**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики - процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №5**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Кластеризация данных»**

**Студент гр. 23Б15-пу**

**Кубякин Н.А.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2025 г.**

***Оглавление***

[1. Цель работы 3](#_Toc178797423)

[2. Описание задачи (формализация задачи) 3](#_Toc178797424)

[3. Теоретическая часть 3](#_Toc178797431)

[4. Основные шаги программы 4](#_Toc178797432)

[5. Блок-схема программы 5](#_Toc178797433)

[6. Описание программы 6](#_Toc178797442)

[7. Рекомендации пользователя 7](#_Toc178797443)

[8. Рекомендации программиста 7](#_Toc178797444)

[9. Исходный код программы 7](#_Toc178797445)

[10. Контрольный пример 8](#_Toc178797447)

[11. Вывод 14](#_Toc178797448)

[12. Источники 15](#_Toc178797448)

[13. Листинг 16](#_Toc178797450)

# Цель работы

# Исследовать и сравнить качество кластеризации табличных данных до и после отбора признаков и обезличивания с применением алгоритма максиминного расстояния. Оценить влияние отбора признаков (по критерию разнесённости образов) и обезличивания на эффективность кластеризации с использованием евклидова расстояния в качестве метрики расстояния и индекса Rand для оценки качества кластеризации. Разработать программное обеспечение, визуализирующее кластеризацию и вычисляющее нужные значения метрик.

# Описание задачи (формализация задачи)

# Сформировать синтетический датасет с 15 столбцами.

# Выполнить кластеризацию по всем столбцам, вычислить индекс Rand.

# Отобрать информативные признаки и выполнить кластеризацию по ним.

# Обезличить датасет и снова выполнить кластеризацию.

# Сравнить эффективность кластеризации в этих трёх случаях.

# Сделать выводы.

# Теоретическая часть

1. Метод кластеризации: алгоритм максиминного расстояния

Идея Maximin-кластера — последовательно выбирать центры кластеров так, чтобы каждый новый центр был максимально удалён от уже выбранных.

Инициализация: первым центром берётся произвольная точка (например, индекс 0).

Шаг выбора i-го центра: для каждого образца вычисляется расстояние до ближайшего из уже выбранных центров, затем за новый центр принимается образец с максимальным расстоянием.

2. Метод выбора наиболее информативных признаков: Разнесённость образов в пространстве характеристик.

Разнесённость каждой координаты (признака) в пространстве образов оценивается через дисперсию. Среди всех признаков выбирают тот с наибольшей дисперсией.

3. Метод измерения расстояния: евклидово расстояние.

Евклидово расстояние между двумя точками – длина отрезка между ними.



4. Метод оценки качества кластеризации: индекс Rand.

Индекс Rand оценивает, насколько много из тех пар элементов, которые находились в одном классе, и тех пар элементов, которые находились в разных классах, сохранили это состояние после кластеризации алгоритмом. Имеет область определения от 0 до 1, где 1 — полное совпадение кластеров с заданными классами, а 0 — отсутствие совпадений.

# Основные шаги программы

1. Задать количество строк датасета, количество информативных признаков, количество кластеров

2. Выполнить кластеризацию на всех столбцах, оценить качество.

3. Выполнить кластеризацию на информативных столбцах, оценить качество.

4. Выполнить анонимизацию, оценить качество.

5. Сравнить качество анонимизации.

# Блок-схема программы

# На рисунке представлена блок-схема алгоритма.

# 

# Рис. 5 Блок-схема алгоритма.

# Описание программы

# Алгоритм реализован на языке Python 3.10 с использованием библиотек tkinter [[1]](#tk), numpy [[2]](#nx), matplotlib [[3]](#mt), pandas [[4]](#pd), scipy [[5]](#rn), sklearn [[6]](#sk). Логика и интерфейс программы реализованы в следующих функциях:

# Алгоритм и интерфейс реализованы через модуль cluster.py, содержащий следующие функции:

Таблица 6.1 Описание программы

|  |  |
| --- | --- |
| Имя функции | Назначение |
| generate\_dataset | Сгенерировать синтетический датасет |
| preprocess\_data | Выполняет предобработку данных |
| maximin\_clustering | Выполняет кластеризацию |
| select\_informative\_features | Отбирает информативные признаки |
| anonymize | Выполняет анонимизацию |
| cluster\_and\_plot | Создаёт визуализацию кластеров |
| process\_clustering | Основная функция программы |

# Рекомендации пользователя

1. Запустить cluster.py для запуска программы.
2. Задать размер датасета, кол-во кластеров и кол-во информативных признаков.
3. Нажать кнопку “Выполнить кластеризацию”.
4. В основном окне выведутся значения индекса Rand для каждого случая, отдельно откроются три окна с визуализацией кластеров.

# Рекомендации программиста

Для запуска программы необходима 64-битная операционная система Windows, Linux или macOS. Для работы с кодом необходима среда разработки, совместимая с python 3.1 и библиотеки tkinter [[1]](#tk), numpy [[2]](#nx), matplotlib [[3]](#mt), pandas [[4]](#pd), scipy [[5]](#rn), sklearn [[6]](#sk).

Для запуска программы необходимо установить Python версии не ниже 3.10, а так же библиотеки tkinter [[1]](#tk), numpy [[2]](#nx), matplotlib [[3]](#mt), pandas [[4]](#pd), scipy [[5]](#rn), sklearn [[6]](#sk).

# Исходный код программы

# Исходный код программы и необходимые текстовые файлы доступны по ссылке: <https://github.com/NikiTaku1/spbu_alg_2/tree/main/lab5>

# Контрольный пример

В данном разделе представлен пример, демонстрирующий работу программы.

1) Запуск cluster.py

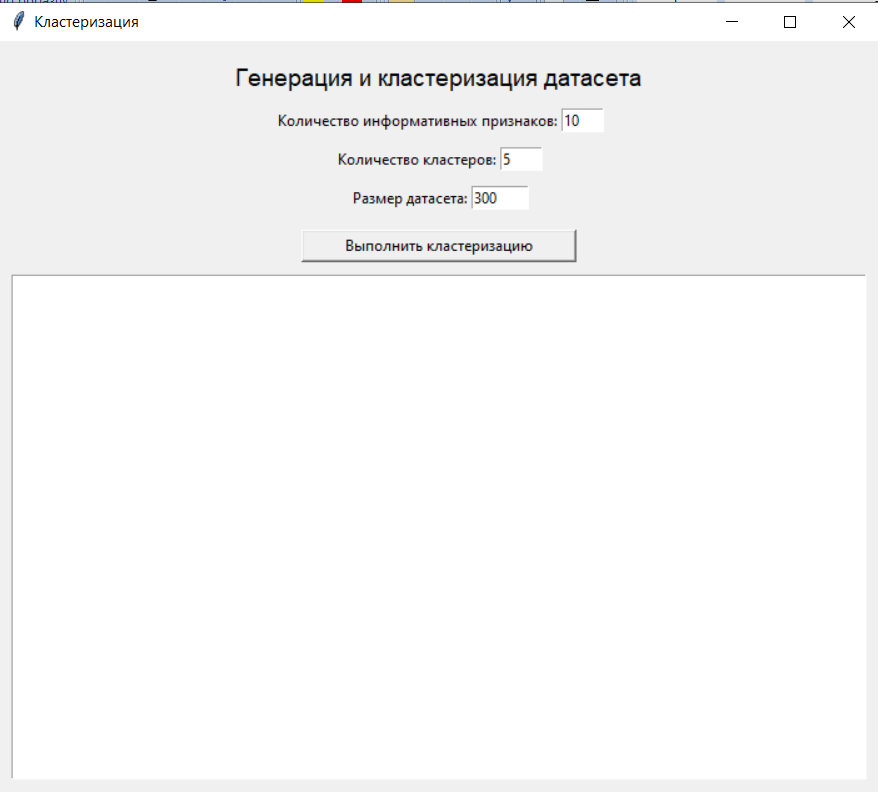


Рис 10.1 Графический интерфейс программы.

2) Основное окно

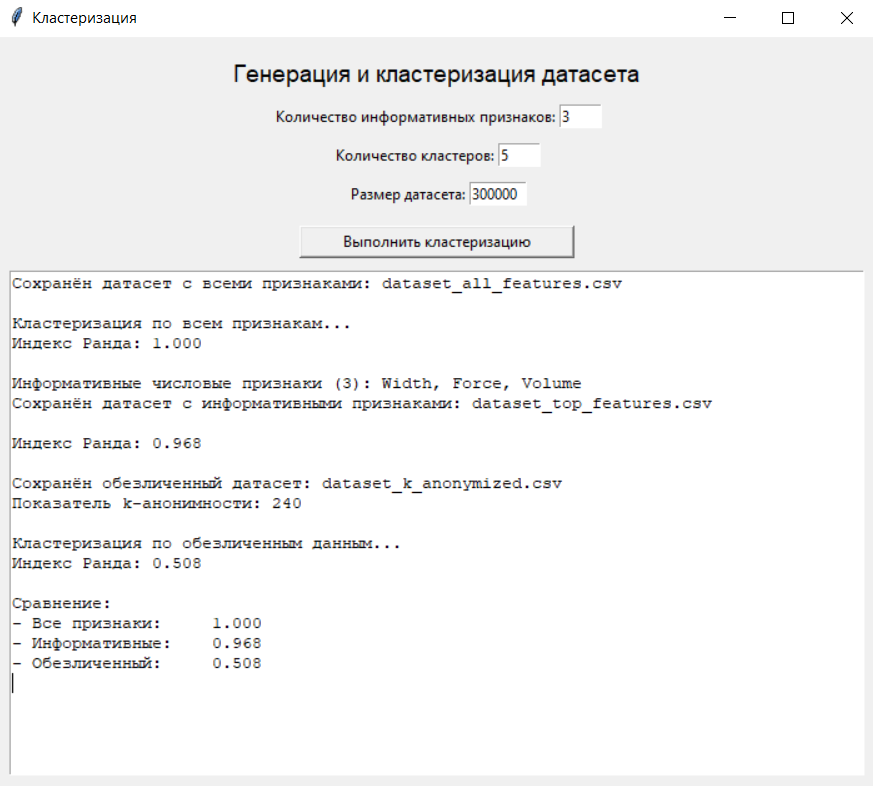


Рис 10.2 Основное окно после нажатия кнопки.

3) Кластеры по всем признакам

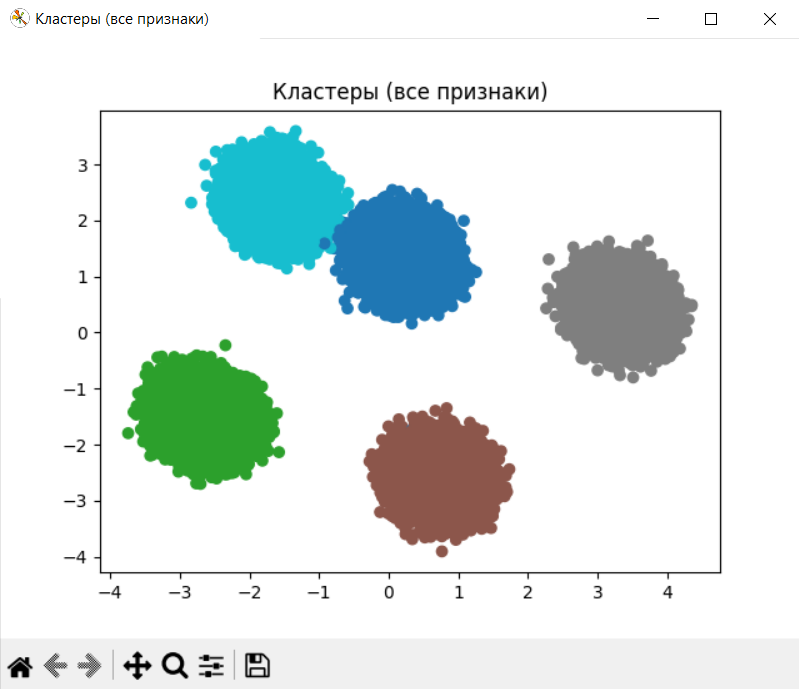


Рис 10.3 Кластеры по всем признакам.

4) Кластеры по информативным признакам

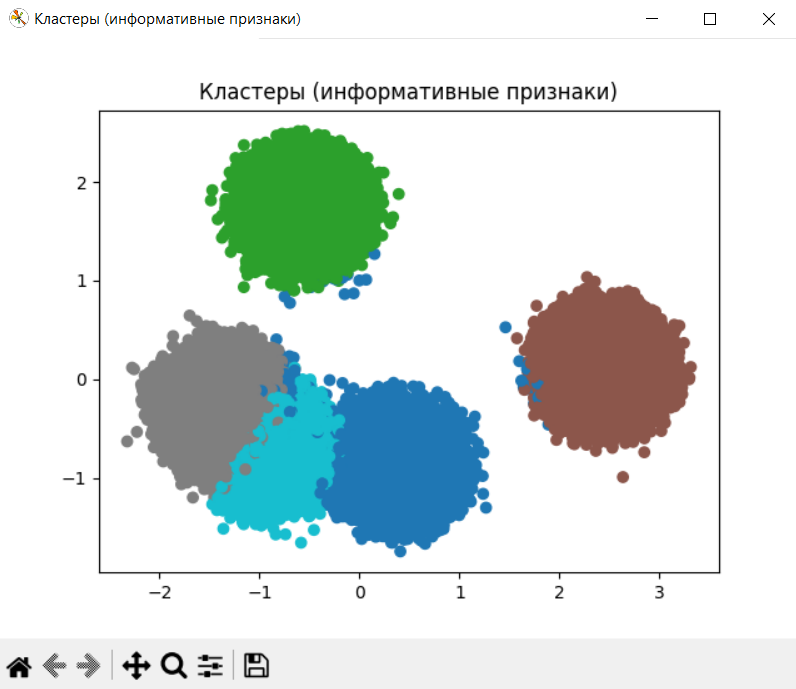


Рис 10.4 Кластеры по информативным признакам.

5) Кластеры по обезличенному датасету

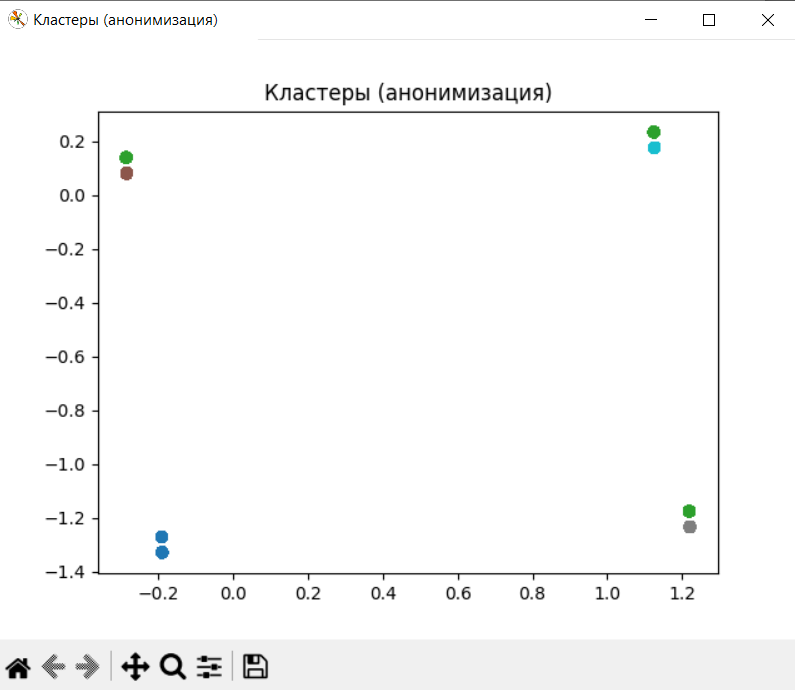


Рис 10.5 Кластеры по обезличенному датасету.

6) Сравнение различных комбинаций

Таблица 10.6 Сравнение

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер датасета | Кол-во кластеров | Кол-во информативных признаков | Rand по всему | Rand по изм. | Rand по обезл. всему | Rand по обезл. изм. |
| 50000 | 3 | 3 | 1.000 | 1.000 | 0.533 | 0.420 |
| 5 | 1.000 | 0.367 |
| 10 | 1.000 | 0.366 |
| 5 | 3 | 1.000 | 0.962 | 0.473 | 0.508 |
| 5 | 1.000 | 0.277 |
| 10 | 1.000 | 0.313 |
| 7 | 3 | 0.995 | 0.926 | 0.340 | 0.431 |
| 5 | 0.990 | 0.224 |
| 10 | 0.992 | 0.173 |
| 100000 | 3 | 3 | 1.000 | 1.000 | 0.477 | 0.493 |
| 5 | 1.000 | 0.381 |
| 10 | 1.000 | 0.396 |
| 5 | 3 | 1.000 | 0.981 | 0.440 | 0.505 |
| 5 | 1.000 | 0.404 |
| 10 | 1.000 | 0.279 |
| 7 | 3 | 0.996 | 0.936 | 0.346 | 0.431 |
| 5 | 0.998 | 0.226 |
| 10 | 0.999 | 0.174 |
| 500000 | 3 | 3 | 1.000 | 1.000 | 0.377 | 0.421 |
| 5 | 1.000 | 0.354 |
| 10 | 1.000 | 0.349 |
| 5 | 3 | 0.999 | 0.983 | 0.469 | 0.508 |
| 5 | 1.000 | 0.262 |
| 10 | 1.000 | 0.304 |
| 7 | 3 | 0.999 | 0.945 | 0.500 | 0.432 |
| 5 | 1.000 | 0.227 |
| 10 | 1.000 | 0.178 |

Таким образом, индекс Ранда по всему датасету и по информативным признакам в среднем остаётся примерно одинаковым, около единицы. Он может быть меньше по информативным признакам из-за удаления части важных данных, но может быть больше из-за удаления шумовых данных. По обезличенному датасету индекс Ранда гораздо ниже из-за сильного обобщения данных и отсутствия возможности создать хорошо выделенные кластеры.

# Вывод

В процессе исследования методов кластеризации данных в датасете была написана программа, создающая датасет заданного размера, создающая заданное количество кластеров по выбранному количеству информативных признаков, выполняющая кластеризацию по всему датасету, по выбранным признакам и по обезличенному датасету. Программа вычисляет индекс Ранда эффективности кластеризации и позволяет сравнить эффективность в разных случах. Проведён анализ эффективности для разных комбинаций размера датасета, количества кластеров и количества информативных признаков.

# Источники

# tkinter — tkinter documentation // Documentation URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 23.05.2025).

# numpy — numpy documentation // Documentation URL: <https://numpy.org/doc/> (дата обращения: 23.05.2025).

# matplotlib — matplotlib documentation // Documentation URL: <https://matplotlib.org/stable/index.html> (дата обращения: 23.05.2025).

# pandas — pandas documentation // Documentation URL: [https://pandas.pydata.org/docs/](https://pandas.pydata.org/docs/%20) (дата обращения: 23.05.2025).

# scipy — scipy documentation // Documentation URL: <https://docs.scipy.org/doc/scipy/> (дата обращения: 23.05.2025).

# sklearn — sklearn documentation // Documentation URL: <https://scikit-learn.org/stable/index.html> (дата обращения: 23.05.2025).

# Листинг

import tkinter as tk

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, OneHotEncoder

from sklearn.decomposition import PCA

from sklearn.metrics import rand\_score

from sklearn.datasets import make\_blobs

from scipy.spatial.distance import cdist

# ---------- Генерация датасета с числовыми и категориальными признаками ----------

def generate\_dataset(n\_samples=300, n\_num\_features=12, n\_cat\_features=5, n\_clusters=5):

    num\_feature\_names = [

        'Length', 'Width', 'Height', 'Weight', 'Density',

        'Temperature', 'Pressure', 'Volume', 'Speed', 'Acceleration',

        'Force', 'Energy'

    ][:n\_num\_features]

    X\_num, y = make\_blobs(n\_samples=n\_samples, centers=n\_clusters, n\_features=n\_num\_features,

                         cluster\_std=1.5, random\_state=42)

    df\_num = pd.DataFrame(X\_num, columns=num\_feature\_names)

    np.random.seed(42)

    cat\_feature\_values = {

        'Color': ['Red', 'Green', 'Blue', 'Yellow', 'Purple'],

        'Shape': ['Circle', 'Square', 'Triangle', 'Hexagon', 'Pentagon', 'Ellipse'],

        'Size': ['Small', 'Medium', 'Large', 'Extra Large'],

        'Material': ['Wood', 'Metal', 'Plastic', 'Glass'],

        'Texture': ['Smooth', 'Rough', 'Matte', 'Glossy']

    }

    cat\_keys = list(cat\_feature\_values.keys())[:n\_cat\_features]

    df\_cat = pd.DataFrame({

        key: np.random.choice(cat\_feature\_values[key], size=n\_samples)

        for key in cat\_keys

    })

    df = pd.concat([df\_num, df\_cat], axis=1)

    df['Target'] = y

    return df

# ---------- Предобработка данных ----------

def preprocess\_data(df, anonymized=False):

    if not anonymized:

        num\_cols = df.select\_dtypes(include=[np.number]).columns.drop('Target')

        cat\_cols = df.select\_dtypes(include=['object', 'category']).columns

        scaler = StandardScaler()

        scaled\_num = scaler.fit\_transform(df[num\_cols])

        try:

            encoder = OneHotEncoder(sparse\_output=False)

        except TypeError:

            encoder = OneHotEncoder(sparse=False)

        encoded\_cat = encoder.fit\_transform(df[cat\_cols])

        X = np.hstack([scaled\_num, encoded\_cat])

        return X, scaler, encoder, num\_cols, cat\_cols

    else:

        cat\_cols = df.columns.drop('Target')

        try:

            encoder = OneHotEncoder(sparse\_output=False)

        except TypeError:

            encoder = OneHotEncoder(sparse=False)

        X = encoder.fit\_transform(df[cat\_cols])

        return X, None, encoder, [], cat\_cols

# ---------- Кластеризация maximin ----------

def maximin\_clustering(X, k=None):

    centers\_idx = [np.random.randint(0, len(X))]

    centers = [X[centers\_idx[0]]]

    while True:

        dists = cdist(X, np.array(centers), metric='euclidean')

        min\_dists = np.min(dists, axis=1)

        new\_center\_idx = np.argmax(min\_dists)

        if k and len(centers) >= k:

            break

        if any(np.array\_equal(X[new\_center\_idx], c) for c in centers):

            break

        centers.append(X[new\_center\_idx])

        centers\_idx.append(new\_center\_idx)

    dists = cdist(X, np.array(centers), metric='euclidean')

    labels = np.argmin(dists, axis=1)

    return labels, centers

# ---------- Выбор информативных признаков ----------

def select\_informative\_features(df\_num, top\_n=10):

    center = df\_num.mean(axis=0)

    spread = ((df\_num - center) \*\* 2).mean(axis=0)

    return spread.sort\_values(ascending=False).head(top\_n).index.tolist()

# ---------- Анонимизация с округлением числовых и смысловым обобщением категориальных ----------

def anonymize(df, k=5, text\_output=None):

    df\_anon = df.copy()

    num\_cols = df\_anon.select\_dtypes(include=[np.number]).columns.drop('Target')

    cat\_cols = df\_anon.select\_dtypes(include=['object', 'category']).columns

    def transform\_number(x):

        if pd.isnull(x):

            return 'Missing'

        sign = 1

        int\_part = abs(int(x))

        digits\_count = len(str(int\_part)) if int\_part != 0 else 1

        return str(sign \* digits\_count)

    for col in num\_cols:

        df\_anon[col] = df\_anon[col].apply(transform\_number)

    cat\_threshold = max(k, 10)

    for col in cat\_cols:

        counts = df\_anon[col].value\_counts()

        rare = counts[counts < cat\_threshold].index

        df\_anon.loc[df\_anon[col].isin(rare), col] = 'Other'

    df\_anon.fillna('Missing', inplace=True)

    df\_anon.replace('', 'Missing', inplace=True)

    quasi\_id = df\_anon.drop(columns='Target').apply(lambda row: tuple(row), axis=1)

    freq = quasi\_id.value\_counts()

    k\_anon = freq.min()

    #if text\_output:

    #    text\_output.insert(tk.END, f"Минимальная частота квази-идентификаторов после обобщения: {k\_anon}\n")

    #    rare\_ids = freq[freq < k].index.tolist()

    #    if rare\_ids:

    #        text\_output.insert(tk.END, f"Количество уникальных (меньше k) квази-идентификаторов: {len(rare\_ids)}\n")

    #    else:

    #        text\_output.insert(tk.END, "Все квази-идентификаторы имеют частоту >= k\n")

        # --- Новый вывод: частоты всех уникальных строк ---

    #    text\_output.insert(tk.END, "\nЧастоты всех уникальных строк:\n")

    #    for row\_val, count in freq.items():

    #        text\_output.insert(tk.END, f"{row\_val} : {count}\n")

    #else:

    #    print(f"Минимальная частота квази-идентификаторов после обобщения: {k\_anon}")

    #    rare\_ids = freq[freq < k].index.tolist()

    #    if rare\_ids:

    #        print(f" Количество уникальных (меньше k) квази-идентификаторов: {len(rare\_ids)}")

    #    else:

    #        print("Все квази-идентификаторы имеют частоту >= k")

    #    print("\nЧастоты всех уникальных строк:")

    #    for row\_val, count in freq.items():

    #        print(f"{row\_val} : {count}")

    return df\_anon, k\_anon

# ---------- Построение графика ----------

def cluster\_and\_plot(data, label, text\_output, color='tab10'):

    labels, centers = maximin\_clustering(data, k=current\_k)

    n\_features = data.shape[1]

    n\_components = 2 if n\_features >= 2 else n\_features

    reduced = PCA(n\_components=n\_components).fit\_transform(data)

    fig, ax = plt.subplots()

    if n\_components == 1:

        ax.scatter(reduced[:, 0], np.zeros\_like(reduced[:, 0]), c=labels, cmap=color)

        ax.set\_ylim(-1, 1)

    else:

        ax.scatter(reduced[:, 0], reduced[:, 1], c=labels, cmap=color)

    ax.set\_title(label)

    fig.canvas.manager.set\_window\_title(label)

    fig.show()

    return labels

# ---------- Основной обработчик ----------

def process\_clustering():

    text\_output.delete(1.0, tk.END)

    try:

        top\_n = int(entry\_top\_n.get())

        if top\_n < 1:

            raise ValueError

    except ValueError:

        text\_output.insert(tk.END, "Некорректное число информативных признаков, используется значение по умолчанию 10\n")

        top\_n = 10

    global current\_k

    try:

        current\_k = int(entry\_clusters.get())

        if current\_k < 1:

            raise ValueError

    except ValueError:

        text\_output.insert(tk.END, "Некорректное число кластеров, используется значение по умолчанию 5\n")

        current\_k = 5

    try:

        n\_samples = int(entry\_samples.get())

        if n\_samples < 10:

            raise ValueError

    except ValueError:

        text\_output.insert(tk.END, "Некорректный размер датасета, используется 300\n")

        n\_samples = 300

    df = generate\_dataset(n\_samples=n\_samples, n\_cat\_features=5, n\_clusters=current\_k)

    y\_true = df['Target']

    df.to\_csv('dataset\_all\_features.csv', index=False)

    text\_output.insert(tk.END, "Сохранён датасет с всеми признаками: dataset\_all\_features.csv\n\n")

    X\_full, scaler, encoder, num\_cols, cat\_cols = preprocess\_data(df)

    text\_output.insert(tk.END, "Кластеризация по всем признакам...\n")

    labels\_all = cluster\_and\_plot(X\_full, "Кластеры (все признаки)", text\_output)

    rand\_all = rand\_score(y\_true, labels\_all)

    text\_output.insert(tk.END, f"Индекс Ранда: {rand\_all:.3f}\n\n")

    top\_num\_features = select\_informative\_features(df[num\_cols], top\_n=top\_n)

    text\_output.insert(tk.END, f"Информативные числовые признаки ({top\_n}): {', '.join(top\_num\_features)}\n")

    scaled\_top\_num = scaler.fit\_transform(df[top\_num\_features])

    X\_top = scaled\_top\_num

    df\_top = pd.concat([df[top\_num\_features], df['Target']], axis=1)

    df\_top.to\_csv('dataset\_top\_features.csv', index=False)

    text\_output.insert(tk.END, "Сохранён датасет с информативными признаками: dataset\_top\_features.csv\n\n")

    labels\_top = cluster\_and\_plot(X\_top, "Кластеры (информативные признаки)", text\_output)

    rand\_top = rand\_score(y\_true, labels\_top)

    text\_output.insert(tk.END, f"Индекс Ранда: {rand\_top:.3f}\n\n")

    df\_anon1, k\_anon\_val1 = anonymize(df, k=5)

    df\_anon1.to\_csv('dataset\_k\_anonymized.csv', index=False)

    text\_output.insert(tk.END, f"Сохранён обезличенный датасет: dataset\_anonymized\_orig.csv\n")

    text\_output.insert(tk.END, f"Показатель k-анонимности: {k\_anon\_val1}\n\n")

    X\_anon1, \_, \_, \_, \_ = preprocess\_data(df\_anon1, anonymized=True)

    text\_output.insert(tk.END, "Кластеризация по обезличенным данным всего датасета...\n")

    labels\_anon1 = cluster\_and\_plot(X\_anon1, "Кластеры (анонимизация всего датасета)", text\_output)

    rand\_anon1 = rand\_score(y\_true, labels\_anon1)

    text\_output.insert(tk.END, f"Индекс Ранда: {rand\_anon1:.3f}\n\n")

    df\_anon2, k\_anon\_val2 = anonymize(df\_top, k=5)

    df\_anon2.to\_csv('dataset\_k\_anonymized.csv', index=False)

    text\_output.insert(tk.END, f"Сохранён обезличенный датасет: dataset\_anonymized\_mod.csv\n")

    text\_output.insert(tk.END, f"Показатель k-анонимности: {k\_anon\_val2}\n\n")

    X\_anon2, \_, \_, \_, \_ = preprocess\_data(df\_anon2, anonymized=True)

    text\_output.insert(tk.END, "Кластеризация по обезличенным данным...\n")

    labels\_anon2 = cluster\_and\_plot(X\_anon2, "Кластеры (анонимизация для измененного датасета)", text\_output)

    rand\_anon2 = rand\_score(y\_true, labels\_anon2)

    text\_output.insert(tk.END, f"Индекс Ранда: {rand\_anon2:.3f}\n\n")

    text\_output.insert(tk.END, "Сравнение:\n")

    text\_output.insert(tk.END, f"- Все признаки:     {rand\_all:.3f}\n")

    text\_output.insert(tk.END, f"- Информативные:    {rand\_top:.3f}\n")

    text\_output.insert(tk.END, f"- Обезличенный (весь):     {rand\_anon1:.3f}\n")

    text\_output.insert(tk.END, f"- Обезличенный (изм):     {rand\_anon2:.3f}\n")

# ---------- GUI ----------

root = tk.Tk()

root.title("Кластеризация")

frame = tk.Frame(root, padx=10, pady=10)

frame.pack()

tk.Label(frame, text="Генерация и кластеризация датасета", font=("Arial", 14)).pack(pady=5)

frame\_top\_n = tk.Frame(frame)

frame\_top\_n.pack(pady=5)

tk.Label(frame\_top\_n, text="Количество информативных признаков:").pack(side=tk.LEFT)

entry\_top\_n = tk.Entry(frame\_top\_n, width=5)

entry\_top\_n.insert(0, "10")

entry\_top\_n.pack(side=tk.LEFT)

frame\_clusters = tk.Frame(frame)

frame\_clusters.pack(pady=5)

tk.Label(frame\_clusters, text="Количество кластеров:").pack(side=tk.LEFT)

entry\_clusters = tk.Entry(frame\_clusters, width=5)

entry\_clusters.insert(0, "5")

entry\_clusters.pack(side=tk.LEFT)

frame\_samples = tk.Frame(frame)

frame\_samples.pack(pady=5)

tk.Label(frame\_samples, text="Размер датасета:").pack(side=tk.LEFT)

entry\_samples = tk.Entry(frame\_samples, width=7)

entry\_samples.insert(0, "300")

entry\_samples.pack(side=tk.LEFT)

btn = tk.Button(frame, text="Выполнить кластеризацию", command=process\_clustering, width=30)

btn.pack(pady=10)

text\_output = tk.Text(frame, height=25, width=85, font=("Courier", 10))

text\_output.pack()

current\_k = 5

root.mainloop()