Кластеризация

<https://scikit-learn.ru/clustering/?ysclid=ls7i6a0d14289758130>

# 2.3. Кластеризация

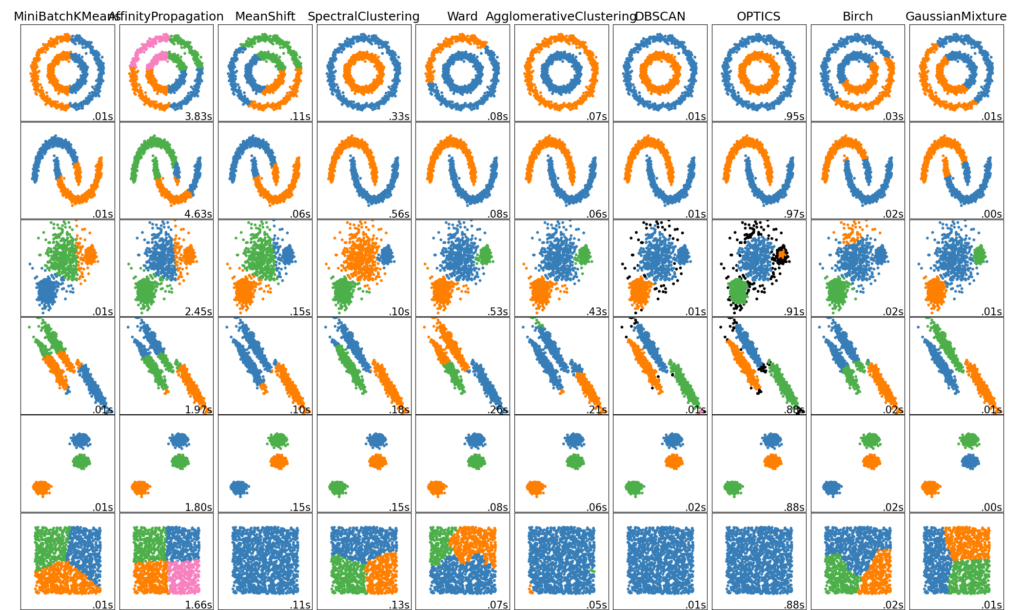
[Кластеризация](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%D0%BD%D1%8B%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7)  неразмеченных данных можно выполнить с помощью модуля **sklearn.cluster**.

Каждый алгоритм кластеризации имеет два варианта: класс, реализующий fit метод изучения кластеров на данных поезда, и функция, которая, учитывая данные поезда, возвращает массив целочисленных меток, соответствующих различным кластерам. Для класса метки над обучающими данными можно найти в labels\_ атрибуте.

Входные данные

Важно отметить, что алгоритмы, реализованные в этом модуле, могут принимать в качестве входных данных различные типы матриц. Все методы принимают стандартные матрицы данных формы . Их можно получить из классов модуля. Для , и можно также ввести матрицы подобия формы . Их можно получить из функций в модуле.(n\_samples, n\_features)[sklearn.feature\_extraction](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction) [AffinityPropagation](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html#sklearn.cluster.AffinityPropagation) [SpectralClustering](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html#sklearn.cluster.SpectralClustering) [DBSCAN](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN)(n\_samples, n\_samples)[sklearn.metrics.pairwise](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics.pairwise)

## 2.3.1. Обзор методов кластеризации



| **Название метода** | **Параметры** | **Масштабируемость** | **Использование** | **Геометрия (используемая метрика)** |
| --- | --- | --- | --- | --- |
| К-средник | число кластеров | Очень большое значение **n симплов** среднее **n\_clusters**вместе с Мини батчи K-средних | Универсальный, любой размер кластеров, плоская геометрия, не слишком много кластеров | Дистанция между точками |
| Афинное распространение | дамфинг, предпочтение выборки | Не масштабируется с помощью **n\_clusters** | Много кластеров, не равномерный размер кластера, неплоская геометрия | Дистанция графа (например граф ближайших соседей) |
| Средний сдвиг | пропускная способность | Не масштабируется с помощью **n\_clusters** | Много кластеров, не равномерный размер кластера, неплоская геометрия | Дистанция между точками |
| Спектральная кластеризация | число кластеров | Средняя n симплов маленькое **n\_clasters** | Мало кластеров, или размер кластера, неплоская геометрия | Дистанция графа (например граф ближайших соседей) |
| Иерархическая кластеризация | чилсо кластеров или порог расстояния | Большое n симплов и **n\_clasters** | Мало кластеров, возможно ограничене связей | Дистанция между точками |
| Агломеративная кластеризация | число кластеров, порог дистанции, тип связи, дистанция | Большое n симплов и **n\_clasters** | Мало кластеров, возможно ограничене связей и не Евклидовое расстояние | Любая попарная дистанция |
| DBSCAN | размер окрестности | Очень большое n симплов и среднее  **n\_clasters** | не плоская геометрия, неравномерный размер кластеров | Дистанция между ближайшими точками |
| OPTICS | Минимальное количество элементов в кластере | Очень большое n симплов и большое **n\_clasters** | не плоская геометрия, неравномерный размер кластеров, переменная плотность кластеров | Дистанция между точками |
| Гауссовская Смешаянная модель | много параметров | не масштабируемо | плоская геометрия, подходит для оценки плотности | Расстояния до центров Махаланобиса |
| Birch | факторы ветвления, порог, не обязательный глобальный кластер | Большое n симплов и  **n\_clasters** | Большой объем данных, удаление выбросов, сокращение данных | Евклидовое расстояние между точками |

Неплоская геометрия кластеризации полезно когда кластеры имеют специфичную форму, то есть многообразие и стандартное евклидовое расстояние в качестве метрики не подходят. Это случай возникает в двух верхних строках рисунка.

Гауссовская Смешаянная модель полезна для кластеризации описанные в другой статье документации, посвященная смешанным моделям. Метод K-средних можно расматривать как частный случай Гауссовской смешанной модели с равной ковариации для каждого компонента.

Методы[трансдуктивной](https://scikit-learn.org/stable/glossary.html#term-transductive) кластеризации (в отличие отметодов[индуктивной](https://scikit-learn.org/stable/glossary.html#term-inductive) кластеризации) не предназначены для применения к новым, невидимым данным.

## 2.3.2. K-средних

Эти [KMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans) данные алгоритмы кластеров пытаются отдельными образцы в п групп одинаковой дисперсии, сводя к минимуму критерия , известный как инерция или внутри-кластера сумм квадратов (см ниже). Этот алгоритм требует указания количества кластеров. Он хорошо масштабируется для большого количества образцов и используется в широком диапазоне областей применения во многих различных областях.

Алгоритм k-средних делит набор

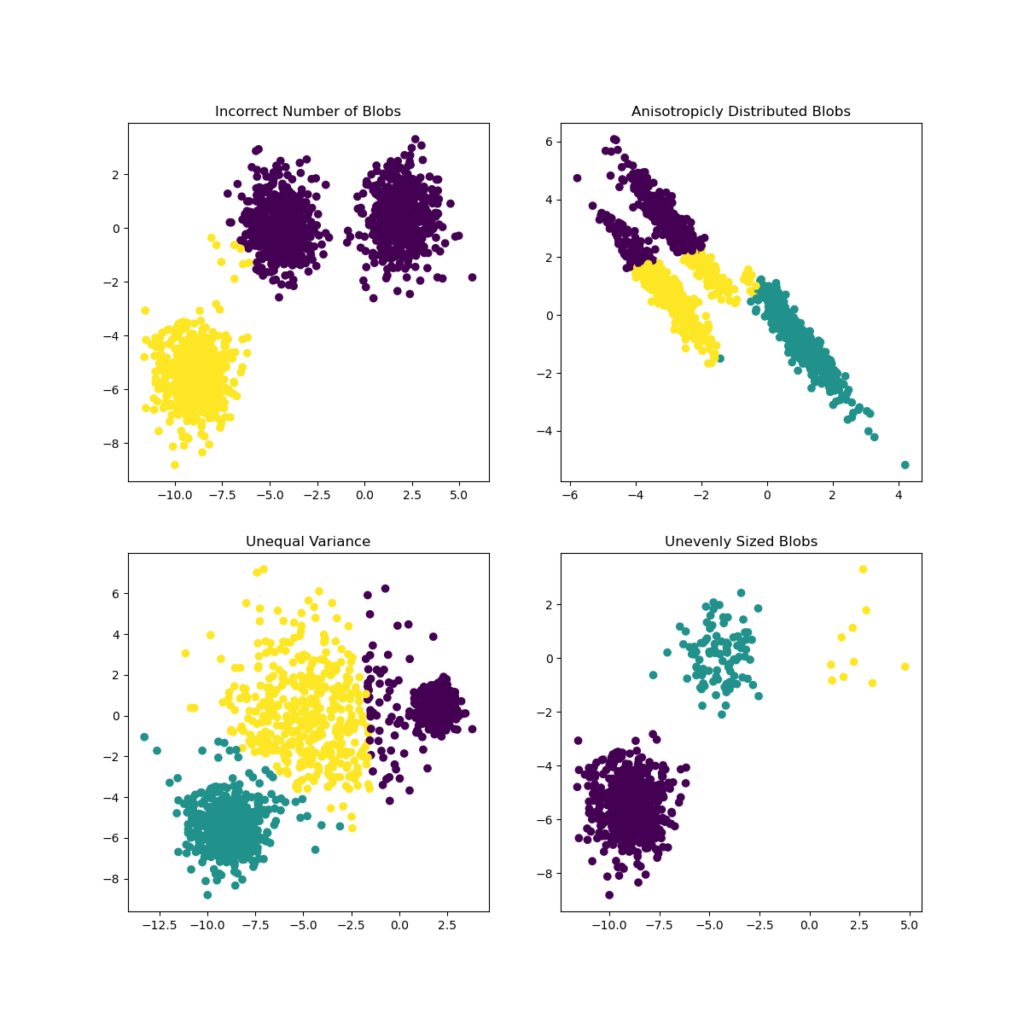
 образцы  в  непересекающиеся кластеры , каждый из которых описывается средним  образцов в кластере. Средние значения обычно называют «центроидами» кластера; обратите внимание, что это, как правило, не баллы из

, хотя они живут в одном пространстве.

Алгоритм K-средних нацелен на выбор центроидов, которые минимизируют **инерцию** или **критерий суммы квадратов внутри кластера** :

Инерцию можно определить как меру того, насколько кластеры внутренне связаны. Он страдает различными недостатками:

* Инерция предполагает, что кластеры выпуклые и изотропные, что не всегда так. Он плохо реагирует на удлиненные кластеры или коллекторы неправильной формы.
* Инерция — это не нормализованная метрика: мы просто знаем, что более низкие значения лучше, а ноль — оптимально. Но в очень многомерных пространствах евклидовы расстояния имеют тенденцию становиться раздутыми (это пример так называемого «проклятия размерности»). Выполнение алгоритма уменьшения размерности, такого как [анализ главных компонентов (PCA)](https://scikit-learn.ru/decomposition/) перед кластеризацией k-средних, может облегчить эту проблему и ускорить вычисления.

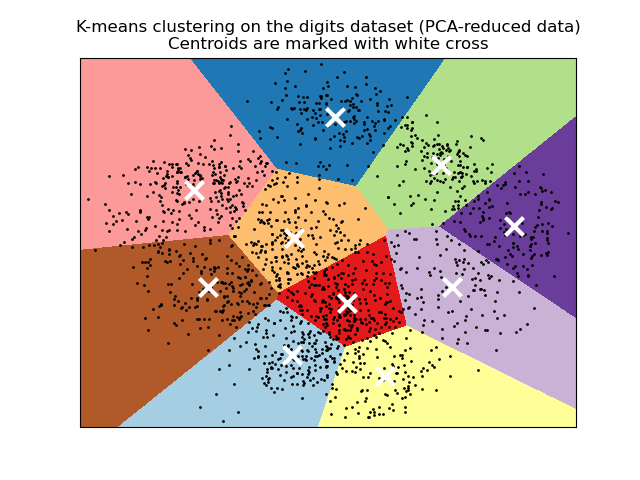


K-средних часто называют алгоритмом Ллойда. В общих чертах алгоритм состоит из трех шагов. На первом этапе выбираются начальные центроиды, а самый простой метод — выбрать

 образцы из набора данных

. После инициализации K-средних состоит из цикла между двумя другими шагами. Первый шаг присваивает каждой выборке ближайший центроид. На втором этапе создаются новые центроиды, взяв среднее значение всех выборок, назначенных каждому предыдущему центроиду. Вычисляется разница между старым и новым центроидами, и алгоритм повторяет эти последние два шага, пока это значение не станет меньше порогового значения. Другими словами, это повторяется до тех пор, пока центроиды не переместятся значительно.

K-средних эквивалентно алгоритму максимизации ожидания с маленькой, все равной диагональной ковариационной матрицей.



Алгоритм также можно понять через концепцию [диаграмм Вороного](https://en.wikipedia.org/wiki/Voronoi_diagram). Сначала рассчитывается диаграмма Вороного точек с использованием текущих центроидов. Каждый сегмент на диаграмме Вороного становится отдельным кластером. Во-вторых, центроиды обновляются до среднего значения каждого сегмента. Затем алгоритм повторяет это до тех пор, пока не будет выполнен критерий остановки. Обычно алгоритм останавливается, когда относительное уменьшение целевой функции между итерациями меньше заданного значения допуска. В этой реализации дело обстоит иначе: итерация останавливается, когда центроиды перемещаются меньше допуска.

По прошествии достаточного времени K-средние всегда будут сходиться, однако это может быть локальным минимумом. Это сильно зависит от инициализации центроидов. В результате вычисление часто выполняется несколько раз с разными инициализациями центроидов. Одним из способов решения этой проблемы является схема инициализации k-means++, которая была реализована в scikit-learn (используйте init='k-means++'параметр). Это инициализирует центроиды (как правило) удаленными друг от друга, что, вероятно, приводит к лучшим результатам, чем случайная инициализация, как показано в справочнике.

K-means++ также может вызываться независимо для выбора начальных значений для других алгоритмов кластеризации, [sklearn.cluster.kmeans\_plusplus](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.kmeans_plusplus.html#sklearn.cluster.kmeans_plusplus) подробности и примеры использования см. В разделе .

Алгоритм поддерживает выборочные веса, которые могут быть заданы параметром sample\_weight. Это позволяет присвоить некоторым выборкам больший вес при вычислении центров кластеров и значений инерции. Например, присвоение веса 2 выборке эквивалентно добавлению дубликата этой выборки в набор данных

.

Метод K-средних может использоваться для векторного квантования. Это достигается с помощью метода преобразования обученной модели [KMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans).

### 2.3.2.1. Низкоуровневый параллелизм

[KMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans) преимущества параллелизма на основе OpenMP через Cython. Небольшие порции данных (256 выборок) обрабатываются параллельно, что, кроме того, снижает объем памяти. Дополнительные сведения о том, как контролировать количество потоков, см. В наших заметках о [параллелизме](https://scikit-learn.ru/8-3-parallelism-resource-management-and-configuration/) .

Примеры:

* [Демонстрация предположений k-средних](https://scikit-learn.ru/example/demonstration-of-k-means-assumptions/) : демонстрация того, когда k-средних работает интуитивно, а когда нет
* [Демонстрация кластеризации K-средних на данных рукописных цифр](https://scikit-learn.ru/example/a-demo-of-k-means-clustering-on-the-handwritten-digits-data/) : Кластеризация рукописных цифр

Рекомендации:

* [«K-means ++: преимущества тщательного посева»](http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf) Артур, Дэвид и Сергей Васильвицкий, Труды восемнадцатого ежегодного симпозиума ACM-SIAM по дискретным алгоритмам , Общество промышленной и прикладной математики (2007)

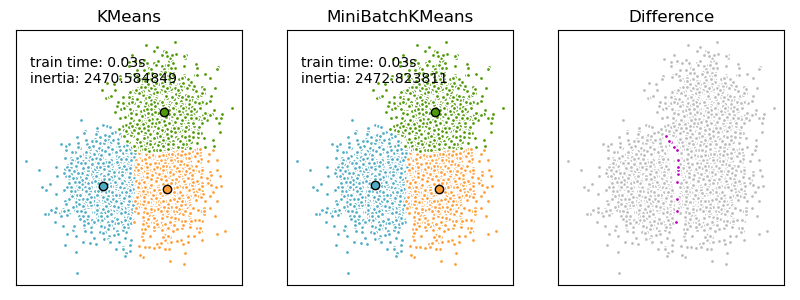
### 2.3.2.2. Мини-партия K-средних

Это [MiniBatchKMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans) вариант [KMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans) алгоритма, который использует мини-пакеты для сокращения времени вычислений, но при этом пытается оптимизировать ту же целевую функцию. Мини-пакеты — это подмножества входных данных, которые выбираются случайным образом на каждой итерации обучения. Эти мини-пакеты резко сокращают объем вычислений, необходимых для схождения к локальному решению. В отличие от других алгоритмов, которые сокращают время сходимости k-средних, мини-пакетные k-средние дают результаты, которые, как правило, лишь немного хуже, чем стандартный алгоритм.

Алгоритм повторяется между двумя основными шагами, аналогично обычным k-средним. На первом этапе

образцы выбираются случайным образом из набора данных, чтобы сформировать мини-пакет. Затем они присваиваются ближайшему центроиду. На втором этапе обновляются центроиды. В отличие от k-средних, это делается для каждой выборки. Для каждой выборки в мини-пакете назначенный центроид обновляется путем взятия среднего потокового значения выборки и всех предыдущих выборок, назначенных этому центроиду. Это приводит к снижению скорости изменения центроида с течением времени. Эти шаги выполняются до тех пор, пока не будет достигнута сходимость или заранее определенное количество итераций.

[MiniBatchKMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MiniBatchKMeans.html#sklearn.cluster.MiniBatchKMeans) сходится быстрее, чем [KMeans](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans), но качество результатов снижается. На практике эта разница в качестве может быть довольно небольшой, как показано в примере и цитированной ссылке.



Примеры:

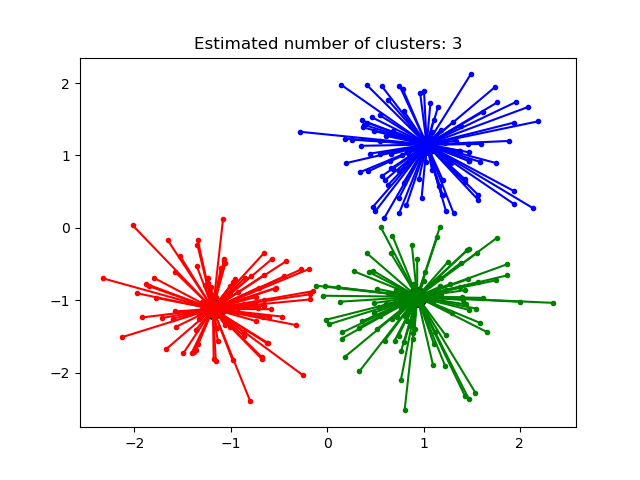
* [Сравнение алгоритмов кластеризации K-средних и MiniBatchKMeans](https://scikit-learn.ru/example/comparison-of-the-k-means-and-minibatchkmeans-clustering-algorithms/) : сравнение KMeans и MiniBatchKMeans
* [Кластеризация текстовых документов с использованием k-средних](https://scikit-learn.ru/example/clustering-text-documents-using-k-means/) : кластеризация документов с использованием разреженных MiniBatchKMeans
* [Онлайн обучение словаря частей лиц](https://scikit-learn.ru/example/online-learning-of-a-dictionary-of-parts-of-faces/)

Рекомендации:

* [«Кластеризация K-средних в веб-масштабе»](https://www.eecs.tufts.edu/~dsculley/papers/fastkmeans.pdf) Д. Скалли, Труды 19-й международной конференции по всемирной паутине (2010 г.)

## 2.3.3. Распространения близости (Affinity Propagation)

[AffinityPropagation](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AffinityPropagation.html#sklearn.cluster.AffinityPropagation) создает кластеры, отправляя сообщения между парами образцов до схождения. Затем набор данных описывается с использованием небольшого количества образцов, которые определяются как наиболее репрезентативные для других образцов. Сообщения, отправляемые между парами, представляют пригодность одного образца быть образцом другого, который обновляется в ответ на значения из других пар. Это обновление происходит итеративно до сходимости, после чего выбираются окончательные образцы и, следовательно, дается окончательная кластеризация.



Метод Распространения близости может быть интересным, поскольку он выбирает количество кластеров на основе предоставленных данных. Для этой цели двумя важными параметрами являются предпочтение , которое контролирует, сколько экземпляров используется, и коэффициент демпфирования, который снижает ответственность и сообщения о доступности, чтобы избежать числовых колебаний при обновлении этих сообщений.

Главный недостаток метода Распространения близости — его сложность. Алгоритм имеет временную сложность порядка

, где  количество образцов и  — количество итераций до сходимости. Далее, сложность памяти порядка

если используется плотная матрица подобия, но может быть сокращена, если используется разреженная матрица подобия. Это делает метод Распространения близости наиболее подходящим для наборов данных малого и среднего размера.

Примеры:

* [Демонстрация алгоритма кластеризации распространения близости](https://scikit-learn.ru/example/demo-of-affinity-propagation-clustering-algorithm/): метода Распространения близости на синтетических наборах данных 2D с 3 классами.
* [Визуализация структуры фондового рынка](https://scikit-learn.ru/example/visualizing-the-stock-market-structure/) Финансовые временные ряды метода Распространения близости для поиска групп компаний

**Описание алгоритма:** сообщения, отправляемые между точками, относятся к одной из двух категорий. Во-первых, это ответственность

, которое является накопленным свидетельством того, что образец  должен быть образцом для образца . Второе — доступность  что является накопленным свидетельством того, что образец  следует выбрать образец  быть его образцом, и учитывает значения для всех других образцов, которые

должен быть образцом. Таким образом, образцы выбираются по образцам, если они (1) достаточно похожи на многие образцы и (2) выбираются многими образцами, чтобы быть репрезентативными.

Более формально ответственность за образец

 быть образцом образца i дан кем-то: Где  сходство между образцами  а также . Наличие образца  быть образцом образца  дан кем-то: Начнем с того, что все значения для  и  aустановлены в ноль, и расчет каждой итерации повторяется до сходимости. Как обсуждалось выше, чтобы избежать числовых колебаний при обновлении сообщений, коэффициент демпфирования  вводится в итерационный процесс: где

 указывает время итерации.

## 2.3.4. Средний сдвиг

[MeanShift](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html#sklearn.cluster.MeanShift) кластеризация направлена ​​на обнаружение капель в образцах с плавной плотностью. Это алгоритм на основе центроидов, который работает, обновляя кандидатов в центроиды, чтобы они были средними точками в данном регионе. Затем эти кандидаты фильтруются на этапе постобработки, чтобы исключить почти дубликаты и сформировать окончательный набор центроидов.

Учитывая центроид кандидата

 для итерации , кандидат обновляется в соответствии со следующим уравнением: Где  это соседство образцов на заданном расстоянии вокруг  а также  — вектор среднего сдвига, который вычисляется для каждого центроида, который указывает на область максимального увеличения плотности точек. Это вычисляется с использованием следующего уравнения, эффективно обновляющего центроид до среднего значения выборок в его окрестности:

Алгоритм автоматически устанавливает количество кластеров, вместо того, чтобы полагаться на параметр bandwidth, который определяет размер области для поиска. Этот параметр можно установить вручную, но можно оценить с помощью предоставленной estimate\_bandwidth функции, которая вызывается, если полоса пропускания не задана.

Алгоритм не отличается высокой масштабируемостью, так как он требует многократного поиска ближайшего соседа во время выполнения алгоритма. Алгоритм гарантированно сходится, однако алгоритм прекратит итерацию, когда изменение центроидов будет небольшим.

Маркировка нового образца выполняется путем нахождения ближайшего центроида для данного образца.

Примеры:

* [Демонстрация алгоритма кластеризации среднего сдвига: кластеризация](https://scikit-learn.ru/example/a-demo-of-the-mean-shift-clustering-algorithm/) среднего сдвига на синтетических наборах данных 2D с 3 классами.

Рекомендации:

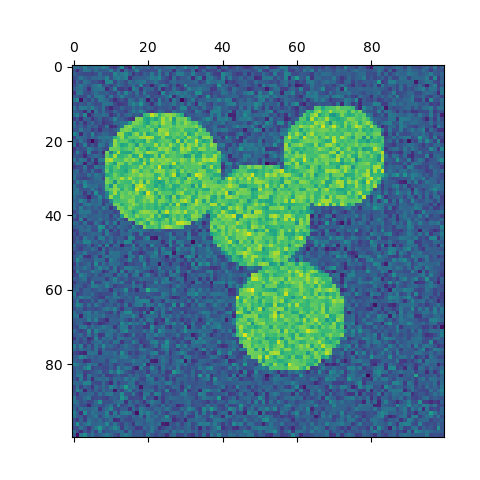
* [«Средний сдвиг: надежный подход к анализу пространства признаков».](http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.8968&rep=rep1&type=pdf)Д. Команичиу и П. Меер, IEEE Transactions on Pattern Analysis and Machine Intelligence (2002)

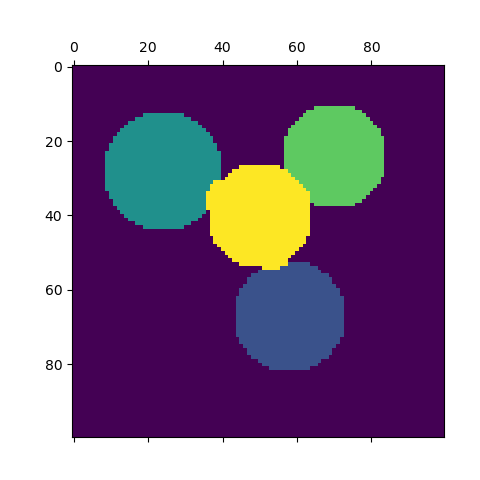
## 2.3.5. Спектральная кластеризация

[SpectralClustering](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html#sklearn.cluster.SpectralClustering) выполняет низкоразмерное встраивание матрицы аффинности между выборками с последующей кластеризацией, например, с помощью K-средних, компонентов собственных векторов в низкоразмерном пространстве. Это особенно эффективно с точки зрения вычислений, если матрица аффинности является разреженной, а amgрешатель используется для проблемы собственных значений (обратите внимание, amg что решающая программа требует, чтобы был установлен модуль [pyamg](https://github.com/pyamg/pyamg) ).

Текущая версия SpectralClustering требует, чтобы количество кластеров было указано заранее. Это хорошо работает для небольшого количества кластеров, но не рекомендуется для многих кластеров.

Для двух кластеров SpectralClustering решает выпуклую релаксацию проблемы [нормализованных разрезов](https://people.eecs.berkeley.edu/~malik/papers/SM-ncut.pdf) на графе подобия: разрезание графа пополам так, чтобы вес разрезаемых рёбер был мал по сравнению с весами рёбер внутри каждого кластера. Этот критерий особенно интересен при работе с изображениями, где вершинами графа являются пиксели, а веса ребер графа подобия вычисляются с использованием функции градиента изображения.





Предупреждение:

Преобразование расстояния в хорошее сходство

Обратите внимание, что если значения вашей матрицы подобия плохо распределены, например, с отрицательными значениями или с матрицей расстояний, а не с подобием, спектральная проблема будет сингулярной, а проблема неразрешима. В этом случае рекомендуется применить преобразование к элементам матрицы. Например, в случае матрицы расстояний со знаком обычно применяется тепловое ядро:

similarity = np.exp(-beta \* distance / distance.std())

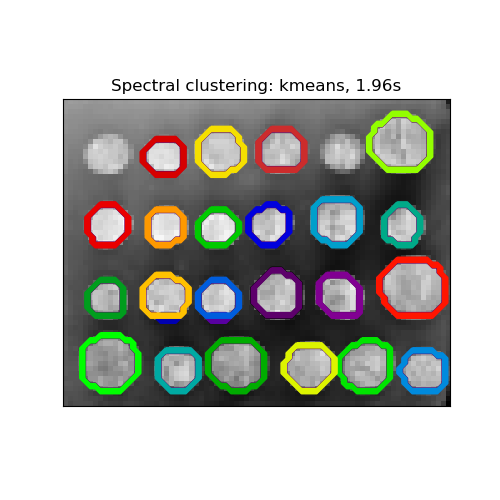
См. Примеры такого приложения.

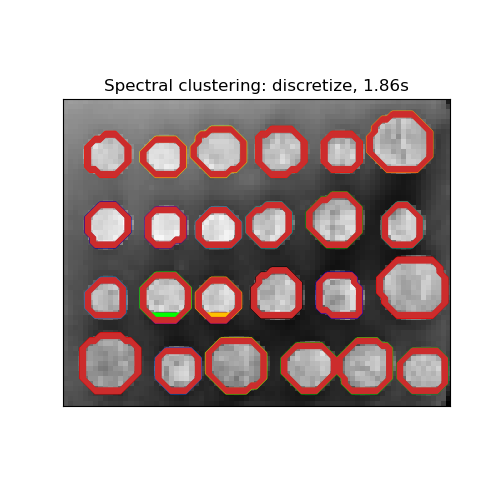
Примеры:

* [Спектральная кластеризация для сегментации изображения](https://scikit-learn.ru/example/spectral-clustering-for-image-segmentation/) : сегментирование объектов на шумном фоне с использованием спектральной кластеризации.
* [Сегментирование изображения греческих монет по регионам](https://scikit-learn.ru/example/segmenting-the-picture-of-greek-coins-in-regions/) : спектральная кластеризация для разделения изображения монет по регионам.

### 2.3.5.1. Различные стратегии присвоения меток

Могут использоваться различные стратегии присвоения меток, соответствующие assign\_labels параметру [SpectralClustering](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html#sklearn.cluster.SpectralClustering). "kmeans" стратегия может соответствовать более тонким деталям, но может быть нестабильной. В частности, если вы не контролируете random\_state, он может не воспроизводиться от запуска к запуску, так как это зависит от случайной инициализации. Альтернативная "discretize" стратегия воспроизводима на 100%, но имеет тенденцию создавать участки довольно ровной и геометрической формы.

**assign\_labels=»kmeans»**

**assign\_labels=»discretize»**

### 2.3.5.2. Графики спектральной кластеризации

Спектральная кластеризация также может использоваться для разбиения графов через их спектральные вложения. В этом случае матрица аффинности является матрицей смежности графа, а SpectralClustering инициализируется с помощью affinity='precomputed':



1

>>> from sklearn.cluster import SpectralClustering

2

>>> sc = SpectralClustering(3, affinity='precomputed', n\_init=100,

3

... assign\_labels='discretize')

4

>>> sc.fit\_predict(adjacency\_matrix)

Рекомендации:

* [«Учебное пособие по спектральной кластеризации»](http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.165.9323) Ульрике фон Люксбург, 2007 г.
* [«Нормализованные разрезы и сегментация изображения»](http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.160.2324) Джианбо Ши, Джитендра Малик, 2000 г.
* [«Случайный взгляд на спектральную сегментацию»](http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.33.1501) Марина Мейла, Цзяньбо Ши, 2001
* [«О спектральной кластеризации: анализ и алгоритм»](http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.8100) Эндрю Й. Нг, Майкл И. Джордан, Яир Вайс, 2001 г.
* [«Предварительно обусловленная спектральная кластеризация для задачи потокового графа стохастического разбиения блоков»](https://arxiv.org/abs/1708.07481) Давид Жужунашвили, Андрей Князев

## 2.3.6. Иерархическая кластеризация

Иерархическая кластеризация — это общее семейство алгоритмов кластеризации, которые создают вложенные кластеры путем их последовательного слияния или разделения. Эта иерархия кластеров представлена ​​в виде дерева (или дендрограммы). Корень дерева — это уникальный кластер, который собирает все образцы, а листья — это кластеры только с одним образцом. См. Страницу в [Википедии](https://en.wikipedia.org/wiki/Hierarchical_clustering) для получения более подробной информации.

В [AgglomerativeClustering](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering) объекте выполняет иерархическую кластеризацию с использованием подхода снизу вверх: каждый начинает наблюдения в своем собственном кластере, и кластеры последовательно объединены вместе. Критерии связывания определяют метрику, используемую для стратегии слияния:

* **Ward** минимизирует сумму квадратов разностей во всех кластерах. Это подход с минимизацией дисперсии, и в этом смысле он аналогичен целевой функции k-средних, но решается с помощью агломеративного иерархического подхода.
* **Максимальное** (**Maximum**) или **полное связывание** (**complete linkage**) сводит к минимуму максимальное расстояние между наблюдениями пар кластеров.
* **Среднее связывание** (**Average linkage**) минимизирует среднее расстояние между всеми наблюдениями пар кластеров.
* **Одиночная связь** (**Single linkage**) минимизирует расстояние между ближайшими наблюдениями пар кластеров.

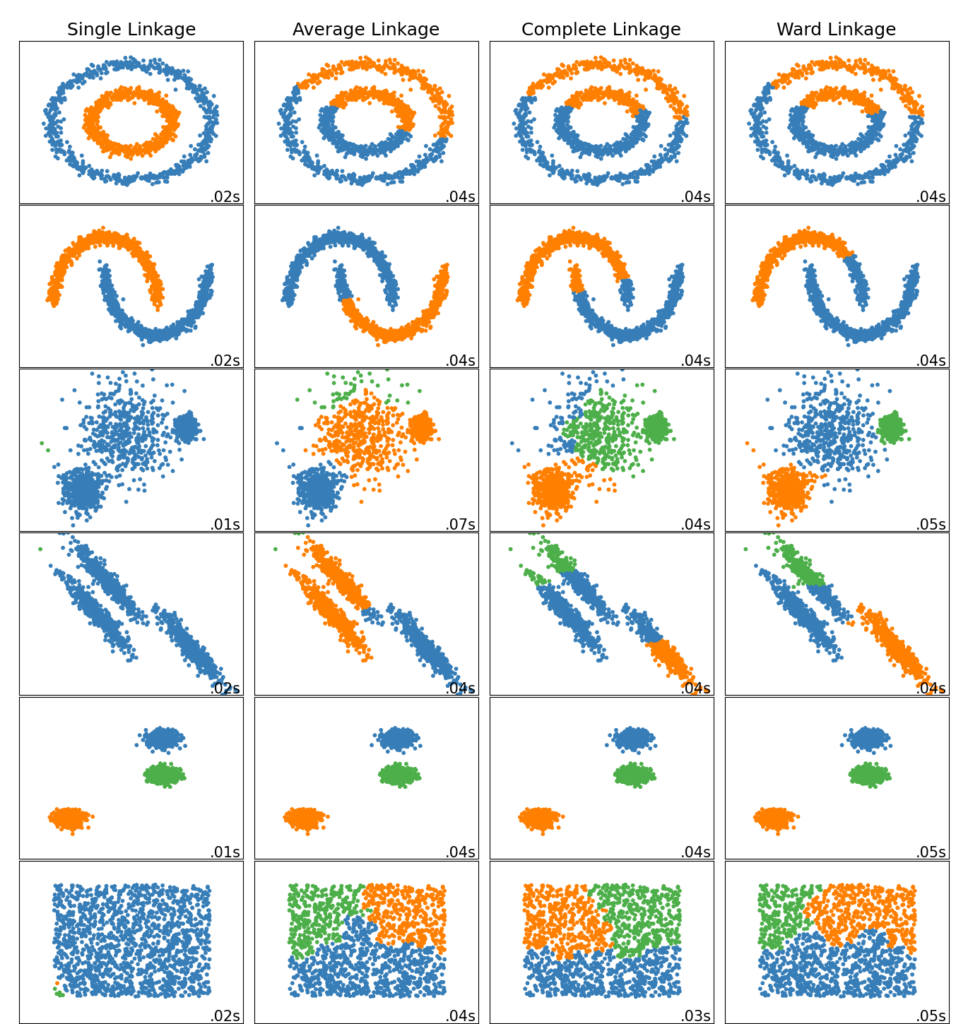
[AgglomerativeClustering](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering) может также масштабироваться до большого количества выборок, когда он используется вместе с матрицей связности, но требует больших вычислительных затрат, когда между выборками не добавляются ограничения связности: он рассматривает на каждом шаге все возможные слияния.

FeatureAgglomeration

В [FeatureAgglomeration](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.FeatureAgglomeration.html#sklearn.cluster.FeatureAgglomeration)использует агломерационную кластеризацию группироваться функции , которые очень похоже, тем самым уменьшая количество функций. Это инструмент уменьшения размерности, см. « [Уменьшение размерности без учителя»](https://scikit-learn.org/stable/modules/unsupervised_reduction.html#data-reduction) .

### 2.3.6.1. Различные типы связи: Ward, полный, средний и одиночный связь

[AgglomerativeClustering](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering) поддерживает стратегии **Ward**, одиночного, среднего и полного связывания.



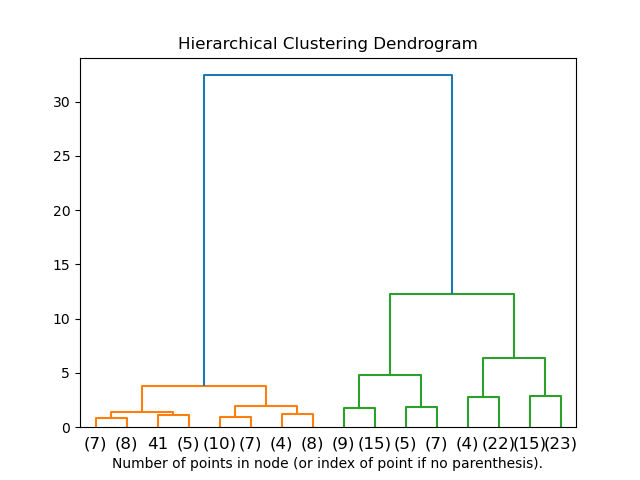
Агломеративный кластер ведет себя по принципу «богатый становится богатее», что приводит к неравномерному размеру кластера. В этом отношении одинарная связь — худшая стратегия, и Ward дает самые обычные размеры. Однако сродство (или расстояние, используемое при кластеризации) нельзя изменять с помощью Уорда, поэтому для неевклидовых показателей хорошей альтернативой является среднее связывание. Одиночная связь, хотя и не устойчива к зашумленным данным, может быть вычислена очень эффективно и поэтому может быть полезна для обеспечения иерархической кластеризации больших наборов данных. Одиночная связь также может хорошо работать с неглобулярными данными.

Примеры:

* [Различная агломеративная кластеризация по двумерному встраиванию цифр](https://scikit-learn.ru/example/various-agglomerative-clustering-on-a-2d-embedding-of-digits/) : изучение различных стратегий связывания в реальном наборе данных.

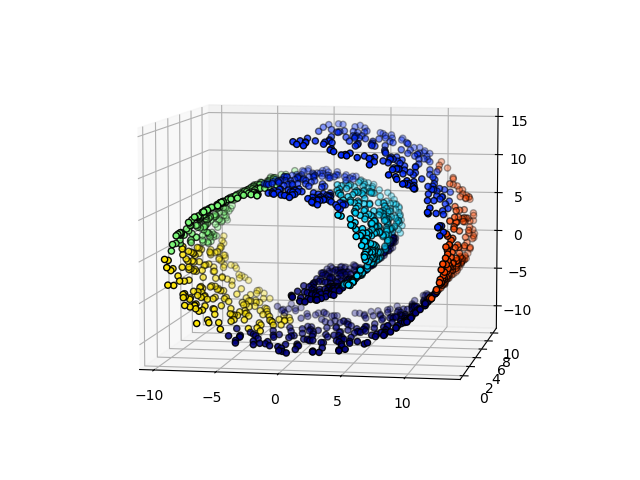
### 2.3.6.2. Визуализация кластерной иерархии

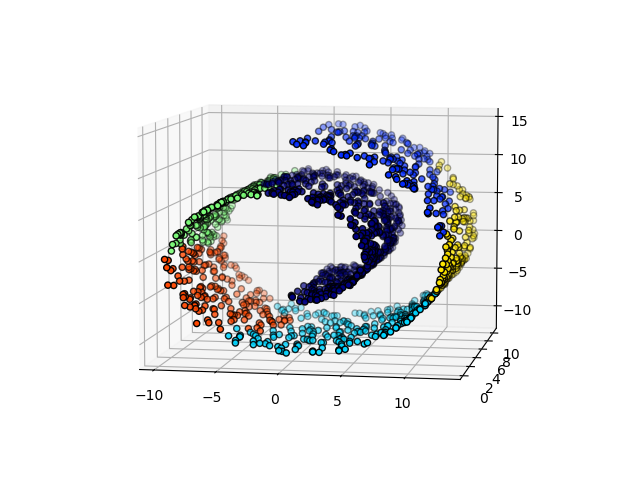
Можно визуализировать дерево, представляющее иерархическое слияние кластеров, в виде дендрограммы. Визуальный осмотр часто может быть полезен для понимания структуры данных, особенно в случае небольших размеров выборки.



### 2.3.6.3. Добавление ограничений связи

Интересным аспектом [AgglomerativeClustering](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering) является то, что к этому алгоритму могут быть добавлены ограничения связности (только соседние кластеры могут быть объединены вместе) через матрицу связности, которая определяет для каждой выборки соседние выборки, следующие заданной структуре данных. Например, в приведенном ниже примере швейцарского рулона ограничения связности запрещают объединение точек, которые не являются смежными на швейцарском рулоне, и, таким образом, избегают образования кластеров, которые проходят через перекрывающиеся складки рулона.





Эти ограничения полезны для наложения определенной локальной структуры, но они также ускоряют алгоритм, особенно когда количество выборок велико.

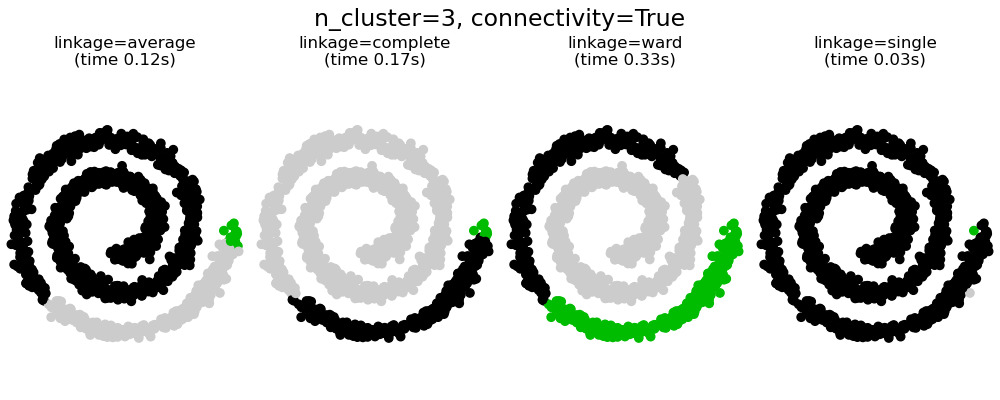
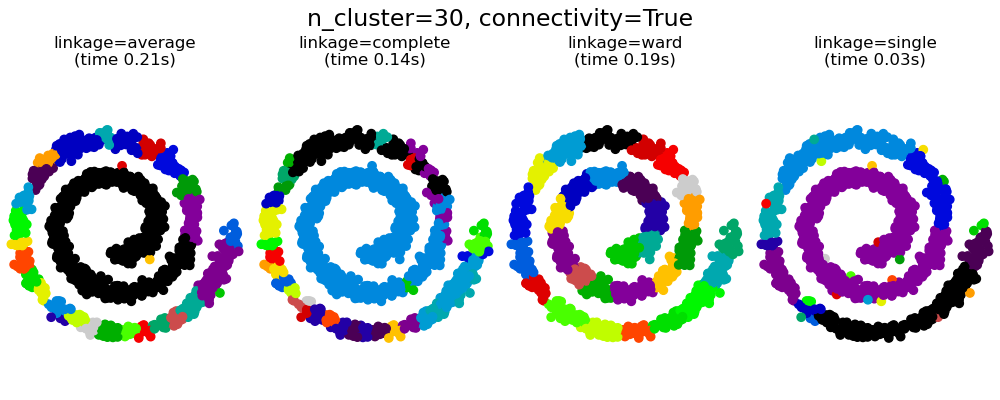
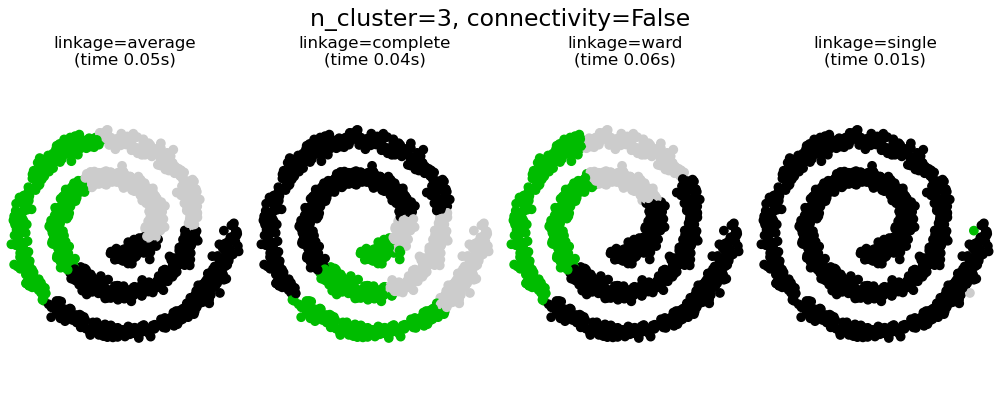
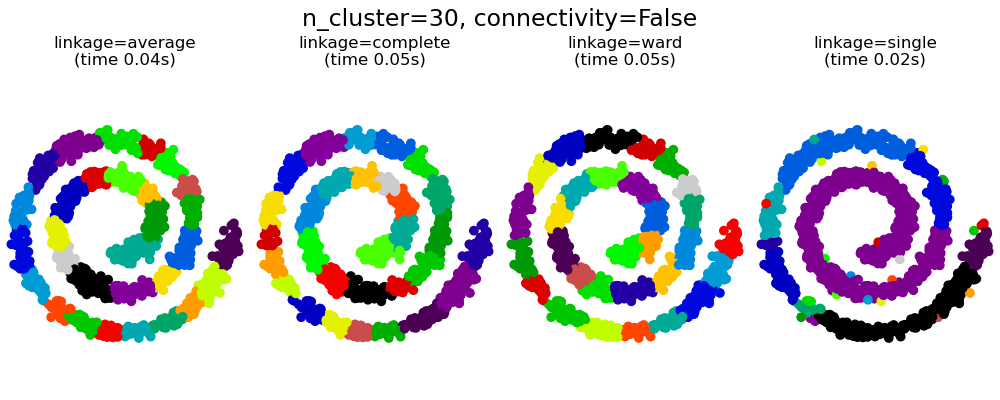
Ограничения связности накладываются через матрицу связности: scipy разреженную матрицу, которая имеет элементы только на пересечении строки и столбца с индексами набора данных, которые должны быть связаны. Эта матрица может быть построена из априорной информации: например, вы можете захотеть сгруппировать веб-страницы, объединяя только страницы со ссылкой, указывающей одну на другую. Это также можно узнать из данных, например, используя [sklearn.neighbors.kneighbors\_graph](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.kneighbors_graph.html#sklearn.neighbors.kneighbors_graph) для ограничения слияния до ближайших соседей, как в [этом примере](https://scikit-learn.ru/example/agglomerative-clustering-with-and-without-structure/) , или с помощью, [sklearn.feature\_extraction.image.grid\_to\_graph](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.image.grid_to_graph.html#sklearn.feature_extraction.image.grid_to_graph)чтобы разрешить слияние только соседних пикселей на изображении, как в примере с [монетой](https://scikit-learn.ru/example/a-demo-of-structured-ward-hierarchical-clustering-on-an-image-of-coins/).

Примеры:

* [Демонстрация структурированной иерархической кластеризации Варда на изображении монет](https://scikit-learn.ru/example/a-demo-of-structured-ward-hierarchical-clustering-on-an-image-of-coins/) : Кластеризация Варда для разделения изображения монет на регионы.
* [Иерархическая кластеризация: структурированная и неструктурированная палата](https://scikit-learn.ru/example/hierarchical-clustering-structured-vs-unstructured-ward/) : пример алгоритма Уорда в швейцарской системе, сравнение структурированных подходов и неструктурированных подходов.
* [Агломерация признаков против одномерного выбора](https://scikit-learn.ru/example/feature-agglomeration-vs-univariate-selection/): пример уменьшения размерности с агломерацией признаков на основе иерархической кластеризации Уорда.
* [Агломеративная кластеризация со структурой и без нее](https://scikit-learn.ru/example/agglomerative-clustering-with-and-without-structure/)

Предупреждение Ограничения по подключению с одиночным, средним и полным подключением

Ограничения связности и одиночная, полная или средняя связь могут усилить аспект агломеративной кластеризации «богатый становится еще богаче», особенно если они построены на [sklearn.neighbors.kneighbors\_graph](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.kneighbors_graph.html#sklearn.neighbors.kneighbors_graph). В пределе небольшого числа кластеров они имеют тенденцию давать несколько макроскопически занятых кластеров и почти пустые. (см. обсуждение в разделе «[Агломеративная кластеризация со структурой и без нее»](https://scikit-learn.ru/example/agglomerative-clustering-with-and-without-structure/) ). Одиночная связь — самый хрупкий вариант связи в этом вопросе.



### 2.3.6.4. Варьируя метрику

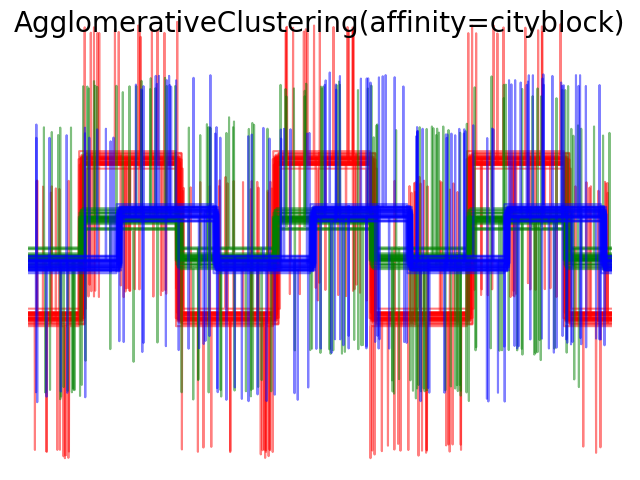
Одиночная, средняя и полная связь может использоваться с различными расстояниями (или сходствами), в частности евклидовым расстоянием ( l2 ), манхэттенским расстоянием (или Cityblock, или l1 ), косинусным расстоянием или любой предварительно вычисленной матрицей аффинности.

* Расстояние l1 часто хорошо для разреженных функций или разреженного шума: то есть многие из функций равны нулю, как при интеллектуальном анализе текста с использованием вхождений редких слов.
* косинусное расстояние интересно тем, что оно инвариантно к глобальному масштабированию сигнала.

Рекомендации по выбору метрики — использовать метрику, которая максимизирует расстояние между выборками в разных классах и минимизирует его внутри каждого класса.







Примеры:

* [Агломеративная кластеризация с разными метриками](https://scikit-learn.ru/example/agglomerative-clustering-with-different-metrics/)

## 2.3.7. DBSCAN

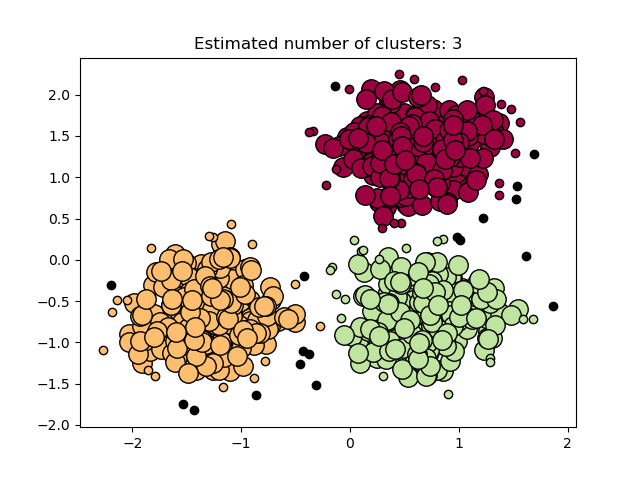
Алгоритм [DBSCAN](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN) рассматривает кластеры , как участки высокой плотности , разделенных районах с низкой плотностью. Из-за этого довольно общего представления кластеры, обнаруженные с помощью DBSCAN, могут иметь любую форму, в отличие от k-средних, которое предполагает, что кластеры имеют выпуклую форму. Центральным компонентом DBSCAN является концепция образцов керна , то есть образцов, находящихся в областях с высокой плотностью. Таким образом, кластер представляет собой набор образцов керна, каждый из которых находится близко друг к другу (измеряется с помощью некоторой меры расстояния), и набор образцов, не относящихся к керну, которые близки к образцу керна (но сами не являются образцами керна). В алгоритме есть два параметра, min\_samples и eps, которые формально определяют, что мы имеем в виду, когда говорим «плотный» . Выше min\_samples или ниже eps указывают на более высокую плотность, необходимую для формирования кластера.

Более формально мы определяем образец керна как образец в наборе данных, так что существуют min\_samples другие образцы на расстоянии eps, которые определены как соседи образца керна. Это говорит нам о том, что основной образец находится в плотной области векторного пространства. Кластер — это набор образцов керна, который можно построить путем рекурсивного взятия образца керна, поиска всех его соседей, которые являются образцами керна, поиска всех их соседей, которые являются образцами керна, и т. Кластер также имеет набор неосновных выборок, которые представляют собой выборки, которые являются соседями керновой выборки в кластере, но сами не являются основными выборками. Интуитивно эти образцы находятся на периферии кластера.

Любой образец керна по определению является частью кластера. Любая выборка, не являющаяся образцом керна и находящаяся по крайней мере eps на расстоянии от любой выборки керна, считается алгоритмом выбросом.

Хотя параметр в min\_samples первую очередь контролирует устойчивость алгоритма к шуму (на зашумленных и больших наборах данных может быть желательно увеличить этот параметр), параметр epsимеет решающее значение для правильного выбора для набора данных и функции расстояния и обычно не может быть оставлен на значение по умолчанию. Он контролирует локальное окружение точек. Если выбран слишком маленький размер, большая часть данных вообще не будет кластеризована (и помечена как -1 для «шума»). Если выбран слишком большой, близкие кластеры будут объединены в один кластер, и в конечном итоге весь набор данных будет возвращен как единый кластер. Некоторые эвристики для выбора этого параметра обсуждались в литературе, например, на основе перегиба на графике расстояний до ближайших соседей (как обсуждается в ссылках ниже).

На рисунке ниже цвет указывает на принадлежность к кластеру, а большие кружки обозначают образцы керна, найденные алгоритмом. Меньшие кружки — это неосновные образцы, которые все еще являются частью кластера. Более того, выбросы обозначены ниже черными точками.



Примеры:

* [Демонстрация алгоритма кластеризации DBSCAN](https://scikit-learn.ru/example/demo-of-dbscan-clustering-algorithm/)

Выполнение

Алгоритм DBSCAN является детерминированным, всегда генерируя одни и те же кластеры, когда им предоставляются одни и те же данные в одном порядке. Однако результаты могут отличаться, если данные предоставляются в другом порядке. Во-первых, даже если основные образцы всегда будут назначаться одним и тем же кластерам, метки этих кластеров будут зависеть от порядка, в котором эти образцы встречаются в данных. Во-вторых, что более важно, кластеры, которым назначены неосновные выборки, могут различаться в зависимости от порядка данных. Это может произойти, если образец неосновного керна находится на расстоянии меньше, чем epsдва образца керна в разных кластерах. Согласно треугольному неравенству эти два образца керна должны быть дальше, чемepsдруг от друга, иначе они были бы в одном кластере. Неосновная выборка назначается тому кластеру, который сгенерирован первым при передаче данных, поэтому результаты будут зависеть от порядка данных.

Текущая реализация использует шаровые деревья и kd-деревья для определения окрестности точек, что позволяет избежать вычисления полной матрицы расстояний (как это было сделано в версиях scikit-learn до 0.14). Сохранена возможность использования пользовательских метрик; подробности см NearestNeighbors.

Потребление памяти для больших размеров выборки

Эта реализация по умолчанию неэффективна с точки зрения памяти, поскольку она строит полную матрицу попарного сходства в случае, когда нельзя использовать kd-деревья или шаровые деревья (например, с разреженными матрицами). Эта матрица будет потреблять

плавает. Вот несколько способов обойти это:

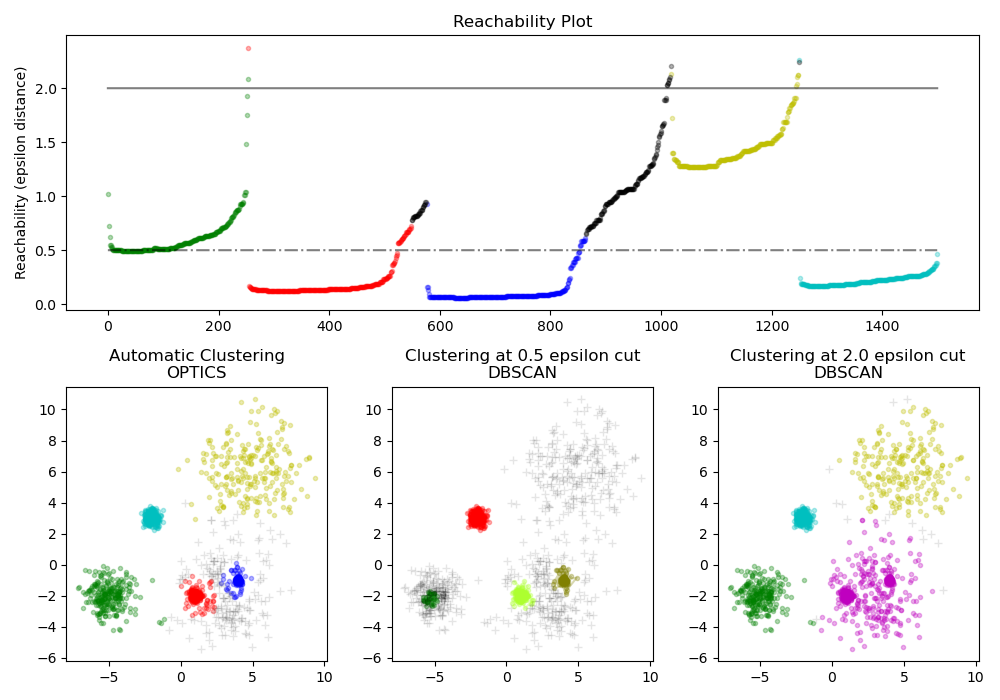
* Используйте кластеризацию [OPTICS](https://scikit-learn.org/stable/modules/clustering.html#optics) вместе с параметром **cluster\_method = ‘dbscan’**. Кластеризация OPTICS также вычисляет полную попарную матрицу, но сохраняет в памяти только одну строку (сложность памяти n).
* График окрестностей с разреженным радиусом (где предполагается, что отсутствующие записи находятся вне eps) можно предварительно вычислить эффективным с точки зрения памяти способом, и dbscan можно обработать с помощью metric='precomputed'. Смотрите [sklearn.neighbors.NearestNeighbors.radius\_neighbors\_graph](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.NearestNeighbors.html#sklearn.neighbors.NearestNeighbors.radius_neighbors_graph).
* Набор данных можно сжать, удалив точные дубликаты, если они встречаются в ваших данных, или используя BIRCH. Тогда у вас будет относительно небольшое количество представителей для большого количества точек. Затем вы можете sample\_weight указать при установке DBSCAN.

Рекомендации:

* «Алгоритм на основе плотности для обнаружения кластеров в больших пространственных базах данных с шумом» Эстер, М., Х. П. Кригель, Дж. Сандер и X. Сюй, в материалах 2-й Международной конференции по открытию знаний и интеллектуальному анализу данных, Портленд, штат Орегон , AAAI Press, стр. 226–231. 1996 г.
* «Новый взгляд на DBSCAN: почему и как вы должны (по-прежнему) использовать DBSCAN. Шуберт, Э., Сандер, Дж., Эстер, М., Кригель, HP, и Сюй, X. (2017). В транзакциях ACM в системах баз данных (TODS), 42 (3), 19.

## 2.3.8. ОПТИКА

В [OPTICS](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html#sklearn.cluster.OPTICS) акции алгоритма много общих с [DBSCAN](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#sklearn.cluster.DBSCAN) алгоритмом, и можно рассматривать как обобщение DBSCAN, что расслабляет epsтребование от одного значения до диапазона значений. Ключевое различие между DBSCAN и OPTICS заключается в том, что алгоритм OPTICS строит граф достижимости , который назначает каждой выборке reachability\_ расстояние и точку в ordering\_ атрибуте кластера ; эти два атрибута назначаются при подборе модели и используются для определения принадлежности к кластеру. Если OPTICS запускается со значением по умолчанию inf, установленным для max\_eps, то извлечение кластера в стиле DBSCAN может выполняться повторно за линейное время для любого заданного eps значения с использованием этого cluster\_optics\_dbscan метода. Параметр max\_eps более низкое значение приведет к сокращению времени выполнения, и его можно рассматривать как максимальный радиус окрестности от каждой точки для поиска других потенциальных достижимых точек.



Расстояния достижимости, генерируемые OPTICS, позволяют извлекать кластеры с переменной плотностью в пределах одного набора данных. Как показано на приведенном выше графике, объединение расстояний достижимости и набора данных ordering\_ дает график достижимости , где плотность точек представлена ​​на оси Y, а точки упорядочены таким образом, что соседние точки являются смежными. «Вырезание» графика достижимости по одному значению дает результаты, подобные DBSCAN; все точки над «вырезом» классифицируются как шум, и каждый раз, когда есть перерыв при чтении слева направо, означает новый кластер. При извлечении кластеров по умолчанию с помощью OPTICS анализируются крутые уклоны на графике, чтобы найти кластеры, и пользователь может определить, что считается крутым уклоном, используя параметр xi. Существуют также другие возможности для анализа на самом графике, такие как создание иерархических представлений данных с помощью дендрограмм графика достижимости, а иерархия кластеров, обнаруженных алгоритмом, может быть доступна через cluster\_hierarchy\_ параметр. Приведенный выше график имеет цветовую кодировку, поэтому цвета кластеров в плоском пространстве соответствуют кластерам линейных сегментов на графике достижимости. Обратите внимание, что синий и красный кластеры находятся рядом на графике достижимости и могут быть иерархически представлены как дочерние элементы более крупного родительского кластера.

Примеры:

* [Демонстрация алгоритма кластеризации OPTICS](https://scikit-learn.ru/example/demo-of-optics-clustering-algorithm/)

Сравнение с DBSCAN

Результаты cluster\_optics\_dbscan метода OPTICS и DBSCAN очень похожи, но не всегда идентичны; в частности, маркировка периферийных и шумовых точек. Отчасти это связано с тем, что первые образцы каждой плотной области, обработанной OPTICS, имеют большое значение достижимости, будучи близкими к другим точкам в своей области, и поэтому иногда будут помечены как шум, а не периферия. Это влияет на соседние точки, когда они рассматриваются как кандидаты на то, чтобы их пометить как периферию или как шум.

Обратите внимание, что для любого отдельного значения eps DBSCAN будет иметь более короткое время выполнения, чем OPTICS; однако для повторных запусков с разными eps значениями один запуск OPTICS может потребовать меньше совокупного времени выполнения, чем DBSCAN. Также важно отметить , что выход ОПТИКА» близок к DBSCAN, только если epsи max\_eps близки.

Вычислительная сложность

Деревья пространственной индексации используются, чтобы избежать вычисления полной матрицы расстояний и обеспечить эффективное использование памяти для больших наборов выборок. С помощью metric ключевого слова можно указать различные метрики расстояния .

Для больших наборов данных аналогичные (но не идентичные) результаты можно получить с помощью [HDBSCAN](https://hdbscan.readthedocs.io/). Реализация HDBSCAN является многопоточной и имеет лучшую алгоритмическую сложность во время выполнения, чем OPTICS, за счет худшего масштабирования памяти. Для очень больших наборов данных, которые исчерпывают системную память с помощью HDBSCAN, OPTICS будет поддерживатьn (в отличие от

) масштабирование памяти; тем не менее, max\_eps для получения решения в разумные сроки, вероятно, потребуется настройка параметра.

Рекомендации:

* «ОПТИКА: точки упорядочивания для определения структуры кластеризации». Анкерст, Михаэль, Маркус М. Бройниг, Ханс-Петер Кригель и Йорг Зандер. В ACM Sigmod Record, vol. 28, вып. 2. С. 49-60. ACM, 1999.

## 2.3.9. ****BIRCH**** (сбалансированное итеративное сокращение и кластеризация с использованием иерархий — balanced iterative reducing and clustering using hierarchies)

Для [Birch](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.Birch.html#sklearn.cluster.Birch) заданных данных он строит дерево, называемое деревом функций кластеризации (CFT). Данные по существу сжимаются с потерями до набора узлов Clustering Feature (CF Nodes). Узлы CF имеют ряд подкластеров, называемых подкластерами функций кластеризации (подкластеры CF), и эти подкластеры CF, расположенные в нетерминальных узлах CF, могут иметь узлы CF в качестве дочерних.

Подкластеры CF содержат необходимую информацию для кластеризации, что избавляет от необходимости хранить все входные данные в памяти. Эта информация включает:

* Количество образцов в подкластере.
* Линейная сумма — n-мерный вектор, содержащий сумму всех выборок.
* Squared Sum — сумма квадратов нормы L2 всех выборок.
* Центроиды — чтобы избежать пересчета линейной суммы / n\_samples.
* Квадратная норма центроидов.

Алгоритм BIRCH имеет два параметра: порог и коэффициент ветвления. Фактор ветвления ограничивает количество подкластеров в узле, а порог ограничивает расстояние между входящей выборкой и существующими подкластерами.

Этот алгоритм можно рассматривать как экземпляр или метод сокращения данных, поскольку он сокращает входные данные до набора подкластеров, которые получаются непосредственно из конечных точек CFT. Эти сокращенные данные могут быть дополнительно обработаны путем подачи их в глобальный кластеризатор. Этот глобальный кластеризатор может быть установлен с помощью n\_clusters. Если n\_clusters установлено значение None, подкластеры из листьев считываются напрямую, в противном случае шаг глобальной кластеризации помечает эти подкластеры в глобальные кластеры (метки), а выборки сопоставляются с глобальной меткой ближайшего подкластера.

**Описание алгоритма:**

* Новый образец вставляется в корень дерева CF, который является узлом CF. Затем он объединяется с подкластером корня, который имеет наименьший радиус после объединения, ограниченный пороговыми условиями и условиями фактора ветвления. Если у подкластера есть дочерний узел, то это повторяется до тех пор, пока он не достигнет листа. После нахождения ближайшего подкластера в листе свойства этого подкластера и родительских подкластеров рекурсивно обновляются.
* Если радиус подкластера, полученного путем слияния новой выборки и ближайшего подкластера, больше квадрата порога, и если количество подкластеров больше, чем коэффициент ветвления, то для этой новой выборки временно выделяется пространство. Берутся два самых дальних подкластера, и подкластеры делятся на две группы на основе расстояния между этими подкластерами.
* Если у этого разбитого узла есть родительский подкластер и есть место для нового подкластера, то родительский подкластер разбивается на два. Если места нет, то этот узел снова делится на две части, и процесс продолжается рекурсивно, пока не достигнет корня.

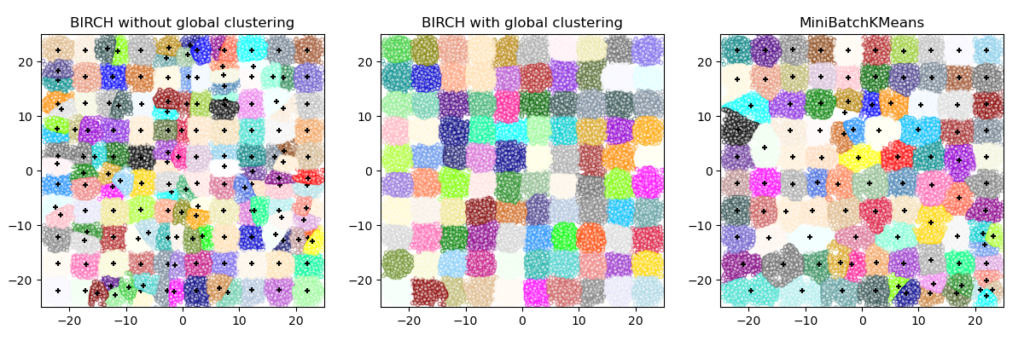
**BIRCH или MiniBatchKMeans?**

* BIRCH не очень хорошо масштабируется для данных большого размера. Как правило, если n\_features оно больше двадцати, лучше использовать MiniBatchKMeans.
* Если количество экземпляров данных необходимо уменьшить или если требуется большое количество подкластеров в качестве этапа предварительной обработки или иным образом, BIRCH более полезен, чем MiniBatchKMeans.

**Как использовать partial\_fit?**

Чтобы избежать вычисления глобальной кластеризации, при каждом обращении partial\_fit пользователя рекомендуется

1. Установить n\_clusters=None изначально
2. Обучите все данные несколькими вызовами partial\_fit.
3. Установите n\_clusters необходимое значение с помощью brc.set\_params(n\_clusters=n\_clusters.
4. Вызов, partial\_fit наконец, без аргументов, т.е. brc.partial\_fit() который выполняет глобальную кластеризацию.



Рекомендации:

* Тиан Чжан, Рагху Рамакришнан, Марон Ливни БЕРЕЗА: эффективный метод кластеризации данных для больших баз данных. <https://www.cs.sfu.ca/CourseCentral/459/han/papers/zhang96.pdf>
* Роберто Пердиши JBirch — Java-реализация алгоритма кластеризации BIRCH <https://code.google.com/archive/p/jbirch>

## 2.3.10. Оценка эффективности кластеризации

Оценка производительности алгоритма кластеризации не так тривиальна, как подсчет количества ошибок или точности и отзыва контролируемого алгоритма классификации. В частности, любая метрика оценки должна учитывать не абсолютные значения меток кластера, а, скорее, если эта кластеризация определяет разделения данных, аналогичные некоторому основному набору истинных классов или удовлетворяющему некоторому предположению, так что члены, принадлежащие к одному классу, более похожи чем члены разных классов в соответствии с некоторой метрикой сходства.

### 2.3.10.1. Индекс Rand

Учитывая знания о назначениях базовых классов истинности labels\_true и назначениях нашим алгоритмом кластеризации одних и тех же выборок labels\_pred, **(скорректированный или нескорректированный) индекс Рэнда** — это функция, которая измеряет **сходство** двух назначений, игнорируя перестановки:



1

>>> from sklearn import metrics

2

>>> labels\_true = [0, 0, 0, 1, 1, 1]

3

>>> labels\_pred = [0, 0, 1, 1, 2, 2]

4

>>> metrics.rand\_score(labels\_true, labels\_pred)

5

0.66...

Индекс Rand не гарантирует получение значения, близкого к 0,0 при случайной маркировке. Скорректированный индекс Rand **вносит поправку на случайность** и дает такую ​​основу.



1

>>> metrics.adjusted\_rand\_score(labels\_true, labels\_pred)

2

0.24...

Как и в случае со всеми метриками кластеризации, можно переставить 0 и 1 в предсказанных метках, переименовать 2 в 3 и получить ту же оценку:



1

>>> labels\_pred = [1, 1, 0, 0, 3, 3]

2

>>> metrics.rand\_score(labels\_true, labels\_pred)

3

0.66...

4

>>> metrics.adjusted\_rand\_score(labels\_true, labels\_pred)

5

0.24...

Более того, оба [rand\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.rand_score.html#sklearn.metrics.rand_score) [adjusted\_rand\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html#sklearn.metrics.adjusted_rand_score) они **симметричны** : замена аргумента не меняет оценок. Таким образом, их можно использовать в качестве **критериев консенсуса**:



1

>>> metrics.rand\_score(labels\_pred, labels\_true)

2

0.66...

3

>>> metrics.adjusted\_rand\_score(labels\_pred, labels\_true)

4

0.24...

Идеальная маркировка оценивается в 1,0:



1

>>> labels\_pred = labels\_true[:]

2

>>> metrics.rand\_score(labels\_true, labels\_pred)

3

1.0

4

>>> metrics.adjusted\_rand\_score(labels\_true, labels\_pred)

5

1.0

Плохо согласованные ярлыки (например, независимые ярлыки) имеют более низкие оценки, а для скорректированного индекса Rand оценка будет отрицательной или близкой к нулю. Однако для нескорректированного индекса Rand оценка, хотя и ниже, не обязательно будет близка к нулю:



1

>>> labels\_true = [0, 0, 0, 0, 0, 0, 1, 1]

2

>>> labels\_pred = [0, 1, 2, 3, 4, 5, 5, 6]

3

>>> metrics.rand\_score(labels\_true, labels\_pred)

4

0.39...

5

>>> metrics.adjusted\_rand\_score(labels\_true, labels\_pred)

6

-0.07...

### 2.3.10.1.1. Преимущества

* **Интерпретируемость** : нескорректированный индекс Рэнда пропорционален количеству пар выборок, метки которых одинаковы в обоих labels\_predи / labels\_true или различаются в обоих.
* **Случайные (единообразные) присвоения меток имеют скорректированный балл индекса Rand, близкий к 0,0** для любого значения n\_clustersи n\_samples (что не относится, например, к нескорректированному индексу Rand или V-мере).
* **Ограниченный диапазон** : более низкие значения указывают на разные маркировки, аналогичные кластеры имеют высокий (скорректированный или нескорректированный) индекс Rand, 1,0 — это оценка идеального соответствия. Диапазон оценок составляет [0, 1] для нескорректированного индекса Rand и [-1, 1] для скорректированного индекса Rand.
* **Не делается никаких предположений о структуре кластера**: (скорректированный или нескорректированный) индекс Rand может использоваться для сравнения всех видов алгоритмов кластеризации и может использоваться для сравнения алгоритмов кластеризации, таких как k-средних, который предполагает изотропные формы капли с результатами спектрального анализа. алгоритмы кластеризации, которые могут найти кластер со «сложенными» формами.

### 2.3.10.1.2. Недостатки

* Вопреки инерции, **(скорректированный или нескорректированный) индекс Рэнда требует знания основных классов истинности,** что почти никогда не доступно на практике или требует ручного назначения аннотаторами-людьми (как в условиях контролируемого обучения). Однако (скорректированный или нескорректированный) индекс Rand также может быть полезен в чисто неконтролируемой настройке в качестве строительного блока для консенсусного индекса, который может использоваться для выбора модели кластеризации (TODO).
* **Нескорректированный индекс Rand часто близко к 1,0** , даже если самим кластеризациям существенно различаются. Это можно понять, интерпретируя индекс Рэнда как точность маркировки пар элементов, полученную в результате кластеризации: на практике часто существует большинство пар элементов, которым присваивается different метка пары как при прогнозируемой, так и при базовой кластеризации истинности, что приводит к высокой доля парных меток, которые согласны, что впоследствии приводит к высокой оценке.

Примеры:

* [Поправка на случайность при оценке производительности кластеризации](https://scikit-learn.ru/example/adjustment-for-chance-in-clustering-performance-evaluation/) : анализ влияния размера набора данных на значение показателей кластеризации для случайных назначений.

### 2.3.10.1.3. Математическая постановка

Если C — это наземное присвоение класса истинности, а K — кластеризация, давайте определим

 и

 в виде:

 , количество пар элементов, которые находятся в одном наборе в C и в одном наборе в K

*  , количество пар элементов, которые находятся в разных наборах в C и в разных наборах в K

Затем нескорректированный индекс Рэнда рассчитывается следующим образом:

где

— общее количество возможных пар в наборе данных. Не имеет значения, выполняется ли расчет для упорядоченных пар или неупорядоченных пар, если расчет выполняется последовательно.

Однако индекс Rand не гарантирует, что случайные присвоения меток получат значение, близкое к нулю (особенно, если количество кластеров имеет тот же порядок величины, что и количество выборок).

Чтобы противостоять этому эффекту, мы можем дисконтировать ожидаемый RI.

 случайных маркировок путем определения скорректированного индекса Rand следующим образом:

Рекомендации

* [Сравнение разделов](https://link.springer.com/article/10.1007%2FBF01908075) Л. Хуберт и П. Араби, Журнал классификации 1985 г.
* [Свойства скорректированного индекса Рэнда Хьюберта-](https://psycnet.apa.org/record/2004-17801-007) Араби Д. Стейнли, Психологические методы, 2004 г.
* [Запись в Википедии для индекса Rand](https://en.wikipedia.org/wiki/Rand_index)
* [Запись в Википедии о скорректированном индексе Рэнда](https://en.wikipedia.org/wiki/Rand_index#Adjusted_Rand_index)

### 2.3.10.2. Оценки на основе взаимной информации

Учитывая знания о назначениях базовых классов истинности labels\_true и назначениях нашим алгоритмом кластеризации одних и тех же выборок labels\_pred, **взаимная информация** — это функция, которая измеряет **согласованность** двух назначений, игнорируя перестановки. Доступны две различные нормализованные версии этой меры: **нормализованная взаимная информация (NMI)** и **скорректированная взаимная информация (AMI)** . NMI часто используется в литературе, в то время как AMI был предложен совсем недавно и **нормируется на случайность** :



1

>>> from sklearn import metrics

2

>>> labels\_true = [0, 0, 0, 1, 1, 1]

3

>>> labels\_pred = [0, 0, 1, 1, 2, 2]

4

​

5

>>> metrics.adjusted\_mutual\_info\_score(labels\_true, labels\_pred)

6

0.22504...

Можно переставить 0 и 1 в предсказанных метках, переименовать 2 в 3 и получить тот же результат:



1

>>> labels\_pred = [1, 1, 0, 0, 3, 3]

2

>>> metrics.adjusted\_mutual\_info\_score(labels\_true, labels\_pred)

3

0.22504...

Все, [mutual\_info\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mutual_info_score.html#sklearn.metrics.mutual_info_score), [adjusted\_mutual\_info\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html#sklearn.metrics.adjusted_mutual_info_score)и [normalized\_mutual\_info\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.normalized_mutual_info_score.html#sklearn.metrics.normalized_mutual_info_score)является симметричным: замена аргумента не меняет рейтинг. Таким образом, их можно использовать в качестве **меры консенсуса** :



1

>>> metrics.adjusted\_mutual\_info\_score(labels\_pred, labels\_true)

2

0.22504...

Идеальная маркировка оценивается в 1,0:



1

>>> labels\_pred = labels\_true[:]

2

>>> metrics.adjusted\_mutual\_info\_score(labels\_true, labels\_pred)

3

1.0

4

​

5

>>> metrics.normalized\_mutual\_info\_score(labels\_true, labels\_pred)

6

1.0

Это неверно mutual\_info\_score, поэтому судить о них труднее:



1

>>> metrics.mutual\_info\_score(labels\_true, labels\_pred)

2

0.69...

Плохие (например, независимые маркировки) имеют отрицательные оценки:



1

>>> labels\_true = [0, 1, 2, 0, 3, 4, 5, 1]

2

>>> labels\_pred = [1, 1, 0, 0, 2, 2, 2, 2]

3

>>> metrics.adjusted\_mutual\_info\_score(labels\_true, labels\_pred)

4

-0.10526...

### 2.3.10.2.1. Преимущества

* **Случайные (унифицированные) присвоения меток имеют оценку AMI, близкую к 0,0** для любого значения n\_clusters и n\_samples (что не относится, например, к необработанной взаимной информации или V-мере).
* **Верхняя граница 1** : значения, близкие к нулю, указывают на два назначения меток, которые в значительной степени независимы, а значения, близкие к единице, указывают на значительное согласие. Кроме того, AMI, равный ровно 1, указывает, что два присвоения меток равны (с перестановкой или без нее).

### 2.3.10.2.2. Недостатки

* Вопреки инерции, **меры на основе MI требуют знания основных классов истинности,** хотя на практике они почти никогда не доступны или требуют ручного назначения аннотаторами-людьми (как в условиях контролируемого обучения).

Однако показатели на основе MI также могут быть полезны в чисто неконтролируемой настройке в качестве строительного блока для согласованного индекса, который можно использовать для выбора модели кластеризации

* NMI и MI не подвергаются случайной корректировке.

Примеры:

* [Поправка на случайность при оценке производительности кластеризации](https://scikit-learn.ru/example/adjustment-for-chance-in-clustering-performance-evaluation/): анализ влияния размера набора данных на значение показателей кластеризации для случайных назначений. Этот пример также включает скорректированный индекс ранда.

### 2.3.10.2.3. Математическая постановка

Предположим, что два присвоения метки (одних и тех же N объектов),

 а также

. Их энтропия — это величина неопределенности для набора разбиений, определяемая следующим образом:

где  вероятность того, что объект выбран наугад из  попадает в класс

. Аналогично дляV:

С участием . Взаимная информация (MI) между  и

 рассчитывается по:

где  вероятность того, что случайно выбранный объект попадает в оба класса  а также

.

Это также может быть выражено в формулировке множества элементов:

Нормализованная взаимная информация определяется как

Это значение взаимной информации, а также нормализованного варианта не скорректировано на случайность и будет иметь тенденцию к увеличению по мере увеличения количества различных меток (кластеров), независимо от фактического количества «взаимной информации» между назначениями меток.

Ожидаемое значение для взаимной информации можно рассчитать с помощью следующего уравнения [VEB2009] . В этом уравнении

 (количество элементов в ) а также  (количество элементов в

).

Используя ожидаемое значение, скорректированная взаимная информация может быть затем рассчитана с использованием формы, аналогичной форме скорректированного индекса Rand:

Для нормализованной взаимной информации и скорректированной взаимной информации нормализующее значение обычно представляет собой некоторое обобщенное среднее энтропий каждой кластеризации. Существуют различные обобщенные средства, и не существует твердых правил предпочтения одного по сравнению с другим. Решение в основном принимается отдельно для каждого поля; например, при обнаружении сообществ чаще всего используется среднее арифметическое. Каждый метод нормализации обеспечивает «качественно похожее поведение» [YAT2016] . В нашей реализации это контролируется average\_methodпараметром.

Vinh et al. (2010) назвали варианты NMI и AMI методом их усреднения [VEB2010] . Их средние «sqrt» и «sum» являются средними геометрическими и арифметическими; мы используем эти более общие имена.

Рекомендации

* Штрел, Александр и Джойдип Гош (2002). «Кластерные ансамбли — структура повторного использования знаний для объединения нескольких разделов». Журнал исследований машинного обучения 3: 583–617. [DOI: 10,1162 / 153244303321897735](http://strehl.com/download/strehl-jmlr02.pdf) .
* [Запись в Википедии для (нормализованной) взаимной информации](https://en.wikipedia.org/wiki/Mutual_Information)
* [Запись в Википедии для Скорректированной взаимной информации](https://en.wikipedia.org/wiki/Adjusted_Mutual_Information)

VEB2009 Винь, Эппс и Бейли (2009). «Теоретико-информационные меры для сравнения кластеризации». Материалы 26-й ежегодной международной конференции по машинному обучению — ICML ’09. [DOI: 10.1145 / 1553374.1553511](https://dl.acm.org/citation.cfm?doid=1553374.1553511) . ISBN 9781605585161.

ВЭБ2010 Винь, Эппс и Бейли (2010). «Теоретико-информационные меры для сравнения кластеризации: варианты, свойства, нормализация и поправка на случайность». JMLR <<http://jmlr.csail.mit.edu/papers/volume11/vinh10a/vinh10a.pdf>>

YAT2016 Ян, Альгешаймер и Тессон (2016). «Сравнительный анализ алгоритмов обнаружения сообществ в искусственных сетях». Научные отчеты 6: 30750. DOI [: 10.1038 / srep30750](https://www.nature.com/articles/srep30750) .

### 2.3.10.3. Однородность, полнота и V-мера

Зная о назначении основных классов истинности выборкам, можно определить некоторую интуитивно понятную метрику, используя условный энтропийный анализ.

В частности, Розенберг и Хиршберг (2007) определяют следующие две желательные цели для любого кластерного задания:

* **однородность** : каждый кластер содержит только членов одного класса.
* **полнота** : все члены данного класса относятся к одному кластеру.

Мы можем превратить эти концепции в партитуры [homogeneity\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html#sklearn.metrics.homogeneity_score) и [completeness\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html#sklearn.metrics.completeness_score). Оба ограничены снизу 0,0 и выше 1,0 (чем выше, тем лучше):



1

>>> from sklearn import metrics

2

>>> labels\_true = [0, 0, 0, 1, 1, 1]

3

>>> labels\_pred = [0, 0, 1, 1, 2, 2]

4

​

5

>>> metrics.homogeneity\_score(labels\_true, labels\_pred)

6

0.66...

7

​

8

>>> metrics.completeness\_score(labels\_true, labels\_pred)

9

0.42...

Их гармоническое среднее значение, называемое **V-мерой** , вычисляется по формуле [v\_measure\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html#sklearn.metrics.v_measure_score):



1

>>> metrics.v\_measure\_score(labels\_true, labels\_pred)

2

0.51...

Формула этой функции выглядит следующим образом:

beta по умолчанию используется значение 1.0, но для использования значения менее 1 для бета-версии:



1

>>> metrics.v\_measure\_score(labels\_true, labels\_pred, beta=0.6)

2

0.54...

больший вес будет приписан однородности, а использование значения больше 1:



1

>>> metrics.v\_measure\_score(labels\_true, labels\_pred, beta=1.8)

2

0.48...

больший вес будет приписан полноте.

V-мера фактически эквивалентна описанной выше взаимной информации (NMI), при этом функция агрегирования является средним арифметическим [B2011] .

Однородность, полнота и V-мера могут быть вычислены сразу, используя [homogeneity\_completeness\_v\_measure](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_completeness_v_measure.html#sklearn.metrics.homogeneity_completeness_v_measure) следующее:



1

>>> metrics.homogeneity\_completeness\_v\_measure(labels\_true, labels\_pred)

2

(0.66..., 0.42..., 0.51...)

Следующее назначение кластеризации немного лучше, поскольку оно однородное, но не полное:



1

>>> labels\_pred = [0, 0, 0, 1, 2, 2]

2

>>> metrics.homogeneity\_completeness\_v\_measure(labels\_true, labels\_pred)

3

(1.0, 0.68..., 0.81...)

Примечание:

[v\_measure\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.v_measure_score.html#sklearn.metrics.v_measure_score) является **симметричным** : его можно использовать для оценки **соответствия** двух независимых присвоений одному и тому же набору данных.

Это не относится к [completeness\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.completeness_score.html#sklearn.metrics.completeness_score) и [homogeneity\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_score.html#sklearn.metrics.homogeneity_score): оба связаны отношениями:

homogeneity\_score(a, b) == completeness\_score(b, a)

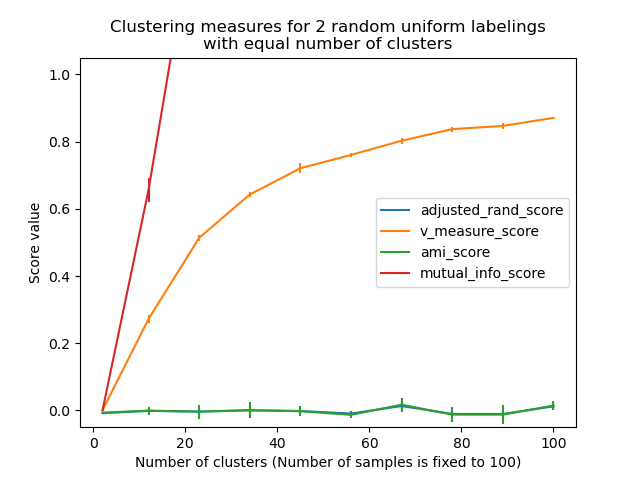
### 2.3.10.3.1. Преимущества

* **Ограниченные баллы** : 0,0 — это настолько плохо, насколько это возможно, 1,0 — идеальный балл.
* Интуитивная интерпретация: кластеризацию с плохой V-мерой можно **качественно проанализировать с точки зрения однородности и полноты,** чтобы лучше понять, какие «ошибки» допускаются при задании.
* **Не делается никаких предположений о структуре кластера** : может использоваться для сравнения алгоритмов кластеризации, таких как k-среднее, которое предполагает изотропные формы капли, с результатами алгоритмов спектральной кластеризации, которые могут находить кластер со «сложенными» формами.

### 2.3.10.3.2. Недостатки

* Ранее введенные метрики **не нормализованы в отношении случайной маркировки** : это означает, что в зависимости от количества выборок, кластеров и основных классов истинности полностью случайная маркировка не всегда будет давать одинаковые значения для однородности, полноты и, следовательно, v-меры.

В частности, **случайная маркировка не даст нулевых оценок, особенно когда количество кластеров велико**. Эту проблему можно спокойно игнорировать, если количество выборок больше тысячи, а количество кластеров меньше 10. **Для меньших размеров выборки или большего количества кластеров безопаснее использовать скорректированный индекс, такой как Скорректированный индекс ранда (ARI)** .



* Эти метрики **требуют знания основных классов истинности,** хотя почти никогда не доступны на практике или требуют ручного назначения аннотаторами-людьми (как в условиях контролируемого обучения).

Примеры:

* [Поправка на случайность при оценке производительности кластеризации](https://scikit-learn.ru/example/adjustment-for-chance-in-clustering-performance-evaluation/) : анализ влияния размера набора данных на значение показателей кластеризации для случайных назначений.

### 2.3.10.3.3. Математическая постановка

Оценки однородности и полноты формально даются по формуле:

где  является **условной энтропией классов с учетом кластерных назначений** и определяется выражением:   
а также  — **энтропия классов** и определяется выражением:  
с участием n общее количество образцов,  а также  количество образцов, соответственно принадлежащих к классу c и кластер , и наконец

 количество образцов из класса c назначен кластеру k.

**Условная энтропия кластеров данного класса**

и **энтропия кластеров**

 определены симметричным образом.

Розенберг и Хиршберг далее определяют **V-меру** как **гармоническое среднее однородности и полноты**:

Рекомендации

* [V-Measure: мера оценки внешнего кластера на основе условной энтропии](https://aclweb.org/anthology/D/D07/D07-1043.pdf) Эндрю Розенберг и Джулия Хиршберг, 2007 г.

B2011 [Идентификация и характеристика событий в социальных сетях](http://www.cs.columbia.edu/~hila/hila-thesis-distributed.pdf) , докторская диссертация Хила Беккер.

### 2.3.10.4. Результаты Фаулкса-Мэллоуса

Индекс Фаулкса-Мэллоуса ( [sklearn.metrics.fowlkes\_mallows\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fowlkes_mallows_score.html#sklearn.metrics.fowlkes_mallows_score)) может использоваться, когда известны наземные присвоения классов истинности выборкам. FMI по шкале Fowlkes-Mallows определяется как среднее геометрическое для попарной точности и отзыва:

Где TP— количество **истинных положительных результатов** (**True Positive**) (т.е. количество пар точек, принадлежащих одним и тем же кластерам как в истинных, так и в предсказанных метках), FP— это количество **ложных положительных результатов (False Positive)** (то есть количество пар точек, принадлежащих к одинаковые кластеры в истинных метках, а не в предсказанных метках) и FN является количеством **ложно-отрицательных** (**False Negative**) (т. е. количеством пар точек, принадлежащих одним и тем же кластерам в предсказанных метках, а не в истинных метках).

Оценка варьируется от 0 до 1. Высокое значение указывает на хорошее сходство между двумя кластерами.



1

>>> from sklearn import metrics

2

>>> labels\_true = [0, 0, 0, 1, 1, 1]

3

>>> labels\_pred = [0, 0, 1, 1, 2, 2]



1

>>> metrics.fowlkes\_mallows\_score(labels\_true, labels\_pred)

2

0.47140...

Можно переставить 0 и 1 в предсказанных метках, переименовать 2 в 3 и получить тот же результат:



1

>>> labels\_pred = [1, 1, 0, 0, 3, 3]

2

​

3

>>> metrics.fowlkes\_mallows\_score(labels\_true, labels\_pred)

4

0.47140...

Идеальная маркировка оценивается в 1,0:



1

>>> labels\_pred = labels\_true[:]

2

>>> metrics.fowlkes\_mallows\_score(labels\_true, labels\_pred)

3

1.0

Плохие (например, независимые маркировки) имеют нулевые баллы:



1

>>> labels\_true = [0, 1, 2, 0, 3, 4, 5, 1]

2

>>> labels\_pred = [1, 1, 0, 0, 2, 2, 2, 2]

3

>>> metrics.fowlkes\_mallows\_score(labels\_true, labels\_pred)

4

0.0

### 2.3.10.4.1. Преимущества

* **Случайное (единообразное) присвоение меток имеет оценку FMI, близкую к 0,0** для любого значения n\_clusters и n\_samples (что не относится, например, к необработанной взаимной информации или V-мере).
* **Верхний предел — 1** : значения, близкие к нулю, указывают на два назначения меток, которые в значительной степени независимы, а значения, близкие к единице, указывают на значительное согласие. Кроме того, значения ровно 0 указывают на **чисто** независимые присвоения меток, а FMI ровно 1 указывает, что два назначения меток равны (с перестановкой или без нее).
* **Не делается никаких предположений о структуре кластера** : может использоваться для сравнения алгоритмов кластеризации, таких как k-среднее, которое предполагает изотропные формы капли, с результатами алгоритмов спектральной кластеризации, которые могут находить кластер со «сложенными» формами.

### 2.3.10.4.2. Недостатки

* Вопреки инерции, **меры, основанные на FMI, требуют знания основных классов истинности,** хотя почти никогда не доступны на практике или требуют ручного назначения аннотаторами-людьми (как в условиях контролируемого обучения).

Рекомендации

* EB Fowkles и CL Mallows, 1983. «Метод сравнения двух иерархических кластеров». Журнал Американской статистической ассоциации. <http://wildfire.stat.ucla.edu/pdflibrary/fowlkes.pdf>
* [Запись в Википедии об Индексе Фаулкса-Маллоуза](https://en.wikipedia.org/wiki/Fowlkes-Mallows_index)

### 2.3.10.5. Коэффициент силуэта

Если наземные метки достоверности неизвестны, оценка должна выполняться с использованием самой модели. Коэффициент силуэта ( [sklearn.metrics.silhouette\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html#sklearn.metrics.silhouette_score)) является примером такой оценки, где более высокий показатель коэффициента силуэта относится к модели с лучше определенными кластерами. Коэффициент силуэта определяется для каждого образца и состоит из двух баллов:

* **a** : Среднее расстояние между образцом и всеми другими точками того же класса.
* **b** : Среднее расстояние между образцом и всеми другими точками в следующем ближайшем кластере .

Тогда коэффициент силуэта s для одного образца определяется как:

Коэффициент силуэта для набора образцов дается как среднее значение коэффициента силуэта для каждого образца.



1

>>> from sklearn import metrics

2

>>> from sklearn.metrics import pairwise\_distances

3

>>> from sklearn import datasets

4

>>> X, y = datasets.load\_iris(return\_X\_y=True)

При обычном использовании коэффициент силуэта применяется к результатам кластерного анализа.



1

>>> import numpy as np

2

>>> from sklearn.cluster import KMeans

3

>>> kmeans\_model = KMeans(n\_clusters=3, random\_state=1).fit(X)

4

>>> labels = kmeans\_model.labels\_

5

>>> metrics.silhouette\_score(X, labels, metric='euclidean')

6

0.55...

Рекомендации

* Питер Дж. Руссеу (1987). «Силуэты: графическое средство для интерпретации и проверки кластерного анализа». Вычислительная и прикладная математика 20: 53–65. [DOI: 10.1016 / 0377-0427 (87) 90125-7](https://doi.org/10.1016/0377-0427(87)90125-7) .

### 2.3.10.5.1. Преимущества

* Оценка ограничена от -1 за неправильную кластеризацию до +1 за высокоплотную кластеризацию. Баллы около нуля указывают на перекрывающиеся кластеры.
* Оценка выше, когда кластеры плотные и хорошо разделенные, что относится к стандартной концепции кластера.

### 2.3.10.5.2. Недостатки

* Коэффициент силуэта обычно выше для выпуклых кластеров, чем для других концепций кластеров, таких как кластеры на основе плотности, подобные тем, которые получены с помощью DBSCAN.

Примеры:

* [Выбор количества кластеров с анализом силуэта в кластеризации KMeans](https://scikit-learn.ru/example/selecting-the-number-of-clusters-with-silhouette-analysis-on-kmeans-clustering/) : в этом примере анализ силуэта используется для выбора оптимального значения для n\_clusters.

### 2.3.10.6. Индекс Калински-Харабаса

Если наземные метки достоверности неизвестны, индекс Калински-Харабаса ( [sklearn.metrics.calinski\_harabasz\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski_harabasz_score.html#sklearn.metrics.calinski_harabasz_score)) — также известный как критерий отношения дисперсии — можно использовать для оценки модели, где более высокий балл Калински-Харабаса относится к модели с более определенными кластерами.

Индекс представляет собой отношение суммы дисперсии между кластерами и дисперсии внутри кластера для всех кластеров (где дисперсия определяется как сумма квадратов расстояний):



1

>>> from sklearn import metrics

2

>>> from sklearn.metrics import pairwise\_distances

3

>>> from sklearn import datasets

4

>>> X, y = datasets.load\_iris(return\_X\_y=True)

При обычном использовании индекс Калински-Харабаса применяется к результатам кластерного анализа:



1

>>> import numpy as np

2

>>> from sklearn.cluster import KMeans

3

>>> kmeans\_model = KMeans(n\_clusters=3, random\_state=1).fit(X)

4

>>> labels = kmeans\_model.labels\_

5

>>> metrics.calinski\_harabasz\_score(X, labels)

6

561.62...

### 2.3.10.6.1. Преимущества

* Оценка выше, когда кластеры плотные и хорошо разделенные, что относится к стандартной концепции кластера.
* Счет быстро подсчитывается.

### 2.3.10.6.2. Недостатки

* Индекс Калински-Харабаса обычно выше для выпуклых кластеров, чем для других концепций кластеров, таких как кластеры на основе плотности, подобные тем, которые получены с помощью DBSCAN.

### 2.3.10.6.3. Математическая постановка

Для набора данных

 размера  который был сгруппирован в  кластеры, оценка Калински-Харабаса  определяется как отношение средней дисперсии между кластерами и дисперсии внутри кластера:  
где  является следом межгрупповой дисперсионной матрицы и  является следом матрицы дисперсии внутри кластера, определяемой следующим образом:  
  
с участием  набор точек в кластере ,  центр кластера ,  центр , а также  количество точек в кластере

.

Рекомендации

* Калински Т. и Харабаш Дж. (1974). [«Дендритный метод кластерного анализа»](https://www.researchgate.net/publication/233096619_A_Dendrite_Method_for_Cluster_Analysis) . Коммуникации в теории статистики и методах 3: 1-27. [DOI: 10.1080 / 03610927408827101](https://doi.org/10.1080/03610927408827101) .

### 2.3.10.7. Индекс Дэвиса-Болдина

Если наземные метки истинности неизвестны, для оценки модели можно использовать индекс Дэвиса-Болдина ([sklearn.metrics.davies\_bouldin\_score](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.davies_bouldin_score.html#sklearn.metrics.davies_bouldin_score)), где более низкий индекс Дэвиса-Болдина относится к модели с лучшим разделением между кластерами.

Этот индекс означает среднее «сходство» между кластерами, где сходство — это мера, которая сравнивает расстояние между кластерами с размером самих кластеров.

Ноль — это наименьший возможный результат. Значения, близкие к нулю, указывают на лучшее разделение.

При обычном использовании индекс Дэвиса-Болдина применяется к результатам кластерного анализа следующим образом:



1

>>> from sklearn import datasets

2

>>> iris = datasets.load\_iris()

3

>>> X = iris.data

4

>>> from sklearn.cluster import KMeans

5

>>> from sklearn.metrics import davies\_bouldin\_score

6

>>> kmeans = KMeans(n\_clusters=3, random\_state=1).fit(X)

7

>>> labels = kmeans.labels\_

8

>>> davies\_bouldin\_score(X, labels)

9

0.6619...

### 2.3.10.7.1. Преимущества

* Вычисление Дэвиса-Боулдина проще, чем оценка Силуэта.
* В индексе вычисляются только количества и характеристики, присущие набору данных.

### 2.3.10.7.2. Недостатки

* Индекс Дэвиса-Боулдинга обычно выше для выпуклых кластеров, чем для других концепций кластеров, таких как кластеры на основе плотности, подобные тем, которые получены из DBSCAN.
* Использование центроидного расстояния ограничивает метрику расстояния евклидовым пространством.

### 2.3.10.7.3. Математическая постановка

Индекс определяется как среднее сходство между каждым кластером.

 для  и его самый похожий . В контексте этого индекса сходство определяется как мера

 что торгуется:

, среднее расстояние между каждой точкой кластера

  и центроид этого кластера — также известный как диаметр кластера.

 , расстояние между центрами скоплений  а также

* .

Простой выбор для построения

 так что он неотрицателен и симметричен:   
Тогда индекс Дэвиса-Болдина определяется как:

Рекомендации

* Дэвис, Дэвид Л .; Боулдин, Дональд В. (1979). «Мера разделения кластеров» IEEE-транзакции по анализу шаблонов и машинному интеллекту. ПАМИ-1 (2): 224-227. [DOI: 10.1109 / TPAMI.1979.4766909](https://doi.org/10.1109/TPAMI.1979.4766909) .
* Халкиди, Мария; Батистакис, Яннис; Вазиргианнис, Михалис (2001). «О методах кластеризации валидации» Журнал интеллектуальных информационных систем, 17 (2-3), 107-145. [DOI: 10,1023 / А: 1012801612483](https://doi.org/10.1023/A:1012801612483) .
* [Запись в Википедии для индекса Дэвиса-Болдина](https://en.wikipedia.org/wiki/Davies%E2%80%93Bouldin_index) .

### 2.3.10.8. Матрица непредвиденных обстоятельств

Матрица непредвиденных обстоятельств ( [sklearn.metrics.cluster.contingency\_matrix](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.cluster.contingency_matrix.html#sklearn.metrics.cluster.contingency_matrix)) сообщает мощность пересечения для каждой истинной / прогнозируемой пары кластеров. Матрица непредвиденных обстоятельств обеспечивает достаточную статистику для всех метрик кластеризации, где выборки независимы и одинаково распределены, и нет необходимости учитывать некоторые экземпляры, которые не были кластеризованы.

Вот пример:



1

>>> from sklearn.metrics.cluster import contingency\_matrix

2

>>> x = ["a", "a", "a", "b", "b", "b"]

3

>>> y = [0, 0, 1, 1, 2, 2]

4

>>> contingency\_matrix(x, y)

5

array([[2, 1, 0],

6

[0, 1, 2]])

Первая строка выходного массива указывает, что есть три образца, истинный кластер которых равен «a». Из них два находятся в предсказанном кластере 0, один — в 1 и ни один — в 2. И вторая строка указывает, что есть три выборки, истинный кластер которых равен «b». Из них ни один не находится в прогнозируемом кластере 0, один — в 1, а два — в 2.

[Матрица неточностей](https://scikit-learn.org/stable/modules/model_evaluation.html#confusion-matrix) для классификации является квадратной матрицей непредвиденной где порядок строк и столбцов соответствует списку классов.

### 2.3.10.8.1. Преимущества

* Позволяет изучить распространение каждого истинного кластера по прогнозируемым кластерам и наоборот.
* Вычисленная таблица непредвиденных обстоятельств обычно используется при вычислении статистики сходства (как и другие, перечисленные в этом документе) между двумя кластерами.

### 2.3.10.8.2. Недостатки

* Матрицу непредвиденных обстоятельств легко интерпретировать для небольшого количества кластеров, но становится очень трудно интерпретировать для большого количества кластеров.
* Он не дает ни одной метрики для использования в качестве цели для оптимизации кластеризации.

Рекомендации

* [Запись в Википедии о матрице непредвиденных обстоятельств](https://en.wikipedia.org/wiki/Contingency_table)

### 2.3.10.9. Матрица неточности пар

Матрица смешения пар ( [sklearn.metrics.cluster.pair\_confusion\_matrix](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.cluster.pair_confusion_matrix.html#sklearn.metrics.cluster.pair_confusion_matrix)) представляет собой матрицу подобия 2×2

между двумя кластерами, вычисленными путем рассмотрения всех пар выборок и подсчета пар, которые назначены в один и тот же или в разные кластеры в рамках истинной и прогнозируемой кластеризации.

В нем есть следующие записи:

: количество пар с обеими кластерами, в которых образцы не сгруппированы вместе  
: количество пар с истинной кластеризацией меток, в которых образцы сгруппированы вместе, но в другой кластеризации образцы не сгруппированы вместе  
: количество пар с истинной кластеризацией меток, в которых образцы не сгруппированы вместе, а другая кластеризация содержит образцы, сгруппированные вместе

: количество пар с обеими кластерами, имеющими образцы, сгруппированные вместе

Если рассматривать пару образцов, сгруппированных вместе как положительную пару, то, как и в бинарной классификации, количество истинных отрицательных значений равно

, ложноотрицательные , истинные положительные стороны  а ложные срабатывания

.

Идеально совпадающие метки имеют все ненулевые записи на диагонали независимо от фактических значений меток:



1

>>> from sklearn.metrics.cluster import pair\_confusion\_matrix

2

>>> pair\_confusion\_matrix([0, 0, 1, 1], [0, 0, 1, 1])

3

array([[8, 0],

4

[0, 4]])



1

>>> pair\_confusion\_matrix([0, 0, 1, 1], [1, 1, 0, 0])

2

array([[8, 0],

3

[0, 4]])

Метки, которые назначают все члены классов одним и тем же кластерам, являются полными, но не всегда могут быть чистыми, следовательно, наказываются и имеют некоторые недиагональные ненулевые записи:



1

>>> pair\_confusion\_matrix([0, 0, 1, 2], [0, 0, 1, 1])

2

array([[8, 2],

3

[0, 2]])

Матрица не симметрична:



1

>>> pair\_confusion\_matrix([0, 0, 1, 1], [0, 0, 1, 2])

2

array([[8, 0],

3

[2, 2]])

Если члены классов полностью разделены по разным кластерам, назначение полностью неполное, следовательно, матрица имеет все нулевые диагональные элементы:



1

>>> pair\_confusion\_matrix([0, 0, 0, 0], [0, 1, 2, 3])

2

array([[ 0, 0],

3

[12, 0]])

Рекомендации

* Л. Хьюберт и П. Араби, Сравнение разделов, журнал классификации 1985 < <https://link.springer.com/article/10.1007%2FBF01908075> > \_

Если вы хотите помочь проекту с переводом, то можно обращаться по следующему адресу [support@scikit-learn.ru](mailto:support@scikit-learn.ru)

<https://bpostance.github.io/posts/clustering-mixed-data/>

<https://bpostance.github.io/posts/clustering-mixed-data/>

# <https://newtechaudit.ru/gruppirovka-klasterizaczia/?ysclid=ls7ihozlv2994350521>

# Группировка пользователей методами кластеризации данных

#### 22.07.2020 Крехно Роман, г. Екатеринбург

Время прочтения: 6 мин.

Данные не только не всегда очевидно указывают на определенную группу, порой непонятно даже сколько таких групп. Для решения подобного рода проблем существует целый класс задач машинного обучения — кластеризация. Данный класс задач относится к категории unsupervised learning, то есть обучение без размеченных данных, поэтому является очень ценным в условиях, когда исследование только начинается. Эта статья посвящена решению проблемы группировки пользователей по собранной статистике методами кластеризации.

Возьмем для примера данные пользователей, которые заходили на сайт компании и кликали на ссылки. При помощи подходящего link tracker посчитаем количество кликов по тем или иным элементам сайта за определенное время от каждого зарегистрированного пользователя, просуммируем, например, по неделям и пронормируем на среднее. Для примера рассмотрим сгенерированный датасет.

Начнем с генерации сета, имитирующего поведение трех групп пользователей, кликающих по 5 ссылкам с 1000 разных аккаунтов. Для этого воспользуемся методом make\_blobs из пакета sklearn.

from sklearn.datasets import make\_blobs

import numpy as np

data, pregenerated= make\_blobs(1000,n\_features=5,cluster\_std=4)

В массив data будут записаны пять фич, каждая из которых является суммой трех гауссиан с центрами в трех разных точках, которые и являются кластерами. Имена этих заранее известных кластеров записаны в переменной  pregenerated. Построим фичи друг относительно друга:

from pandas.plotting import scatter\_matrix

import pandas as pd

scatter\_matrix(pd.DataFrame(data), alpha=0.2, figsize=(6, 6), )

pd.Series(pregenerated).value\_counts()

Видно, что по фичам ничего невозможно понять — даже сколько там кластеров. Это весьма похоже на реальные данные. Преобразуем массив данных в датасет и назовем фичи так, как будто это и есть реальные данные.

result\_df = pd.DataFrame(data)

result\_df.columns = ["каталог","доставка","о нас","реклама","адреса"]

result\_df["группа"] = pregenerated

result\_df.head()

Далее понаблюдаем, как даже самый базовый метод кластеризации легко справляется с задачей группировки данных. Прогоним полученный датасет через популярную модель для кластеризации — k-means. Ее главный минус в том, что нужно знать заранее количество кластеров. Обычно это решается либо эмпирически, либо моделями способными на поиск количества кластеров, таких как DBSCAN. У нас же количество кластеров известно заранее.

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=3).fit(data)

Для валидации построим распределение как изначальных, так и предсказанных кластеров.

result\_df["предсказанные"] = kmeans.labels\_

print(result\_df["группа"].value\_counts())

print(result\_df["предсказанные"].value\_counts())

result\_df.head()

0    334

2    333

1    333

Name: группа, dtype: int64

0    341

1    333

2    326

Name: предсказанные, dtype: int64

Поскольку мы заранее знали к какому кластеру принадлежит тот или иной пользователь — мы ведь их сами создавали — есть возможность проверить насколько хорошо справился k-means со своей задачей. В соответствии с вышеприведенной частью датасета переназначим найденные алгоритмом кластеры и сравним их с сгенерированными:

temp = result\_df["предсказанные"].map({2:0,0:1,1:2}) - result\_df["группа"]

temp.value\_counts(normalize=True)

0    0.971

-1    0.018

 1    0.011

dtype: float64

Впечатляющее качество распознавания! К сожалению, в реальных данных этой информации как правило нет, а если есть, то это уже supervised learning и изначально нужно было решать задачу классификации, а не кластеризации. Проблему сложности восприятия табличных данных о кластерах принято решать графически при помощи dimensional reduction, что позволяет все фичи ужать до двух, x и y, и построить их на графике. Существует много алгоритмов dr, мы покажем один из самых популярных из-за своей красоты t-sne.

from matplotlib import pyplot as plt

from sklearn.manifold import TSNE

tsne\_df = TSNE(n\_components=2,perplexity=20).fit\_transform(data)

result\_df["tsneX"] = tsne\_df[:,0]

result\_df["tsneY"] = tsne\_df[:,1]

plt.scatter(result\_df["tsneX"], result\_df["tsneY"],s=2,)

plt.savefig("blobs\_gray.jpg",dpi=1200,transparent=True)

Как мы видим, пять фич сжались до двух, чтобы уместиться на плоскости. Кроме того, точки разделились на несколько различимых взглядом агломератов. Для проверки выделим точки справа по координатам и посмотрим какие из изначально созданных групп пользователей в него входят.

result\_df[(result\_df['tsneX']>15) & (result\_df['tsneY']<5)]["группа"].value\_counts()

0    325

1     19

Name: группа, dtype: int64

Действительно, соответствует сгенерированному кластеру с кодом 0.

Если бы это были реальные данные, мы бы многое поняли о пользователях из этого графика — во первых группы пользователей действительно есть, их как минимум две, во вторых — все пользователи пользуются всеми ссылками, просто с разной интенсивностью, ведь кластеры связаны. Чтобы извлечь больше выводов из этого графика, можно добавить в него больше внешней информации. Проще всего изменению поддаются цвет точек, их фон и размер. Красить точки и фон лучше в категориальные фичи, размер же отведен для численных. В данном примере у нас нет никаких категориальных фич, кроме, собственно, кластеров к которым принадлежат точки. Покрасим точки на графике в цвета согласно известным нам заранее группам пользователей, для валидации.  Этой информации у модели не было.

def colorer(row,column):

if row[column] == 1:

return "Green"

if row[column] == 2:

return "Brown"

return "Magenta"

result\_df["цвет"] = result\_df.apply(colorer,axis = 1,column = "группа")

plt.scatter(result\_df["tsneX"], result\_df["tsneY"],s=2,c=list(result\_df['цвет']))

plt.savefig("blobs\_colored.jpg",dpi=1200,transparent=True)

Метод t-sne сумел отделить фиолетовый кластер от двух других. В реальной ситуации в качестве цвета обычно выступает собранная помимо статистики переходов по ссылкам эвристика, например user agent их браузера, или их участие/неучастие в опросе, или проведенное на странице время. Бывает так, что кластеры видны плохо и выводы делать трудно. Для визуального определения помимо цвета можно использовать размер точек, делая более важные точки больше. При анализе поведения пользователей одной из самых важных для визуализации фич является активность.

Для примера сгенерируем ее при помощи случайных чисел, зависящих от номера кластера

result\_df["активность"] = np.random.randint(1,10, len(result\_df))\*(result\_df["группа"]+0.1)

plt.scatter(result\_df["tsneX"], result\_df["tsneY"],s=list(result\_df['активность']),c=list(result\_df['цвет']))

plt.savefig("blobs\_size.jpg",dpi=1200,transparent=True)

Видно, что в фиолетовый кластер входили самые неактивные пользователи, а в коричневый — самые активные. При этом случайные вбросы фиолетового цвета в зеленый стали менее видны, и это плюс, ведь нам важнее поведение активных пользователей, нежели чем неактивных.

Параметр размера также можно использовать для создания фона под картинкой, добавляя второй scatter на тот же график, с большим размером точки s и низкой прозрачностью alpha

result\_df["фон"] = result\_df.apply(colorer,axis = 1,column = "предсказанные")

plt.scatter(result\_df["tsneX"], result\_df["tsneY"],s = 100, c=list(result\_df['фон'])

,marker = "o",alpha = 0.05)

plt.scatter(result\_df["tsneX"], result\_df["tsneY"],s=list(result\_df['активность']),c=list(result\_df['цвет']))

plt.savefig("blobs\_background.jpg",dpi=1200,transparent=True)

Использование фона очень полезно, если есть возможность эвристически распределить пользователей в другие группы, не связанные с поведением, например на фон можно поместить категорию размера города для каждой точки.

На этом этап построения графиков заканчивается, и начинается анализ результатов, включающий демонстрацию графика коллегам, поиск скрытых в нем инсайдов и многого другого, выходящего за рамки статьи. Совокупность вышеперечисленных подходов применима к любым неразмеченным данным, и была успешно использована для выявления интереса (и отсутствия интереса) к продуктам среди посетителей определенного сайта. Подобное исследование может дать пищу для размышлений в рамках выполнения обязанностей аналитиков, маркетологов и аудиторов.

Код можно скачать по ссылке <https://github.com/RomanKrekhno/table_recognition_example/blob/master/clustering%20example.ipynb>

Начало формы

Конец формы

3. <https://teletype.in/@dt_analytic/A9eoW4ZPeMu?ysclid=ls7ijyxr64250608748>

[[](https://yandex.ru/an/count/WdWejI_zOoVX2LdU0bKJ04CWVIOQbKgbKga4mGHzFfSxUxRVkVDTNpjVl-FrXZlS8g4HrgCRIjOXgz9F7bKwak5yWl21uZuWdDCfBdNY-NIAMvEGbeo2h5QawqyDwb2IK9KwLCE4DGbfoIZeH-b8XX9rHj3FqH299QGHDvI4af2iDOIAbkeXAjAIjUZ83fYoYO67f8m2SKelG5V8J64333gE9fveQMbfQMdfx9XUyIf0Exq2f03jz4fN0Thf5Qu2j5CcSGLefqpW2j1EcNmBq4wPnELyJKGHI_og85tBwTA0Uehx43OA0IY3Em_2mPD2QKQ75ayNJIOGeDB5PUFjm8yOGstLlcZSHpIlD5z3_Z242UAupE-F2-BKJiETCeaBO8l1MqIho5gXe5SFBsOCsWCCmJwCG663T68gWiNEa14GiZQpfQ_q8WDZo35Gp4CcHDhG1EZ2T0ymeZnRO4YEsTJheCdjeBbJCCH-yGPF37-OHZI4-k01dWR_Tu2dmUyTu6dmSmku0lPtf21DNwlYov6VHGhIn3HJa8LPNe2iwxgEfSqix22Fq8qRhhQSgNszEG5VzaB2kc1vieyxB0BenxG76w16PPZblqQ5V4Ao_9RMTHrB8P-GBA_rQcvYOP4-wm-d_CAAQALOHT5mFRQvKgbKMksaDB2zhkiObeJA3u1b4RKB2GsXAlougmp0r2kuwfTFekBa4sgkAL_dxQ4j7_7hMUxOLWCq4_kM9rr39MaDRGonu8Z-Tv2IcqMK00TU_j9o4-bfJmGO-4HurfMJeGPyBaPRXcURUhn3bTPnebjk7ucSp8g9H1xHfSCd38o4-ZABn-crSW83t000~2?stat-id=1&test-tag=262783279300625&banner-sizes=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjcwMHgzMTAifQ%3D%3D&actual-format=10&pcodever=958942&banner-test-tags=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjQ1NjQxODEwMDkifQ%3D%3D&constructor-rendered-assets=eyI3MjA1NzYwOTAzNjMzNTQzNSI6MTA1OTM0MX0&width=700&height=310&pcode-active-testids=938403%2C0%2C92%3B955911%2C0%2C14&subDesignId=1000696000)](https://yandex.ru/an/count/WdWejI_zOoVX2LdU0bKJ04CWVIOQbKgbKga4mGHzFfSxUxRVkVDTNpjVl-FrXZlS8g4HrgCRIjOXgz9F7bKwak5yWl21uZuWdDCfBdNY-NIAMvEGbeo2h5QawqyDwb2IK9KwLCE4DGbfoIZeH-b8XX9rHj3FqH299QGHDvI4af2iDOIAbkeXAjAIjUZ83fYoYO67f8m2SKelG5V8J64333gE9fveQMbfQMdfx9XUyIf0Exq2f03jz4fN0Thf5Qu2j5CcSGLefqpW2j1EcNmBq4wPnELyJKGHI_og85tBwTA0Uehx43OA0IY3Em_2mPD2QKQ75ayNJIOGeDB5PUFjm8yOGstLlcZSHpIlD5z3_Z242UAupE-F2-BKJiETCeaBO8l1MqIho5gXe5SFBsOCsWCCmJwCG663T68gWiNEa14GiZQpfQ_q8WDZo35Gp4CcHDhG1EZ2T0ymeZnRO4YEsTJheCdjeBbJCCH-yGPF37-OHZI4-k01dWR_Tu2dmUyTu6dmSmku0lPtf21DNwlYov6VHGhIn3HJa8LPNe2iwxgEfSqix22Fq8qRhhQSgNszEG5VzaB2kc1vieyxB0BenxG76w16PPZblqQ5V4Ao_9RMTHrB8P-GBA_rQcvYOP4-wm-d_CAAQALOHT5mFRQvKgbKMksaDB2zhkiObeJA3u1b4RKB2GsXAlougmp0r2kuwfTFekBa4sgkAL_dxQ4j7_7hMUxOLWCq4_kM9rr39MaDRGonu8Z-Tv2IcqMK00TU_j9o4-bfJmGO-4HurfMJeGPyBaPRXcURUhn3bTPnebjk7ucSp8g9H1xHfSCd38o4-ZABn-crSW83t000~2?stat-id=1&test-tag=262783279300625&banner-sizes=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjcwMHgzMTAifQ%3D%3D&actual-format=10&pcodever=958942&banner-test-tags=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjQ1NjQxODEwMDkifQ%3D%3D&constructor-rendered-assets=eyI3MjA1NzYwOTAzNjMzNTQzNSI6MTA1OTM0MX0&width=700&height=310&pcode-active-testids=938403%2C0%2C92%3B955911%2C0%2C14&subDesignId=1000696000" \t "_blank)

[[https://favicon.yandex.net/favicon/neural-university.ru?size=32&stub=2](https://yandex.ru/an/count/WdWejI_zOoVX2LdU0bKJ04CWVIOQbKgbKga4mGHzFfSxUxRVkVDTNpjVl-FrXZlS8g4HrgCRIjOXgz9F7bKwak5yWl21uZuWdDCfBdNY-NIAMvEGbeo2h5QawqyDwb2IK9KwLCE4DGbfoIZeH-b8XX9rHj3FqH299QGHDvI4af2iDOIAbkeXAjAIjUZ83fYoYO67f8m2SKelG5V8J64333gE9fveQMbfQMdfx9XUyIf0Exq2f03jz4fN0Thf5Qu2j5CcSGLefqpW2j1EcNmBq4wPnELyJKGHI_og85tBwTA0Uehx43OA0IY3Em_2mPD2QKQ75ayNJIOGeDB5PUFjm8yOGstLlcZSHpIlD5z3_Z242UAupE-F2-BKJiETCeaBO8l1MqIho5gXe5SFBsOCsWCCmJwCG663T68gWiNEa14GiZQpfQ_q8WDZo35Gp4CcHDhG1EZ2T0ymeZnRO4YEsTJheCdjeBbJCCH-yGPF37-OHZI4-k01dWR_Tu2dmUyTu6dmSmku0lPtf21DNwlYov6VHGhIn3HJa8LPNe2iwxgEfSqix22Fq8qRhhQSgNszEG5VzaB2kc1vieyxB0BenxG76w16PPZblqQ5V4Ao_9RMTHrB8P-GBA_rQcvYOP4-wm-d_CAAQALOHT5mFRQvKgbKMksaDB2zhkiObeJA3u1b4RKB2GsXAlougmp0r2kuwfTFekBa4sgkAL_dxQ4j7_7hMUxOLWCq4_kM9rr39MaDRGonu8Z-Tv2IcqMK00TU_j9o4-bfJmGO-4HurfMJeGPyBaPRXcURUhn3bTPnebjk7ucSp8g9H1xHfSCd38o4-ZABn-crSW83t000~2?stat-id=1&test-tag=262783279300625&banner-sizes=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjcwMHgzMTAifQ%3D%3D&actual-format=10&pcodever=958942&banner-test-tags=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjQ1NjQxODEwMDkifQ%3D%3D&constructor-rendered-assets=eyI3MjA1NzYwOTAzNjMzNTQzNSI6MTA1OTM0MX0&width=700&height=310&pcode-active-testids=938403%2C0%2C92%3B955911%2C0%2C14&subDesignId=1000696000)](https://yandex.ru/an/count/WdWejI_zOoVX2LdU0bKJ04CWVIOQbKgbKga4mGHzFfSxUxRVkVDTNpjVl-FrXZlS8g4HrgCRIjOXgz9F7bKwak5yWl21uZuWdDCfBdNY-NIAMvEGbeo2h5QawqyDwb2IK9KwLCE4DGbfoIZeH-b8XX9rHj3FqH299QGHDvI4af2iDOIAbkeXAjAIjUZ83fYoYO67f8m2SKelG5V8J64333gE9fveQMbfQMdfx9XUyIf0Exq2f03jz4fN0Thf5Qu2j5CcSGLefqpW2j1EcNmBq4wPnELyJKGHI_og85tBwTA0Uehx43OA0IY3Em_2mPD2QKQ75ayNJIOGeDB5PUFjm8yOGstLlcZSHpIlD5z3_Z242UAupE-F2-BKJiETCeaBO8l1MqIho5gXe5SFBsOCsWCCmJwCG663T68gWiNEa14GiZQpfQ_q8WDZo35Gp4CcHDhG1EZ2T0ymeZnRO4YEsTJheCdjeBbJCCH-yGPF37-OHZI4-k01dWR_Tu2dmUyTu6dmSmku0lPtf21DNwlYov6VHGhIn3HJa8LPNe2iwxgEfSqix22Fq8qRhhQSgNszEG5VzaB2kc1vieyxB0BenxG76w16PPZblqQ5V4Ao_9RMTHrB8P-GBA_rQcvYOP4-wm-d_CAAQALOHT5mFRQvKgbKMksaDB2zhkiObeJA3u1b4RKB2GsXAlougmp0r2kuwfTFekBa4sgkAL_dxQ4j7_7hMUxOLWCq4_kM9rr39MaDRGonu8Z-Tv2IcqMK00TU_j9o4-bfJmGO-4HurfMJeGPyBaPRXcURUhn3bTPnebjk7ucSp8g9H1xHfSCd38o4-ZABn-crSW83t000~2?stat-id=1&test-tag=262783279300625&banner-sizes=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjcwMHgzMTAifQ%3D%3D&actual-format=10&pcodever=958942&banner-test-tags=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjQ1NjQxODEwMDkifQ%3D%3D&constructor-rendered-assets=eyI3MjA1NzYwOTAzNjMzNTQzNSI6MTA1OTM0MX0&width=700&height=310&pcode-active-testids=938403%2C0%2C92%3B955911%2C0%2C14&subDesignId=1000696000" \t "_blank)

[neural-university.ru](https://yandex.ru/an/count/WdWejI_zOoVX2LdU0bKJ04CWVIOQbKgbKga4mGHzFfSxUxRVkVDTNpjVl-FrXZlS8g4HrgCRIjOXgz9F7bKwak5yWl21uZuWdDCfBdNY-NIAMvEGbeo2h5QawqyDwb2IK9KwLCE4DGbfoIZeH-b8XX9rHj3FqH299QGHDvI4af2iDOIAbkeXAjAIjUZ83fYoYO67f8m2SKelG5V8J64333gE9fveQMbfQMdfx9XUyIf0Exq2f03jz4fN0Thf5Qu2j5CcSGLefqpW2j1EcNmBq4wPnELyJKGHI_og85tBwTA0Uehx43OA0IY3Em_2mPD2QKQ75ayNJIOGeDB5PUFjm8yOGstLlcZSHpIlD5z3_Z242UAupE-F2-BKJiETCeaBO8l1MqIho5gXe5SFBsOCsWCCmJwCG663T68gWiNEa14GiZQpfQ_q8WDZo35Gp4CcHDhG1EZ2T0ymeZnRO4YEsTJheCdjeBbJCCH-yGPF37-OHZI4-k01dWR_Tu2dmUyTu6dmSmku0lPtf21DNwlYov6VHGhIn3HJa8LPNe2iwxgEfSqix22Fq8qRhhQSgNszEG5VzaB2kc1vieyxB0BenxG76w16PPZblqQ5V4Ao_9RMTHrB8P-GBA_rQcvYOP4-wm-d_CAAQALOHT5mFRQvKgbKMksaDB2zhkiObeJA3u1b4RKB2GsXAlougmp0r2kuwfTFekBa4sgkAL_dxQ4j7_7hMUxOLWCq4_kM9rr39MaDRGonu8Z-Tv2IcqMK00TU_j9o4-bfJmGO-4HurfMJeGPyBaPRXcURUhn3bTPnebjk7ucSp8g9H1xHfSCd38o4-ZABn-crSW83t000~2?stat-id=1&test-tag=262783279300625&banner-sizes=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjcwMHgzMTAifQ%3D%3D&actual-format=10&pcodever=958942&banner-test-tags=eyI3MjA1NzYwOTAzNjMzNTQzNSI6IjQ1NjQxODEwMDkifQ%3D%3D&constructor-rendered-assets=eyI3MjA1NzYwOTAzNjMzNTQzNSI6MTA1OTM0MX0&width=700&height=310&pcode-active-testids=938403%2C0%2C92%3B955911%2C0%2C14&subDesignId=1000696000" \t "_blank)

Реклама

4. После того, как все точки данных назначены кластеру, вычислите новый центроид и поместите его в каждый кластер. На основе нового центроида найдите, нужно ли переназначить какие-либо точки данных новому кластеру на основе его близости к центроидам.

**Метод K-средних итеративно находит лучшие центроиды для кластера, чередуя их**

* **Назначение точек данных кластерам на основе их близости к текущим центроидам и**
* **Повторное вычисление центроидов на основе текущего назначения точек данных кластерам.**

Алгоритм K-средних останавливает уточнение кластера, когда

* **Центроиды кластера стабилизировались, когда больше нет изменений в точке данных кластера или**
* **Алгоритм выполнил указанное количество итераций.**

Алгоритм K-средних используется для

* **Обнаружение аномалий, таких как обнаружение мошенничества**
* **Сегментация, такая как сегментация клиентов, сегментация изображений**
* **Интеллектуальный анализ данных**

#### **Внедрение K-средних**

[Набор данных клиентов торгового центра](https://www.kaggle.com/shwetabh123/mall-customers)

Прочтите файл CSV во фрейме данных, преобразуйте категориальную переменную в числовой тип данных, а затем уменьшите размер с помощью PCA (анализ главных компонентов)

# Import the required libraries

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.cluster import KMeans**

**from sklearn.preprocessing import LabelBinarizer**

**from sklearn.decomposition import PCA**

# Read he dataset into a dataframe

**dataset = pd.read\_csv('Mall\_Customers.csv',index\_col='CustomerID')**

# Drop duplicates

**dataset.drop\_duplicates(inplace=True)**

# Converting categorical column Genre to onehot vector

**label\_binarizer = LabelBinarizer()**

#use LabelBinarizer for gender

**label\_binarizer\_output = label\_binarizer.fit\_transform( dataset['Genre'])**

#Adding the categorical and the main dataframe into one dataframe

**result\_df = pd.DataFrame(label\_binarizer\_output, columns=['Gender\_1'])**

**dataset\_1= pd.concat([dataset, result\_df], axis=1, join='inner')**

# Creating the input variable

**X= dataset\_1.iloc[:, [1,2,3,4]].values**

# reducing the input dimension using PCA

**reduced\_data = PCA(n\_components=2).fit\_transform(X)**

Когда мы запускаем KMeans, нам необходимо предоставить **n\_clusters, которые представляют собой оптимальное количество кластеров**, которые необходимо сформировать, и количество центроидов, которые необходимо сгенерировать для набора данных.

**kmeans = KMeans(init="k-means++", n\_clusters=5, n\_init=4)**

**kmeans.fit(X)**

**Как лучше всего найти оптимальные кластеры?**

Существует 3 популярных метода поиска оптимального кластера для алгоритма кластеризации KMeans.

#### **Метод локтя**

В методе локтя алгоритм K-средних запускается для разного количества кластеров, чтобы найти сумму квадратов расстояний каждой точки данных от центроида кластера, также называемую **суммой квадратов внутри кластера**.

Постройте WCSS вместе с количеством кластеров K. **Выберите значение K на графике, где WCSS начинает сглаживаться или внезапно падает, а добавление еще одного кластера не приводит к значительному улучшению, образуя изгиб**. Изгиб считается индикатором оптимального кластера для набора данных.

# Using the elbow method to find the optimal number of clusters

**max\_k=11** # max no. of clusters to be evaluated

**wcss = []**

**for i in range(1, max\_k):**

**kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)**

**kmeans.fit(reduced\_data)**

# inertia method returns wcss for that model

**wcss.append(kmeans.inertia\_)**

#plotting the data

**plt.figure(figsize=(5,3))**

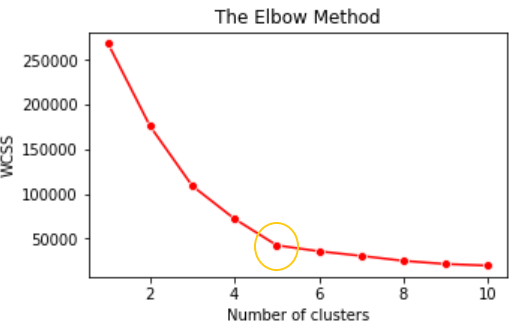
**sns.lineplot(range(1, max\_k), wcss,marker='o',color='red')**

**plt.title('The Elbow Method')**

**plt.xlabel('Number of clusters')**

**plt.ylabel('WCSS')**

**plt.show()**



#### **Оценка силуэта**

**Оценка силуэта - это еще один показатель, позволяющий проверить компактность кластера, чтобы определить, является ли кластер хорошим**.

Оценка силуэта **вычисляет среднее расстояние внутри кластера**, как метод локтя, а также **среднее значение ближайшего кластера**. Это показывает, насколько точка данных похожа на ее собственный кластер по сравнению со всеми другими центроидами кластера. Следовательно, **показатель силуэта измеряет как компактность одного и того же кластера, так и разделение с другими кластерами**.

**Если значение показателя силуэта для всех точек данных значительно велико, это дает оптимальное значение для кластеров**. Оценка силуэта использует расстояние Минковского или евклидово расстояние, и его значение находится в диапазоне [-1, 1].

Приведенный ниже код адаптирован из [https://scikit-learn.org/stable/auto\_examples/cluster/plot\_kmeans\_sil Silhouette\_analysis.html#](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html).

Код модифицирован для использования сокращенного измерения данных с помощью PCA.

**from sklearn.metrics import silhouette\_samples, silhouette\_score**

**import matplotlib.cm as cm**

**for n\_clusters in range(2, max\_k):**

# Create a subplot with 1 row and 2 columns

**fig, (ax1, ax2) = plt.subplots(1, 2)**

**fig.set\_size\_inches(18, 7)**

# The 1st subplot is the silhouette plot

# The silhouette coefficient can range from -1, 1 but in this example all

# lie within [-0.1, 1]

**ax1.set\_xlim([-0.1, 1])**

# The (n\_clusters+1)\*10 is for inserting blank space between silhouette

# plots of individual clusters, to demarcate them clearly.

**ax1.set\_ylim([0, len(reduced\_data) + (n\_clusters + 1) \* 10])**

# Initialize the clusterer with n\_clusters value and a random generator

# seed of 10 for reproducibility.

**clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)**

**cluster\_labels = clusterer.fit\_predict(reduced\_data)**

# The silhouette\_score gives the average value for all the samples.

# This gives a perspective into the density and separation of the formed

# clusters

**silhouette\_avg = silhouette\_score(reduced\_data, cluster\_labels)**

**print("For n\_clusters =", n\_clusters,"The average silhouette\_score is :", silhouette\_avg)**

# Compute the silhouette scores for each sample

**sample\_silhouette\_values = silhouette\_samples(reduced\_data, cluster\_labels)**

**y\_lower = 10**

**for i in range(n\_clusters):**

# Aggregate the silhouette scores for samples belonging to

# cluster i, and sort them

**ith\_cluster\_silhouette\_values = \**

**sample\_silhouette\_values[cluster\_labels == i]**

**ith\_cluster\_silhouette\_values.sort()**

**size\_cluster\_i = ith\_cluster\_silhouette\_values.shape[0]**

**y\_upper = y\_lower + size\_cluster\_i**

**color = cm.nipy\_spectral(float(i) / n\_clusters)**

**ax1.fill\_betweenx(np.arange(y\_lower, y\_upper),**

**0, ith\_cluster\_silhouette\_values,**

**facecolor=color, edgecolor=color, alpha=0.7)**

# Label the silhouette plots with their cluster numbers at the middle

**ax1.text(-0.05, y\_lower + 0.5 \* size\_cluster\_i, str(i))**

# Compute the new y\_lower for next plot

**y\_lower = y\_upper + 10** # 10 for the 0 samples

**ax1.set\_title("The silhouette plot for the various clusters.")**

**ax1.set\_xlabel("The silhouette coefficient values")**

**ax1.set\_ylabel("Cluster label")**

# The vertical line for average silhouette score of all the values

**ax1.axvline(x=silhouette\_avg, color="red", linestyle="--")**

**ax1.set\_yticks([]) # Clear the yaxis labels / ticks**

**ax1.set\_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])**

# 2nd Plot showing the actual clusters formed

**colors = cm.nipy\_spectral(cluster\_labels.astype(float) / n\_clusters)**

**ax2.scatter(reduced\_data[:, 0], reduced\_data[:, 1], marker='.', s=30, lw=0, alpha=0.7,**

**c=colors, edgecolor='k')**

# Labeling the clusters

**centers = clusterer.cluster\_centers\_**

# Draw white circles at cluster centers

**ax2.scatter(centers[:, 0], centers[:, 1], marker='o',**

**c="white", alpha=1, s=200, edgecolor='k')**

**for i, c in enumerate(centers):**

**ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,**

**s=50, edgecolor='k')**

**ax2.set\_title("The visualization of the clustered data.")**

**ax2.set\_xlabel("Feature space for the 1st feature")**

**ax2.set\_ylabel("Feature space for the 2nd feature")**

**plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "**

**"with n\_clusters = %d" % n\_clusters),**

**fontsize=14, fontweight='bold')**

**plt.show()**

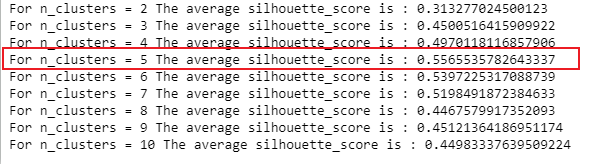
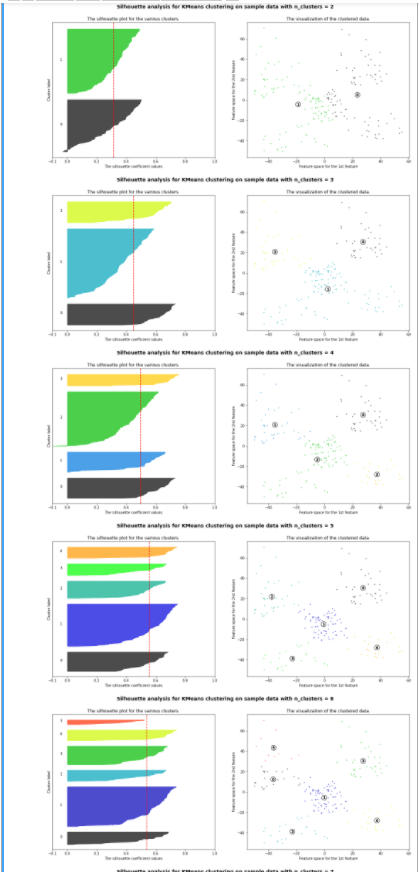


График силуэта предполагает, что 5 является хорошим выбором для оптимального кластера для заданных данных. Оптимальные кластеры оцениваются по наличию кластеров с оценкой силуэта выше среднего и соответствующими колебаниями в размере участков силуэта.



**И метод локтя, и оценка по силуэту дают оптимальные кластеры 5**

Запуск KMeans с оптимальным кластером, равным 5, а затем построение диаграммы рассеяния с использованием оптимальных кластеров по уменьшенным\_data для визуализации кластера

# Fitting K-Means to the dataset

**kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42)**

**y\_kmeans = kmeans.fit\_predict(reduced\_data)**

#Visualising the clusters

**plt.figure(figsize=(15,7))**

**sns.scatterplot(reduced\_data[y\_kmeans == 0, 0], reduced\_data[y\_kmeans == 0, 1], color = 'yellow', label = 'Cluster 1',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 1, 0], reduced\_data[y\_kmeans == 1, 1], color = 'blue', label = 'Cluster 2',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 2, 0], reduced\_data[y\_kmeans == 2, 1], color = 'green', label = 'Cluster 3',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 3, 0], reduced\_data[y\_kmeans == 3, 1], color = 'grey', label = 'Cluster 4',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 4, 0], reduced\_data[y\_kmeans == 4, 1], color = 'orange', label = 'Cluster 5',s=50)**

**sns.scatterplot(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], color = 'red',**

**label = 'Centroids',s=300,marker=',')**

**plt.grid(False)**

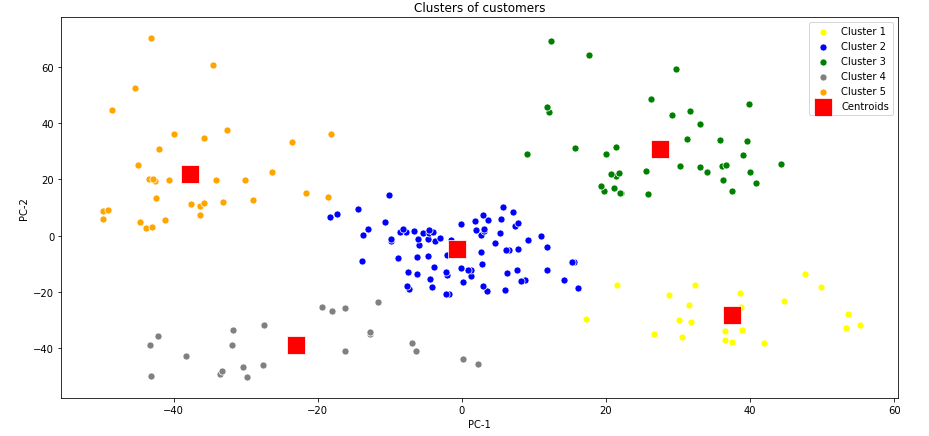
**plt.title('Clusters of customers')**

**plt.xlabel('PC-1')**

**plt.ylabel('PC-2')**

**plt.legend()**

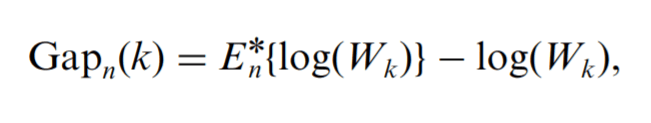
**plt.show()**



#### **Статистика разрывов**

Статистика разрыва - еще один популярный метод определения оптимального количества кластеров, который можно применить к любому методу кластеризации.

**Идея статистики разрыва состоит в том, чтобы сравнить инерцию кластера в отношении данных с ее ожиданием в соответствующем наборе данных с нулевой ссылкой.** Наборы данных с нулевой ссылкой могут быть взяты из нормального или равномерного распределения. **Оптимальный выбор K - это значение, которое максимизирует статистику разрыва между внутрикластерной инерцией набора данных и набором данных с нулевой ссылкой**.



**Статистика разрыва - это разница между E ₙ, который представляет собой ожидание данных с нулевой ссылкой, вычисленное как логарифм общей внутрикластерной вариации и итоговой внутрикластерной вариации для набора данных**

Приведенный ниже код вдохновлен и адаптирован из <https://glowingpython.blogspot.com/2019/01/a-visual-introduction-to-gap-statistics.html>

Набор данных с нулевой ссылкой - это равномерно распределенный набор точек.

**from sklearn.datasets import make\_blobs**

**from sklearn.metrics import pairwise\_distances**

# creating a uniform distributed null refernced dataset

**reference = np.random.rand(100, 2)**

**plt.figure(figsize=(12, 3))**

**for k in range(1,6):**

**kmeans = KMeans(n\_clusters=k)**

**a = kmeans.fit\_predict(reference)**

**plt.subplot(1,5,k)**

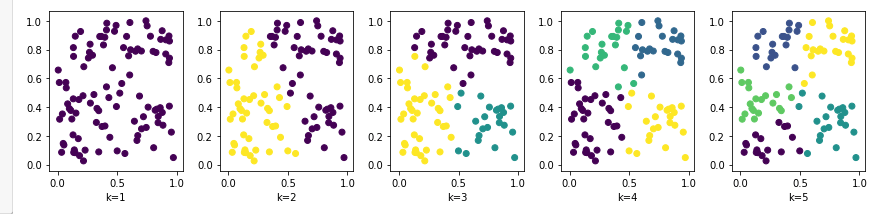
**plt.scatter(reference[:, 0], reference[:, 1], c=a)**

**plt.xlabel('k='+str(k))**

**plt.tight\_layout()**

**plt.show()**

Поскольку точки распределены равномерно, алгоритм KMeans равномерно разбивает точки на K кластеров, даже если между ними нет разделения.



**def compute\_inertia(a, X):**

**W = [np.mean(pairwise\_distances(X[a == c, :])) for c in np.unique(a)]**

**return np.mean(W)**

**def compute\_gap(clustering, data, k\_max=5, n\_references=5):**

**if len(data.shape) == 1:**

**data = data.reshape(-1, 1)**

# The Null Reference dataset that is uniformly distributed

**reference = np.random.rand(\*data.shape)**

**reference\_inertia = []**

# Calculate the WCSS for the null refrenced data

**for k in range(1, k\_max+1):**

**local\_inertia = []**

**for \_ in range(n\_references):**

**clustering.n\_clusters = k**

**assignments = clustering.fit\_predict(reference)**

**local\_inertia.append(compute\_inertia(assignments, reference))**

**reference\_inertia.append(np.mean(local\_inertia))**

# Calculate the WCSS for the data

**ondata\_inertia = []**

**for k in range(1, k\_max+1):**

**clustering.n\_clusters = k**

**assignments = clustering.fit\_predict(data)**

**ondata\_inertia.append(compute\_inertia(assignments, data))**

# Calculate the gao statistics between the WCSS for the null referenced data and the WCSS for the data

**gap = np.log(reference\_inertia)-np.log(ondata\_inertia)**

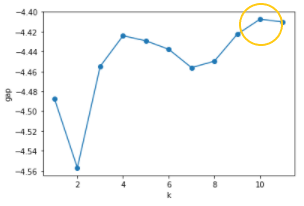
**return gap, np.log(reference\_inertia), np.log(ondata\_inertia)**

**gap, reference\_inertia, ondata\_inertia = compute\_gap(KMeans(), reduced\_data, k\_max)**

**plt.plot(range(1, k\_max+1), gap, '-o')**

**plt.ylabel('gap')**

**plt.xlabel('k')**



Статистика разрывов дает оптимальное количество кластеров, равное 10, на основе максимального разрыва между **инерцией кластера данных** и данными с нулевой ссылкой.

Запуск KMeans для 10 кластеров, а затем визуализация кластеров на диаграмме рассеяния.

# Visualising the clusters

**plt.figure(figsize=(15,7))**

**sns.scatterplot(reduced\_data[y\_kmeans == 0, 0], reduced\_data[y\_kmeans == 0, 1], color = 'yellow', label = 'Cluster 1',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 1, 0], reduced\_data[y\_kmeans == 1, 1], color = 'blue', label = 'Cluster 2',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 2, 0], reduced\_data[y\_kmeans == 2, 1], color = 'green', label = 'Cluster 3',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 3, 0], reduced\_data[y\_kmeans == 3, 1], color = 'grey', label = 'Cluster 4',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 4, 0], reduced\_data[y\_kmeans == 4, 1], color = 'orange', label = 'Cluster 5',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 5, 0], reduced\_data[y\_kmeans == 5, 1], color = 'pink', label = 'Cluster 6',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 6, 0], reduced\_data[y\_kmeans == 6, 1], color = 'magenta', label = 'Cluster 7',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 7, 0], reduced\_data[y\_kmeans == 7, 1], color = 'cyan', label = 'Cluster 8',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 8, 0], reduced\_data[y\_kmeans == 8, 1], color = 'purple', label = 'Cluster 9',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 9, 0], reduced\_data[y\_kmeans == 9, 1], color = 'brown', label = 'Cluster 10',s=50)**

**sns.scatterplot(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], color = 'red',**

**label = 'Centroids',s=300,marker=',')**

**plt.grid(False)**

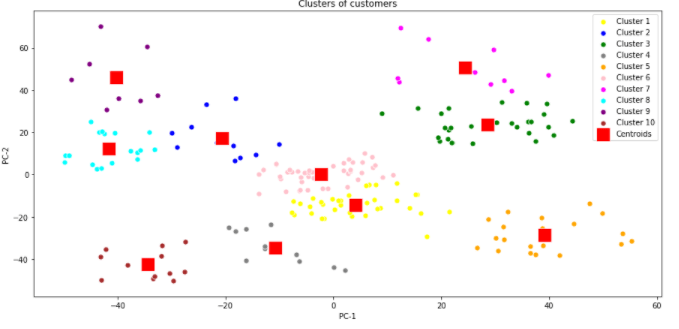
**plt.title('Clusters of customers')**

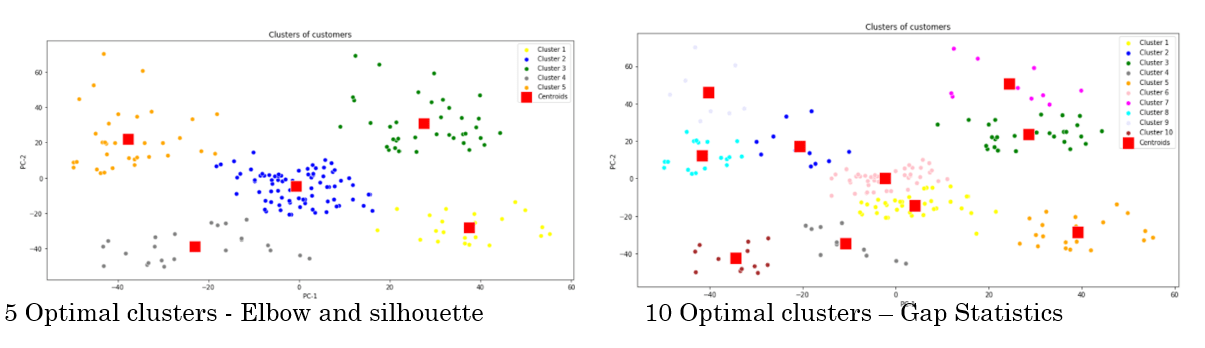
**plt.xlabel('PC-1')**

**plt.ylabel('PC-2')**

**plt.legend()**

**plt.show()**





#### Вывод:

Кластеризация K-средних - это

* **Алгоритм машинного обучения без учителя**
* **Итерационный алгоритм для поиска групп данных со схожими характеристиками для немаркированного набора данных**

Популярные методы определения оптимальных кластеров

* **Метод изгиба**: когда WCSS внезапно начинает сглаживаться, и добавление еще одного кластера не приводит к значительному улучшению, образуя изгиб.
* **Оценка силуэта**: измеряет как компактность с одним и тем же кластером, так и разделение с другими кластерами.
* **Статистика разрыва.** Максимальный разрыв между внутрикластерной инерцией набора данных и набором данных, на который имеется ссылка на null, распределяется равномерно.

#### Использованная литература:

[Https://stanford.edu/~cpiech/cs221/handouts/kmeans.html](https://stanford.edu/~cpiech/cs221/handouts/kmeans.html)

[**Кластеризация K-средних: Справочник по модулю - Машинное обучение Azure**   
В этой статье описывается, как использовать модуль кластеризации K-средних в конструкторе машинного обучения Azure для создания… docs.microsoft.com](https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/k-means-clustering)

[Https://arxiv.org/ftp/arxiv/papers/1912/1912.00643.pdf](https://arxiv.org/ftp/arxiv/papers/1912/1912.00643.pdf)

После того, как все точки данных назначены кластеру, вычислите новый центроид и поместите его в каждый кластер. На основе нового центроида найдите, нужно ли переназначить какие-либо точки данных новому кластеру на основе его близости к центроидам.

**Метод K-средних итеративно находит лучшие центроиды для кластера, чередуя их**

* **Назначение точек данных кластерам на основе их близости к текущим центроидам и**
* **Повторное вычисление центроидов на основе текущего назначения точек данных кластерам.**

Алгоритм K-средних останавливает уточнение кластера, когда

* **Центроиды кластера стабилизировались, когда больше нет изменений в точке данных кластера или**
* **Алгоритм выполнил указанное количество итераций.**

Алгоритм K-средних используется для

* **Обнаружение аномалий, таких как обнаружение мошенничества**
* **Сегментация, такая как сегментация клиентов, сегментация изображений**
* **Интеллектуальный анализ данных**

#### **Внедрение K-средних**

[Набор данных клиентов торгового центра](https://www.kaggle.com/shwetabh123/mall-customers)

Прочтите файл CSV во фрейме данных, преобразуйте категориальную переменную в числовой тип данных, а затем уменьшите размер с помощью PCA (анализ главных компонентов)

# Import the required libraries

**import numpy as np**

**import pandas as pd**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**from sklearn.cluster import KMeans**

**from sklearn.preprocessing import LabelBinarizer**

**from sklearn.decomposition import PCA**

# Read he dataset into a dataframe

**dataset = pd.read\_csv('Mall\_Customers.csv',index\_col='CustomerID')**

# Drop duplicates

**dataset.drop\_duplicates(inplace=True)**

# Converting categorical column Genre to onehot vector

**label\_binarizer = LabelBinarizer()**

#use LabelBinarizer for gender

**label\_binarizer\_output = label\_binarizer.fit\_transform( dataset['Genre'])**

#Adding the categorical and the main dataframe into one dataframe

**result\_df = pd.DataFrame(label\_binarizer\_output, columns=['Gender\_1'])**

**dataset\_1= pd.concat([dataset, result\_df], axis=1, join='inner')**

# Creating the input variable

**X= dataset\_1.iloc[:, [1,2,3,4]].values**

# reducing the input dimension using PCA

**reduced\_data = PCA(n\_components=2).fit\_transform(X)**

Когда мы запускаем KMeans, нам необходимо предоставить **n\_clusters, которые представляют собой оптимальное количество кластеров**, которые необходимо сформировать, и количество центроидов, которые необходимо сгенерировать для набора данных.

**kmeans = KMeans(init="k-means++", n\_clusters=5, n\_init=4)**

**kmeans.fit(X)**

**Как лучше всего найти оптимальные кластеры?**

Существует 3 популярных метода поиска оптимального кластера для алгоритма кластеризации KMeans.

#### **Метод локтя**

В методе локтя алгоритм K-средних запускается для разного количества кластеров, чтобы найти сумму квадратов расстояний каждой точки данных от центроида кластера, также называемую **суммой квадратов внутри кластера**.

Постройте WCSS вместе с количеством кластеров K. **Выберите значение K на графике, где WCSS начинает сглаживаться или внезапно падает, а добавление еще одного кластера не приводит к значительному улучшению, образуя изгиб**. Изгиб считается индикатором оптимального кластера для набора данных.

# Using the elbow method to find the optimal number of clusters

**max\_k=11** # max no. of clusters to be evaluated

**wcss = []**

**for i in range(1, max\_k):**

**kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)**

**kmeans.fit(reduced\_data)**

# inertia method returns wcss for that model

**wcss.append(kmeans.inertia\_)**

#plotting the data

**plt.figure(figsize=(5,3))**

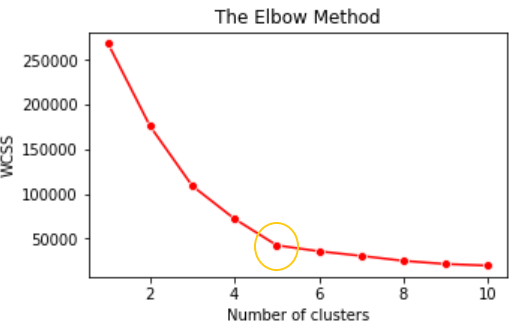
**sns.lineplot(range(1, max\_k), wcss,marker='o',color='red')**

**plt.title('The Elbow Method')**

**plt.xlabel('Number of clusters')**

**plt.ylabel('WCSS')**

**plt.show()**



#### **Оценка силуэта**

**Оценка силуэта - это еще один показатель, позволяющий проверить компактность кластера, чтобы определить, является ли кластер хорошим**.

Оценка силуэта **вычисляет среднее расстояние внутри кластера**, как метод локтя, а также **среднее значение ближайшего кластера**. Это показывает, насколько точка данных похожа на ее собственный кластер по сравнению со всеми другими центроидами кластера. Следовательно, **показатель силуэта измеряет как компактность одного и того же кластера, так и разделение с другими кластерами**.

**Если значение показателя силуэта для всех точек данных значительно велико, это дает оптимальное значение для кластеров**. Оценка силуэта использует расстояние Минковского или евклидово расстояние, и его значение находится в диапазоне [-1, 1].

Приведенный ниже код адаптирован из [https://scikit-learn.org/stable/auto\_examples/cluster/plot\_kmeans\_sil Silhouette\_analysis.html#](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html).

Код модифицирован для использования сокращенного измерения данных с помощью PCA.

**from sklearn.metrics import silhouette\_samples, silhouette\_score**

**import matplotlib.cm as cm**

**for n\_clusters in range(2, max\_k):**

# Create a subplot with 1 row and 2 columns

**fig, (ax1, ax2) = plt.subplots(1, 2)**

**fig.set\_size\_inches(18, 7)**

# The 1st subplot is the silhouette plot

# The silhouette coefficient can range from -1, 1 but in this example all

# lie within [-0.1, 1]

**ax1.set\_xlim([-0.1, 1])**

# The (n\_clusters+1)\*10 is for inserting blank space between silhouette

# plots of individual clusters, to demarcate them clearly.

**ax1.set\_ylim([0, len(reduced\_data) + (n\_clusters + 1) \* 10])**

# Initialize the clusterer with n\_clusters value and a random generator

# seed of 10 for reproducibility.

**clusterer = KMeans(n\_clusters=n\_clusters, random\_state=10)**

**cluster\_labels = clusterer.fit\_predict(reduced\_data)**

# The silhouette\_score gives the average value for all the samples.

# This gives a perspective into the density and separation of the formed

# clusters

**silhouette\_avg = silhouette\_score(reduced\_data, cluster\_labels)**

**print("For n\_clusters =", n\_clusters,"The average silhouette\_score is :", silhouette\_avg)**

# Compute the silhouette scores for each sample

**sample\_silhouette\_values = silhouette\_samples(reduced\_data, cluster\_labels)**

**y\_lower = 10**

**for i in range(n\_clusters):**

# Aggregate the silhouette scores for samples belonging to

# cluster i, and sort them

**ith\_cluster\_silhouette\_values = \**

**sample\_silhouette\_values[cluster\_labels == i]**

**ith\_cluster\_silhouette\_values.sort()**

**size\_cluster\_i = ith\_cluster\_silhouette\_values.shape[0]**

**y\_upper = y\_lower + size\_cluster\_i**

**color = cm.nipy\_spectral(float(i) / n\_clusters)**

**ax1.fill\_betweenx(np.arange(y\_lower, y\_upper),**

**0, ith\_cluster\_silhouette\_values,**

**facecolor=color, edgecolor=color, alpha=0.7)**

# Label the silhouette plots with their cluster numbers at the middle

**ax1.text(-0.05, y\_lower + 0.5 \* size\_cluster\_i, str(i))**

# Compute the new y\_lower for next plot

**y\_lower = y\_upper + 10** # 10 for the 0 samples

**ax1.set\_title("The silhouette plot for the various clusters.")**

**ax1.set\_xlabel("The silhouette coefficient values")**

**ax1.set\_ylabel("Cluster label")**

# The vertical line for average silhouette score of all the values

**ax1.axvline(x=silhouette\_avg, color="red", linestyle="--")**

**ax1.set\_yticks([]) # Clear the yaxis labels / ticks**

**ax1.set\_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])**

# 2nd Plot showing the actual clusters formed

**colors = cm.nipy\_spectral(cluster\_labels.astype(float) / n\_clusters)**

**ax2.scatter(reduced\_data[:, 0], reduced\_data[:, 1], marker='.', s=30, lw=0, alpha=0.7,**

**c=colors, edgecolor='k')**

# Labeling the clusters

**centers = clusterer.cluster\_centers\_**

# Draw white circles at cluster centers

**ax2.scatter(centers[:, 0], centers[:, 1], marker='o',**

**c="white", alpha=1, s=200, edgecolor='k')**

**for i, c in enumerate(centers):**

**ax2.scatter(c[0], c[1], marker='$%d$' % i, alpha=1,**

**s=50, edgecolor='k')**

**ax2.set\_title("The visualization of the clustered data.")**

**ax2.set\_xlabel("Feature space for the 1st feature")**

**ax2.set\_ylabel("Feature space for the 2nd feature")**

**plt.suptitle(("Silhouette analysis for KMeans clustering on sample data "**

**"with n\_clusters = %d" % n\_clusters),**

**fontsize=14, fontweight='bold')**

**plt.show()**

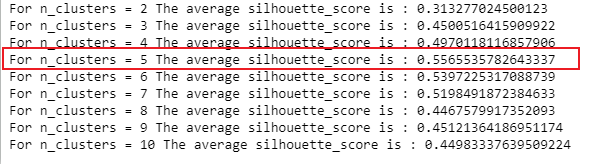
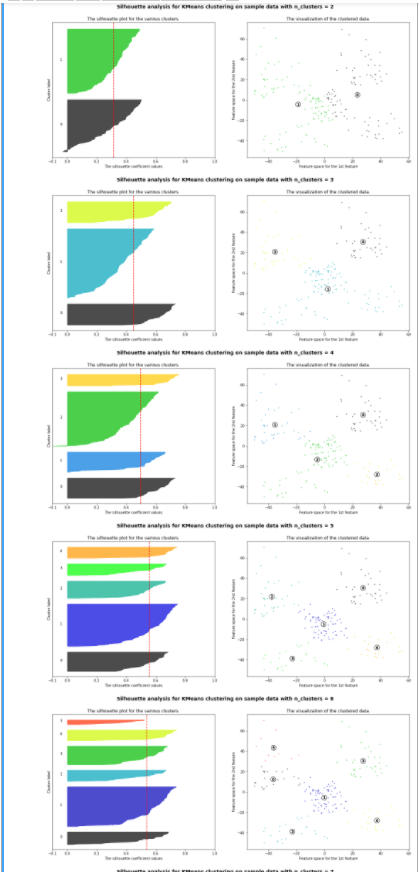


График силуэта предполагает, что 5 является хорошим выбором для оптимального кластера для заданных данных. Оптимальные кластеры оцениваются по наличию кластеров с оценкой силуэта выше среднего и соответствующими колебаниями в размере участков силуэта.



**И метод локтя, и оценка по силуэту дают оптимальные кластеры 5**

Запуск KMeans с оптимальным кластером, равным 5, а затем построение диаграммы рассеяния с использованием оптимальных кластеров по уменьшенным\_data для визуализации кластера

# Fitting K-Means to the dataset

**kmeans = KMeans(n\_clusters = 5, init = 'k-means++', random\_state = 42)**

**y\_kmeans = kmeans.fit\_predict(reduced\_data)**

#Visualising the clusters

**plt.figure(figsize=(15,7))**

**sns.scatterplot(reduced\_data[y\_kmeans == 0, 0], reduced\_data[y\_kmeans == 0, 1], color = 'yellow', label = 'Cluster 1',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 1, 0], reduced\_data[y\_kmeans == 1, 1], color = 'blue', label = 'Cluster 2',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 2, 0], reduced\_data[y\_kmeans == 2, 1], color = 'green', label = 'Cluster 3',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 3, 0], reduced\_data[y\_kmeans == 3, 1], color = 'grey', label = 'Cluster 4',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 4, 0], reduced\_data[y\_kmeans == 4, 1], color = 'orange', label = 'Cluster 5',s=50)**

**sns.scatterplot(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], color = 'red',**

**label = 'Centroids',s=300,marker=',')**

**plt.grid(False)**

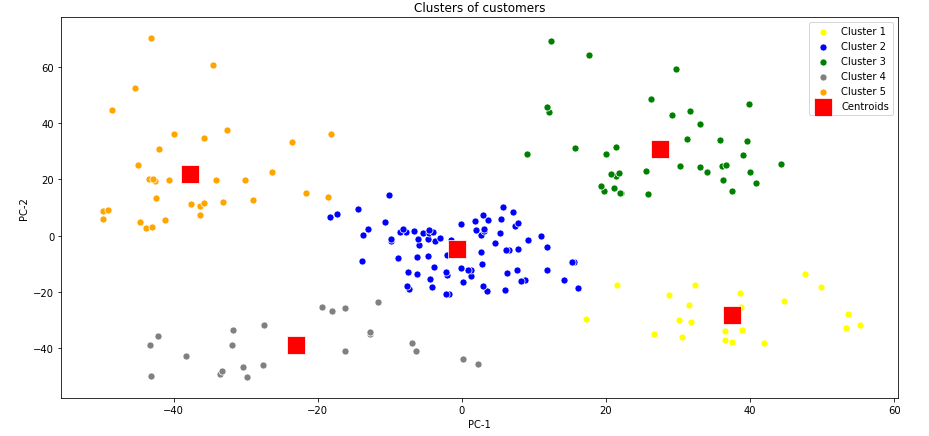
**plt.title('Clusters of customers')**

**plt.xlabel('PC-1')**

**plt.ylabel('PC-2')**

**plt.legend()**

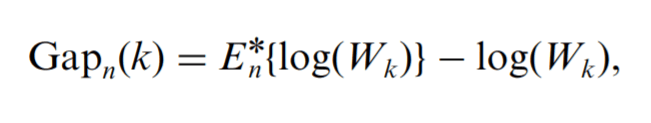
**plt.show()**



#### **Статистика разрывов**

Статистика разрыва - еще один популярный метод определения оптимального количества кластеров, который можно применить к любому методу кластеризации.

**Идея статистики разрыва состоит в том, чтобы сравнить инерцию кластера в отношении данных с ее ожиданием в соответствующем наборе данных с нулевой ссылкой.** Наборы данных с нулевой ссылкой могут быть взяты из нормального или равномерного распределения. **Оптимальный выбор K - это значение, которое максимизирует статистику разрыва между внутрикластерной инерцией набора данных и набором данных с нулевой ссылкой**.



**Статистика разрыва - это разница между E ₙ, который представляет собой ожидание данных с нулевой ссылкой, вычисленное как логарифм общей внутрикластерной вариации и итоговой внутрикластерной вариации для набора данных**

Приведенный ниже код вдохновлен и адаптирован из <https://glowingpython.blogspot.com/2019/01/a-visual-introduction-to-gap-statistics.html>

Набор данных с нулевой ссылкой - это равномерно распределенный набор точек.

**from sklearn.datasets import make\_blobs**

**from sklearn.metrics import pairwise\_distances**

# creating a uniform distributed null refernced dataset

**reference = np.random.rand(100, 2)**

**plt.figure(figsize=(12, 3))**

**for k in range(1,6):**

**kmeans = KMeans(n\_clusters=k)**

**a = kmeans.fit\_predict(reference)**

**plt.subplot(1,5,k)**

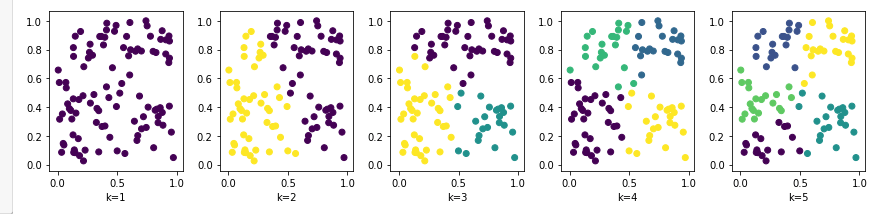
**plt.scatter(reference[:, 0], reference[:, 1], c=a)**

**plt.xlabel('k='+str(k))**

**plt.tight\_layout()**

**plt.show()**

Поскольку точки распределены равномерно, алгоритм KMeans равномерно разбивает точки на K кластеров, даже если между ними нет разделения.



**def compute\_inertia(a, X):**

**W = [np.mean(pairwise\_distances(X[a == c, :])) for c in np.unique(a)]**

**return np.mean(W)**

**def compute\_gap(clustering, data, k\_max=5, n\_references=5):**

**if len(data.shape) == 1:**

**data = data.reshape(-1, 1)**

# The Null Reference dataset that is uniformly distributed

**reference = np.random.rand(\*data.shape)**

**reference\_inertia = []**

# Calculate the WCSS for the null refrenced data

**for k in range(1, k\_max+1):**

**local\_inertia = []**

**for \_ in range(n\_references):**

**clustering.n\_clusters = k**

**assignments = clustering.fit\_predict(reference)**

**local\_inertia.append(compute\_inertia(assignments, reference))**

**reference\_inertia.append(np.mean(local\_inertia))**

# Calculate the WCSS for the data

**ondata\_inertia = []**

**for k in range(1, k\_max+1):**

**clustering.n\_clusters = k**

**assignments = clustering.fit\_predict(data)**

**ondata\_inertia.append(compute\_inertia(assignments, data))**

# Calculate the gao statistics between the WCSS for the null referenced data and the WCSS for the data

**gap = np.log(reference\_inertia)-np.log(ondata\_inertia)**

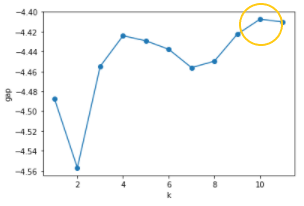
**return gap, np.log(reference\_inertia), np.log(ondata\_inertia)**

**gap, reference\_inertia, ondata\_inertia = compute\_gap(KMeans(), reduced\_data, k\_max)**

**plt.plot(range(1, k\_max+1), gap, '-o')**

**plt.ylabel('gap')**

**plt.xlabel('k')**



Статистика разрывов дает оптимальное количество кластеров, равное 10, на основе максимального разрыва между **инерцией кластера данных** и данными с нулевой ссылкой.

Запуск KMeans для 10 кластеров, а затем визуализация кластеров на диаграмме рассеяния.

# Visualising the clusters

**plt.figure(figsize=(15,7))**

**sns.scatterplot(reduced\_data[y\_kmeans == 0, 0], reduced\_data[y\_kmeans == 0, 1], color = 'yellow', label = 'Cluster 1',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 1, 0], reduced\_data[y\_kmeans == 1, 1], color = 'blue', label = 'Cluster 2',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 2, 0], reduced\_data[y\_kmeans == 2, 1], color = 'green', label = 'Cluster 3',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 3, 0], reduced\_data[y\_kmeans == 3, 1], color = 'grey', label = 'Cluster 4',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 4, 0], reduced\_data[y\_kmeans == 4, 1], color = 'orange', label = 'Cluster 5',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 5, 0], reduced\_data[y\_kmeans == 5, 1], color = 'pink', label = 'Cluster 6',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 6, 0], reduced\_data[y\_kmeans == 6, 1], color = 'magenta', label = 'Cluster 7',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 7, 0], reduced\_data[y\_kmeans == 7, 1], color = 'cyan', label = 'Cluster 8',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 8, 0], reduced\_data[y\_kmeans == 8, 1], color = 'purple', label = 'Cluster 9',s=50)**

**sns.scatterplot(reduced\_data[y\_kmeans == 9, 0], reduced\_data[y\_kmeans == 9, 1], color = 'brown', label = 'Cluster 10',s=50)**

**sns.scatterplot(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], color = 'red',**

**label = 'Centroids',s=300,marker=',')**

**plt.grid(False)**

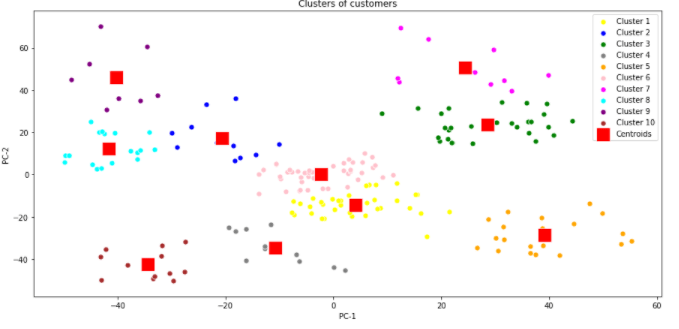
**plt.title('Clusters of customers')**

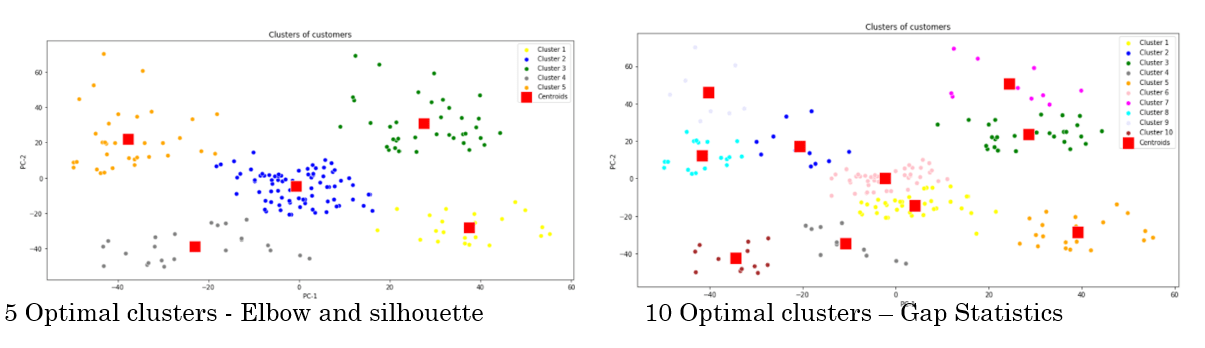
**plt.xlabel('PC-1')**

**plt.ylabel('PC-2')**

**plt.legend()**

**plt.show()**





#### Вывод:

Кластеризация K-средних - это

* **Алгоритм машинного обучения без учителя**
* **Итерационный алгоритм для поиска групп данных со схожими характеристиками для немаркированного набора данных**

Популярные методы определения оптимальных кластеров

* **Метод изгиба**: когда WCSS внезапно начинает сглаживаться, и добавление еще одного кластера не приводит к значительному улучшению, образуя изгиб.
* **Оценка силуэта**: измеряет как компактность с одним и тем же кластером, так и разделение с другими кластерами.
* **Статистика разрыва.** Максимальный разрыв между внутрикластерной инерцией набора данных и набором данных, на который имеется ссылка на null, распределяется равномерно.

#### Использованная литература:

[Https://stanford.edu/~cpiech/cs221/handouts/kmeans.html](https://stanford.edu/~cpiech/cs221/handouts/kmeans.html)

[**Кластеризация K-средних: Справочник по модулю - Машинное обучение Azure**   
В этой статье описывается, как использовать модуль кластеризации K-средних в конструкторе машинного обучения Azure для создания… docs.microsoft.com](https://docs.microsoft.com/en-us/azure/machine-learning/algorithm-module-reference/k-means-clustering)

[Https://arxiv.org/ftp/arxiv/papers/1912/1912.00643.pdf](https://arxiv.org/ftp/arxiv/papers/1912/1912.00643.pdf)

[**Выбор количества кластеров с помощью анализа силуэта в кластеризации KMeans - scikit-learn…**   
Анализ силуэта можно использовать для изучения расстояния разделения между результирующими кластерами. Сюжет силуэта… scikit-learn.org](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

[Оценка количества кластеров в наборе данных с помощью статистики разрывов](https://statweb.stanford.edu/~gwalther/gap) [Р. Тибширани, Г. Вальтер и Т. Хасти (Стэндфордский университет, 2001 г.)](http://web.stanford.edu/~hastie/Papers/gap.pdf) .

[Https://glowingpython.blogspot.com/2019/01/a-visual-introduction-to-gap-statistics.html](https://glowingpython.blogspot.com/2019/01/a-visual-introduction-to-gap-statistics.html)

[**Выбор количества кластеров с помощью анализа силуэта в кластеризации KMeans - scikit-learn…**   
Анализ силуэта можно использовать для изучения расстояния разделения между результирующими кластерами. Сюжет силуэта… scikit-learn.org](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)

[Оценка количества кластеров в наборе данных с помощью статистики разрывов](https://statweb.stanford.edu/~gwalther/gap) [Р. Тибширани, Г. Вальтер и Т. Хасти (Стэндфордский университет, 2001 г.)](http://web.stanford.edu/~hastie/Papers/gap.pdf) .

[Https://glowingpython.blogspot.com/2019/01/a-visual-introduction-to-gap-statistics.html](https://glowingpython.blogspot.com/2019/01/a-visual-introduction-to-gap-statistics.html)