаков в отдельных кластерах (группах) со средними значениями в целом по всей совокупности объектов. Если групповые средние существенно отличаются от общего среднего значения, то это может являться признаком хорошего разбиения.

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 k-means. Полученные результаты содержательно проинтерпретировать.

Порядок выполнения работы

1. Нормализовать множество точек, отобразить полученное множество.
2. Определить верхнюю оценку количества кластеров по формуле: $\bar{k} = \lfloor\sqrt{N/2}\rfloor$, где N – число точек.
3. Реализовать алгоритм k-means в двух вариантах:
    - Пересчет центра кластера осуществляется после каждого изменения его состава
    - Пересчет центра кластера осуществляется после того, как будет завершен шаг процедуры
4. На каждом шаге процедуры разбиения методом k-means вычислять функционалы качества полученного разбиения:
    - F_1 – сумма по всем кластерам квадратов расстояний элементов кластеров до центров соответствующих кластеров
    - F_2 – сумма по всем кластерам внутрикластерных расстояний между элементами кластера
    - F_3 – сумма по всем кластерам внутрикластерных дисперсий (относительно центров кластеров)
5. Отобразить полученные кластеры, выделить каждый кластер разным цветом, отметить центроиды.
6. Содержательно проинтерпретировать полученные результаты.

Выполнение работы

Нормировка данных

Первоначальная выборка представлена на рис. 1.

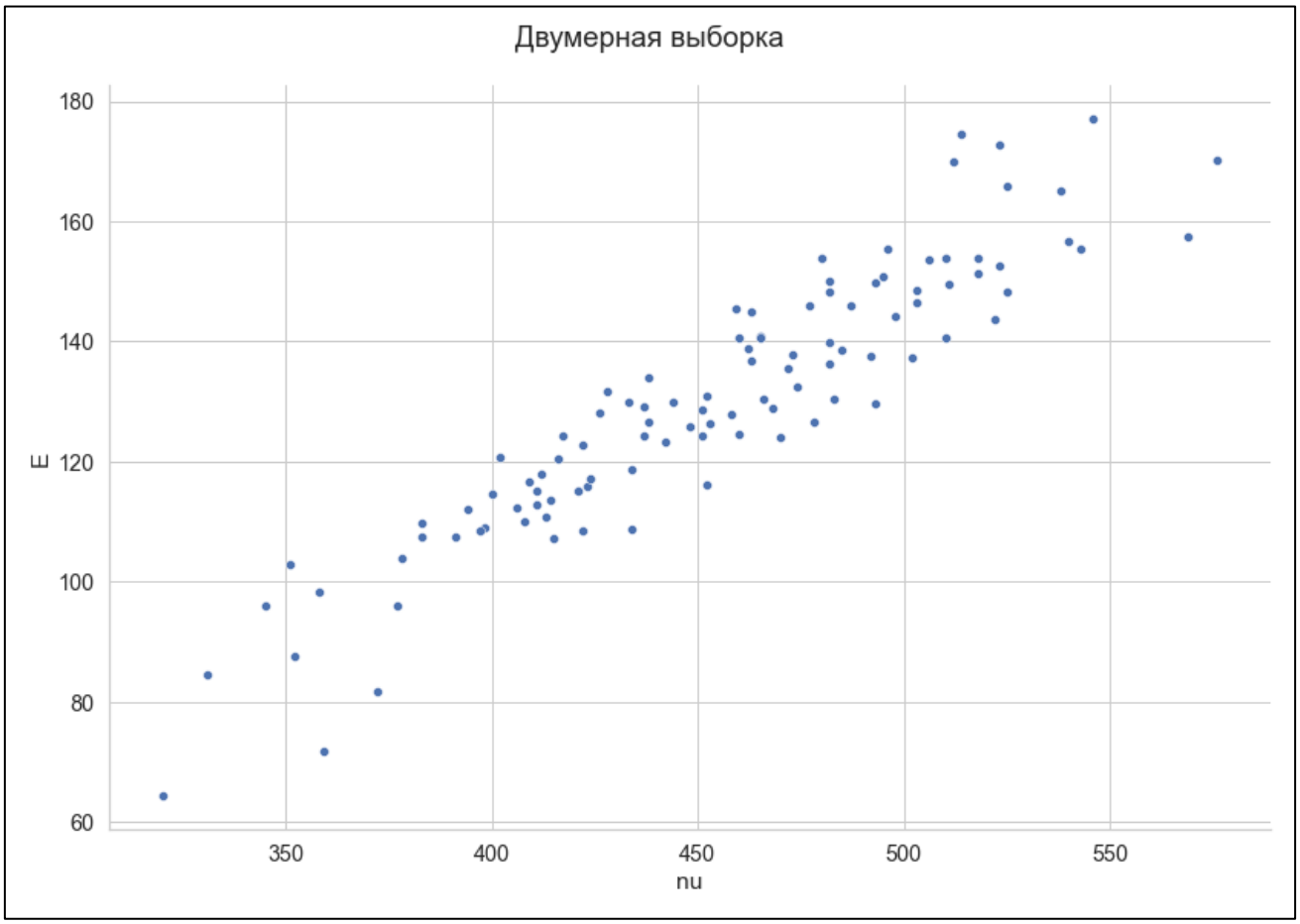

Рисунок 1 – Первоначальная выборка

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.

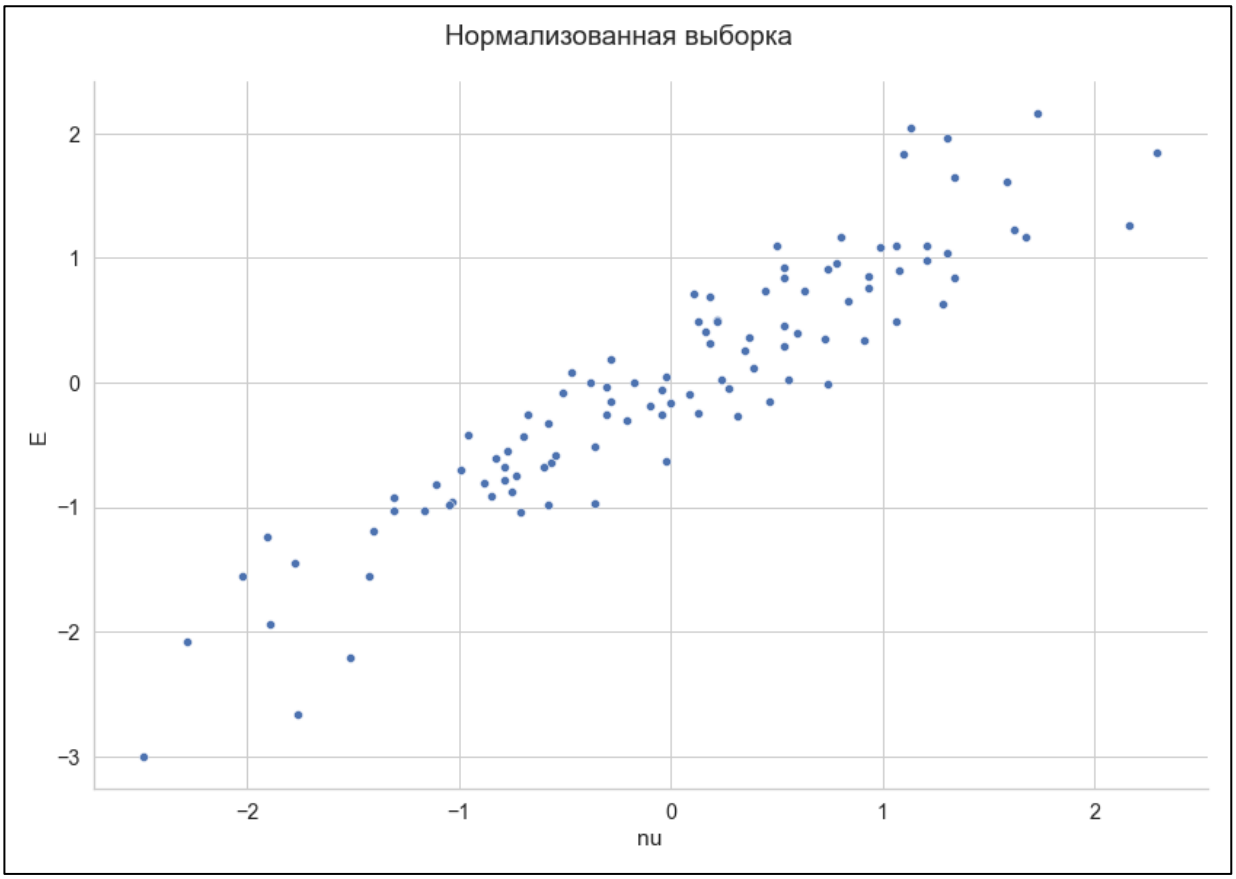

Рисунок 2 – Нормализованная выборка

Грубая верхняя оценка количества кластеров:

$$\bar{k} = \lfloor \sqrt{N/2} \rfloor = \lfloor \sqrt{104/2} \rfloor = 7$$

Алгоритм k-means

- Пересчет центра после шага процедуры

Реализован алгоритм k-means, где пересчет центра кластера осуществляется после шага процедуры (после просмотра всех данных). Количество кластеров от 2 до 7. Полученные кластеры были отображены, выделены свои цветом, были отмечены центроиды. На каждом шаге процедуры вычислены функционалы качества полученного разбиения:

- F_1 – сумма по всем кластерам квадратов расстояний элементов кластеров до центров соответствующих кластеров
- F_2 – сумма по всем кластерам внутрикластерных расстояний между элементами кластера
- F_3 – сумма по всем кластерам внутрикластерных дисперсий

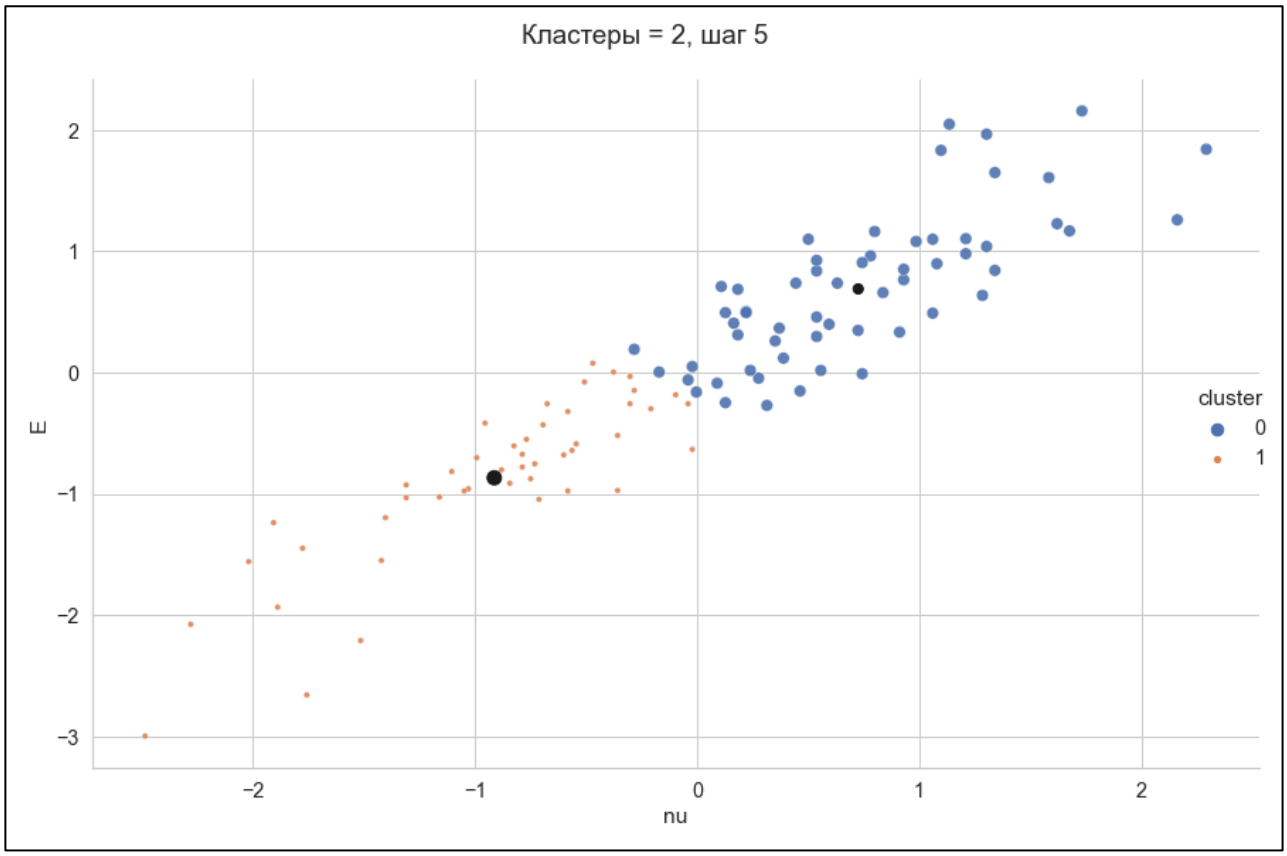

Рисунок 3 – Кластеризация методом k-средних (2 кластера)

Таблица 1

Центр кластера	Количество элементов
(0.7249; 0.6889)	58
(-0.914; -0.8687)	46

Таблица 2

F_1	F_2	F_3
79.619	4555.727	1.534
78.349	4307.859	1.531
77.567	4134.173	1.53
76.855	4016.464	1.52
76.855	4016.464	1.52

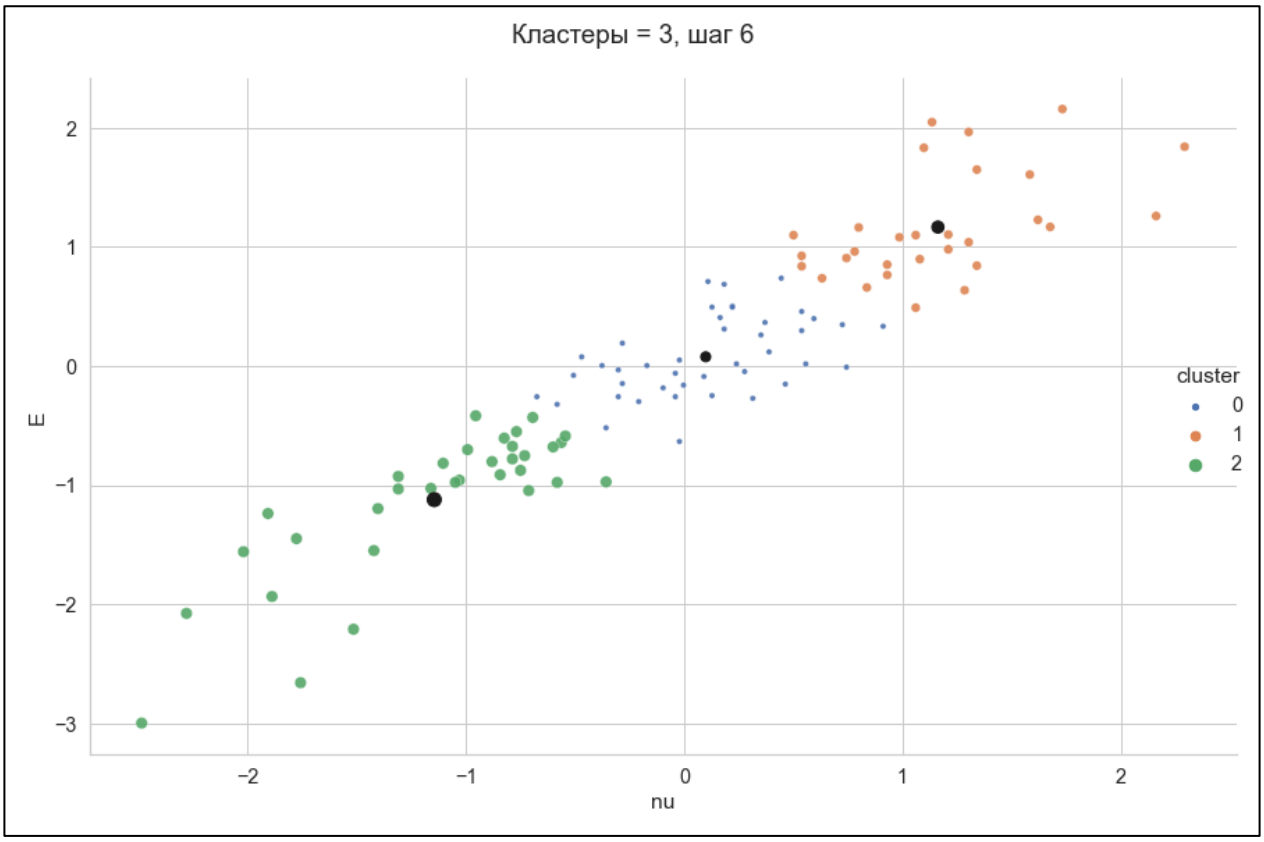

Рисунок 4 – Кластеризация методом k-средних (3 кластера)

Таблица 3

Центр кластера	Количество элементов
(0.098; 0.0769)	42
(1.162; 1.166)	29
(-1.1459; -1.1225)	33

Таблица 4

F_1	F_2	F_3
51.075	2185.045	1.466
46.297	1719.227	1.385
45.085	1599.229	1.356
44.544	1550.091	1.351
43.857	1497.05	1.349
43.857	1497.05	1.349

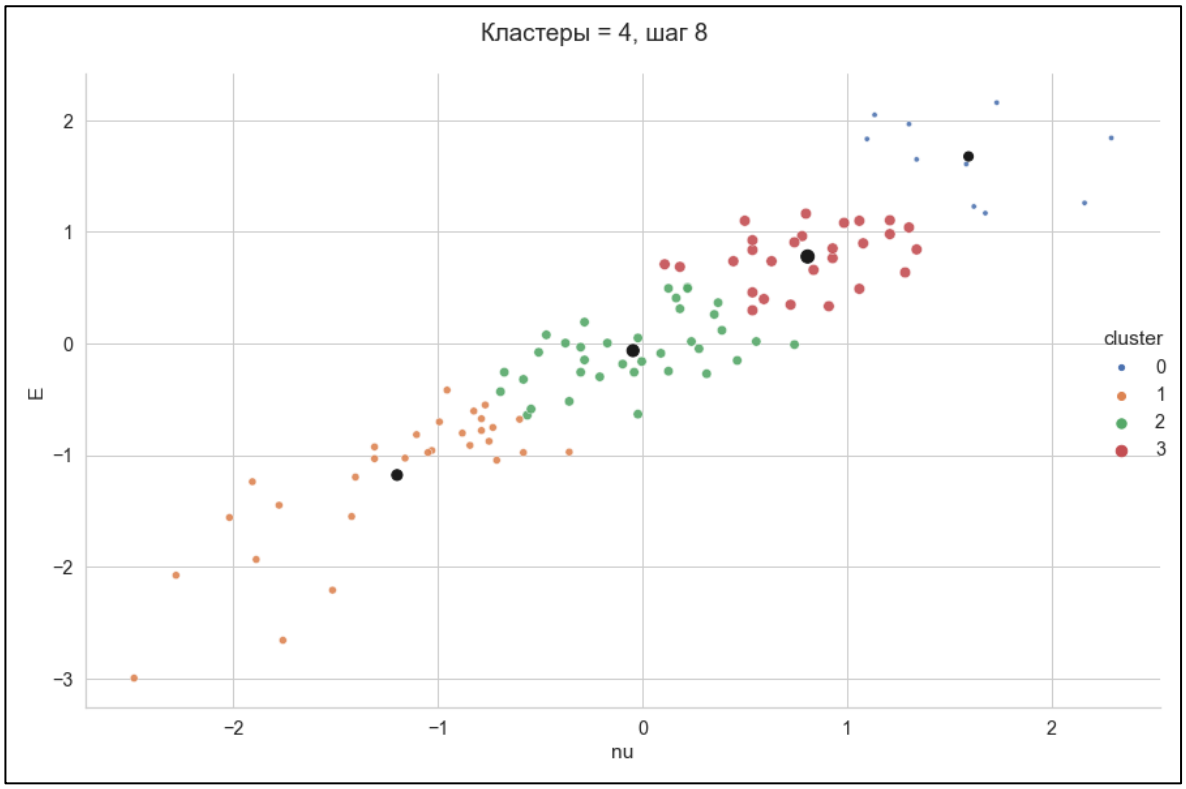

Рисунок 5 – Кластеризация методом k-средних (4 кластера)

Таблица 5

Центр кластера	Количество элементов
(1.5941; 1.6751)	10
(-1.2003; -1.1793)	30
(-0.0467; -0.0648)	37
(0.8072; 0.7787)	27

Таблица 6

F_1	F_2	F_3
61.594	2888.433	1.784
54.657	2272.014	1.688
46.384	1750.805	1.48
37.374	1186.478	1.354
35.815	1085.379	1.357
35.551	1058.432	1.379
35.379	1053.881	1.381
35.379	1053.881	1.381

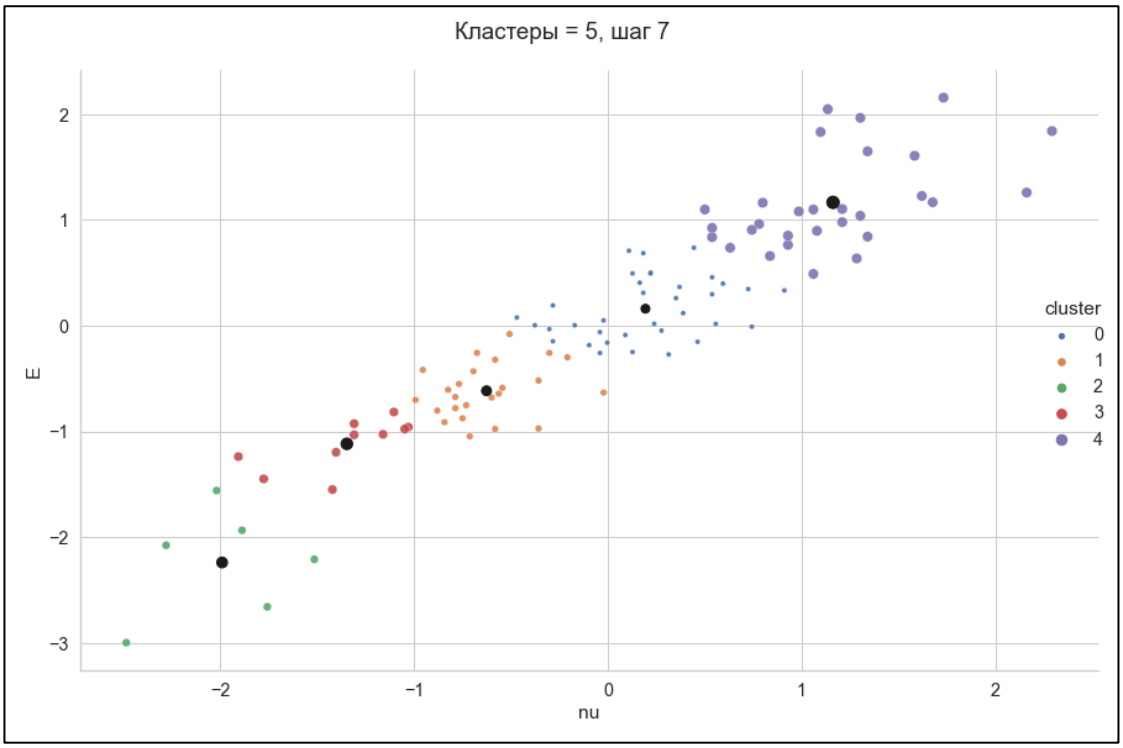

Рисунок 6 – Кластеризация методом k-средних (5 кластеров)

Таблица 7

Центр кластера	Количество элементов
(0.1936; 0.16)	35
(-0.6268; -0.6164)	24
(-1.9918; -2.2403)	6
(-1.3478; -1.1179)	10
(1.162; 1.166)	29

Таблица 8

F_1	F_2	F_3
32.94	1326.678	1.284
27.323	859.938	1.269
25.084	733.119	1.244
24.407	681.996	1.254
24.16	664.971	1.257
24.084	654.669	1.259
24.084	654.669	1.259

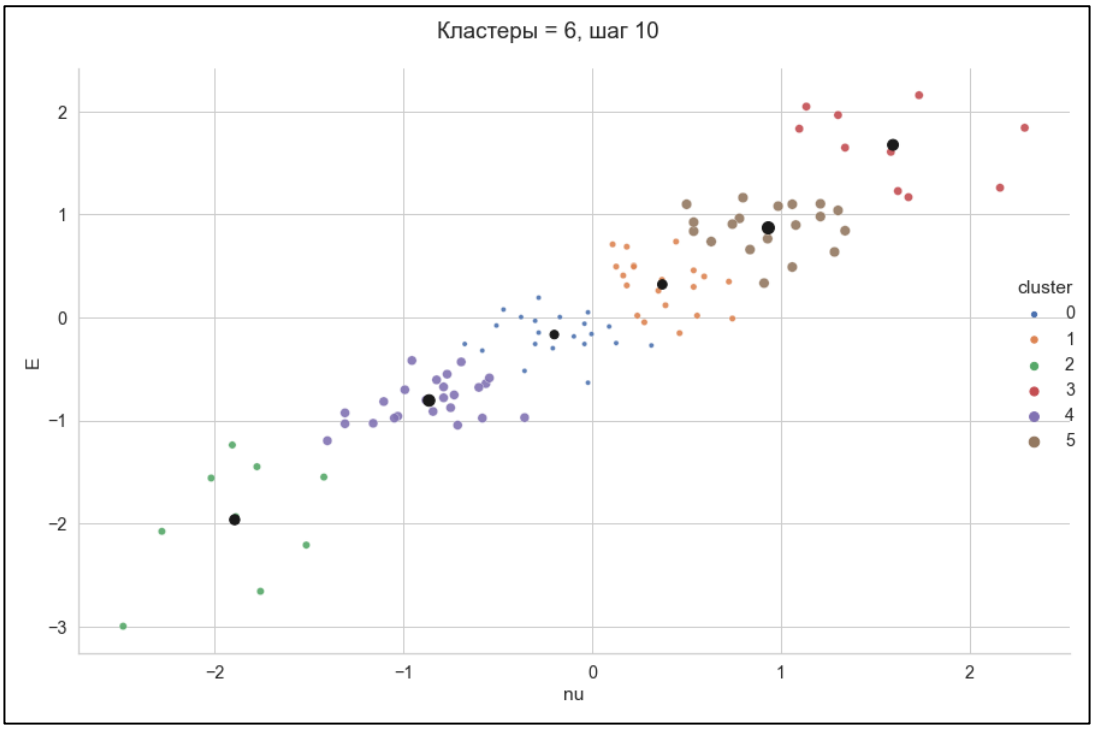

Рисунок 7 – Кластеризация методом k-средних (6 кластеров)

Таблица 9

Центр кластера	Количество элементов
(-0.2011; -0.1669)	21
(0.3714; 0.3201)	20
(-1.8954; -1.9646)	9
(1.5941; 1.6751)	10
(-0.8648; -0.8068)	24
(0.9333; 0.8698)	20

Таблица 10

F_1	F_2	F_3
44.763	1673.801	1.425
32.729	920.885	1.354
29.655	692.789	1.415
25.679	497.495	1.433
20.715	342.549	1.346
17.233	269.68	1.276
15.541	253.261	1.201
15.174	251.847	1.183
15.096	248.091	1.182
15.096	248.091	1.182

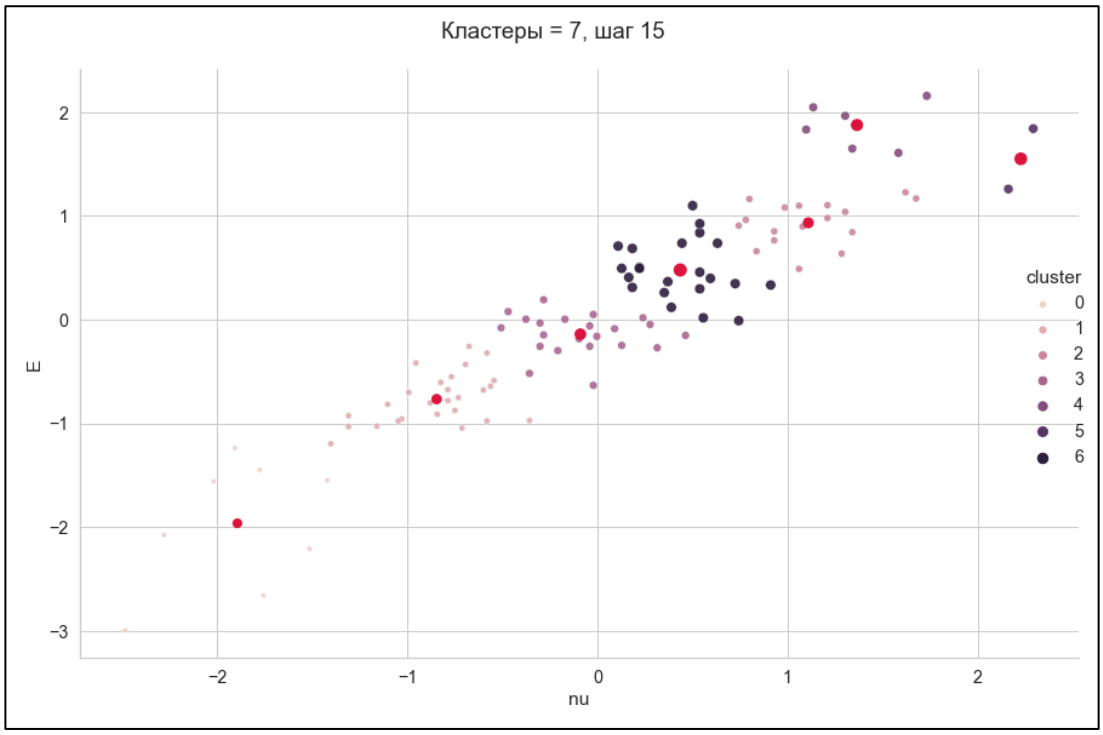

Рисунок 8 – Кластеризация методом k-средних (7 кластеров)

Таблица 11

Центр кластера	Количество элементов
(-1.8954; -1.9646)	9
(-0.8467; -0.767)	26
(1.1086; 0.933)	17
(-0.0903; -0.1412)	22
(1.3652; 1.8761)	6
(2.227; 1.5499)	2
(0.4349; 0.4779)	22

Таблица 12

F_1	F_2	F_3
29.274	768.748	1.488
25.75	607.845	1.437
23.891	518.596	1.451
21.365	453.482	1.428
19.513	416.434	1.391
...		
14.481	261.669	1.231

В таблице 13 представлены значения функционалов качества на последней итерации и количество итераций для различных значений количества кластеров.

Таблица 13

Количество кластеров	***Количество итераций***	F_1	F_2	F_3
2	5	76.855	4016.464	1.52
3	6	43.857	1497.05	1.349
4	8	35.379	1053.881	1.381
5	7	24.084	654.669	1.259
6	10	15.096	248.091	1.182
7	15	14.481	261.669	1.231

Можно заметить, что при увеличении количества кластеров увеличивается число итераций и минимизируются значения функционалов качества.

- Пересчет центра после каждого изменения состава кластера

При реализации алгоритма в случае изменения центра кластера после каждого изменения его состава были получены следующие значения количества итераций:

Таблица 13

Количество кластеров	***Количество итераций (центр меняется в конце итерации)***	***Количество итераций (центр меняется после каждого объекта)***
2	5	2
3	6	2
4	8	2
5	7	3
6	10	4
7	15	4

Из таблицы можно увидеть, что количество итераций второй версии алгоритма меньше, чем первой, так как центр корректируется больше раз, что лучше минимизирует функционал качества.

Найдем оптимальное количество кластеров:

1. С помощью метода локтя

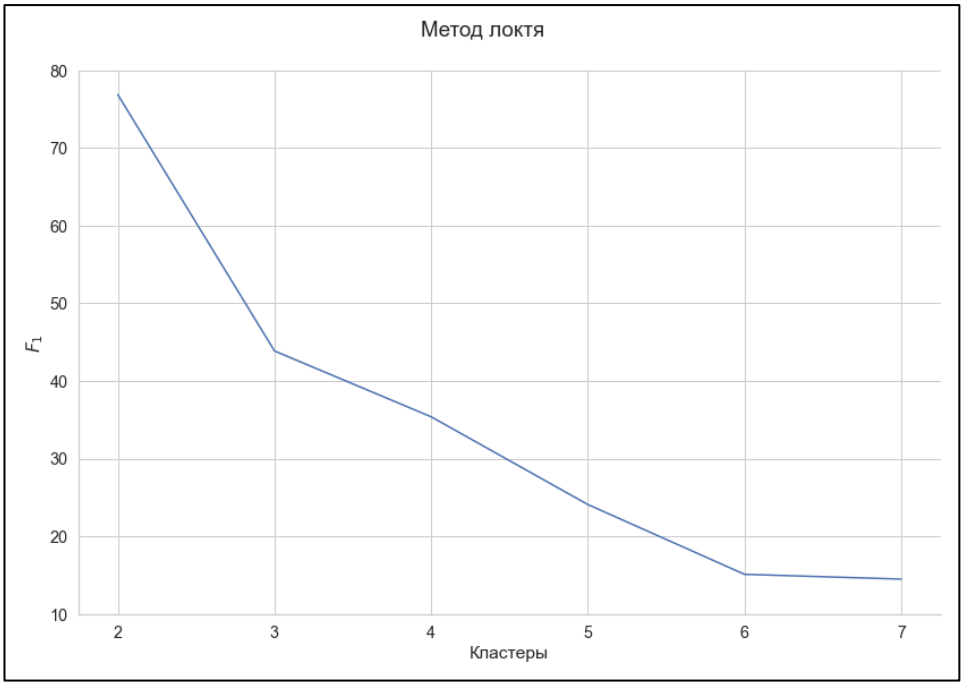

Рисунок 9 – Метод локтя

Значение числа кластеров можно получить в точке сгиба, после которой значение функционала качества изменяется медленно. Исходя из рис. 9 можно предположить, что оптимальное число кластеров – 6.

2. С помощью метода силуэтов

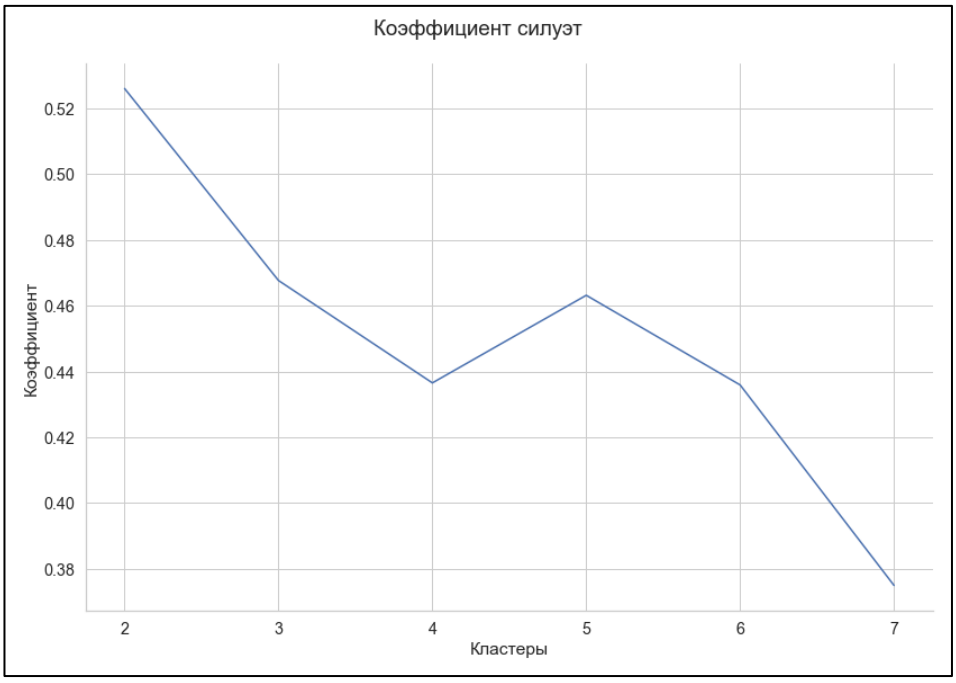

Рисунок 10 – Метод силуэтов

Коэффициент «силуэт» вычисляется с помощью среднего внутрикластерного расстояния и среднего расстояния до ближайшего кластера по каждому образцу. Можно вычислить среднее значение силуэта по всем образцам и использовать его как метрику для оценки количества кластеров.

График силуэта имеет пиковый характер, в отличие от мягко изогнутого графика при использовании метода локтя. Исходя из рис. 10 можно предположить, что оптимальное количество кластеров – 2 или 5.

Выводы

В ходе выполнения лабораторной работы были освоены основные понятия кластерного анализа, в частности, метода k-средних. Исходная двумерная выборка была нормализована и отображена на рисунке. Была определена грубая верхняя оценка количества кластеров $\bar{k} = 7$.

Реализован алгоритм k-means в двух вариантах: пересчет центра осуществляется после каждого изменения состава кластера, либо же после просмотра всех данных. Первый вариант имеет меньшее число шагов процедуры, так как центр корректируется больше раз, что лучше минимизирует функционал качества.

Разбиение проводилось для разного количества кластеров, от 2 до 7. Можно заметить, что при увеличении количества кластеров увеличивается число итераций алгоритма и минимизируются значения функционалов качества.

Было найдено оптимальное значение количества кластеров с помощью метода локтя и метода силуэтов. В первом случае значение равно шести, во втором же пяти.

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 sklearn.cluster import KMeans
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
from scipy.spatial import distance
from sklearn.metrics import silhouette_score


# 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,
aspect=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,
aspect=11.7/8.27)
ax.set_axis_labels('nu', 'E')
ax.fig.suptitle('Нормализованная выборка')
plt.tight_layout()
plt.savefig('pics/1.png')

# In[6]:

N = len(df)
sup_k = np.floor(np.sqrt(N/2))
sup_k

# ## Кластеры = 3

# In[7]:

clusterN = 3
k_means = KMeans(init='k-means++', n_clusters=clusterN, n_init=15)
k_means.fit(X_norm)
labels = k_means.labels_
df['cluster'] = labels
means = k_means.cluster_centers_

# In[8]:

pd.DataFrame(np.concatenate((means,
df.groupby('cluster')['nu'].count().values.reshape(-1,1)), axis=1),
             columns=['nu_mean', 'E_mean', 'num'])

# In[9]:

ax = sns.relplot(data=df, x='nu', y='E', hue='cluster', kind='scatter',
palette='deep', alpha=0.9,
                 size='cluster', height=8.27, aspect=11.7/8.27)
plt.scatter(means[:,0],means[:,1], c='crimson', s=70)

ax.set_axis_labels('nu', 'E')
ax.fig.suptitle('Кластеры 3')
plt.tight_layout()
# plt.savefig('pics/2.png')
df = df.drop('cluster', axis=1)
```

```
# ## Алгоритм

# In[10]:

def sc_plots(data, means, Ncl, step):
    if Ncl > 6:
        ax = sns.relplot(data=data, x='nu', y='E', hue='cluster',
kind='scatter', alpha=0.9,
                 size='cluster', height=8.27, aspect=11.7/8.27)
        plt.scatter(means[:,0],means[:,1], c='crimson',
s=np.linspace(50,100,Ncl))
    else:
        ax = sns.relplot(data=data, x='nu', y='E', hue='cluster',
kind='scatter', palette='deep', alpha=0.9,
                 size='cluster', height=8.27, aspect=11.7/8.27)
        plt.scatter(means[:,0],means[:,1], c='k', s=np.linspace(50,100,Ncl))

    ax.set_axis_labels('nu', 'E')
    ax.fig.suptitle(f'Кластеры = {Ncl}, шаг {step}')
    plt.tight_layout()
    plt.savefig(f'pics/2_{Ncl}.png')
    plt.close()

# In[11]:

def nearest_center(data, cts):
    distl = np.array([], dtype=np.float64)
    for i in cts:
        distl = np.append(distl, np.linalg.norm(i[:-1]-data)) # евклидово
расстояние
    min_dist = np.argmin(distl)
    return min_dist

# In[12]:

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.values[curr_data.iloc[i,2]])
        F1 += dist_F1**2

    # F2 - сумма кв. расст. до всех точек соотв. кластеров
    for i in range(len(cts)):
```

```
        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()

    return F1,F2,F3

# In[13]:

def custKM(dataf, n_clusters, chng_ctr=1, max_iter=30, tol=0.01):
    data = dataf.copy()
    centers = data.sample(n_clusters) # рандомные центры
    data['cluster'] = -1 # нет принадлежности кластерам
    cts = np.array([], dtype=np.float64)
    F1,F2,F3 = 0,0,0
    df_Fs = pd.DataFrame(columns=['F1', 'F2', 'F3'])

    for i in range(n_clusters):
        data.loc[centers.index[i],'cluster'] = i # кластеры для центров
        cts = np.append(cts, [data.loc[centers.index[i]].values])
    centers = cts.reshape((n_clusters,3))

    for j in range(max_iter):
        for i in range(len(data)): # ближ. центр для каждой точки
            curr_clust = nearest_center(data.iloc[i,:-1].values, centers)
            data.loc[i,'cluster'] = curr_clust # соотносим кластер
            if chng_ctr: # пересчет центра при новой точке
                centers[curr_clust][:2] =
data[data['cluster']==curr_clust].iloc[:,:2].mean()

        if chng_ctr == 0: # пересчет центра на каждой итерации
            for i in range(n_clusters):
                centers[i][:2] = data[data['cluster']==i].iloc[:,:2].mean()

        cur_F1,cur_F2,cur_F3 = Fs(data) # функционалы
        df_Fs = df_Fs.append({'F1':cur_F1,'F2':cur_F2,'F3':cur_F3},
ignore_index=True)

        if np.abs(F1-cur_F1) < tol:
            data['cluster'].astype('int')
            sc_plots(data, centers, n_clusters, j+1)
            break
        F1,F2,F3 = cur_F1,cur_F2,cur_F3
        data['cluster'] = -1

    df_ctrs = pd.DataFrame(np.concatenate((centers[:,:2],
```

```
data.groupby('cluster')['nu'].count().values.reshape(-1,1)), axis=1),
                            columns=['nu_mean', 'E_mean', 'num'])
    silhouette_avg = silhouette_score(data.values[:,:2], data.values[:,2])

    return df_Fs, df_ctrs, silhouette_avg

# In[71]:

sse = []
sils = []
for i in range(2,8):
    F, ctrs, sil = custKM(df, n_clusters=i, chng_ctr=1)
#     print(len(F))
    sse.append(F.iloc[-1,0])
    sils.append(sil)
    F.round(3).to_csv(f'data/Fs_{i}c.csv', index=False)
    ctrs.round(4).to_csv(f'data/centers_{i}c.csv', index=False)

# ### Локоть

# In[53]:

# sse = [76.855,43.857,35.379,24.084,15.096,14.481]
ax = sns.relplot(x=range(2,8), y=sse, kind='line', height=8.27,
aspect=11.7/8.27)
ax.set(ylim=[10,80])
ax.set_axis_labels('Кластеры', '$F_1$')
ax.fig.suptitle('Метод локтя')
plt.tight_layout()
plt.savefig(f'pics/elbow.png')
plt.show()

# ### Силуэт

# In[54]:

ax = sns.relplot(x=range(2,8), y=sils, kind='line', height=8.27,
aspect=11.7/8.27)
ax.set_axis_labels('Кластеры', 'Коэффициент')
ax.fig.suptitle('Коэффициент силуэт')
plt.tight_layout()
plt.savefig(f'pics/silhouette.png')
plt.show()
```