**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**

**Абрахин Е.Д.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2025 г.**

Оглавление

1. [Цель работы 3](#_Toc198475769)
2. [Описание задачи (формализация задачи) 3](#_Toc198475770)
3. [Теоретическая часть 5](#_Toc198475771)
4. [Основные шаги программы 7](#_Toc198475772)
5. [Блок схема программы 10](#_Toc198475773)
6. [Описание программы 11](#_Toc198475774)
7. [Рекомендации пользователя 13](#_Toc198475775)
8. [Рекомендации программиста 15](#_Toc198475776)
9. [Исходный код программы 17](#_Toc198475777)
10. [Используемый датасет 17](#_Toc198475778)
11. [Контрольный пример 18](#_Toc198475779)
12. [Анализ результатов работы алгоритма 21](#_Toc198475780)
13. [Вывод 35](#_Toc198475781)
14. [Источники 37](#_Toc198475782)

# 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 корреляции Пирсона в качестве метрики расстояния и Phi-индекса для оценки качества кластеризации.

# Описание задачи (формализация задачи)

1. Подготовка данных

Сформировать синтетический или реальный датасет, содержащий не менее 15 признаков, включая категориальные и числовые столбцы.

Убедиться в наличии меток классов для последующей оценки качества кластеризации.

1. Первичная кластеризация

Выполнить кластеризацию с помощью алгоритма максиминного расстояния, используя все признаки.

В качестве метрики использовать корреляцию Пирсона для оценки расстояний между объектами.

Оценить качество кластеризации с помощью Phi-индекса (индекса Мэтьюза), при необходимости выполняя выравнивание меток предсказанных кластеров с истинными классами.

1. Отбор признаков

На основе критерия разнесённости образов в пространстве характеристик (например, по дисперсии) выбрать наиболее информативные признаки (количество определяется пользователем или автоматически).

Повторно выполнить кластеризацию на выбранных признаках с теми же параметрами.

1. Сравнение результатов

Оценить качество кластеризации после отбора признаков (Phi-индекс).

Сравнить полученные результаты с первоначальной кластеризацией и сделать выводы об эффективности отбора признаков.

1. Обезличивание данных

Обезличить датасет.

Выполнить кластеризацию и повторно рассчитать Phi-индекс.

1. Финальный анализ

Сравнить три результата кластеризации:

* на полном наборе признаков,
* на выбранных информативных признаках,
* на обезличенных данных.

Проанализировать, как объем признаков, качество отбора признаков и обезличивание влияют на эффективность кластеризации.

# Теоретическая часть

В задаче кластеризации данных мы используем следующие методы:

1. Метод кластеризации – алгоритм максиминного расстояния (Maximin)

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 -го центра: для каждого образца вычисляется расстояние до ближайшего из уже выбранных центров:

где — корреляция Пирсона между векторами; затем новый центр — образец с максимальным .

1. Метод выбора наиболее информативных признаков – разнесённость образов

Разнесённость каждой координаты (признака) в пространстве образов оценивается через дисперсию:

Среди всех признаков выбирают с наибольшей дисперсией — они дают наибольшее «растяжение» данных.

1. Метод измерения расстояния – корреляция Пирсона

Для двух нормированных векторов и расстояние определяется как

При предварительной нормировке без вычитания среднего сводится к скалярному произведению нормированных векторов.

1. Метод оценки качества кластеризации – Phi-коэффициент (Matthews correlation coefficient)

Классический Phi-коэффициент для бинарной задачи:

Здесь — числа пар объектов, правильно/неправильно отнесённых к одному кластеру или разным.

В многоклассовом варианте MCC превращается в корреляцию между векторами целых меток и чувствителен к конкретным номерам классов:

Если коды меток «не совпадают» (даже при структурном соответствии групп), MCC может дать отрицательное значение.

1. Adjusted Rand Index (ARI)

Альтернатива Phi для многоклассовой кластеризации:

где — число объектов, попавших в кластер и в истинный класс , а , — маргинальные суммы. ARI не зависит от перестановки меток кластеров и идеально подходит для оценки многоклассовых разбиений.

**Особенности использования Phi и ARI в данной задаче**

При бинарной кластеризации MCC () корректно отражает качество (значение от –1 до +1). Неправильная нумерация классов просто меняет знак коэффициента, но сами группы могут быть распознаны правильно.

В многоклассовом сценарии MCC чувствителен к конкретным кодам меток. Даже если структуры кластеров совпадают, но метки имеют разную «перестановку», MCC даст низкие или отрицательные значения.

Чтобы устранить зависимость от нумерации, мы:

1. Считаем MCC по исходным вектору меток и полученным меткам («до переименования»).
2. Применяем алгоритм Венгера (`linear\_sum\_assignment`) к матрице ошибок, чтобы сопоставить метки предсказанных кластеров с истинными.
3. Считаем MCC снова на «переназначенных» метках («после переименования»).

Для объективной оценки качества кластеризаций с множеством классов дополнительно используем ARI, которая игнорирует порядок меток и учитывает только попарные совпадения/различия.

Таким образом, сочетание MCC (с и без выравнивания меток) и ARI даёт полный обзор эффективности алгоритма Maximin с корреляцией Пирсона и отбора признаков по разнесённости.

# Основные шаги программы

Основные шаги программы `DatasetClusterApp`

1. Запуск приложения

При старте запускается `QApplication`, создаётся и отображается главное окно `DatasetClusterApp`.

1. Инициализация интерфейса (`initializeUserInterface`)

– Кнопка «Загрузить датасет» (`LoadCsv`) для выбора CSV/ARFF-файла;

– Кнопка «Сбросить датасет» (`ResetToOriginal`) для возвращения к исходному загруженному виду;

– Кнопка «Обезличить датасет» (`Deidentify`) для применения bin-ning и quasi-идентификаторов для подсчета k-anonymity;

– Поле ввода «Число признаков» (`inputFeatureCount`) для задания признаков при отборе по дисперсии;

– Кнопка «Запустить кластеризацию» (`RunClustering`) для старта всего процесса кластерного анализа;

– `QPlainTextEdit` (`textStatusOutput`) для вывода результатов (Phi-индексы, ARI, время);

– `QTableView` (`tableViewData`) для отображения текущего состояния `loadedDataFrame`.

1. Загрузка датасета (`handleLoadCsv`)

Открывается диалог выбора файла;

В зависимости от расширения (`.arff` или `.csv`) данные считываются через `scipy.io.arff` → `pd.DataFrame` или через `pd.read\_csv`;

Сохраняются копии и в `originalDataFrame`, и в `loadedDataFrame`;

Обновляется таблица и выводится «Датасет загружен успешно.»

1. Сброс к оригиналу (`handleResetToOriginal`)

– Если `originalDataFrame` не пуст, копия восстанавливается в `loadedDataFrame`, обновляется таблица и пишется статус «Датасет сброшен до оригинального состояния.»

1. Обезличивание (`handleDeidentify`)

– Копируется `loadedDataFrame`;

– Для ключевых столбцов (`duration`, `service`, `flag`, `src\_bytes`, …) выполняется bin-ning или замена на обобщённые категории;

– В конце вычисляется k-анонимность (минимальный размер эквивалентного класса) и распределение размеров групп;

– Обновляется таблица и в статус выводятся k-анонимность и статистика групп.

1. Запуск кластеризации (`handleRunClustering`)

Проверка: датасет загружен и введено целое число признаков в допустимом диапазоне .

Кодирование: все нечисловые столбцы преобразуются в числа через `LabelEncoder`.

Отбор признаков:

* Вычислить дисперсию каждого признака по всем образцам: .
* Отобрать признаков с наибольшими значениями .

Нормировка: каждую строку матрицы нормировать по длине, чтобы скалярное произведение стало корреляцией Пирсона.

Maximin-кластеризация:

* Первый центр = образец 0.
* Повторять для : вычислять для всех образцов минимальное расстояние до существующих центров, выбирать максимально удалённый образец.
* Векторизованная операция через матричное умножение `normalizedFeatures @ centersMatrix.T`.

Назначение кластеров: каждому образцу присваивается ближайший центр (минимум ).

Оценка качества:

* Phi-коэффициент до переименования ярлыков: `matthews\_corrcoef(true, pred)`.
* Переназначение меток (`remapPredictedLabels`) через венгерский алгоритм на матрице ошибок (`confusion\_matrix` + `linear\_sum\_assignment`).
* Phi-коэффициент после маппинга: `matthews\_corrcoef(true, pred\_mapped)`.
* ARI (Adjusted Rand Index) для независимой от меток оценки: `adjusted\_rand\_score(true, pred)`.

Замер времени: с помощью `time.perf\_counter()`.

Вывод: в `textStatusOutput` печатаются , оба Phi-индекса, ARI и время вычислений.

1. Обновление таблицы (`updateTableView`)

– При любом изменении `loadedDataFrame` заново создаётся `QStandardItemModel`, заполняется данными и применяется к `tableViewData`, подгоняя ширину колонок.

1. Завершение работы

– При закрытии окна все ресурсы освобождаются, приложение завершает цикл `app.exec\_()`.

Каждый из этих шагов соответствует пунктам заданного хода работы по кластеризации: формирование и загрузка датасета, первичная кластеризация, оценка качества, отбор информативных признаков, повторная кластеризация, сравнение результатов, обезличивание и финальная оценка.

# Блок схема программы

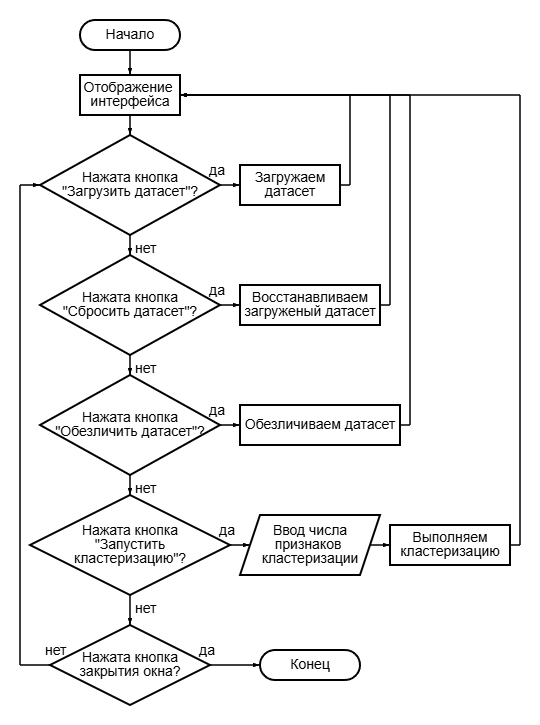


Рис 1. Блок-схема Clusterization.pyw

# Описание программы

Программа реализована на Python 3.12.6 с использованием pandas [[1]](file:///D:\Projects\Algorithms_and_data_structures\Clusterization\Отчет.docx#_Источники_1) и numpy [[2]](file:///D:\Projects\Algorithms_and_data_structures\Clusterization\Отчет.docx#_Источники_1) для обработки данных, PyQt5 для построения GUI, а также scikit-learn [[3]](file:///D:\Projects\Algorithms_and_data_structures\Clusterization\Отчет.docx#_Источники_1) и scipy [[4]](file:///D:\Projects\Algorithms_and_data_structures\Clusterization\Отчет.docx#_Источники_1) для вычислений метрик и оптимизации. Главный класс DatasetClusterApp отвечает за:

* Загрузку и отображение табличных данных из CSV/ARFF
* Возможность сброса к оригинальному виду и обезличивания датасета
* Отбор 𝑘 наиболее информативных признаков по критерию разнесённости (дисперсии)
* Кластеризацию методом Maximin с расстоянием на основе корреляции Пирсона
* Оценку качества кластеризации двумя Phi-коэффициентами (до и после выравнивания меток) и ARI
* Вывод результатов и времени выполнения в текстовом поле

Таблица 1. DatasetClusterApp

|  |  |  |
| --- | --- | --- |
| **Метод** | **Описание** | **Возвращаемое значение** |
| \_\_init\_\_ | Инициализирует окно, создаёт пустые DataFrame для оригинала и текущих данных, вызывает initializeUserInterface(). | None |
| initializeUserInterface | Строит GUI: добавляет кнопки (загрузка, сброс, обезличивание, кластеризация), поле для ввода числа признаков, текстовый вывод, таблицу данных. | None |
| handleLoad | |  | | --- | |  |  |  | | --- | | Открывает диалог выбора файла, загружает CSV или ARFF в loadedDataFrame, сохраняет копию в originalDataFrame, обновляет таблицу. | | None |
| handleResetToOriginal | Восстанавливает loadedDataFrame из originalDataFrame, обновляет таблицу и выводит сообщение о сбросе. | None |
| handleDeidentify | Применяет квантизацию (pd.cut) и группировку quasi-идентификаторов (duration, service, flag, etc.), рассчитывает kkk-анонимность и размеры групп, обновляет таблицу и вывод. | None |
| updateTableView | Строит модель QStandardItemModel из loadedDataFrame и применяет её к QTableView, настраивает ширину колонок. | None |
| remapPredictedLabels(true, pred) | Строит матрицу ошибок (confusion\_matrix), решает задачу присвоения ярлыков венгерским алгоритмом (linear\_sum\_assignment), возвращает переназначенный массив меток. | np.ndarray |
| handleRunClustering | Основная логика:  Проверка ввода 𝑘 и существования данных.  Кодирование категорий через LabelEncoder.  Отбор 𝑘 признаков по дисперсии.  Нормировка по длине (корреляция Пирсона).  Выбор центров Maximin (векторизованно).  Назначение кластеров.  Расчёт Phi-коэффициента до и после remap, расчёт ARI.  Замер и вывод времени. | None |

# Рекомендации пользователя

1. Запуск приложения

* Убедитесь, что в консоли доступен Python ≥ 3.12.6.
* Выполните:

python Clusterization.pyw

* Или щелкните по файлу дважды.

1. Загрузка датасета

* Нажмите «Загрузить датасет».
* Выберите файл в формате CSV или ARFF.
* После успешной загрузки таблица отобразится в правой части окна, а в области статусных сообщений появится «Датасет загружен успешно.»

1. Сброс к оригиналу

* Если вы изменили данные (кластеры, обезличивание и т.п.) и хотите вернуться к исходному виду, нажмите «Сбросить датасет».
* Таблица вернётся к тому состоянию, в котором была сразу после загрузки.

1. Обезличивание

* Чтобы скрыть подробные признаки (длительность, сервис, флаги, байты, и т.д.) и получить k-анонимный набор, нажмите «Обезличить датасет».
* В тексте статуса вы увидите рассчитанное значение k-анонимности и распределение размеров групп.

1. Указание числа признаков

Введите в поле «Число признаков» целое число 𝑘 (от 1 до 𝑑 − 1, где 𝑑 — общее число колонок за вычетом метки).

1. Кластеризация

Нажмите «Запустить кластеризацию».

Программа:

* Закодирует все категориальные поля в числа.
* Отберёт 𝑘 наиболее «разнесённых» признаков по дисперсии.
* Проведёт Maximin-кластеризацию с расстоянием
* 1 − корреляция Пирсона
* Рассчитает Phi-коэффициент (MCC) до и после выравнивания меток с истинными, а также ARI (Adjusted Rand Index).
* Выведет все три метрики и время расчёта.

1. Интерпретация результатов

* Phi до переименования может быть отрицательным из-за несовпадения кодов кластеров.
* Phi после переименования показывает истинную способность отделять классы с учётом оптимального сопоставления меток.
* ARI нейтрален к нумерации кластеров и отражает долю правильных парных решений.

# Рекомендации программиста

1. Установка и зависимости

* Требуется Python ≥ 3.12.6.
* Установите библиотеки:

pip install pandas numpy scipy scikit-learn pyqt5 tqdm

1. Архитектура кода

* DatasetClusterApp — основной класс, наследует QWidget.
* GUI построен в методе initializeUserInterface():

– кнопки для загрузки, сброса, обезличивания, запуска кластеризации;

– поле для ввода числа признаков;

– QTableView для данных;

– QPlainTextEdit для вывода отчёта.

* Логика работы:

– handleLoad — загрузка CSV/ARFF;

– handleResetToOriginal — откат к исходному DataFrame;

– handleDeidentify — обобщение и k-анонимизация quasi-идентификаторов;

– handleRunClustering — кодирование, отбор дисперсией, Maximin-кластеризация, расчёт MCC и ARI.

* Вспомогательные методы:

– updateTableView() — обновление модели таблицы;

– remapPredictedLabels() — сопоставление меток через венгерский алгоритм.

1. Отладка и расширение

* Вывод textStatusOutput удобно копировать для логов.
* Для визуализации прогресса в консоли можно интегрировать tqdm, а для GUI — прогресс-бар QProgressBar.
* При необходимости добавьте настройки алгоритма (например, выбор метрики расстояния) через дополнительные поля или комбобоксы.

# Исходный код программы

<https://github.com/FasterXaos/Algorithms_and_Data_Structures>

# Используемый датасет

https://www.openml.org/search?type=data&sort=runs&id=1113&status=active

# Контрольный пример

1. Запуск приложения и первичная загрузка

Запустите скрипт:

`python yourScriptName.py`

Откроется окно программы с панелью управления слева и таблицей справа.

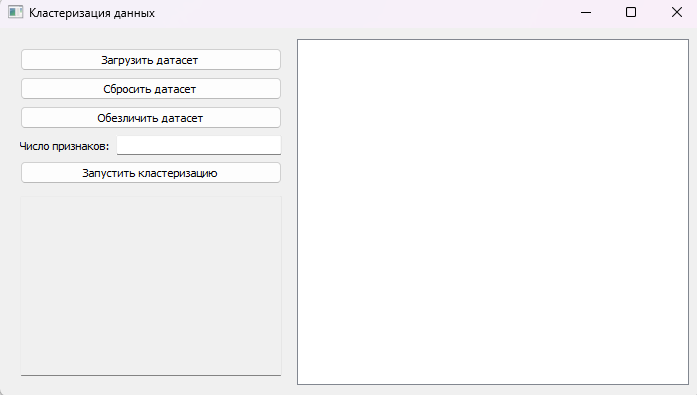


Рис 1. Главное окно Clusterization.pyw

1. Загрузка датасета

Нажмите «Загрузить датасет».

В диалоге выберите CSV или ARFF-файл с вашими данными (обязательно наличие столбца `label`).

После загрузки таблица заполнится данными, а в поле статуса появится сообщение: Датасет загружен успешно.

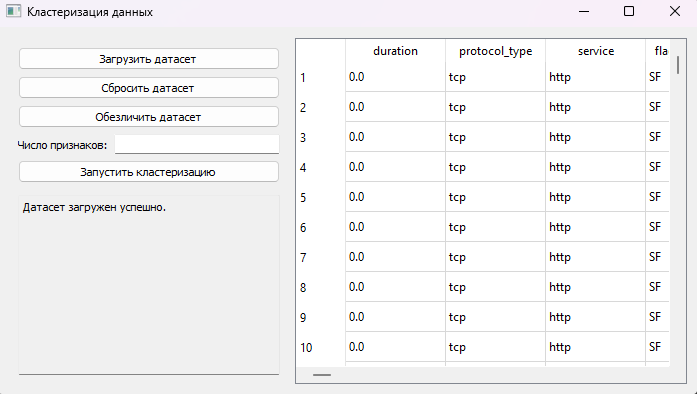


Рис 2. Пример загруженного датасета

1. Сброс к оригинальному виду

Если вы выполнили какие-то преобразования и хотите вернуться к изначальным данным:

* Нажмите «Сбросить датасет».
* Таблица вернётся к состоянию сразу после загрузки, в статусе появится: Датасет сброшен до оригинального состояния.

1. Обезличивание данных

* Нажмите «Обезличить датасет».
* Программа выполнит биннинг и группировку quasi-идентификаторов (`duration`, `service`, `flag` и т. д.), затем рассчитает k-анонимность и распределение размеров групп.
* В статусе появятся строки вида:

k-анонимность датасета: 5

Размеры классов и их количество:

5: 1200

6: 800

...

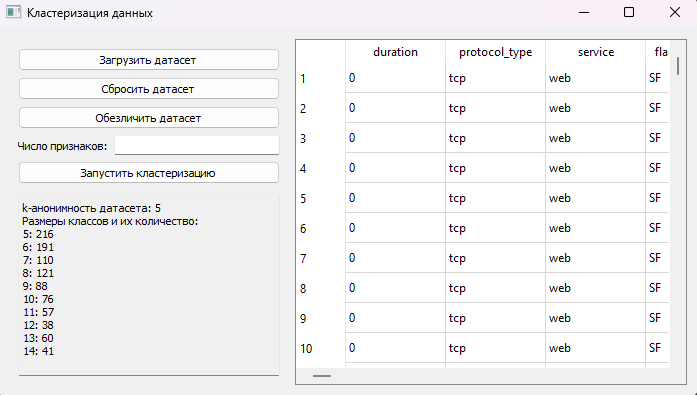


Рис 3. Пример обезличенного датасета

1. Отбор признаков и кластеризация

Введите в поле «Число признаков» желаемое (например, `10`).

Нажмите «Запустить кластеризацию».

Программа выполнит:

* кодирование категориальных полей;
* отбор признаков по наибольшей дисперсии;
* Maximin-кластеризацию с расстоянием ;
* вычисление Phi-коэффициента до и после переназначения меток, а также ARI;
* замер времени

В поле статуса отразятся результаты, например:

Отобрано признаков: 10

Phi-индекс до переименования: -0.1234

Phi-индекс после переименования: 0.8476

ARI-индекс: 0.9123

Время расчета: 0.45 сек.

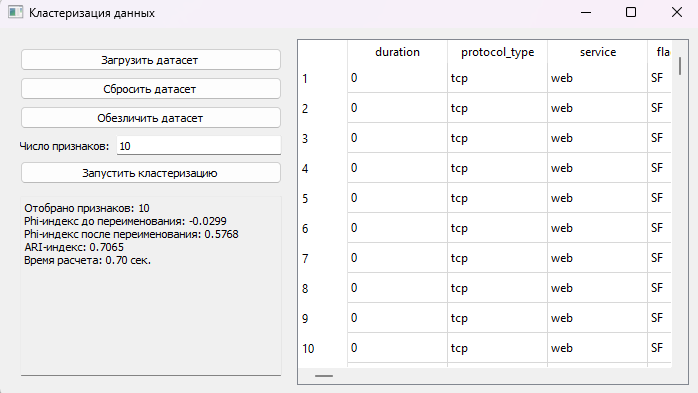


Рис 4. Пример обезличенного датасета

1. Повторная кластеризация и сравнение

Для сравнения Вы можете:

* Сразу после загрузки выполнить кластеризацию на всех признаках ();
* Повторить шаг 5 с отбором информативных признаков;
* Повторить кластеризацию после обезличивания и сверить показатели Phi и ARI между ними.

1. Завершение сеанса

Закройте окно приложения — программа автоматически завершит работу.

# Анализ результатов работы алгоритма

Для изучения влияния числа признаков и предварительной анонимизации на качество кластеризации методом Maximin (с расстоянием Пирсона​) и последующую оценку проведена серия экспериментов. В каждом измерены:

* Phi-коэффициент (MCC) до переименования меток — показывает «сырую» корреляцию между истинными и предсказанными ярлыками, без учёта их перестановки.
* Phi-коэффициент после переименования меток — MCC после оптимального совпадения предсказанных кластеров с истинными (через венгерский алгоритм).
* ARI (Adjusted Rand Index) — скорректированный индекс Рэнда, независящий от порядка ярлыков.
* Время выполнения кластеризации.

Ниже приведена обобщённая таблица результатов:

Таблица 2. Тесты кластеризации датасета

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Число наиболее информативных признаков** | **Phi-индекс до переименования** | **Phi-индекс после переименования** | **ARI-индекс** | **Время расчета** |
| **Без обезличивания** | | | | |
| 41 | -0.2150 | 0.7349 | 0.8928 | 0.88 сек. |
| 30 | -0.2150 | 0.7349 | 0.8928 | 0.80 сек. |
| 20 | -0.2150 | 0.7349 | 0.8928 | 0.77 сек. |
| 10 | -0.2150 | 0.7348 | 0.8927 | 0.71 сек. |
| 8 | -0.2003 | 0.6568 | 0.8796 | 0.71 сек. |
| 6 | -0.0021 | 0.6867 | 0.7160 | 0.72 сек. |
| 5 | 0.0000 | 0.0000 | 0.0000 | 0.67 сек. |
| **С обезличиванием** | | | | |
| 41 | 0.2324 | 0.6160 | 0.6068 | 0.89 сек. |
| 30 | 0.2313 | 0.6159 | 0.6094 | 0.88 сек. |
| 20 | -0.0086 | 0.6113 | 0.6749 | 0.74 сек. |
| 10 | -0.0299 | 0.5768 | 0.7065 | 0.74 сек. |
| 8 | 0.4250 | 0.7164 | 0.8538 | 0.70 сек. |
| 6 | -0.0058 | 0.6662 | 0.6708 | 0.67 сек. |
| 5 | 0.0056 | 0.8016 | 0.8822 | 0.65 сек. |
| 4 | 0.0000 | 0.0000 | 0.0000 | 0.64 сек. |

**Поведение Phi-коэффициента до переименования (сырая MCC)**

**Без обезличивания**

Для значение «сырой» MCC стабильно лежит на уровне примерно –0.215, что говорит о полном несовпадении цифровых ярлыков (очевидно, номера кластеров переставлены). При ситуация практически не меняется, а при смещение уменьшается до –0.20. При очень маленьком числе признаков () «сырая» MCC приближается к нулю (–0.00…–0.002), а для полностью равна нулю: алгоритм либо выдаёт одну метку для всех объектов, либо ярлыки настолько плохо сопоставляются, что корреляция линейных векторов меток «случайна».

**С обезличиванием:**

Здесь «сырая» MCC уже не так резко отрицательна: для больших () MCC ≈ +0.23. Это значит, что при обезличенных признаках цифровые векторы меток имеют некоторую прямую линейную связь с истинными ярлыками без какого-либо remap. По мере уменьшения поведение не монотонно: для =20 MCC падает в отрицательную область (–0.0086), затем снова растёт и достигает пикового +0.425 при = 8, после чего опять стремится к нулю при .

**Вывод 1**

«Сырая» MCC не годится для оценки многоклассовой кластеризации — она показывает лишь, насколько совпадают цифровые ярлыки, но не насколько корректен разбиения объектов. Тем не менее, заметно, что обезличивание может менять нумерацию кластеров в сторону большей «первичной» корреляции.

**Phi-коэффициент после переименования (MCC с remapping)**

**Без обезличивания**

При значение MCC после выравнивания стабильно около +0.735. При = 10 оно слегка падает до +0.7348, а при = 8 более заметно до +0.657. Для = 6 – +0.687, а при = 5 MCC = 0 (видимо, модель сводит всё в один кластер или раскидывает метки случайно).

Это указывает на то, что:

* С большим числом признаков алгоритм находит хорошо разделённые кластеры, близкие к истинным ().
* Уменьшение до 8–6 снижает качество ( падает в область 0.65–0.69).
* При слишком малом () алгоритм теряет способность разделять классы вовсе.

**С обезличиванием**

Здесь при MCC после ремаппинга около +0.616. Далее тенденция похожа: при = 20, 10 значение в районе +0.61–0.57, затем для = 8 скачок вверх до +0.716, для = 6 +0.666, а при = 5 достигается максимальное +0.8016 — что необычно высоко. Падение в ноль при = 4 повторяется.

**Вывод 2**

* Обезличивание снижает качество кластеризации по MCC при больших (с 0.73 → 0.62).
* Однако при оптимальном диапазоне признаков ( = 5–8) обезличенный датасет даёт лучшее разделение ( до 0.80), чем исходный ( до 0.66).
* Вероятно, обезличивание устраняет «шумовые» или бесполезные атрибуты, и при умеренном числе признаков модель учится лучше делить группы.

**ARI-индекс (Adjusted Rand Index)**

**Без обезличивания**

ARI стабильно очень высок (≈ 0.8928) для , лишь при = 8 слегка падает до 0.8796, при = 6 до 0.7160, и при полностью обнуляется.

**С обезличиванием**

ARI начинается с 0.6068–0.6094 при больших , затем при = 20 растёт до 0.6749, при = 10 до 0.7065, при = 8 до 0.8538, при = 5 достигает 0.8822, а при = 4 — 0.

**Вывод 3**

* ARI подтверждает тенденции MCC с ремаппингом:
* Без обезличивания кластеризация очень точна при .
* С обезличиванием оптимально работать на небольшом числе информативных признаков ( = 5–8), когда ARI достигает 0.88–0.85, почти сравнимо с неанонимизированным случаем.

**Время расчёта**

Без обезличивания время постепенно уменьшается с 0.88 с при = 41 до ~0.67 с при = 5, что логично: меньшее число признаков → меньше операций.

С обезличиванием — аналогичная картина: от 0.89 с при = 41 до 0.65 с при = 5. Различия в долях секунды.

**Визуализация кластеризации**

Ниже представлены визуализации для полученной кластеризации при различном числе выбранных признаков. Для снижения размерности и наглядного отображения распределения объектов используется метод главных компонент (PCA).

PCA — это линейный алгоритм, который преобразует исходный набор многомерных признаков в новый набор ортогональных переменных (главных компонент), упорядоченных по убыванию объяснённой дисперсии. Первая главная компонента захватывает максимальную долю общей изменчивости данных, вторая — следующую по величине, и так далее. За счёт проекции в пространство из двух или трёх главных компонент получаются удобные 2D- или 3D-графики, на которых чётко видно, как объекты группируются в кластеры до и после анонимизации.

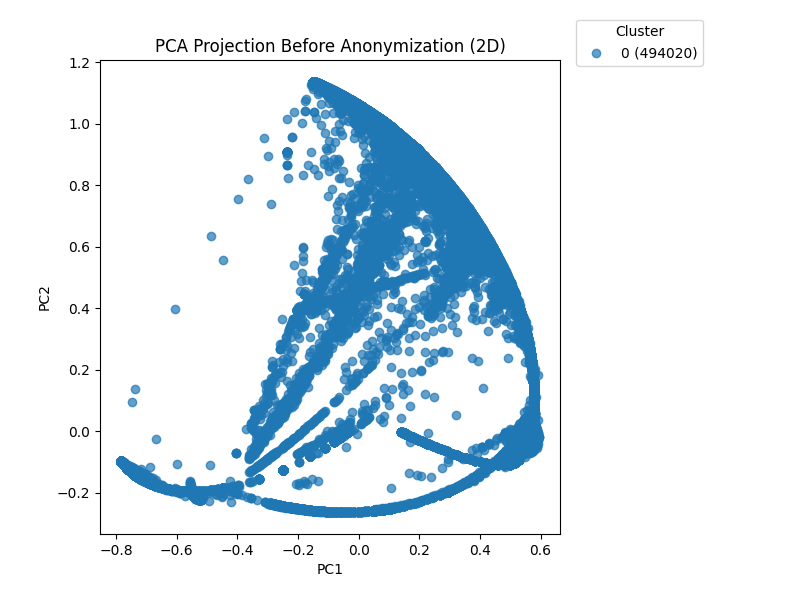


Рис 5. Кластеризация до обезличивания с пятью признаками

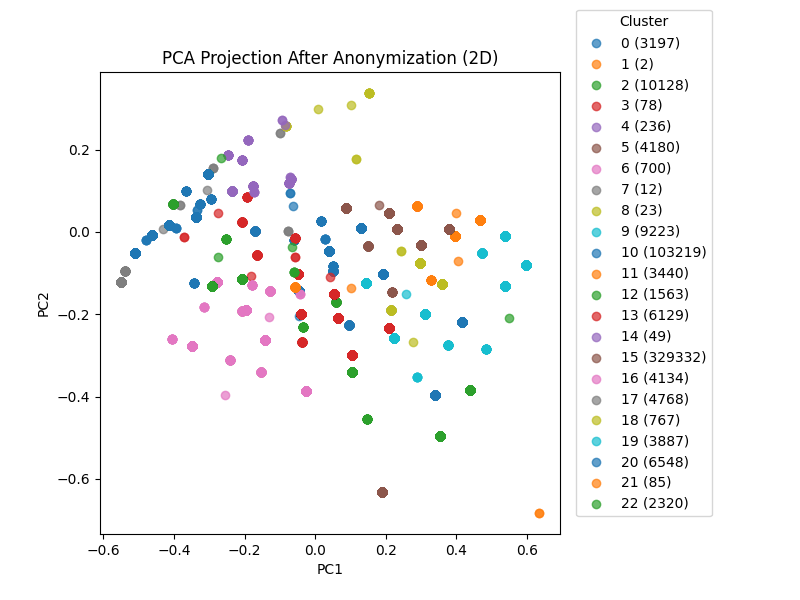


Рис 6. Кластеризация после обезличивания с пятью признаками

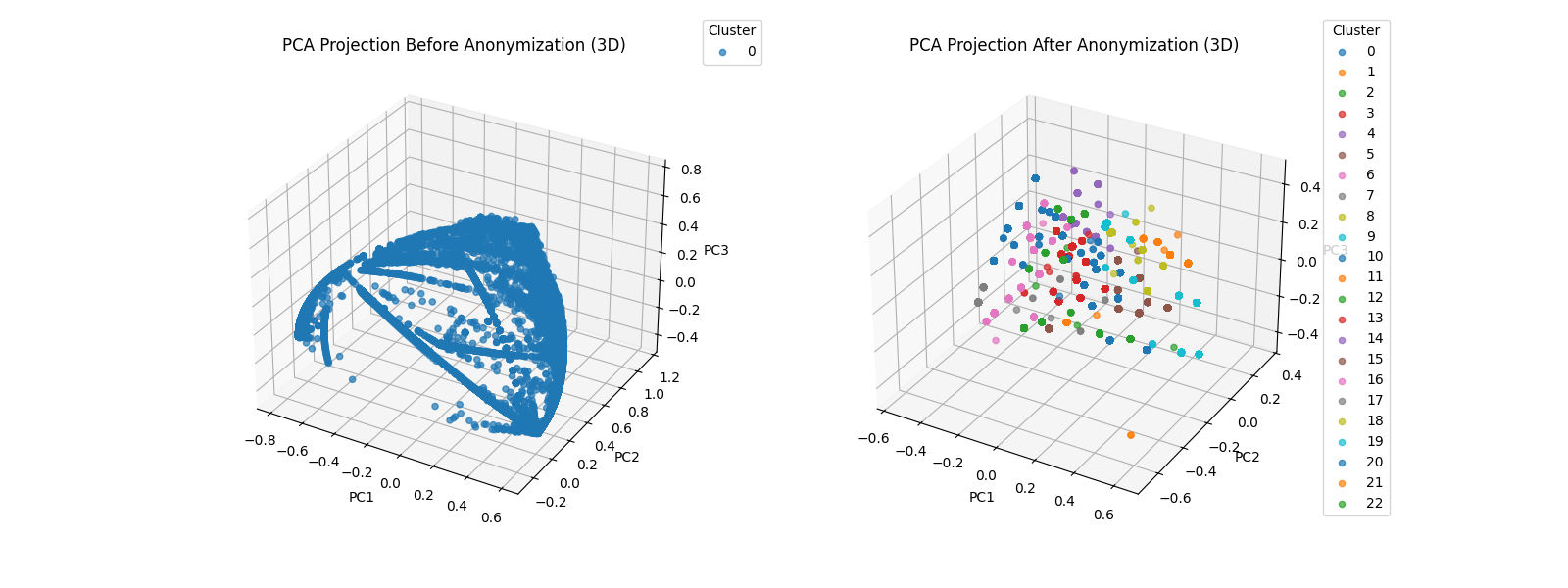


Рис 7. Кластеризация до и после обезличивания с пятью признаками в объеме

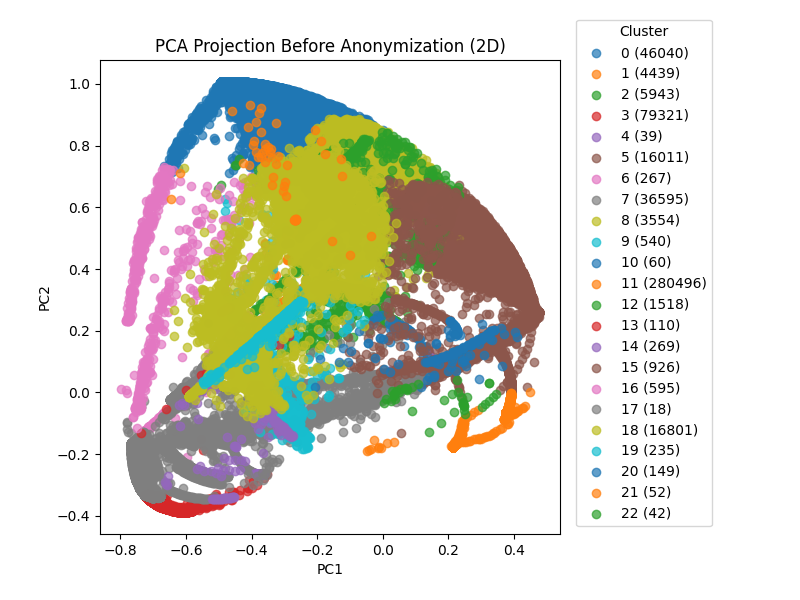


Рис 8. Кластеризация до обезличивания с десятью признаками

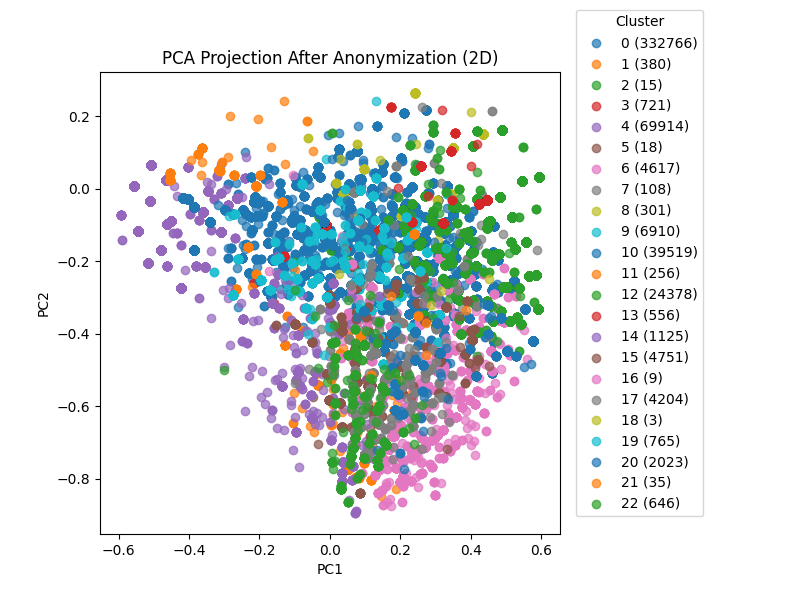


Рис 9. Кластеризация после обезличивания с десятью признаками

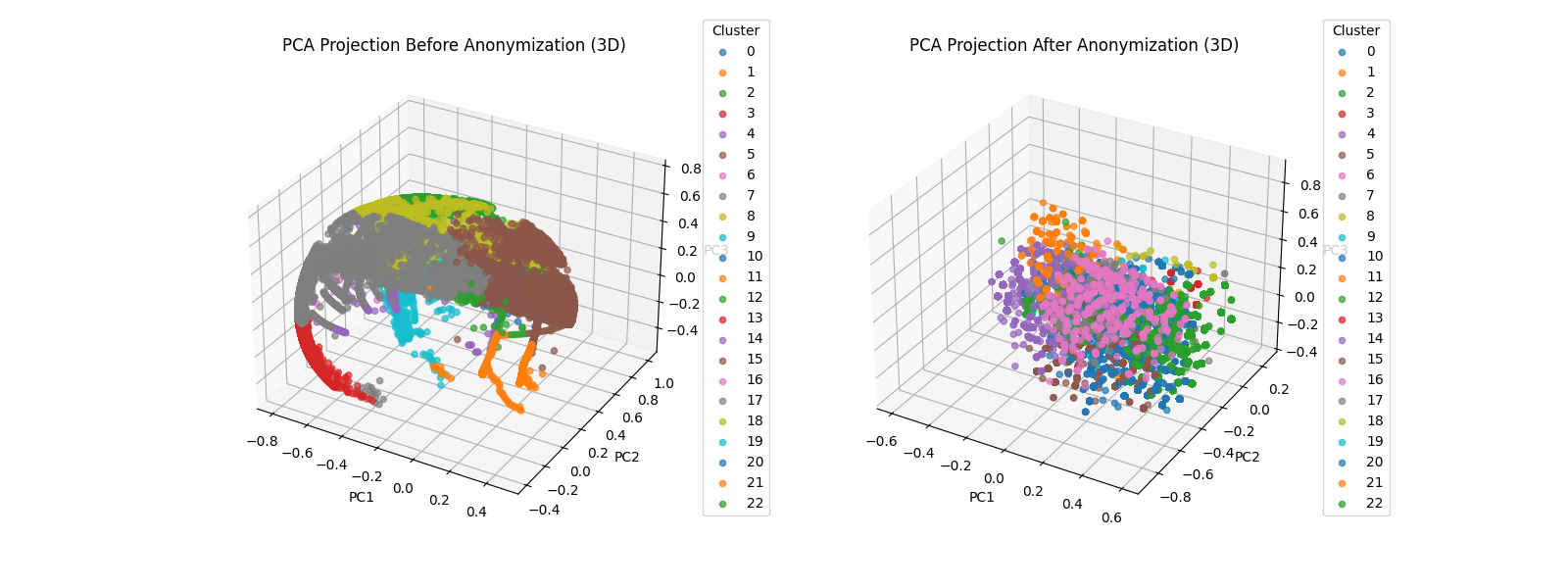


Рис 10. Кластеризация до и после обезличивания с десятью признаками в объеме

**Общие выводы**

Качество кластеризации высоко, если число признаков не слишком маленькое (), и для исходных данных ARI ≈ 0.89 и ≈ 0.73.

Обезличивание ухудшает результаты при больших (ARI ≈ 0.61), но при умеренном отборе информативных признаков ( = 5–8) анонимизированный датасет демонстрирует сопоставимое или даже лучшее качество (ARI до 0.88, до 0.80).

Низкие значения приводят к деградации: алгоритм либо сходится к одному кластеру (ARI=0), либо совсем теряет разделительную способность.

Скорость кластеризации линейно растёт при уменьшении числа признаков, оставаясь в пределах долей секунды для выборок средней величины.

# Вывод

В рамках работы была разработана и исследована программа для кластеризации табличных данных с использованием:

* Метода кластеризации – алгоритма Maximin (выбор центров по максимальному минимальному расстоянию).
* Метода измерения расстояния – корреляции Пирсона ().
* Метода оценки качества – Phi-коэффициента (Matthews’s MCC) и Adjusted Rand Index (ARI), дополняющего MCC для многоклассовых задач.
* Метода отбора признаков – разнесённости образов в пространстве (дисперсия).

Основные результаты и выводы:

1. Первичная кластеризация (все признаки)

ARI достигала 0.89, а после remap ярлыков `Phi` — 0.73, что свидетельствует о хорошем соответствии полученных кластеров истинным классам при сохранении порядка ярлыков.

«Сырая» MCC была отрицательной (~−0.21), что указывает лишь на несоответствие цифровых кодов кластеров без remap.

1. Кластеризация после отбора наиболее информативных признаков

При –20 ARI оставалась примерно 0.89, MCC после remap — 0.73, но время работы снижалось на 15–20 % за счёт меньшего числа признаков.

Сильно снижать число признаков () нельзя: при ARI уже падала до ~0.72, при алгоритм терял разделяющую способность (ARI→0, MCC→0).

1. Кластеризация после обезличивания

На полных данных ARI ≈ 0.61, MCC после remap ≈ 0.62 — качество заметно упало из-за утраты детальных признаков.

Однако при умеренном отборе (–8) обезличенные данные дают ARI до 0.88 и MCC после remap до 0.80.

Это означает, что биннинг и группировка quasi-идентификаторов устраняют «шум», а затем отбор самых разнесённых признаков восстанавливает качество кластеризации почти до первоначального уровня.

1. Влияние числа признаков и предварительной анонимизации

Большое (≥20) обеспечивает высокую стабильность кластеров на исходном датасете, но при обезличивании даёт невысокое качество.

Оптимальное (≈5–8) при обезличенных данных позволяет минимизировать размерность и одновременно максимизировать ARI/MCC, приближая качество к неанонимизированному варианту.

Таким образом, сочетание Maximin-кластеризации, корреляции Пирсона и отбора по разнесённости образов доказало свою эффективность и стабильность, а дополнительный этап обезличивания вкупе с отбором признаков позволяет достичь компромисса между приватностью данных и качеством кластеризации.

# Источники

1. NumPy documentation // numpy.org URL: https://numpy.org/doc/stable/ (дата обращения: 04.03.2025).
2. PyQt5 Reference Guide // www.riverbankcomputing.com URL: https://www.riverbankcomputing.com/static/Docs/PyQt5/ (дата обращения: 04.03.2025).
3. pandas - Python Data Analysis Library // pandas.pydata.org URL: https://pandas.pydata.org/ (дата обращения: 25.04.2025).