Алгоритмы кластеризации

Обучение без учителя (unsupervised learning) – класс методов машинного обучения для поиска шаблонов в наборе данных. Данные, получаемые на вход таких алгоритмов обычно не размечены, то есть передаются только входные переменные X без соответствующих меток y. Если в обучении с учителем (supervised learning) система пытается извлечь уроки из предыдущих примеров, то в обучении без учителя – система старается самостоятельно найти шаблоны непосредственно из приведенного примера.

Одним из наиболее популярных семейств методов машинного обучения без учителя являются **методы кластеризации данных**.

Понятие кластеризации

Кластеризация (или кластерный анализ) — это задача разбиения множества объектов на группы, называемые кластерами. Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны. Модель самостоятельно определяет схожесть некоторых предметов и объединяет их в один сектор.

Кластеризация позволяет разбить объекты на группы, не имея заранее заданной обучающей выборки или знания о природе этих классов. Из-за сходства постановки задачи научным названием кластеризации является unsupervised classification.

Главное отличие кластеризации от классификации состоит в том, что перечень групп четко не задан и определяется в процессе работы алгоритма.

Кластеризацией также называется сегментация данных в некоторых приложениях, поскольку кластеризация разделяет большие наборы данных на группы в соответствии с их сходством.

Кластерный анализ в общем виде сводится к следующим этапам:

- Отбор выборки объектов для кластеризации.
- Определение множества переменных, по которым будут оцениваться объекты в выборке. При необходимости – нормализация значений переменных, например, чтобы они варьировались от 0 до 1.
- Вычисление значений меры сходства между объектами.
- Применение выбранного метода кластерного анализа для создания групп сходных объектов (кластеров).

Метрики расстояний между объектами

Итак, как же определять «похожесть» объектов? Для начала нужно составить вектор характеристик для каждого объекта — как правило, это набор числовых значений, например, рост-вес человека.

После того, как определили вектор характеристик, можно провести нормализацию, чтобы все компоненты давали одинаковый вклад при расчете «расстояния». В процессе нормализации все значения приводятся к некоторому диапазону, например, [-1, -1] или [0, 1].

Наконец, **для каждой пары объектов измеряется «расстояние» между ними — степень похожести.**

Существует множество метрик расстояний, вот лишь основные из них:

1. **Евклидово расстояние**

Наиболее распространенная функция расстояния. Представляет собой геометрическое расстояние в многомерном пространстве, и вычисляется по теореме Пифагора.

$$d(x,y) = \sqrt{\sum_{i}^{n}(x_i - y_i)^2}$$

2. **Квадрат евклидова расстояния**

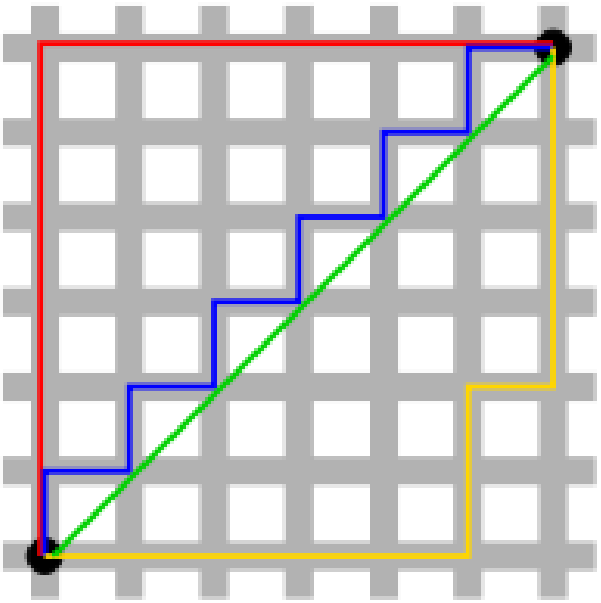
Применяется для придания большего веса более отдаленным друг от друга объектам. Это расстояние вычисляется следующим образом:

$$d(x,y) = \sum_{i}^{n}(x_i - y_i)^2$$

3. **Расстояние городских кварталов (манхэттенское расстояние)**

Это расстояние является средним разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако для этой меры влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат). Формула для расчета манхэттенского расстояния:

$$d(x,y) = \sum_{i}^{n}|x_i - y_i|$$



4. **Расстояние Чебышева**

Это расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате. Расстояние Чебышева вычисляется по формуле:

$$d(x,y) = \max(|x_i - y_i|)$$

Выбор метрики полностью лежит на исследователе, поскольку результаты кластеризации могут существенно отличаться при использовании разных мер.

Использование различных метрик расстояний в Minecraft

Алгоритмы квадратичной ошибки

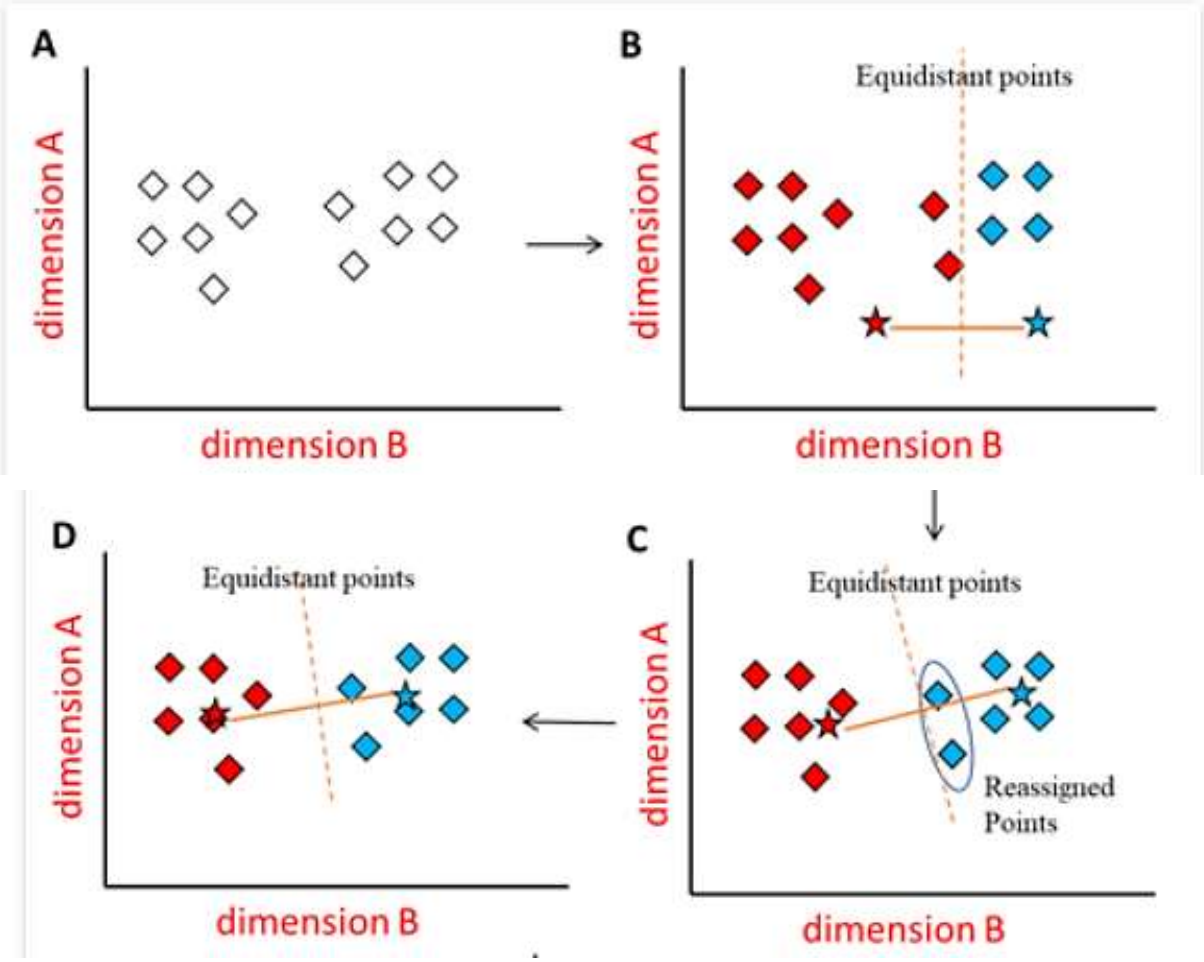
Задачу кластеризации можно рассматривать как построение оптимального разбиения объектов на группы. При этом оптимальность может быть определена как требование минимизации среднеквадратической ошибки разбиения:

$$e^2(X,L) = \sum_{j=1}^{K} \sum_{i=1}^{n_j} \| x_i^{(j)} - c_j \|^2$$

где c_j — «центр масс» кластера j (точка со средними значениями характеристик для данного кластера).

Самым распространенным алгоритмом этой категории является **метод k- means**.

Алгоритм k-means (k-средних) строит заданное число кластеров, расположенных как можно дальше друг от друга.



https://www.blopig.com/blog/2020/07/k-means-clustering-made-simple/

Рассмотрим пример с K = 2, алгоритм:

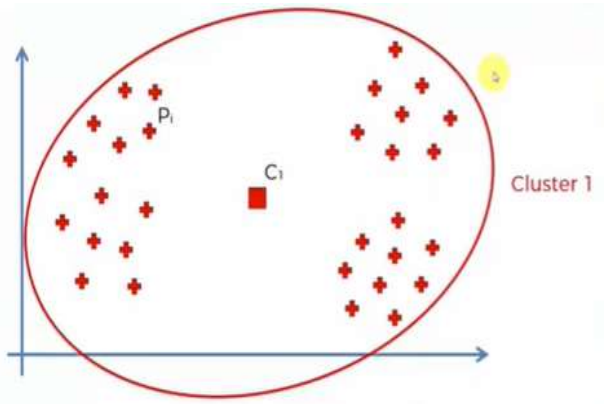
1. Случайным образом создает 2 центроида (звездочки на рис. B).
2. Создаются кластеры (в нашем случае 2 кластера), назначая каждую точку данных ближайшему центроиду (красный и синий квадраты в B).

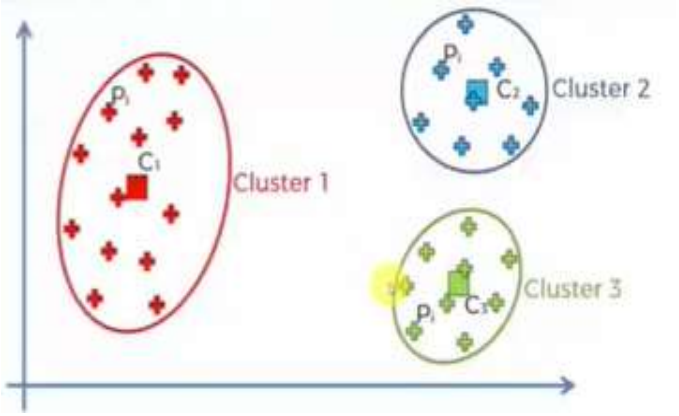
3. Вычисляет новые центроиды для каждого из 2ух кластеров (C).
4. Переназначает точки данных, используя новый набор центроидов (C):
   - если есть новые задания, возвращается к шагу 3;
   - если новых присваиваний не происходит, алгоритм завершается (D).

В качестве критерия остановки работы алгоритма обычно выбирают минимальное изменение среднеквадратической ошибки. Так же возможно останавливать работу алгоритма, если на шаге 2 не было объектов, переместившихся из кластера в кластер.

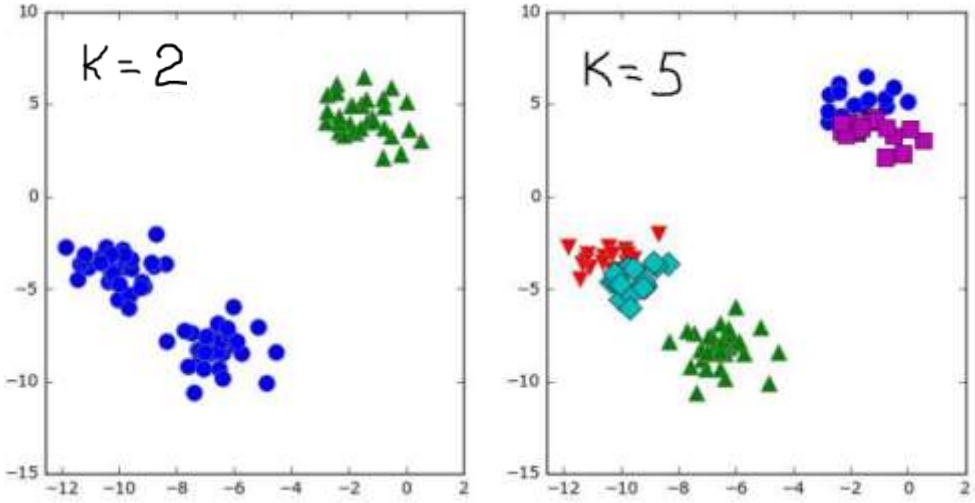
НО есть проблема: как определить количество кластеров?

Как оценить на каком рисунке результат кластеризации лучше?





или



Количество кластеров, которые мы выбираем для конкретного набора данных, можно рассчитать. Каждый кластер формируется путем вычисления и сравнения расстояний между точками данных в кластере и его центроидом. **Идеальным способом определения правильного количества кластеров было бы вычисление внутрикластерной суммы квадратов (WCSS).**

WCSS - это сумма квадратов расстояний от каждой точки данных во всех кластерах до их соответствующих центроидов.

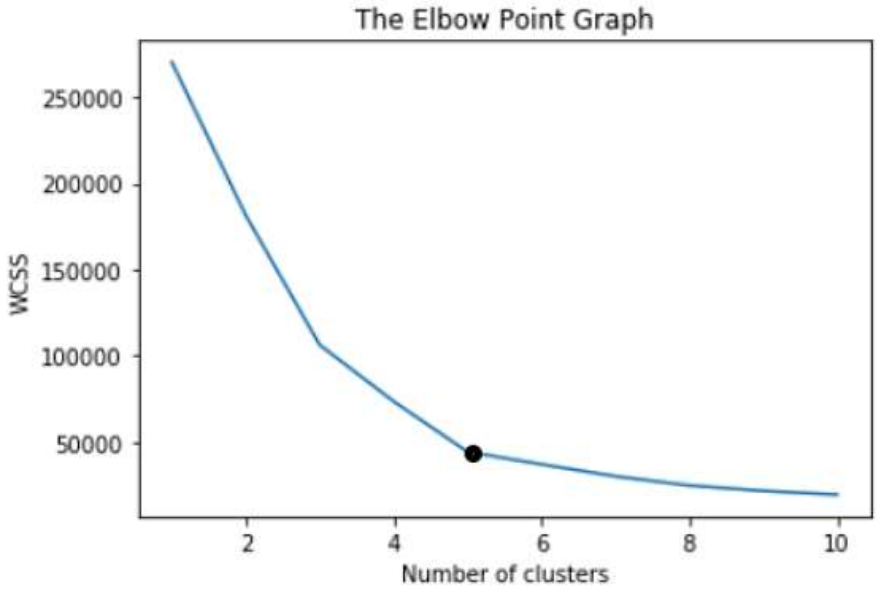
$$\text{WCSS} = \sum_{C_k}^{C_n} \Big(\sum_{d_i in C_i}^{d_m} distance(d_i, C_k)^2 \Big)$$

Where,
C is the cluster centroids and d is the data point in each Cluster.

Из формулы видно, что чем больше расстояния от точек кластера до его центроиды, тем больше будет значение WCSS. Поэтому идея состоит в том, чтобы минимизировать сумму.

Предположим, что в данном наборе данных есть n наблюдений, и мы указываем n количество кластеров (k = n), тогда WCSS станет нулевым, поскольку сами точки данных будут действовать как центроиды, а расстояние будет равно нулю, и в идеале это образует идеальный кластер, однако это не имеет никакого смысла, поскольку у нас столько кластеров, сколько наблюдений.

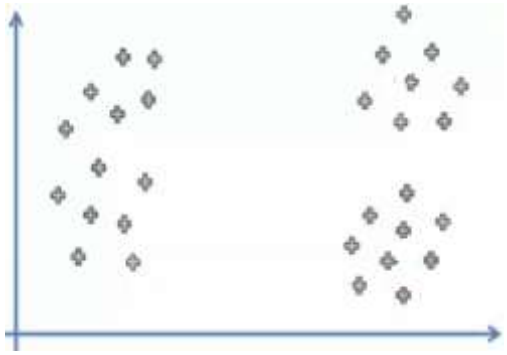
Можем найти **оптимальное значение для K**, используя график точек локтя. Инициализируем алгоритм K-средних для диапазона значений K и наносим его на график относительно WCSS для каждого значения K.



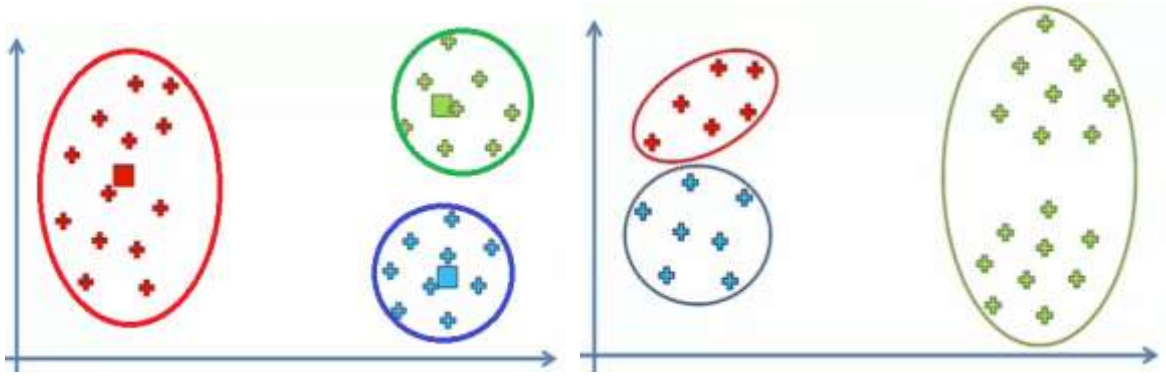
https://analyticsindiamag.com/beginners-guide-to-k-means-clustering/

Из графика видно, что с увеличением количества кластеров значение WCSS уменьшается. Выбираем значение K на основе скорости уменьшения WCSS. Например, в кластере 1–2–3 на приведенном выше графике видно быстрое падение WCSS. После значения 5 падение минимально, поэтому можно выбрать 5 как оптимальное значение для K.

Одним из основных недостатков кластеризации K-средних является случайная инициализация центроидов. Формирование кластеров тесно связано с начальным положением центроида. Случайное расположение центроидов может полностью изменить кластеры и привести к случайному образованию.



Центроиды для представленных на рисунке данных могут быть выбраны по-разному, и тогда меняется результат кластеризации

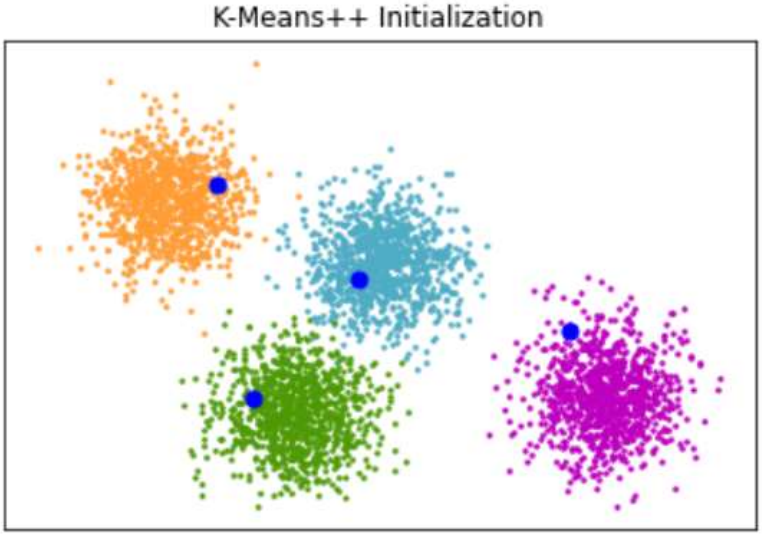


Решение этой проблемы – это применение алгоритма K-means ++.

K-Means ++ - это алгоритм, который используется для инициализации начальных центроидов для алгоритма K-средних. Он выбирает начальные центры кластеров для кластеризации с k-средним разумным способом для ускорения схождения.

В реализации метода K-Means в библиотеке skilearn метод **K-Means++** используется по умолчанию:

```
KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
```



https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_plusplus.html#sphx-glr-auto-examples-cluster-plot-kmeans-plusplus-py

К недостаткам алгоритма K-Means можно отнести:

- ✓ необходимость задавать количество кластеров для разбиения,

  следовательно, необходим предварительный анализ на определение количества кластеров;

- ✓ алгоритм очень чувствителен к выбору начальных центров кластеров.

  Классический вариант подразумевает случайный выбор центров кластеров, что очень часто являлось источником погрешности. Как вариант решения, необходимо проводить исследования объектов для более точного определения центров начальных кластеров, например – в качестве центров самые отдаленные точки кластеров.

Ограничения:

- ✓ кластеризация K- means хорошо работает в тех случаях, когда данные хорошо разделены и имеют сферическую / круглую форму;
- ✓ не справляется с задачей, когда объект принадлежит к разным кластерам в равной степени или не принадлежит ни одному.

На рисунках виден неудовлетворительный результат работы кластеризации K-means на сложных кластерах.

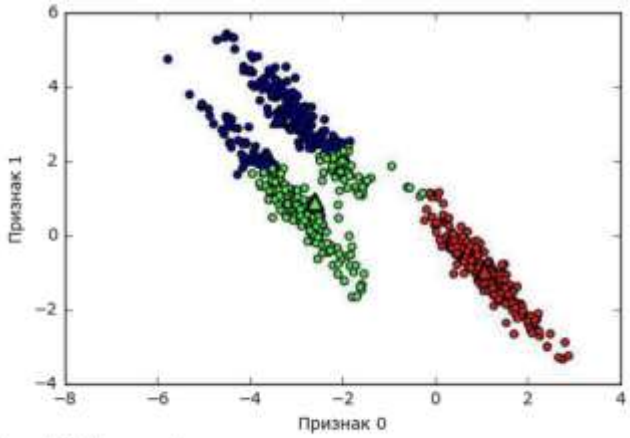


Рис. 3.28 Алгоритм *k*-средних не позволяет выявить несферические кластеры

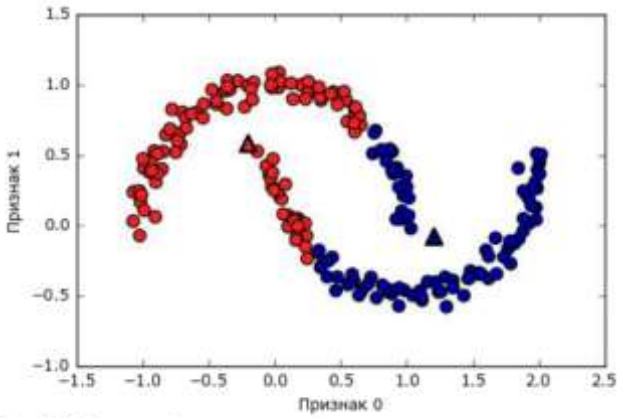


Рис. 3.29 Алгоритм *k*-средних не позволяет выявить кластеры более сложной формы

Реализация при помощи библиотеки **Scikit-Learn**

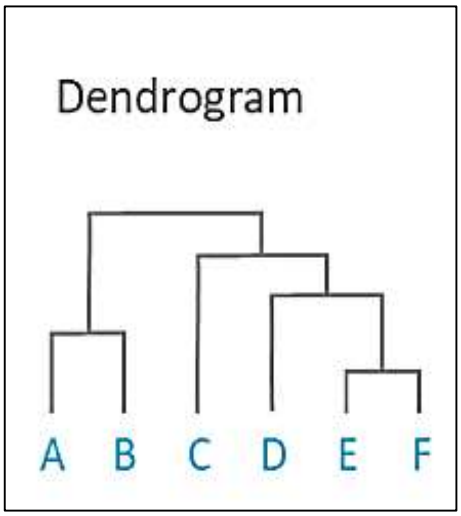
```
from sklearn.cluster import KMeans
# обучение K-means на данных
kmeans = KMeans(n_clusters = 5, init = 'k-means++', random_state = 42)
kmean.fit(X)
```

Алгоритмы иерархической кластеризации

Среди алгоритмов иерархической кластеризации выделяются два основных типа: восходящие и нисходящие алгоритмы.

Нисходящие алгоритмы работают по принципу «сверху-вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры.

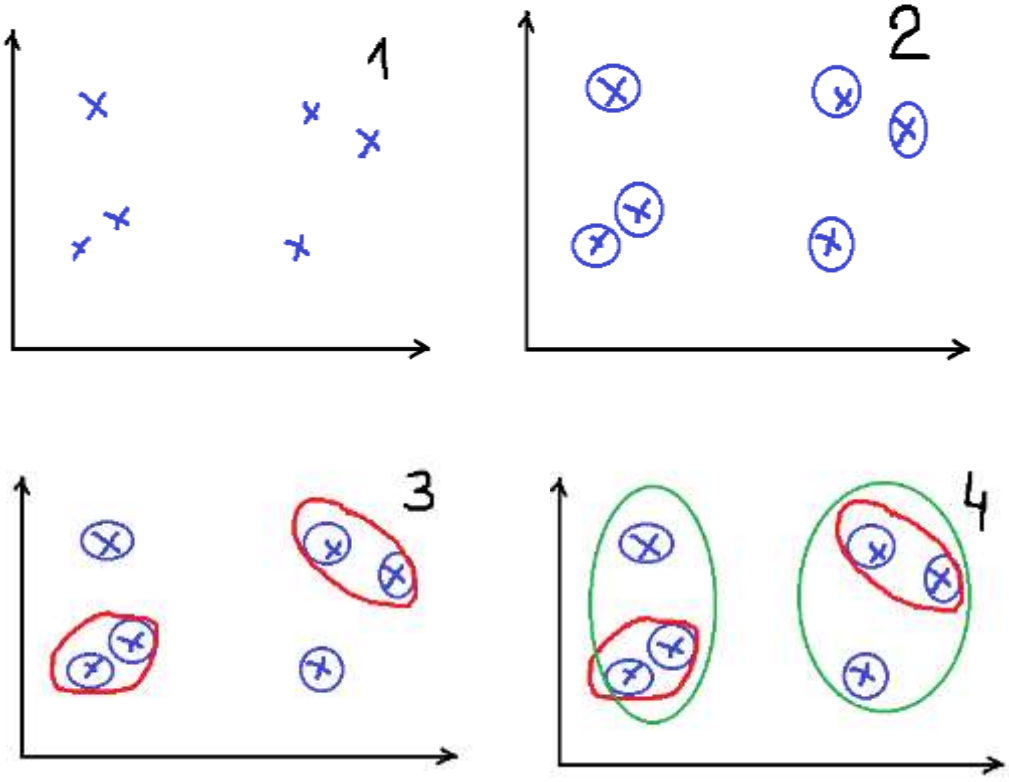
Более распространены ***восходящие*** алгоритмы, которые в начале работы помещают каждый объект в отдельный кластер, а затем объединяют кластеры во все более крупные, пока все объекты выборки не будут содержаться в одном кластере.

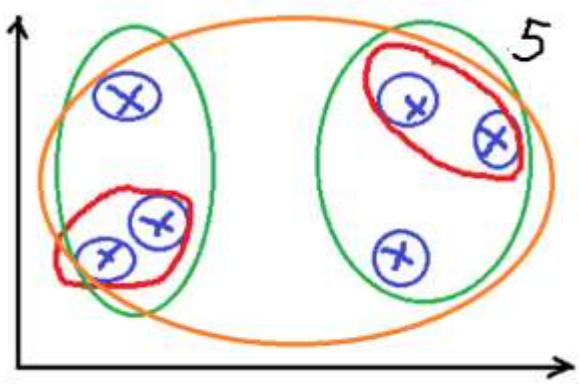


Таким образом строится система вложенных разбиений. Результаты таких алгоритмов обычно представляют в виде дерева – **дендрограммы,** в листьях которого находятся отдельно взятые элементы, а корень дерева — это совокупность всех элементов.

Иерархическая кластеризация

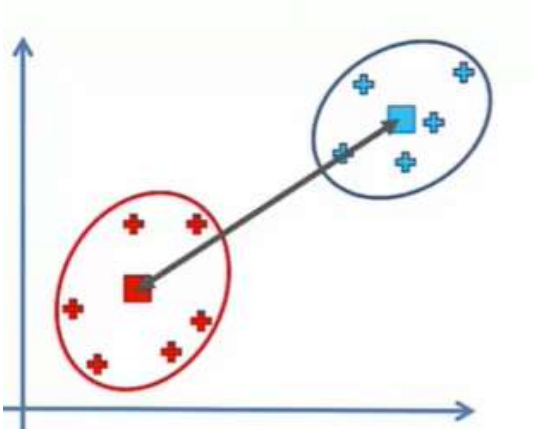
Иерархическая кластеризация представляет собой алгоритм, который строит иерархию кластеров. Этот алгоритм начинает работу с того, что каждому экземпляру данных сопоставляется свой собственный кластер, затем два ближайших кластера объединяются в один и так далее, пока не будет образован один общий кластер.





Кластеры объединяются с ближайшими, пока не будет образован один общий кластер.

В случае использования иерархических алгоритмов встает вопрос, как объединять между собой кластеры, как вычислять «расстояния» между ними.



Есть несколько критериев связи (linkage), которые задают точный способ измерения «наиболее схожего кластера». В основе этих критериев лежит расстояние между двумя существующими кластерами.

Существует несколько метрик расстояний между кластерами:

1. **Одиночная связь (single linkage)**

В этом методе расстояние между двумя кластерами определяется расстоянием между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах. Результирующие кластеры имеют тенденцию объединяться в цепочки.

$$d(u,v) = \min(dist(u[i], v[j]))$$

2. **Полная связь (complete linkage)**

В этом методе расстояния между кластерами определяются наибольшим расстоянием между любыми двумя объектами в различных кластерах (т.е. наиболее удаленными соседями). Если кластеры имеют удлиненную форму или их естественный тип является «цепочечным», то этот метод непригоден.

$$d(u,v) = \max(dist(u[i], v[j]))$$

3. **Невзвешенное попарное среднее**

В этом методе расстояние между двумя различными кластерами вычисляется как среднее расстояние между всеми парами объектов в них. Метод эффективен, когда объекты формируют различные группы, однако он работает одинаково хорошо и в случаях протяженных («цепочечного» типа) кластеров.

4. **Взвешенное попарное среднее**

Метод идентичен методу невзвешенного попарного среднего, за исключением того, что при вычислениях размер соответствующих кластеров (т.е. число объектов, содержащихся в них) используется в качестве весового коэффициента. Поэтому данный метод должен быть использован, когда предполагаются неравные размеры кластеров.

5. **Невзвешенный центроидный метод**

В этом методе расстояние между двумя кластерами определяется как расстояние между их центрами тяжести.

6. **Взвешенный центроидный метод (медиана)**

Этот метод идентичен предыдущему, за исключением того, что при вычислениях используются веса для учета разницы между размерами кластеров. Поэтому, если имеются или подозреваются значительные отличия в размерах кластеров, этот метод оказывается предпочтительнее предыдущего.

В scikit-learn реализованы следующие критерии:

- **Ward** (Метод Варда) метод по умолчанию. Выбирает и объединяет два кластера так, чтобы прирост дисперсии внутри кластеров был минимальным. Часто этот критерий приводит к получению кластеров относительно одинакового размера.
- **Average** (метод средней связи) объединяет два кластера, которые имеют наименьшее среднее значение всех расстояний, измеренных между точками двух кластеров.
- **Single** использует минимум расстояний между всеми наблюдениями двух наборов.
- **Complete** (метод полной связи или метод максимальной связи) объединяет два кластера, которые имеют наименьшее расстояние между двумя их самыми удаленными точками.

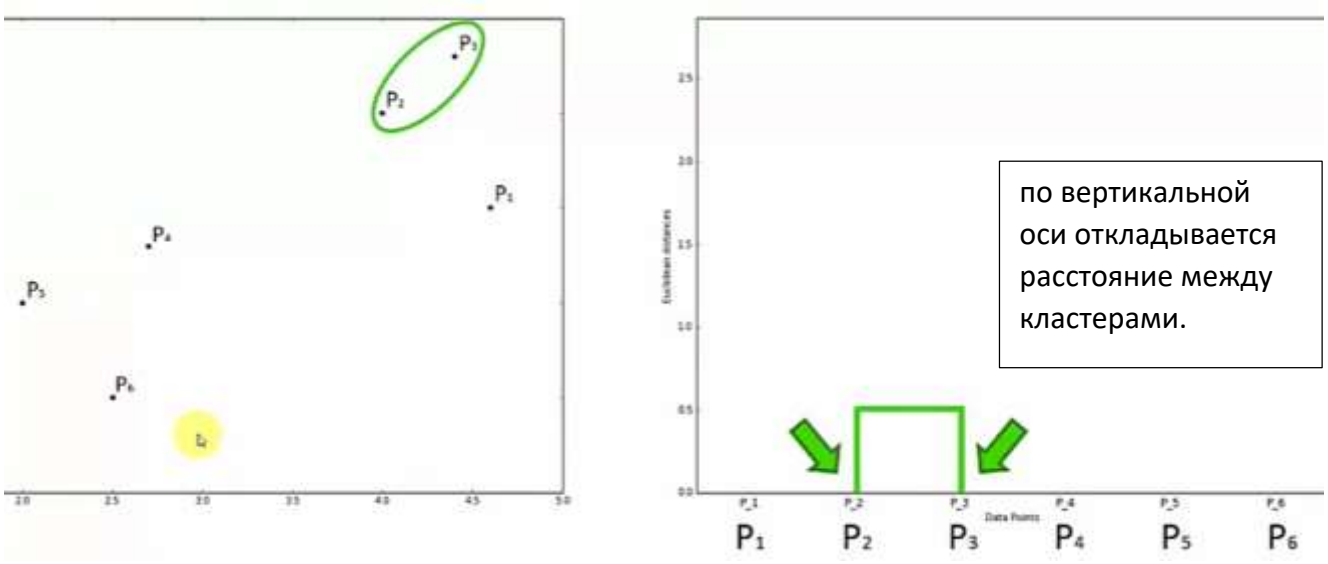
Ward подходит для большинства наборов данных.

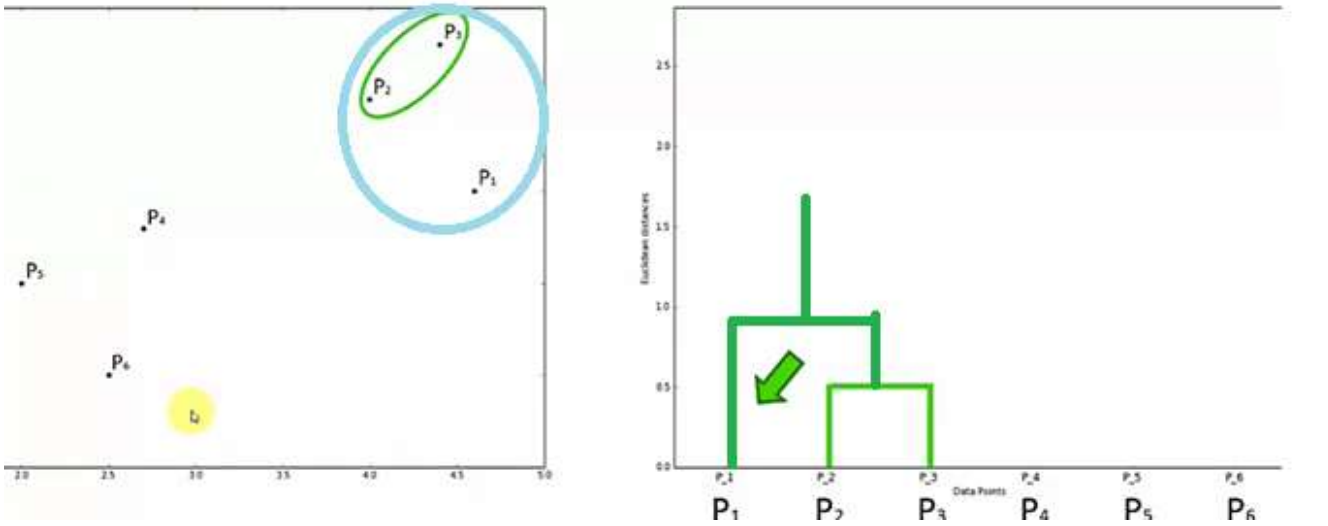
Если кластеры имеют сильно различающиеся размеры (например, один кластер содержит намного больше точек данных, чем все остальные), использование критериев ***average*** или ***complete*** может дать лучший результат.

Если в иерархической кластеризации кластеры объединяются пока не будет образован один общий кластер, то по какому принципу нужно прервать объединение, чтобы получить разумное количество кластеров?

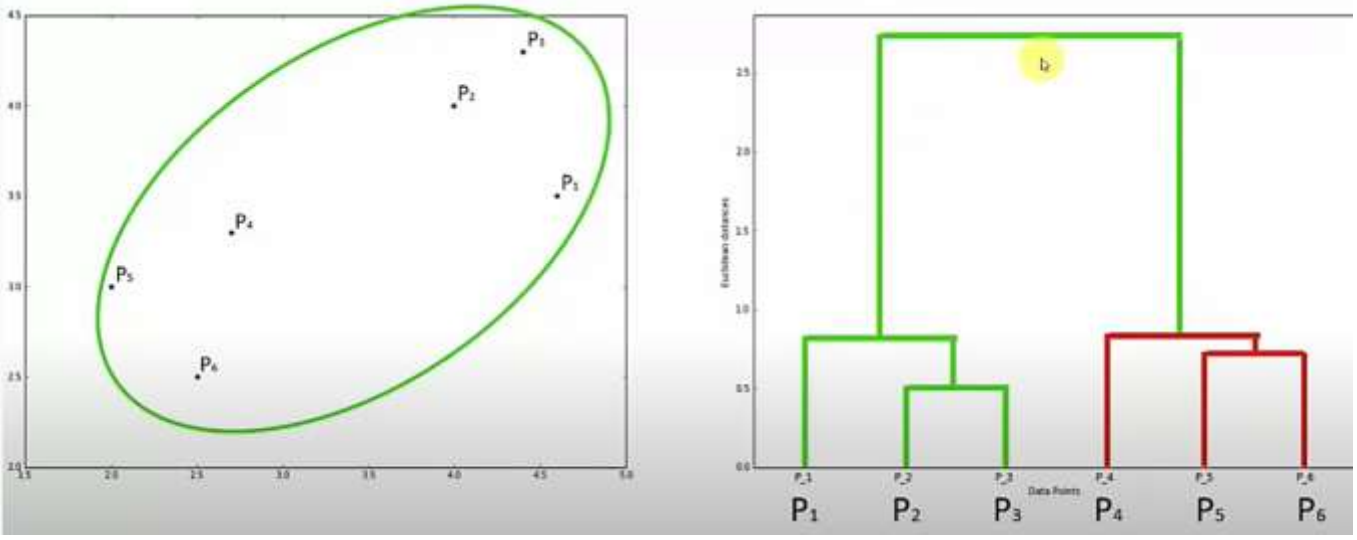
На это вопрос можно ответить при помощи **дендрограммы**, которая визуализирует результат иерархической кластеризации:

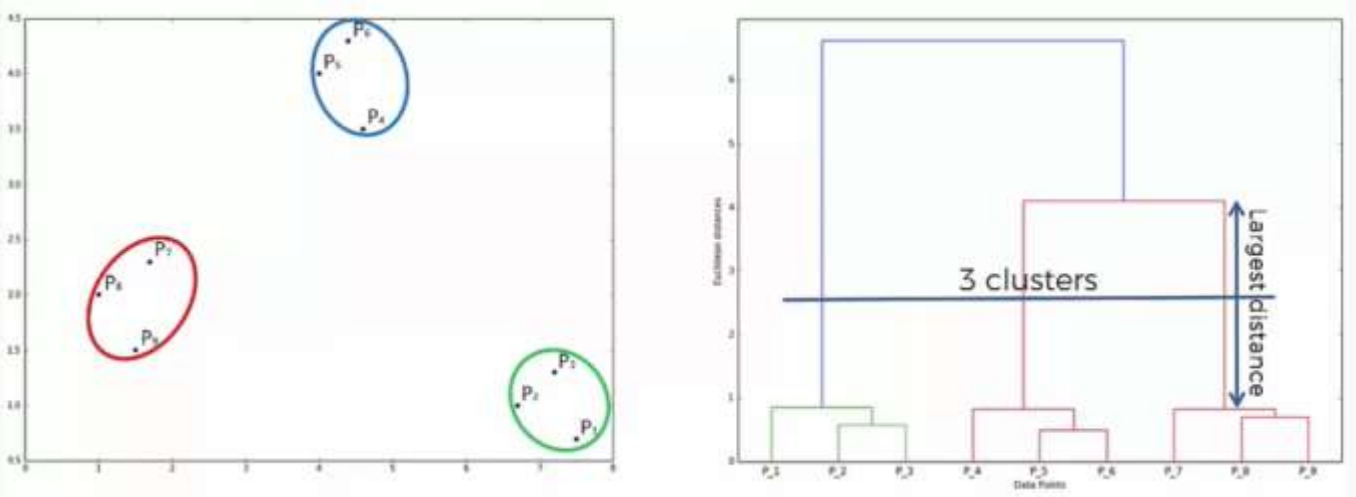
https://www.youtube.com/watch?v=sPVRyeEXQyY

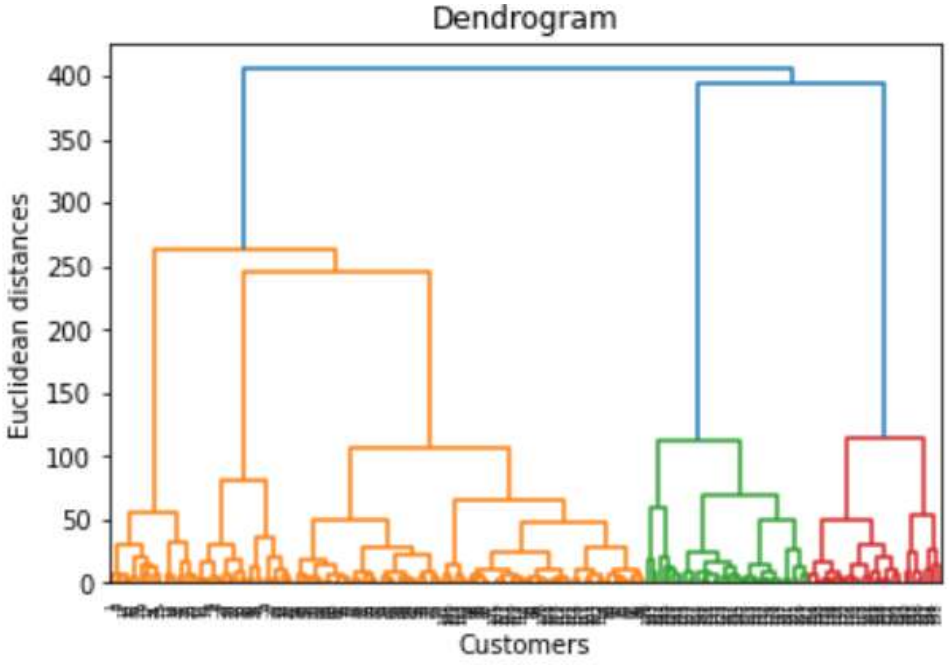




Таким образом, можно определить нужное количество кластеров, выделяя в разные группы те поддеревья, расстояния между которыми достаточно велики.

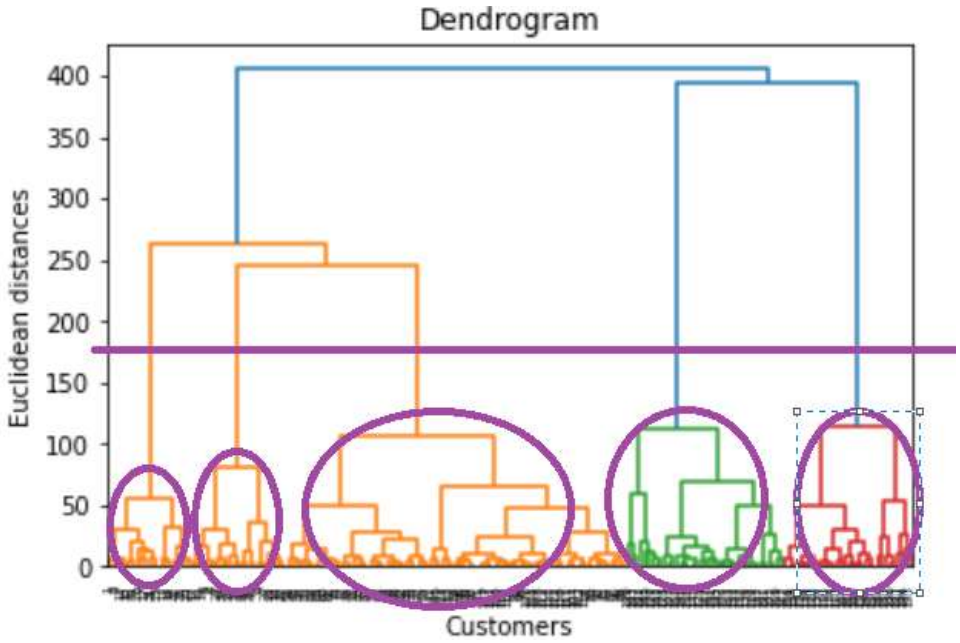






Вопрос:

На сколько кластеров целесообразно разбить эти данные?



Реализация иерархической кластеризации при помощи библиотеки **Scikit-Learn**

```
from sklearn.cluster import AgglomerativeClustering
# обучение на данных
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean', linkage = 'ward')
hc.fit(X)
```

Для построения дендрограммы используем библиотеку **SciPy**

```
from scipy.cluster.hierarchy import linkage, dendrogram
# mergings - массив связей (linkage array) с записанными сходствами между кластерами
mergings = linkage(X, method = 'ward')
dendrogram(mergings)
plt.show()
```

Сравнение метода k-средних с иерархической кластеризацией данных

✓ Иерархическая кластеризация хуже подходит для кластеризации больших объемов данных в сравнении с методом k-средних. Это объясняется тем, что временная сложность алгоритма линейна для метода k-средних (O(n)) и квадратична для метода иерархической кластеризации (O(n2))

✓ В кластеризации при помощи метода k-средних алгоритм начинает построение с произвольного выбора начальных точек, поэтому, результаты, генерируемые при многократном запуске алгоритма, могут отличаться. В то же время в случае иерархической кластеризации результаты воспроизводимы.

✓ Из центроидной геометрии построения метода k-средних следует, что метод хорошо работает, когда форма кластеров является гиперсферической (например, круг в 2D или сфера в 3D).

✓ Метод k-средних более чувствителен к зашумленным данным, чем иерархический метод.

НО, алгоритм иерархической кластеризации как и алгоритм k-means не в состоянии обработать сложные данные типа набора two_moons (на рис.).

С такими задачами способен справится алгоритм DBSCAN.

Алгоритм DBSCAN

DBSCAN (density based spatial clustering of applications with noise – плотностный алгоритм кластеризации пространственных данных с присутствием шума).

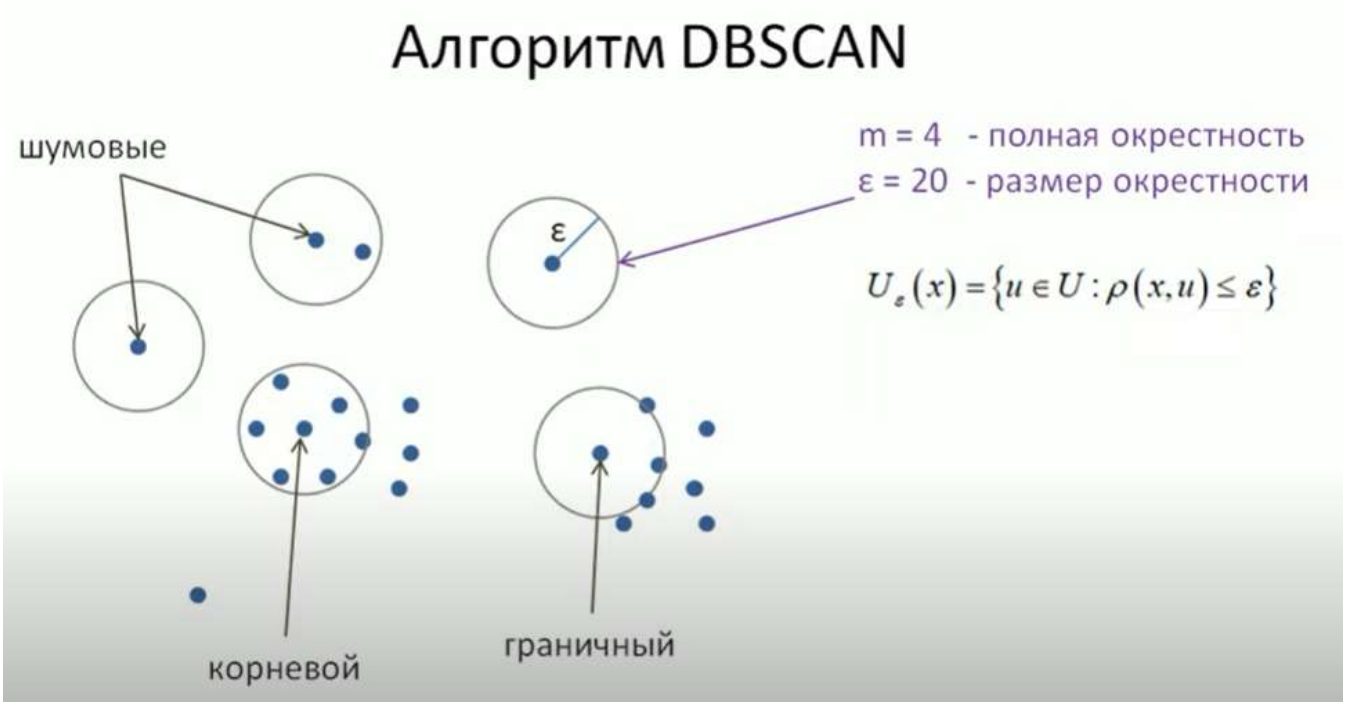
Основные преимущества алгоритма DBSCAN заключаются в том, что пользователю не нужно заранее задавать количество кластеров, алгоритм может выделить кластеры сложной формы и способен определить точки, которые не принадлежат какому-либо кластеру.

DBSCAN работает немного медленнее, чем алгоритм агломеративной кластеризации и алгоритм k-средних, но также может масштабироваться на относительно большие наборы данных.

Принцип работы

DBSCAN определяет точки, расположенные в «густонаселенных» областях пространства характеристик, когда многие точки данных расположены близко друг к другу. Эти области называются плотными(dense) областями пространства характеристик.

Идея алгоритма DBSCAN заключается в том, что кластеры образуют плотные области данных, которые отделены друг от друга относительно пустыми областями.

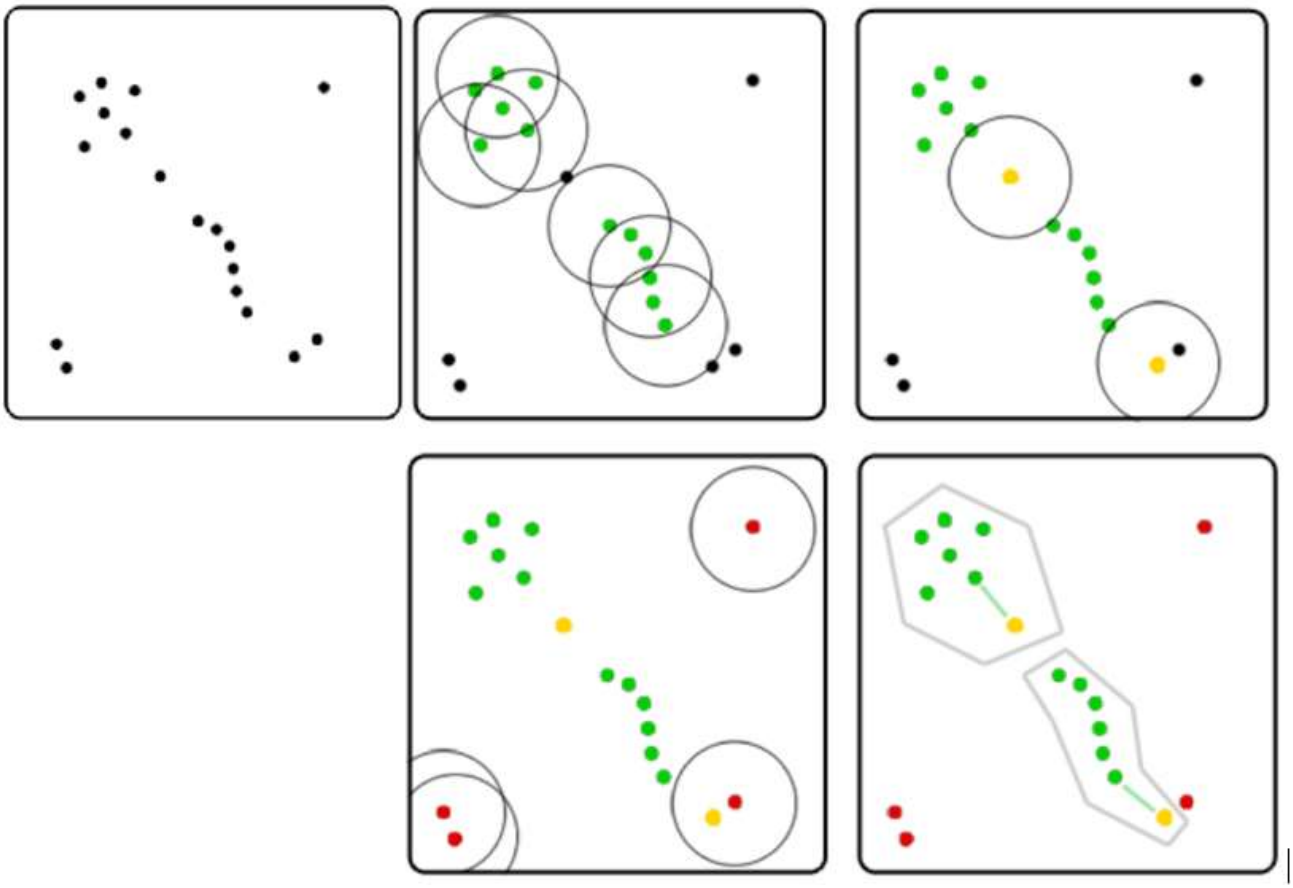


Точки, находящиеся в плотной области, называются ядровыми примерами (core samples) или ядровыми точками (core points). Алгоритм DBSCAN имеет два параметра: **min_samples** (количество соседей) и **eps** (радиус окрестности). Если по крайней мере min_samples точек находятся в радиусе окрестности eps рассматриваемой точки, то эта точка классифицируется как ядровая. Ядровые точки, расстояния между которыми не превышают радиус окрестности eps, помещаются алгоритмом DBSCAN в один и тот же кластер.

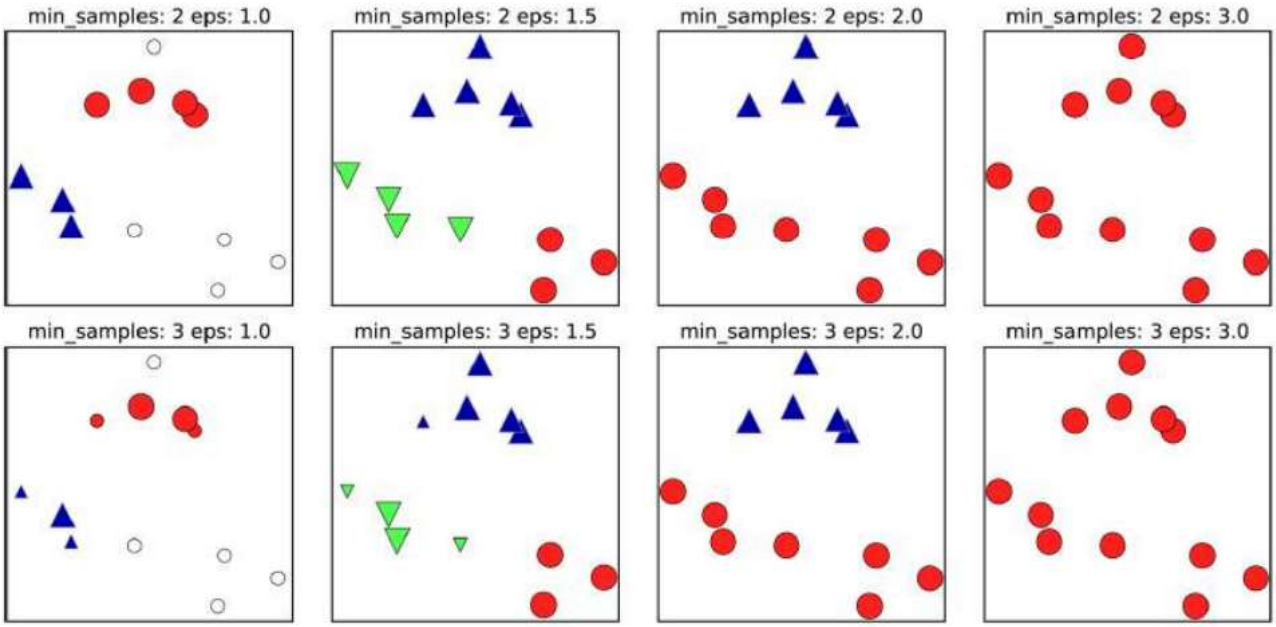
На старте алгоритм выбирает произвольную точку. Затем он находит все точки, удаленные от стартовой точки на расстоянии, не превышающем радиус окрестности eps. Если множество точек, находящихся в пределах радиуса окрестности eps, меньше значения min_samples, стартовая точка помечается как шум (noise), это означает, что она не принадлежит какому-либо кластеру. Если это множество точек больше значения min_samples, стартовая точка помечается как ядровая и ей назначается метка нового кластера. Затем посещаются все соседи этой точки (находящиеся в пределах eps). Если они еще не были присвоены кластеру, им присваивается метка только что созданного кластера. Если они являются ядровыми точками, поочередно посещаются их соседи и т.д. Кластер растет до тех пор, пока не останется ни одной ядерной точки в пределах радиуса окрестности eps. Затем выбирается другая точка, которая еще не была посещена, и повторяется та же самая процедура.

В итоге получаем три вида точек: ядровые точки, точки, которые находятся в пределах радиуса окрестности eps ядровых точек (так называемые пограничные точки или boundary points) и шумовые точки. При многократном применении алгоритма DBSCAN к конкретному набору данных результаты кластеризации ядровых точек будут всегда одинаковыми, при этом одни и те же точки всегда будут помечаться как шумовые. Однако пограничная точка может быть соседом

для ядровых точек из нескольких кластеров. Поэтому кластерная принадлежность пограничных точек зависит от порядка посещения точек. Как правило, существует лишь несколько пограничных точек, поэтому эта слабая зависимость результатов кластеризации от порядка посещения точек не имеет значения



Принадлежность к кластерам для различных значений *min_samples* и *eps* показана в сводке и визуализирована на рис.



На этом графике точки, которые принадлежат кластерам, окрашены сплошным цветом, а шумовые точки – белым цветом. Ядровые точки показаны в виде больших маркеров, тогда как пограничные точки отображаются в виде небольших маркеров. Увеличение значения eps (слева направо на рисунке) означает включение большего количества точек в кластер. Это приводит к росту размеров кластеров, но также может привести к тому, что несколько кластеров будут объединены в один. Увеличение значения min_samples (сверху вниз на рисунке) означает уменьшение количества ядерных точек и увеличение количества шумовых точек. Параметр eps чуть более важен, поскольку он определяет, что подразумевается под «близостью» точек друг к другу. Очень маленькое значение eps будет означать отсутствие ядерных точек и может привести к тому, что все точки будут помечены как шумовые. Очень большое значение eps приведет к тому, что все точки сформируют один кластер.

Значение min_samples главным образом определяет, будут ли точки, расположенные в менее плотных областях, помечены как выбросы или как кластеры. Если увеличить значение min_samples, все, что могло бы стать кластером с количеством точек, не превышающим min_samples, будет помечено как шум.

Несмотря на то что в алгоритме DBSCAN не нужно явно указывать количество кластеров, значение eps неявно задает количество выделяемых кластеров. Иногда подобрать оптимальное значение eps становится проще после масштабирования данных с помощью *StandardScaler* или *MinMaxScaler*, так как использование этих методов масштабирования гарантирует, что все характеристики будут иметь одинаковый масштаб.

Алгоритм показывает хорошие результаты для кластеров сложной формы, например, на наборе данных *two_moons*.

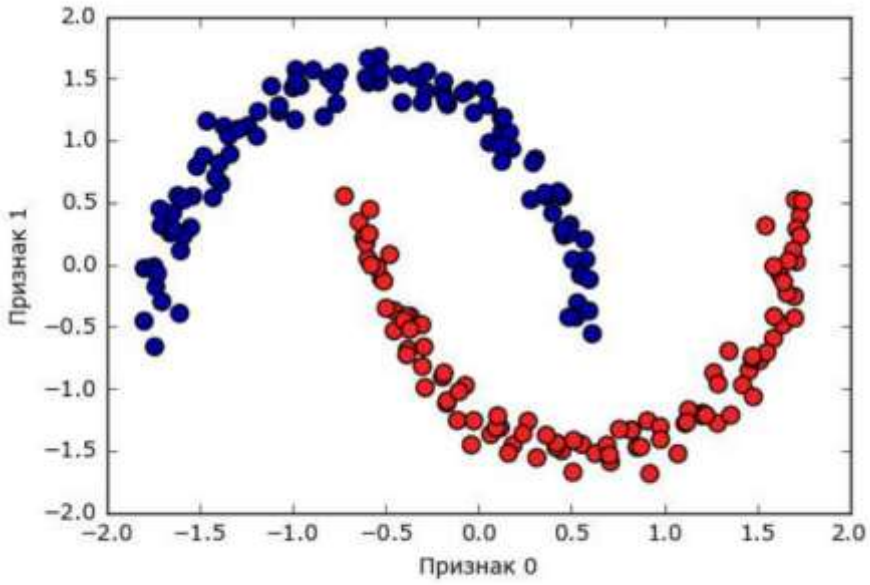
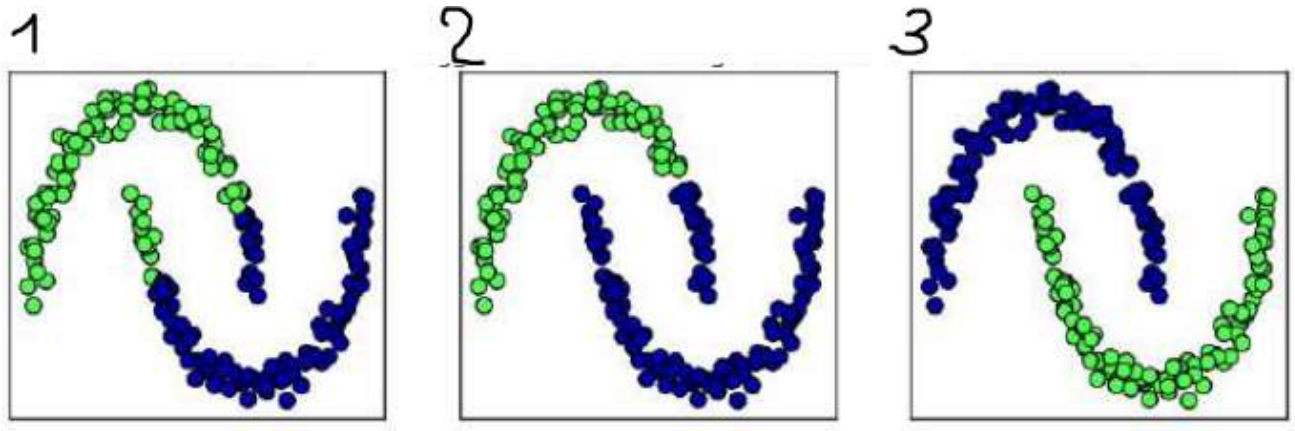
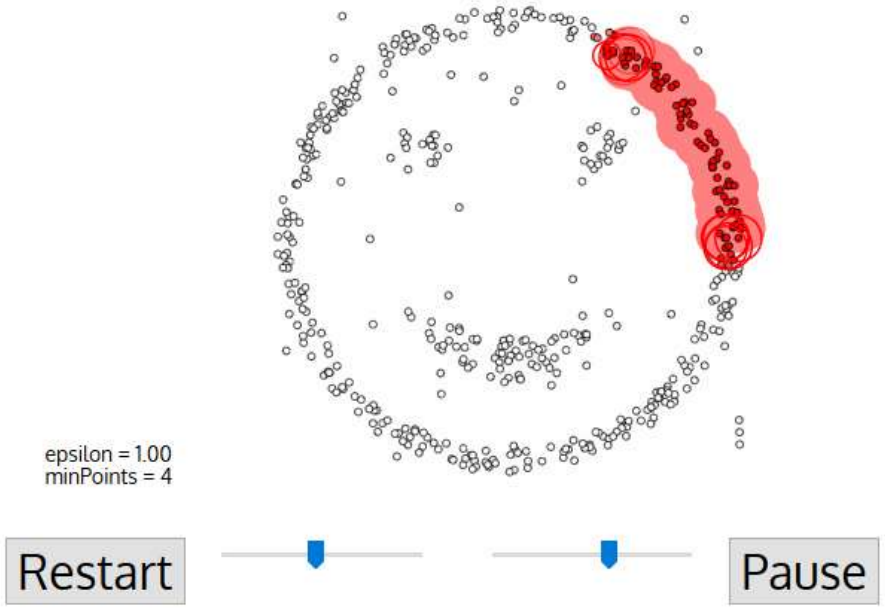


Рис. 3.38 Принадлежность к кластерам, вычисленная с помощью алгоритма DBSCAN, использовалось значение по умолчанию eps=0.5

Сравнение результатов случайной кластеризации, 1- k-средних,

2 - иерархической кластеризации и 3 - DBSCAN для набора данных *two_moons*





sklearn.cluster.DBSCAN

class **sklearn.cluster.DBSCAN**(*eps=0.5, *, min_samples=5, metric='euclidean', metric_params=None, algorithm='auto', leaf_size=30, p=None, n_jobs=None*) [source]

Методы оценки качества кластеризации

Метод оценки качества кластеризации — инструментарий для количественной оценки результатов кластеризации.

Принято выделять две группы методов оценки качества кластеризации:

- ✓ **Внешние** (англ. External) меры основаны на сравнении результата кластеризации с априори известным разделением на классы.
- ✓ **Внутренние** (англ. Internal) меры отображают качество кластеризации только по информации в данных.

Внешние меры оценки качества

Данные меры используют дополнительные знания о кластеризуемом множестве: распределение по кластерам, количество кластеров и т.д.

Внутренние меры оценки качества

Данные меры оценивают качество структуры кластеров опираясь только непосредственно на нее, не используя внешней информации.

Метрик оценки качества существует множество, рассмотрим некоторые.

- ✓ **Компактность кластеров (англ. Cluster Cohesion)**

Идея данного метода в том, что чем ближе друг к другу находятся объекты внутри кластеров, тем лучше разделение.

Таким образом, необходимо минимизировать внутриклассовое расстояние, например, сумму квадратов отклонений:

$$WSS = \sum_{j=1}^{M} \sum_{i=1}^{|C_j|} (x_{ij} - \overline{x_j})^2, \text{ где } M \text{ — количество кластеров.}$$

- ✓ **Разделимость кластеров (англ. Cluster Separation)**

В данном случае идея противоположная — чем дальше друг от друга находятся объекты разных кластеров, тем лучше.

Поэтому здесь стоит задача максимизации суммы квадратов отклонений:

$$BSS = n \cdot \sum_{j=1}^{M} (\overline{x_j} - \overline{x})^2, \text{ где } M \text{ — количество кластеров.}$$

✓ **Силуэт (англ. Silhouette)**

Значение силуэта показывает, насколько объект похож на свой кластер по сравнению с другими кластерами.

$$Sil(C) = \frac{1}{N} \sum_{c_k \in C} \sum_{x_i \in c_k} \frac{b(x_i, c_k) - a(x_i, c_k)}{max\{a(x_i, c_k), b(x_i, c_k)\}},$$

где:

$a(x_i, c_k) = \frac{1}{|c_k|} \sum_{x_j \in c_k} \|x_i - x_j\|$ — среднее расстояние от $x_i \in c_k$ до других объектов из кластера c_k (компактность),

$b(x_i, c_k) = min_{c_l \in C \setminus c_k} \{\frac{1}{|c_l|} \sum_{x_j \in c_l} \|x_i - x_j\|\}$ — среднее расстояние от $x_i \in c_k$ до объектов из другого кластера $c_l : k \neq l$ (отделимость).

Можно заметить, что

$$-1 \leq Sil(C) \leq 1.$$

Чем ближе данная оценка к 1, тем лучше.

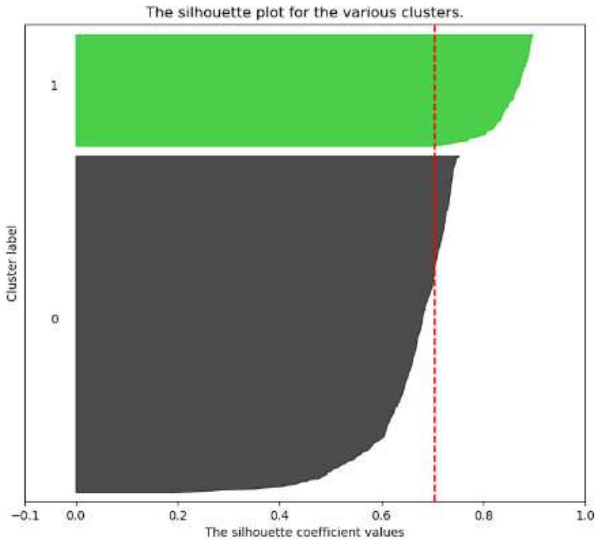
Есть также упрощенная вариация силуэта: $a(x_i, c_k)$ и $b(x_i, c_k)$ вычисляются через центры кластеров.

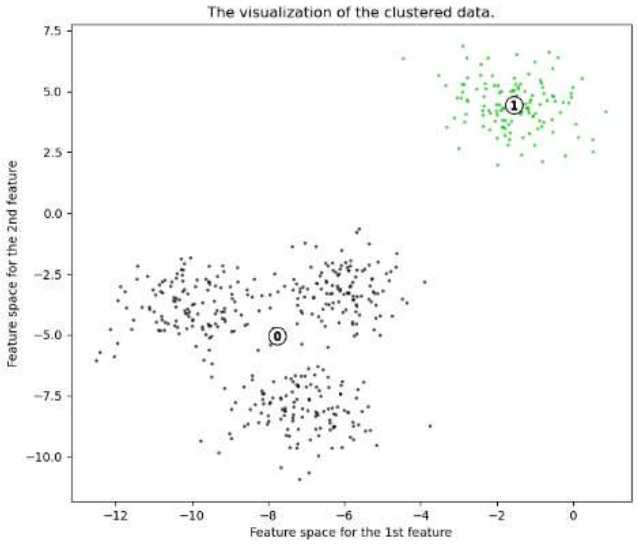
Значения силуэта, близкие к -1, соответствуют плохим (разрозненным) кластеризациям, значения, близкие к нулю, говорят о наложении кластеров, значения, близкие к 1, соответствуют "плотным" четко выделенным кластерам.

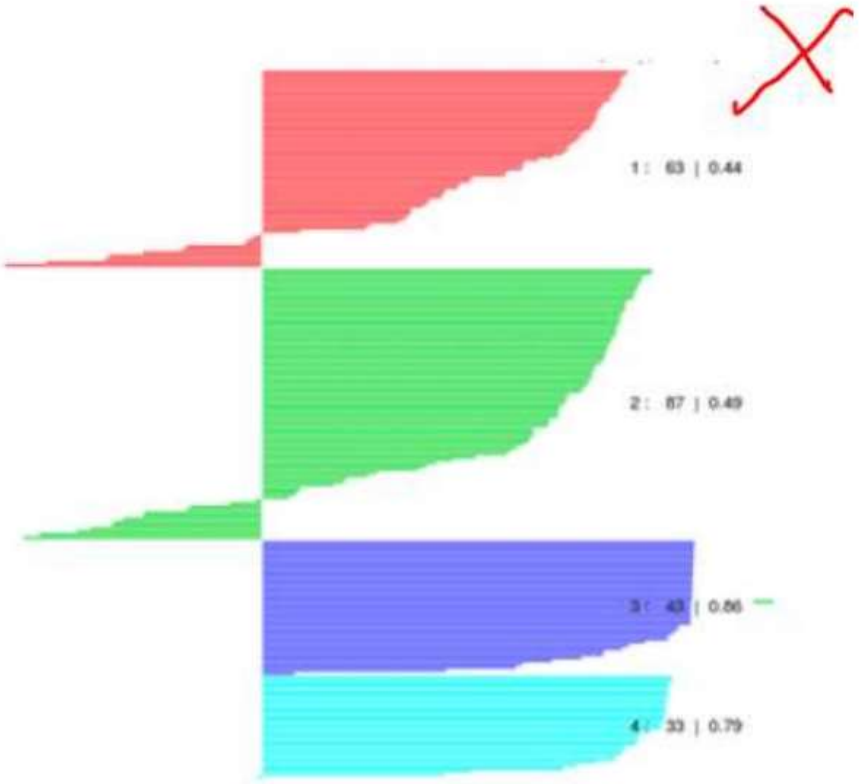
С помощью силуэта можно выбирать оптимальное число кластеров k (если оно заранее неизвестно) — выбирается число кластеров, максимизирующее значение силуэта.

https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

Примеры силуэтов







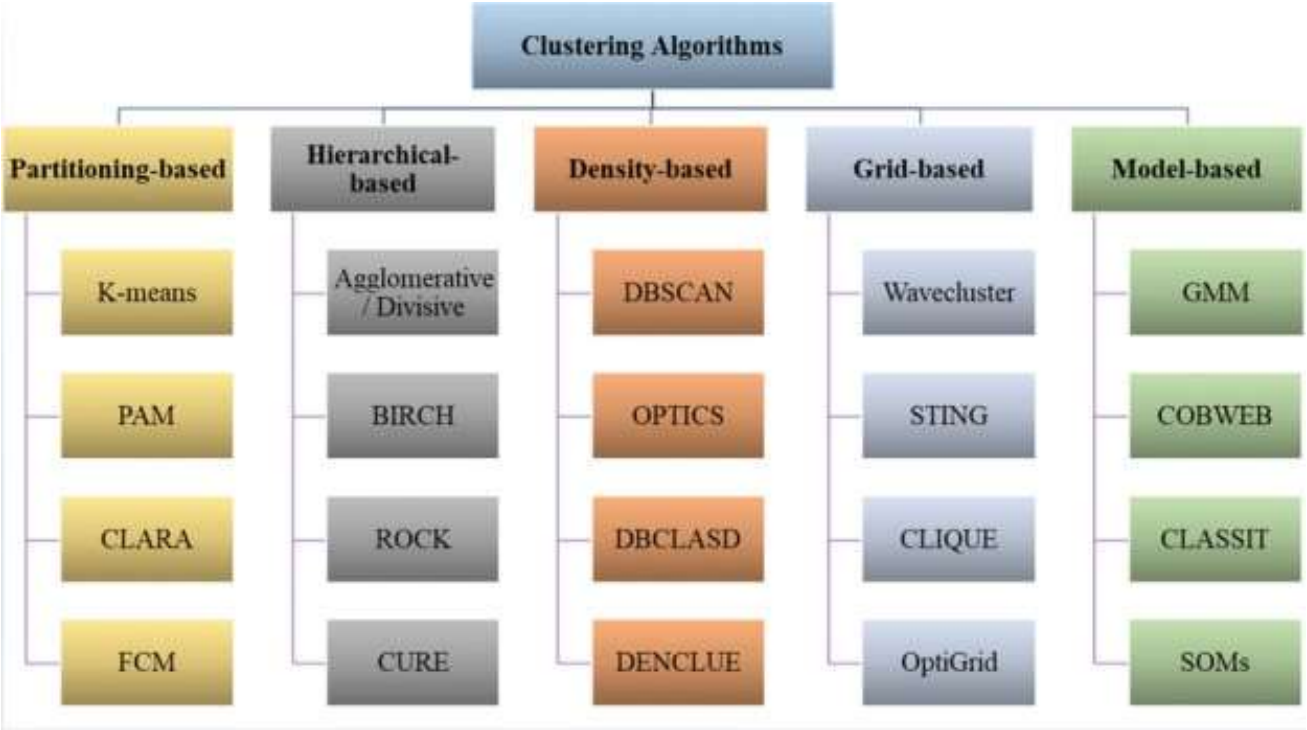
Метрики оценки качества кластеризации реализованы в sklearn.metrics.

Классификация методов кластеризации

Трудно обеспечить четкую категоризацию методов кластеризации, поскольку эти категории могут перекрываться, а метод может иметь функции из нескольких категорий.

Фахад и др. (2014) предложили классификацию кластеризации с точки зрения

разработчика алгоритма. Он пытается разделить различные подходы к кластеризации на основе технических деталей каждого подхода, в результате чего получается дерево, показанное на рис.



- Методы разбиения
- Иерархические методы
- Методы, основанные на плотности
- Грид-методы
- Методы, основанные на моделях

Кластеризацию применяют в:

- ✓ Сегментации рынка (типов покупателей, лояльности)
- ✓ Объединении близких точек на карте
- ✓ Сжатии изображений
- ✓ Обнаружения выбросов, где выбросы могут быть более интересными, чем обычные случаи.
- ✓ Уменьшения объема данных путем категоризации или группировки похожих элементов данных.
- ✓ Анализ ДНК
- ✓ Социальные сети

Один из хороших примеров применения методов кластеризации – **анализ геоданных**. При использовании приложений в мобильных телефонах часто необходимо определить точное местоположение. Погрешность в GPS-данных возникает из-за движения пользователей: часто приходится наблюдать множество точек вместо точного положения. Это особенно актуально при анализе поведения тысяч людей в определенной локации, например, для определения наиболее популярных мест, где пользователи садятся в такси у аэропорта.

Яндекс.Такси применял кластеризацию координат точек заказа, чтобы определить удобные для вызова такси места. Центры классов использовались в качестве кандидатов пикап-пойнтов, подсвечиваемых в приложении. Учитывали и простые фильтры, чтобы исключить точки в зданиях, в воде. Пикап-пойнты, выставленные вручную, использовались, например, рядом с аэропортами.

Еще одним примером, связанным с кластеризацией геоданных, является поиск изображений по месту съёмки (геопозиции). Изображения отображаются на карте, а при масштабировании карты можно увидеть разные кластеры, на которые они делятся.

Также интересным примером является **построение цветовой схемы интерфейса** под выбранное пользователем изображение: необходимо произвести кластеризацию цветов, используя RGB-представление или другие признаки оттенка. Затем использовать их для оформления интерфейса, включая фоновую картинку.