# Об одной метрике эффективности агломеративной кластеризации

В. Г. Мосин

**Аннотация**

Излагается метод оценки эффективности агломеративной кластеризации, основанный на вычислении средней дистанции между центроидами кластеров на обучающей и тестовой выборке. Приводится алгоритм метода. Рассматривается пример подбора оптимальной кластеризации данных о потреблении контента.

## Содержание

# 1 Введение

Кластеризация данных — это метод машинного обучения, осуществляющий группировку данных на основе их схожести (см. [5]). В процессе кластеризации данные разделяются на группы, которые называются кластерами, таким образом, что объекты внутри одного кластера более похожи друг на друга, чем на объекты из других кластеров. Кластеризация позволяет выявить скрытые шаблоны, структуры и взаимосвязи в данных, которые могут быть незаметны при первом взгляде. Путем группировки по схожести можно выделить группы схожих объектов и выделить ключевые особенности каждой группы (см. [5], [6]).

Определение числа кластеров является важным элементом анализа данных. Во-первых, число кластеров напрямую влияет на качество и интерпретируемость кластеризации. Неправильное определение числа кластеров может привести к объединению сильно различающихся групп в один кластер или к разделению однородной группы на несколько кластеров. Правильно определенное число кластеров помогает обнаружить скрытые шаблоны, взаимосвязи и сходства в данных.

Далее, разбиение данных на недостаточное число кластеров может привести к упрощению кластерной структуры, не раскрывающей всю информацию в данных. Хотя, с другой стороны, слишком большое число кластеров может привести к излишней сложности и непонятности кластеризации. Определение оптимального числа кластеров помогает найти баланс между этими двумя крайностями.

И, наконец, существенным является вопрос принятия решений. Определение правильного числа кластеров помогает принимать правильные

решения на основе кластерного анализа, такие как создание сегментов покупателей, выявление характерных поведенческих паттернов или предсказание рисков и мошеннических действий.

В целом, определение числа кластеров является важным этапом в анализе данных, который помогает найти баланс между сложностью и информативностью кластеризации, а также позволяет принимать правильные решения на основе результатов анализа.

## 1.1 Теоретическая часть

Как правило, кластеризация осуществляется на основании мнения исследователя или представителя экспертной группы из конкретной предметной области, к которой относятся анализируемые данные (см. [5], [6]). В таком случае, ответственность за качество кластеризации ложится на исследователя или эксперта, и вопрос о машинно-ориентированном подходе к оценке качества кластеризации не возникает вообще.

Однако, в ряде случаев такой подход невозможен или нецелесообразен. Известные методы кластеризации, такие как KMeans или агломеративная кластеризация, формируют кластеры самостоятельно, отталкиваясь от геометрической близости (или отдаленности) объектов, но они требуют указания априорных значений своих параметров, в первую очередь, числа кластеров, которое опять-таки выбирается субъективно.

Таким образом, есть насущная необходимость в некоторой метрике, которая позволяла бы сравнивать качество кластеризации при изменении количества используемых кластеров, и в настоящей работе мы предлагаем такую метрику.

## 1.2 Постановки задачи

### 1.2.1 Предмет исследования

Предметом исследования служит метод, определяющий оптимальное число кластеров для агломеративной кластеризации.

### 1.2.2 Методика исследования

Мы разбиваем облако данных на две выборки, которые называем обучающей и тестовой, и производим их кластеризации. Если число кластеров подобрано хорошо, то кластеризации на этих двух выборках будут «похожими» друг на друга, и в качестве критерия их сходства мы предлагаем следующую идею.

Несмотря на то, что агломеративная кластеризации не апеллирует к понятию центроида кластера, после кластеризации все равно возникает несколько кластеров, каждый их которых обладает своим центроидом.

Если центроиды на обучающей выборке и тестовой выборке расположены близко друг к другу, то кластеризация качественная, если они менее близки, то она менее качественна и так далее. Это соображение позволяет упорядочить кластеризации с различным количеством кластеров по метрике, в качестве которой мы принимает среднюю дистанцию между центроидами на обучающей и тестовой выборке.

### 1.2.3 Цель исследования

Наша цель — реализовать метрику средних дистанций в виде алгоритма и провести апробацию метода.

## 1.3 Библиотеки

Для выполнения вычислений и анализа данных мы пользуемся средой `Jupyter Notebook`, которая предоставляет удобные средства для работы с языком программирования Python и его главными библиотеками: `NumPy`, `Pandas`, `sklearn` и `matplotlib`. Благодаря этим инструментам, мы можем эффективно работать с данными, выполнять исследования и визуализировать результаты (см. [1], [2]).

Библиотека `numpy` является одной из ключевых библиотек для научных вычислений и обработки массивов данных в языке программирования `Python`. Библиотека `pandas` — одна из наиболее популярных и мощных библиотек для работы с данными в языке программирования `Python` (см. [1]).

Библиотека `scikit-learn`, широко известная как `sklearn`, предоставляет обширный набор инструментов и функций для решения различных задач в языке программирования Python, таких как задачи классификации, регрессии, кластеризации и др. Мы используем эту библиотеку для решения регрессионных задач.

# 2 Описание данных

Для применения нашего алгоритма мы в качестве примера используем данные о потреблении контента пользователями одного из ведущих хостингом. Данные представляют собой записи о 500 датах, начиная с 2021-08-20 и заканчивая 2023-01-01. В каждую из этих дат были зафиксированы показатели потребления контента, такие, как 'Просмотры', 'Время просмотра (часы)', 'Поделились' и т. д. всего 18 признаков. Подробная структура данных описана ниже.

# 3 Алгоритм

## 3.1 Чтение данных

При помощи функции `read_csv` из библиотеки `pandas`, читаем набор данных:

| | Дата | Просмотры | Поделились | ... | Лайки |
|---|---|---|---|---|---|
| **0** | 2023-01-01 | 475.0 | 9.0 | ... | 16.0 |
| **1** | 2022-12-31 | 174.0 | 1.0 | ... | 4.0 |
| **2** | 2022-12-30 | 490.0 | 3.0 | ... | 3.0 |
| **...** | ... | ... | ... | ... | ... |
| **498** | 2021-08-21 | 222.0 | 0.0 | ... | 4.0 |
| **499** | 2021-08-20 | 209.0 | 0.0 | ... | 1.0 |

Записываем датафрейм в переменную `df`. В данных 500 записей, относящихся типу с плавающей запятой, пропущенных данных нет.

## 3.2 Нормализация данных

Применяя метод `describe` библиотеки pandas к датафрему `df`, получаем статистики признаков:

| | min | mean | max | std |
|---|---|---|---|---|
| **Просмотры** | 159.00 | 936.51 | 2200.00 | 418.144 |
| **Время просмотра (часы)** | 5.48 | 37.17 | 96.72 | 16.64 |
| **Поделились** | 0.00 | 6.96 | 71.00 | 6.25 |
| **Постоянные зрители** | 30.00 | 163.34 | 463.00 | 78.90 |
| **Новые комментарии** | 0.00 | 0.53 | 6.00 | 0.83 |
| **Отказались от подписки** | 0.00 | 2.77 | 29.00 | 2.55 |
| **Новые подписчики** | 0.00 | 6.49 | 19.00 | 3.51 |
| **Новые зрители** | 60.00 | 366.83 | 735.00 | 174.01 |
| **Среднее число просмотров одним пользователем** | 1.31 | 1.79 | 2.85 | 0.21 |
| **Уникальные зрители** | 96.00 | 530.18 | 1103.00 | 239.22 |
| **CTR для значков видео (%)** | 1.25 | 5.54 | 8.52 | 1.11 |

| | min | mean | max | std |
|---|---|---|---|---|
| **Показы** | 1938.00 | 8093.78 | 39479.00 | 3816.08 |
| **Подписчики** | 0.00 | 3.72 | 15.00 | 4.02 |
| **Средний процент просмотра (%)** | 18.68 | 26.72 | 41.29 | 3.41 |
| **Процент лайков** | 0.00 | 92.02 | 100.00 | 10.31 |
| **Средняя продолжительность просмотра** | 96.07 | 144.33 | 211.02 | 15.66 |
| **Дизлайки** | 0.00 | 1.28 | 10.00 | 1.34 |
| **Лайки** | 0.00 | 15.80 | 70.00 | 9.13 |

Такие значения говорят о том, что показатели признаков могут отличаться на порядки, что, безусловно, приведет к искажению результатов. Чтобы избежать дисбаланса значений, выполняем стандартную нормализацию данных:

```
df = (df - df.mean())/df.std()
```

где метод `mean` возвращает средние значения признаков, а метод `std` — их средние квадратичные отклонения. Теперь метод `describe` показывает:

| | min | mean | max | std |
|---|---|---|---|---|
| **Просмотры** | –1.85 | 0.00 | 3.02 | 1.00 |
| **Время просмотра (часы)** | –1.90 | 0.00 | 3.57 | 1.00 |
| **Поделились** *–1.11 | 0.00 | 10.23 | 1.00 | |
| **Постоянные зрители** | –1.69 | 0.00 | 3.79 | 1.00 |
| **Новые комментарии** | –0.64 | 0.00 | 6.57 | 1.00 |
| **Отказались от подписки** | –1.08 | 0.00 | 10.27 | 1.00 |
| **Новые подписчики** | –1.84 | 0.00 | 3.55 | 1.00 |
| **Новые зрители** | –1.76 | 0.00 | 2.11 | 1.00 |
| **Среднее число просмотров одним пользователем** | –2.23 | 0.00 | 4.92 | 1.00 |
| **Уникальные зрители** | –1.81 | 0.00 | 2.39 | 1.00 |
| **CTR для значков видео (%)** | –3.84 | 0.00 | 2.67 | 1.00 |
| **Показы** | –1.61 | 0.00 | 8.22 | 1.00 |
| **Подписчики** | –6.63 | 0.00 | 2.80 | 1.00 |
| **Средний процент просмотра (%)** | –2.35 | 0.00 | 4.26 | 1.00 |

| | min | mean | max | std |
|---|---|---|---|---|
| **Процент лайков** | –8.92 | 0.00 | 0.77 | 1.00 |
| **Средняя продолжительность просмотра** | –3.08 | 0.00 | 4.25 | 1.00 |
| **Дизлайки** | –0.95 | 0.00 | 6.49 | 1.00 |
| **Лайки** | –2.38 | 0.00 | 5.93 | 1.00 |

Таким образом, теперь дисбаланс ликвидирован, все признаки приведены к единой шкале, и разница в единицах измерения не повлияет на результаты нашего исследования.

## 3.3 Стартовые значения параметров алгоритма

Мы будем варьировать два параметра: 1) процент `P`, который отводится на объем тестовой выборки, и 2) число кластеров `N`, на которое будут разбиваться обучающая и текстовая выборки. В начале алгоритма мы устанавливаем `P=10` (в цикле мы доведем значение этой переменной до 90 с шагом 10), `N=2` (это значение мы доведем до 10 с шагом 1).

## 3.4 Разбиение данных на train и test

Используя случайное разбиение, мы разделяем датафрей `df` на две части: обучающую выборку `df_train` и тестовую выборку d`f_test`. Для этого мы применяем метод `train_test_split` из модуля `model_selection` библиотеки `sklearn` к датафрейму `df`. В начале алгоритма указываем значение параметра `test_size=0.1` (это как раз соответствует 10% от общего объема выборки), затем будем его увеличивать.

## 3.5 Преобразование данных в массивы numpy

Далее мы будем использовать методы библиотеки `sklearn`, которая взаимодействует с массивам `numpy`. Поэтому прежде чем продолжать работу мы, пользуясь методом `to_numpy` библиотеки `pandas`, трансформируем наши датафремы в массивы `X_train` и `X_test`.

## 3.6 Агломеративная кластеризация

На этом шаге мы применяем методы агломеративной кластеризации к обеим выборкам: к обучающей выборке и к тестовой. Для этого используем метод `AgglomerativeClustering` из модуля `cluster` библиотеки `sklearn`, указывая в качестве значения параметра `n_clusters` нужное

количество кластеров (напомним, что на старте алгоритма `n_clusters=2`, а затем, по мере исполнения алгоритма, параметр будет увеличиваться).

### 3.6.1 Кластеризация обучающей выборки

Мы формируем объект `clust_train`, относящийся к классу агломеративной кластеризации и применяем к нему метод `fit` на массиве обучающих данных `X_train`. Затем, используя обученный объект `clust_train`, при помощи метода `labels_` получаем метки кластеров и записываем их в переменную `labels_test`.

### 3.6.2 Кластеризация тестовой выборки

Действуя аналогично на тестовых данных `X_test`, получаем метки кластеров и записываем их в переменную `labels_test`.

## 3.7 Вычисление центроидов

Для вычисления центроидов мы приписываем метки кластеров в качестве дополнительного признака к датафреймам `df_train` и `df_test`, после чего сами кластеры оказываются доступны после локализации.

### 3.7.1 Центроиды кластеров на обучающих данных

Добавляем к датафрейму `df_train` столбец 'labels' и заносим в него метки кластеров. Затем формируем пустой датафрейм `df_c_train` для того, чтобы заносить в него координаты центроидов. Его столбцы в точности соответствуют столбцам обучающего датафрейма `df_train`, а в качестве индекса мы приписываем ему уникальные значения меток, используя для этого метод `unique`, примененный к столбцу `df_train['labels']`. Сначала этот датафрейм содержит только значения `NaN`, то есть абсолютно пуст.

После этого мы запускаем цикл по всем уникальным значениям меток кластеров. Используя метод `loc` библиотеки pandas, мы локализуем датафрейм `df_train` по тем объектам, которые имеют одинаковые метки и усредняем получившуюся локализацию методом `mean`. В результате возникает строка, значения которой и есть координаты соответствующего центроида. Эту строку при помощи метода `iloc` мы записываем в датафрейм `df_c_train`, переходим к следующей метке и так до тех пор, пока не переберем все метки кластеров.

На выходе из этого цикла имеем датафрейм `df_c_train`, строки которого содержат координаты центроидов, а индекс указывает на номер соответствующей метки.

### 3.7.2 Центроиды кластеров на тестовых данных

Аналогично получаем датафрейм `df_c_test` с центроидами кластеров на тестовой выборке.

## 3.8 Вычисление дистанций между ближайшими центроидами

Заметим, что одна из серьезных проблем состоит в том, что кластеры в датафреймах `df_train` и `df_test` нумеруются произвольным образом. Если, например, какой-то кластер на обучающей выборке имеет номер 0, то это совсем не означает, что на тестовой выборке соответствующий кластер тоже будет иметь номер 0, наоборот: скорее всего, у него будет другой номер. Поэтому сопоставлять центроиды кластеров построчно, используя строки датафреймов `df_c_train` и `df_c_test`, это плохая идея. Сначала нужно получить пары ближайших центроидов и только после этого вычислять расстояния между ними.

Для этого мы применяем метод ближайших соседей. Идея состит в том, чтобы сначала найти пару центроидов, расстояние между которыми является наименьшим из всех возможных пар. После этого найденную пару исключить из рассмотрения, а к оставшимся центроидам применить тот же примем и найти пару с наименьшим расстоянием. И так далее в цикле по всем меткам. Для реализации нашей идеи мы формируем пустой список `dist_list`, в который будем заносить возникающие в процессе вычислений расстояния между ближайшими ценроидами.

### 3.8.1 Поиск ближайших соседей

Используя метод `NearestNeighbors` из модуля `neighbors` библиотеки `sklearn`, создаем объект `knn`. При этом в силу того, что нас интересует одна пара, расположенная на наименьшем расстоянии, устанавливаем значение параметра `n_neighbors=1`. Применяя метод `fit`, обучаем объект `knn` на обучающих данных `df_c_train`, и, наконец, применяем к обученному `knn` метод `kneighbors`, указав в качестве параметра тестовые данные `df_c_test`. Этот метод возвращает два объекта:

1. массив наименьших дистанций от объектов test'а `df_c_test` до объектов train'а `df_c_train`. Мы обозначаем этот массив `dist`,
2. массив с номерами объектов train'а, для которых были вычислены дистанции `dist`. Этот массив мы обозначаем `ind`.

### 3.8.2 Пополнение списка дистанций

Применяем метод `min` к массиву `dist` и, используя метод `append`, добавляем найденную дистанцию в список `dist_list`.

### 3.8.3 Удаление ближайших соседей

Используя метод `drop` библиотеки `pandas`, удаляем из датафреймов с центроидами `df_c_train` и `df_c_test` по номерам из массива ind.

Повторяем для всех меток кластеров, в результате чего получаем список дистанций между ближайшими центроидами (другими словами, созданный выше пустой список `dist_list` после этого оказывается заполненным).

## 3.9 Средняя дистанция между центроидами

К списку `dist_list` применяем метод `mean`, вычисляя, тем самым, среднюю дистанцию между центроидами. Получаем числовое значение, которое заносим в переменную `dist_mean`.

## 3.10 Цикл по случайным разбиениям

Напомним, что разбиение данных на обучающую и тестовую выборки производилось случайным образом. При повторном запуске этого фрагмента алгоритма неизбежно получится другой результат, потому что будет произведено новое случайное разбиение, и оно будет отличаться от предыдущего. Чтобы нивелировать случайные артефакты, мы повторяем действия, то есть, проводим разбиение на обучающую и тестовую выборки достаточное количество раз (с нашем исследовании мы применяли 100 запусков) и усредняем результат. Теперь в переменной `dist_mean` записана средняя дистанция между кластерами, дополнительно усредненная по множеству запусков, то есть, без случайных искажений.

## 3.11 Цикл по количеству кластеров

Увеличиваем число кластеров `N=N+1` и повторяем действия. По итогам этого цикла получаем список усредненных дистанций между центроидами, отвечающий различным числам кластеров, при разбиении датафрейма `df` на обучающие и тестовые данные в пропорции 90%:10% (то есть, первый столбец сводной таблицы, см. ниже, п. 4).

## 3.12 Цикл по объему теста

Увеличиваем процент тестовой выборки `P=P+10` и повторяем действия. По итогам цикла, получаем заполненную сводную таблицу (см. п. 4).

# 4 Результаты

Исполненный алгоритм дает нам сводную таблицу результатов. В ней строки маркируют число кластеров, на которое производится разбиение данных (от 2 до 10), столбцы — процент, который составляет тестовая выборка по отношению ко всему объему данных (от 10% до 90%), а в ячейках записаны усредненные дистанции между центроидами кластеров обучающей и тестовой выборок (от 0 и выше).

| | **10%** | **20%** | **30%** | **40%** | **50%** | **60%** | **70%** | **80%** | **90%** |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 3.8707 | 4.6629 | 5.6559 | 1.7830 | 0.7869 | 1.9051 | 4.5096 | 3.8337 | 2.3577 |
| **3** | 10.510 | 10.385 | 10.493 | 10.606 | 10.702 | 10.643 | 10.387 | 10.489 | 10.442 |
| **4** | 8.5213 | 7.8302 | 7.7736 | 7.8131 | 7.4245 | 7.8788 | 7.5779 | 7.9459 | 8.6633 |
| **5** | 8.3093 | 8.1927 | 7.8731 | 7.9363 | 7.9244 | 8.0607 | 7.7922 | 8.3361 | 8.3682 |
| **6** | 8.3101 | 7.9377 | 7.4084 | 7.2856 | 7.3403 | 7.3498 | 7.4885 | 7.5738 | 8.3224 |
| **7** | 7.4553 | 6.7890 | 6.8119 | 6.8467 | 6.9926 | 6.7211 | 7.0431 | 7.0354 | 7.7350 |
| **8** | 6.9855 | 6.5877 | 6.3294 | 6.3740 | 6.3154 | 6.4042 | 6.4384 | 6.5095 | 6.9303 |
| **9** | 6.4841 | 6.2184 | 6.2126 | 6.1051 | 6.0505 | 6.2150 | 6.0290 | 6.1549 | 6.5130 |
| **10** | 6.1066 | 5.9245 | 5.7804 | 5.8570 | 5.8209 | 5.8978 | 5.8354 | 5.9252 | 6.1896 |

## 4.1 Анализ по числу кластеров

Визуализируем дистанцию между центроидами кластеров, соотнесенную с числом кластеров (см. рис. 1). Это девять кривых, которые соответствуют девяти столбцам сводной таблицы. Каждая кривая получается при фиксированном значении объема тестовой выборки (от 10% до 90%).

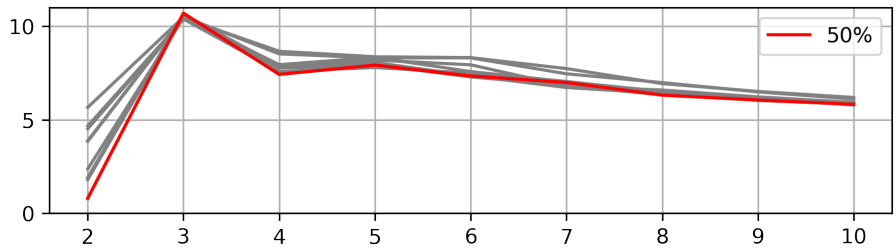


Рис. 1: Дистанция, соотнесенная с числом кластеров

Усредним кривые дистанций. Для этого проведем усреднение по столбцам сводной таблицы:

| | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% |
|---|---|---|---|---|---|---|---|---|---|
| | 7.3948 | 7.1699 | 7.1487 | 6.7341 | 6.5953 | 6.7863 | 7.0113 | 7.0893 | 7.2802 |

После усреднения кривые становятся горизонтальными линиями. Низшая линия соответствует проценту тестовой выборки с наилучшим значением нашей метрики (см. рис. 2). Таким образом, наименьшее от-

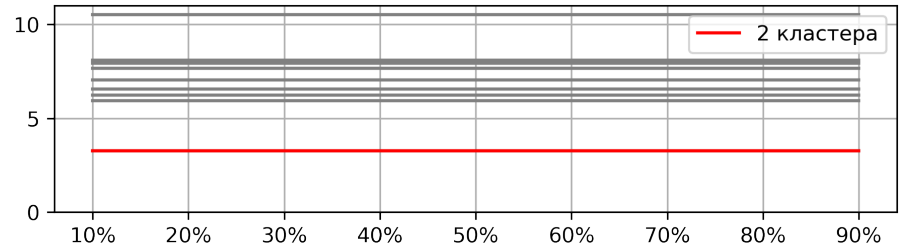


Рис. 2: Дистанция, усредненная по всем процентам test'а

клонение центроидов происходит при разбиении выборки в пропорции 50%:50%.

## 4.2 Анализ по объему тестовой выборки

Аналогично мы можем визуализировать дистанции между центроидами, соотнеся их с процентом тестовой выборки (см. рис. 3). Здесь девять кривых соответствуют девяти строкам сводной таблицы, и каждая кривая описывает ситуацию при фиксированном числе кластеров, на которые производится разбиение данных (от 2 до 10).

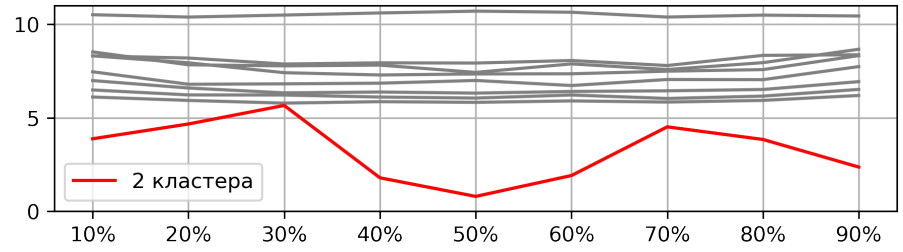


Рис. 3: Дистанция, соотнесенная с процентом тестовой выборки

Так же как и выше усредняем кривые дистанций. Для этого вычисляем средние значения по строкам сводной таблицы:

| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| | 3.2628 | 10.5178 | 7.9365 | 8.0881 | 7.6685 | 7.0478 | 6.5416 | 6.2203 | 5.9264 |

В результате получаем серию горизонтальных линий. Низшая линия соответствует числу кластеров с наилучшим значением нашей метрики.

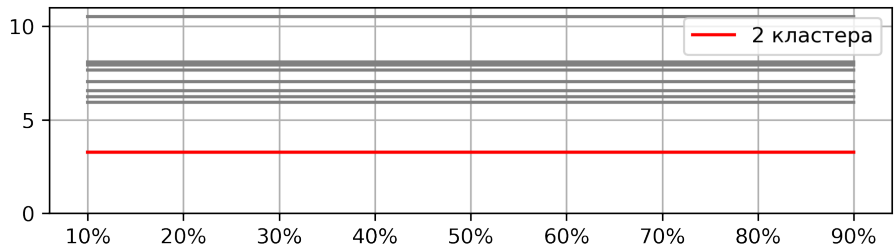


Рис. 4: Дистанция, усредненная по всем кластерам

Таким образом, оптимальное число кластеров равно 2.

# 5 Выводы

В большинстве задач машинного обучения нет заранее известной информации о количестве кластеров в данных, поэтому их число должно быть определено на основе самих данных. Это делает процесс определения оптимального количества кластеров весьма сложным и неоднозначным. Существует множество методов и метрик для определения оптимального числа кластеров (см. [2]), но в данной работе мы предложили и апробировали собственный метод, основанный на средних дистанциях между центроидами кластеров на обучающей и тестовой выборке. Предложенный нами метод подбора оптимального числа кластеров для агломеративной кластеризации показал хорошую работоспособность, четко выделив сразу два показателя: 1) оптимальное количество кластеров, 2) оптимальную пропорцию для разбиения данных на обучающую и тестовую выборку.

# 6 Литература

1. Хейдт М. Изучаем Pandas / М. Хейдт; — Москва: ДМК Пресс, 2018. — 438 с.
2. Бурков А. Машинное обучение без лишних слов / А. Бурков; — СПб: Питер, 2020. — 192 с.

3. .Вьюгин, В. В. Математические основы теории машинного обучения и прогнозирования / В. В. Вьюгин; — М.: МЦИМО. — 2013. — 387 с.
4. Бринк Х. Машинное обучение / Х. Бринк, Дж. Ричардс, М. Феверолф — СПб.: Питер, 2017. — 336 с.
5. Байков И.И., Семерова Е.А., Курмуков А.И. Метод ансамблирования алгоритмов кластеризации для решения задачи совместной кластеризации // Сенсорные системы. 2021. Т. 35. № 1. С. 43–49.
6. Паксашвили С.А. тестирования алгоритма кластеризации k-means в решении задачи кластеризации финансовых операций // В сборнике: СНК-2022. Материалы LXXII открытой международной студенческой научной конференции Московского Политеха. Москва, 2022. С. 347–353.