<https://yandex.ru/search/?text=Dataset+%D0%B4%D0%BB%D1%8F+%D0%BA%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8&clid=1836587&lr=46&p=2&cee=1>

<https://sky.pro/media/kak-ispolzovat-python-dlya-raboty-s-klasterizacziej-dannyh/>

# Как использовать Python для работы с кластеризацией данных

Кластеризация данных — это процесс группирования набора объектов таким образом, чтобы объекты в одной группе (кластере) были более похожи друг на друга, чем на объекты в других кластерах. В этой статье мы рассмотрим, как использовать Python для работы с кластеризацией данных.

## Библиотеки и инструменты

Для работы с кластеризацией данных в Python, мы будем использовать следующие библиотеки:

* NumPy
* pandas
* scikit-learn

Убедитесь, что у вас установлены эти библиотеки. Если нет, вы можете установить их с помощью команды:

pip install numpy pandas scikit-learn

## Загрузка и подготовка данных

Для начала нам нужно загрузить и подготовить данные. В этом примере мы будем использовать набор данных о цветках ириса, который можно загрузить из библиотеки scikit-learn. Загрузим и подготовим данные следующим образом:

import pandas as pd

from sklearn.datasets import load\_iris

iris = load\_iris()

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

Python-разработчик: новая работа через 9 месяцев

Получится, даже если у вас нет опыта в IT

[Получить программу](https://sky.pro/courses/programming/python-web-course?utm_source=skypro_blog&utm_medium=free&utm_campaign=kak-ispolzovat-python-dlya-raboty-s-klasterizacziej-dannyh)



## Выбор алгоритма кластеризации

Есть множество алгоритмов кластеризации, таких как K-means, DBSCAN, иерархическая кластеризация и т. д. В этом примере мы будем использовать алгоритм K-means:

from sklearn.cluster import KMeans

kmeans = KMeans(n\_clusters=3)

kmeans.fit(data)

В этом случае, мы выбрали количество кластеров равное 3. Вы можете изменить это число в зависимости от вашего набора данных и задачи.

## Результаты кластеризации

После выполнения алгоритма кластеризации, мы можем проверить результаты следующим образом:

data['cluster'] = kmeans.labels\_

print(data.head())

Этот код добавляет столбец «cluster» в наш DataFrame, который показывает, к какому кластеру относится каждый объект.

😉 Теперь вы знаете основы использования Python для работы с кластеризацией данных! Это лишь начало, и существует множество других алгоритмов и методов, которые вы можете изучить и применять в своих проектах.

Python-разработчик: новая работа через 9 месяцев

Получится, даже если у вас нет опыта в IT

[Получить программу](https://sky.pro/courses/programming/python-web-course?utm_source=skypro_blog&utm_medium=free&utm_campaign=kak-ispolzovat-python-dlya-raboty-s-klasterizacziej-dannyh)



Не забудьте практиковаться и изучать дополнительные материалы, чтобы углубить свои знания в работе с кластеризацией данных на Python. Удачи вам в вашем обучении!

[[](https://sky.pro/media/author/alexkodov/)](https://sky.pro/media/author/alexkodov/)

[Алексей Кодов](https://sky.pro/media/author/alexkodov/)

[Автор статьи](https://sky.pro/media/author/alexkodov/)

#### Содержание

* + [Библиотеки и инструменты](https://sky.pro/media/kak-ispolzovat-python-dlya-raboty-s-klasterizacziej-dannyh/#title0)
  + [Загрузка и подготовка данных](https://sky.pro/media/kak-ispolzovat-python-dlya-raboty-s-klasterizacziej-dannyh/#title1)
  + [Выбор алгоритма кластеризации](https://sky.pro/media/kak-ispolzovat-python-dlya-raboty-s-klasterizacziej-dannyh/#title2)
  + [Результаты кластеризации](https://sky.pro/media/kak-ispolzovat-python-dlya-raboty-s-klasterizacziej-dannyh/#title3)

### Добавить комментарий

[Войти](https://sky.pro/media/kak-ispolzovat-python-dlya-raboty-s-klasterizacziej-dannyh/)

Начало формы



[Работает на AnyComment](https://anycomment.io)

Конец формы

Войти через

* [Конфиденциальность](https://anycomment.io/legal/privacy)
* [Условия](https://anycomment.io/legal/terms)

### Ещё по теме

[](https://sky.pro/media/chto-nuzhno-znat-i-umet-junior-python-razrabotchiku/)[Программирование](https://sky.pro/media/programmirovanie/)

15 Ноя 2023

8 мин

[Что нужно знать и уметь Junior Python-разработчику, чтобы найти работу](https://sky.pro/media/chto-nuzhno-znat-i-umet-junior-python-razrabotchiku/)

Работать в команде, писать SQL-запросы и знать фреймворки.

[#python](https://sky.pro/media/tag/python/)

[](https://sky.pro/media/skolko-mozhno-zarabativat-rabotaya-bez-vysshego-obrazovaniya/)[Программирование](https://sky.pro/media/programmirovanie/)

24 Окт 2023

13 мин

[Мир IT без «корочки»: как найти работу без высшего образования](https://sky.pro/media/skolko-mozhno-zarabativat-rabotaya-bez-vysshego-obrazovaniya/)

Что делать, если страшно менять профессию, но хочется: для каких сфер не нужен диплом и сколько можно зарабатывать без высшего образования

[#python](https://sky.pro/media/tag/python/)

[Программирование](https://sky.pro/media/programmirovanie/)

29 Сен 2023

2 мин

[Самые востребованные языки программирования: актуальный рейтинг](https://sky.pro/media/samye-vostrebovannye-yazyki-programmirovaniya-aktualnyj-rejting/)

Ищете самые востребованные языки программирования? В этом рейтинге мы откроем вам топ-3.

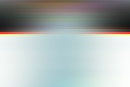
[#python](https://sky.pro/media/tag/python/)

<https://vk.com/wall-166353059_38658>

##### [Искусственный интеллект и нейронауки. AI Новости](https://vk.com/ainewsru)

[30 авг 2021](https://vk.com/wall-166353059_38658)

10 алгоритмов кластеризации с помощью Python  
  
астеризация или кластерный анализ-это проблема обучения без контроля.  
Он часто используется в качестве метода анализа данных для выявления интересных закономерностей в данных, таких как группы клиентов, основанные на их поведении.  
  
Существует множество алгоритмов кластеризации на выбор, и нет единого наилучшего алгоритма кластеризации для всех случаев. Вместо этого рекомендуется изучить ряд алгоритмов кластеризации и различные конфигурации для каждого алгоритма.  
  
В этом уроке вы узнаете, как подобрать и использовать лучшие алгоритмы кластеризации в python.  
  
После завершения этого урока вы узнаете:  
  
Кластеризация-это неконтролируемая проблема поиска естественных групп в пространстве признаков входных данных.  
Существует множество различных алгоритмов кластеризации и нет единого наилучшего метода для всех наборов данных.  
Как реализовать, адаптировать и использовать лучшие алгоритмы кластеризации в Python с помощью библиотеки машинного обучения scikit-learn.  
  
Начните свой проект с моей новой книги "Мастерство машинного обучения с Python", включающей пошаговые руководства и файлы исходного кода Python для всех примеров.  
  
Давайте начнем.  
  
Обзор учебника  
  
Этот учебник разделен на три части; они:  
  
Кластеризация  
Алгоритмы кластеризации  
Примеры алгоритмов кластеризации  
Установка библиотеки  
Набор данных для кластеризации  
Распространение Аффинности  
Агломеративная кластеризация  
БЕРЕЗА  
DBSCAN  
К-Означает  
Мини-Партия K-Означает  
Среднее Смещение  
ОПТИКА  
Спектральная кластеризация  
Модель Гауссовой смеси  
  
Кластеризация  
  
Кластерный анализ, или кластеризация, - это задача машинного обучения без контроля.  
  
Это включает в себя автоматическое обнаружение естественной группировки в данных. В отличие от контролируемого обучения (например, прогностического моделирования), алгоритмы кластеризации интерпретируют только входные данные и находят естественные группы или кластеры в пространстве объектов.  
  
Методы кластеризации применяются, когда нет класса, который нужно предсказать, а скорее, когда экземпляры должны быть разделены на естественные группы.  
  
— Страница 141, Интеллектуальный анализ данных: Практические инструменты и методы машинного обучения, 2016.  
  
Кластер часто представляет собой область плотности в пространстве объектов, где примеры из области (наблюдения или строки данных) находятся ближе к кластеру, чем другие кластеры. Кластер может иметь центр (центроид), который является образцом или пространством точечных объектов, и может иметь границу или экстент.  
  
Эти кластеры, по-видимому, отражают некоторый механизм, работающий в домене, из которого извлекаются экземпляры, механизм, который заставляет некоторые экземпляры иметь большее сходство друг с другом, чем с остальными экземплярами.  
  
— Страницы 141-142, Интеллектуальный анализ данных: Практические инструменты и методы машинного обучения, 2016.  
  
Кластеризация может быть полезна в качестве действия по анализу данных, чтобы узнать больше о проблемной области, так называемом обнаружении шаблонов или обнаружении знаний.  
  
Например:  
  
Филогенетическое дерево можно считать результатом ручного кластерного анализа.  
Разделение нормальных данных от выбросов или аномалий может рассматриваться как проблема кластеризации.  
Разделение кластеров на основе их естественного поведения является проблемой кластеризации, называемой сегментацией рынка.  
  
Кластеризация также может быть полезна как тип проектирования объектов, где существующие и новые примеры могут быть сопоставлены и помечены как принадлежащие к одному из идентифицированных кластеров в данных.  
  
Оценка выявленных кластеров является субъективной и может потребовать участия эксперта по предметной области, хотя существует множество количественных показателей, специфичных для кластеризации. Как правило, алгоритмы кластеризации сравниваются академически на синтетических наборах данных с заранее определенными кластерами, которые, как ожидается, обнаружит алгоритм.  
  
Кластеризация-это метод обучения без контроля, поэтому трудно оценить качество результатов любого данного метода.  
  
— Стр. 534, Машинное обучение: Вероятностная перспектива, 2012.  
  
Алгоритмы кластеризации  
  
Существует множество типов алгоритмов кластеризации.  
  
Многие алгоритмы используют меры сходства или расстояния между примерами в пространстве объектов в попытке обнаружить плотные области наблюдений. Таким образом, часто рекомендуется масштабировать данные до использования алгоритмов кластеризации.  
  
Центральным для всех целей кластерного анализа является понятие степени сходства (или различия) между отдельными объектами, которые группируются. Метод кластеризации пытается сгруппировать объекты на основе предоставленного ему определения сходства.  
  
— Страница 502, Элементы статистического обучения: Интеллектуальный анализ данных, выводы и прогнозирование, 2016.  
  
Некоторые алгоритмы кластеризации требуют, чтобы вы указали или предположили количество кластеров, которые необходимо обнаружить в данных, в то время как другие требуют указания некоторого минимального расстояния между наблюдениями, в которых примеры могут считаться “близкими” или “связанными”.  
  
Таким образом, кластерный анализ представляет собой итерационный процесс, в ходе которого субъективная оценка идентифицированных кластеров возвращается в изменения конфигурации алгоритма до тех пор, пока не будет достигнут желаемый или соответствующий результат.  
  
Библиотека scikit-learn предоставляет набор различных алгоритмов кластеризации на выбор.  
  
Список из 10 наиболее популярных алгоритмов выглядит следующим образом:  
  
Распространение Аффинности  
Агломеративная кластеризация  
БЕРЕЗА  
DBSCAN  
К-Означает  
Мини-Партия K-Означает  
Среднее Смещение  
ОПТИКА  
Спектральная кластеризация  
Смесь гауссов  
  
Каждый алгоритм предлагает свой подход к решению задачи обнаружения естественных групп в данных.  
  
Не существует наилучшего алгоритма кластеризации и простого способа найти наилучший алгоритм для ваших данных без использования контролируемых экспериментов.  
  
В этом уроке мы рассмотрим, как использовать каждый из этих 10 популярных алгоритмов кластеризации из библиотеки scikit-learn.  
  
Приведенные примеры послужат основой для копирования и вставки примеров и тестирования методов на ваших собственных данных.  
  
Мы не будем углубляться в теорию, лежащую в основе работы алгоритмов, или сравнивать их напрямую. Для хорошей отправной точки по этой теме см.:  
  
Кластеризация, API scikit-learn.  
  
Давайте погрузимся в это.  
  
Примеры алгоритмов кластеризации  
  
В этом разделе мы рассмотрим, как использовать 10 популярных алгоритмов кластеризации в scikit-learn.  
  
Это включает в себя пример подгонки модели и пример визуализации результата.  
  
Примеры предназначены для того, чтобы вы могли скопировать и вставить их в свой собственный проект и применить методы к своим собственным данным.  
  
Установка библиотеки  
  
Во-первых, давайте установим библиотеку.  
  
Не пропустите этот шаг, так как вам нужно будет убедиться, что у вас установлена последняя версия.  
  
Вы можете установить библиотеку scikit-learn с помощью установщика pip Python следующим образом:  
  
sudo pip установите scikit-узнайте  
  
1  
  
sudo pip установите scikit-узнайте  
  
Дополнительные инструкции по установке, относящиеся к вашей платформе, см. в разделе:  
  
Установка scikit-узнайте  
  
Затем давайте подтвердим, что библиотека установлена и вы используете современную версию.  
  
Запустите следующий сценарий, чтобы распечатать номер версии библиотеки.  
  
# проверьте scikit-изучите версию импорта печати sklearn(sklearn.\_\_версия\_\_)  
  
1  
  
2  
  
3  
  
# проверьте версию scikit-learn  
  
импорт sklearn  
  
печать(sklearn.\_\_версия\_\_)  
  
Запустив пример, вы должны увидеть следующий номер версии или выше.  
  
0.22.1  
  
1  
  
0.22.1  
  
Набор данных для кластеризации  
  
Мы будем использовать функцию make\_classification() для создания тестового набора данных двоичной классификации.  
  
Набор данных будет содержать 1000 примеров с двумя входными объектами и одним кластером на класс. Кластеры визуально видны в двух измерениях, так что мы можем построить данные с помощью точечной диаграммы и раскрасить точки на графике по назначенному кластеру. Это поможет увидеть, по крайней мере, в тестовой задаче, насколько “хорошо” были идентифицированы кластеры.  
  
Кластеры в этой тестовой задаче основаны на многомерном гауссовском, и не все алгоритмы кластеризации будут эффективны при идентификации этих типов кластеров. Таким образом, результаты, приведенные в этом руководстве, не должны использоваться в качестве основы для сравнения методов в целом.  
  
Ниже приведен пример создания и обобщения набора данных синтетической кластеризации  
  
# synthetic classification dataset from numpy import where from sklearn.datasets import make\_classification from matplotlib import pyplot # define dataset X, y = make\_classification(n\_samples=1000, n\_features=2, n\_informative=2, n\_redundant=0, n\_clusters\_per\_class=1, random\_state=4) # create scatter plot for samples from each class for class\_value in range(2): # get row indexes for samples with this class row\_ix = where(y == class\_value) # create scatter of these samples pyplot.scatter(X[row\_ix, 0], X[row\_ix, 1]) # show the plot [pyplot.show](https://vk.com/away.php?to=http%3A%2F%2Fpyplot.show&post=-166353059_38658&cc_key=)()  
  
1  
  
2  
  
3  
  
4  
  
5  
  
6  
  
7  
  
8  
  
9  
  
10  
  
11  
  
12  
  
13  
  
14  
  
# synthetic classification dataset  
  
from numpy import where  
  
from sklearn.datasets import make\_classification  
  
from matplotlib import pyplot  
  
# define dataset  
  
X, y = make\_classification(n\_samples=1000, n\_features=2, n\_informative=2, n\_redundant=0, n\_clusters\_per\_class=1, random\_state=4)  
  
# create scatter plot for samples from each class  
  
for class\_value in range(2):  
  
# get row indexes for samples with this class  
  
row\_ix = where(y == class\_value)  
  
# create scatter of these samples  
  
pyplot.scatter(X[row\_ix, 0], X[row\_ix, 1])  
  
# show the plot  
  
[pyplot.show](https://vk.com/away.php?to=http%3A%2F%2Fpyplot.show&post=-166353059_38658&cc_key=)()  
  
При выполнении примера создается набор данных синтетической кластеризации, затем создается точечная диаграмма входных данных с точками, окрашенными меткой класса (идеализированные кластеры).  
  
Мы можем четко видеть две различные группы данных в двух измерениях, и есть надежда, что алгоритм автоматической кластеризации сможет обнаружить эти группировки.  
  
Точечная диаграмма набора данных синтетической кластеризации С точками, Окрашенными известным кластером  
  
Затем мы можем начать рассматривать примеры алгоритмов кластеризации, применяемых к этому набору данных.  
  
Я предпринял несколько минимальных попыток настроить каждый метод на набор данных.  
  
Можете ли вы получить лучший результат для одного из алгоритмов?  
Дайте мне знать в комментариях ниже.  
  
Распространение Аффинности  
  
Распространение сходства включает в себя поиск набора примеров, которые наилучшим образом обобщают данные.  
  
Мы разработали метод, называемый “распространением сходства”, который принимает в качестве входных показателей сходство между парами точек данных. Между точками данных осуществляется обмен реальными сообщениями до тех пор, пока постепенно не появится высококачественный набор примеров и соответствующих кластеров  
  
— Кластеризация путем передачи сообщений между точками данных, 2007.  
  
Методика описана в статье:  
  
Кластеризация путем передачи сообщений между точками Данных, 2007.  
  
Он реализован с помощью класса Affinityprop, и основной настройкой для настройки является “демпфирование”, установленное в диапазоне от 0,5 до 1, и, возможно, “предпочтение”.  
  
Полный пример приведен ниже.  
  
# affinity propagation clustering from numpy import unique from numpy import where from sklearn.datasets import make\_classification from sklearn.cluster import AffinityPropagation from matplotlib import pyplot # define dataset X, \_ = make\_classification(n\_samples=1000, n\_features=2, n\_informative=2, n\_redundant=0, n\_clusters\_per\_class=1, random\_state=4) # define the model model = AffinityPropagation(damping=0.9) # fit the model [model.fit](https://vk.com/away.php?to=http%3A%2F%2Fmodel.fit&post=-166353059_38658&cc_key=)(X) # assign a cluster to each example yhat = model.predict(X) # retrieve unique clusters clusters = unique(yhat) # create scatter plot for samples from each cluster for cluster in clusters: # get row indexes for samples with this cluster row\_ix = where(yhat == cluster) # create scatter of these samples pyplot.scatter(X[row\_ix, 0], X[row\_ix, 1]) # show the plot [pyplot.show](https://vk.com/away.php?to=http%3A%2F%2Fpyplot.show&post=-166353059_38658&cc_key=)()  
  
1  
  
2  
  
3  
  
4  
  
5  
  
6  
  
7  
  
8  
  
9  
  
10  
  
11  
  
12  
  
13  
  
14  
  
15  
  
16  
  
17  
  
18  
  
19  
  
20  
  
21  
  
22  
  
23  
  
24  
  
# affinity propagation clustering  
  
from numpy import unique  
  
from numpy import where  
  
from sklearn.datasets import make\_classification  
  
from sklearn.cluster import AffinityPropagation  
  
from matplotlib import pyplot  
  
# define dataset  
  
X, \_ = make\_classification(n\_samples=1000, n\_features=2, n\_informative=2, n\_redundant=0, n\_clusters\_per\_class=1, random\_state=4)  
  
# define the model  
  
model = AffinityPropagation(damping=0.9)  
  
# fit the model  
  
[model.fit](https://vk.com/away.php?to=http%3A%2F%2Fmodel.fit&post=-166353059_38658&cc_key=)(X)  
  
# assign a cluster to each example  
  
yhat = model.predict(X)  
  
# retrieve unique clusters  
  
clusters = unique(yhat)  
  
# create scatter plot for samples from each cluster  
  
for cluster in clusters:  
  
# get row indexes for samples with this cluster  
  
row\_ix = where(yhat == cluster)  
  
# create scatter of these samples  
  
pyplot.scatter(X[row\_ix, 0], X[row\_ix, 1])  
  
# show the plot  
  
[pyplot.show](https://vk.com/away.php?to=http%3A%2F%2Fpyplot.show&post=-166353059_38658&cc_key=)()  
  
Запуск примера соответствует модели в обучающем наборе данных и предсказывает кластер для каждого примера в наборе данных. Затем создается точечная диаграмма с точками, окрашенными в соответствии с назначенным им кластером.  
  
В этом случае я не смог добиться хорошего результата.  
  
Точечная диаграмма набора данных С Кластерами, Идентифицированными С помощью Распространения сходства  
  
Агломеративная кластеризация  
  
Агломеративная кластеризация включает объединение примеров до тех пор, пока не будет достигнуто желаемое количество кластеров.  
  
Это часть более широкого класса методов иерархической кластеризации, и вы можете узнать больше здесь:  
  
Иерархическая кластеризация, Википедия.  
  
Он реализован с помощью класса AgglomerativeClustering, и основной настройкой для настройки является набор “n\_clusters”, оценка количества кластеров в данных, например 2.  
  
Полный пример приведен ниже.  
  
# agglomerative clustering from numpy import unique from numpy import where from sklearn.datasets import make\_classification from sklearn.cluster import AgglomerativeClustering from matplotlib import pyplot # define dataset X, \_ = make\_classification(n\_samples=1000, n\_features=2, n\_informative=2, n\_redundant=0, n\_clusters\_per\_class=1, random\_state=4) # define the model model = AgglomerativeClustering(n\_clusters=2) # fit model and predict clusters yhat = [model.fit](https://vk.com/away.php?to=http%3A%2F%2Fmodel.fit&post=-166353059_38658&cc_key=)\_predict(X) # retrieve unique clusters clusters = unique(yhat) # create scatter plot for samples from each cluster for cluster in clusters: # get row indexes for samples with this cluster row\_ix = where(yhat == cluster) # create scatter of these samples pyplot.scatter(X[row\_ix, 0], X[row\_ix, 1]) # show the plot [pyplot.show](https://vk.com/away.php?to=http%3A%2F%2Fpyplot.show&post=-166353059_38658&cc_key=)()  
  
1  
  
2  
  
3  
  
4  
  
5  
  
6  
  
7  
  
8  
  
9  
  
10  
  
11  
  
12  
  
13  
  
14  
  
15  
  
16  
  
17  
  
18  
  
19  
  
20  
  
21  
  
22  
  
# agglomerative clustering  
  
from numpy import unique  
  
from numpy import where  
  
from sklearn.datasets import make\_classification  
  
from sklearn.cluster import AgglomerativeClustering  
  
from matplotlib import pyplot  
  
# define dataset  
  
X, \_ = make\_classification(n\_sample



<https://www.pvsm.ru/python/371244>

# Кластеризация изображений с помощью нейросети CLIP

В статье пойдёт речь о том, как можно автоматически разделить датасет изображений на кластеры, которые поделены по качественному контекстному признаку, благодаря эмбедингам из нашумевшей нейронной сети CLIP от компании Илона Маска. Расскажу на примере контента из нашего приложения iFunny.

Кластеризация считается unsupervised задачей — это значит, что нет никакой явной разметки целевых значений, то есть нет «учителя». В нашем случае мы загружаем некий датасет картинок и хотим произвольно, но качественно побить его на кластеры.

Например, набор изображений животных может разделиться на кластеры по виду, по полосатости, по количеству лап или другим признакам. В любом случае ожидается понятная логика разбивки, которую можно дальше использовать для других задач.

Под катом расскажу, как мы построили логичную кластеризацию с помощью библиотеки HDBSCAN и векторов из нейронной сети CLIP, и каких результатов добились на выходе.

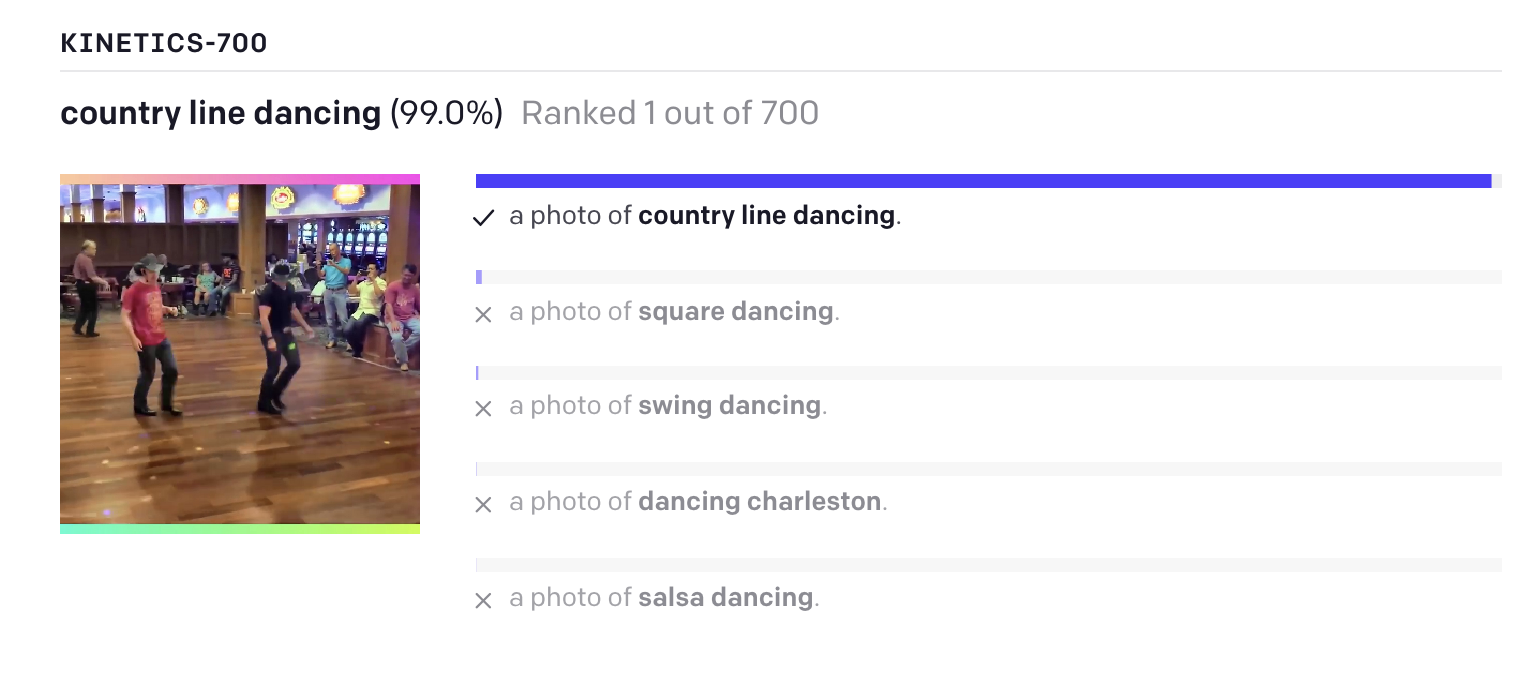
### Что такое нейросеть CLIP

В январе 2021 года компания Илона Маска OpenAI выпустила нейросеть CLIP ([официальный сайт](https://openai.com/blog/clip/) и [код на GitHub](https://github.com/openai/CLIP)). Её обучали обобщать огромное количество категорий, чтобы затем использовать в разных ML-задачах. Разметки классов под конкретную задачу в ней нет, зато есть пары изображений и их текстовые описания в одном пространстве. Отсюда вытекает главный плюс — данную сеть можно использовать в задачах классификации изображений даже без дообучения (zero-shot).

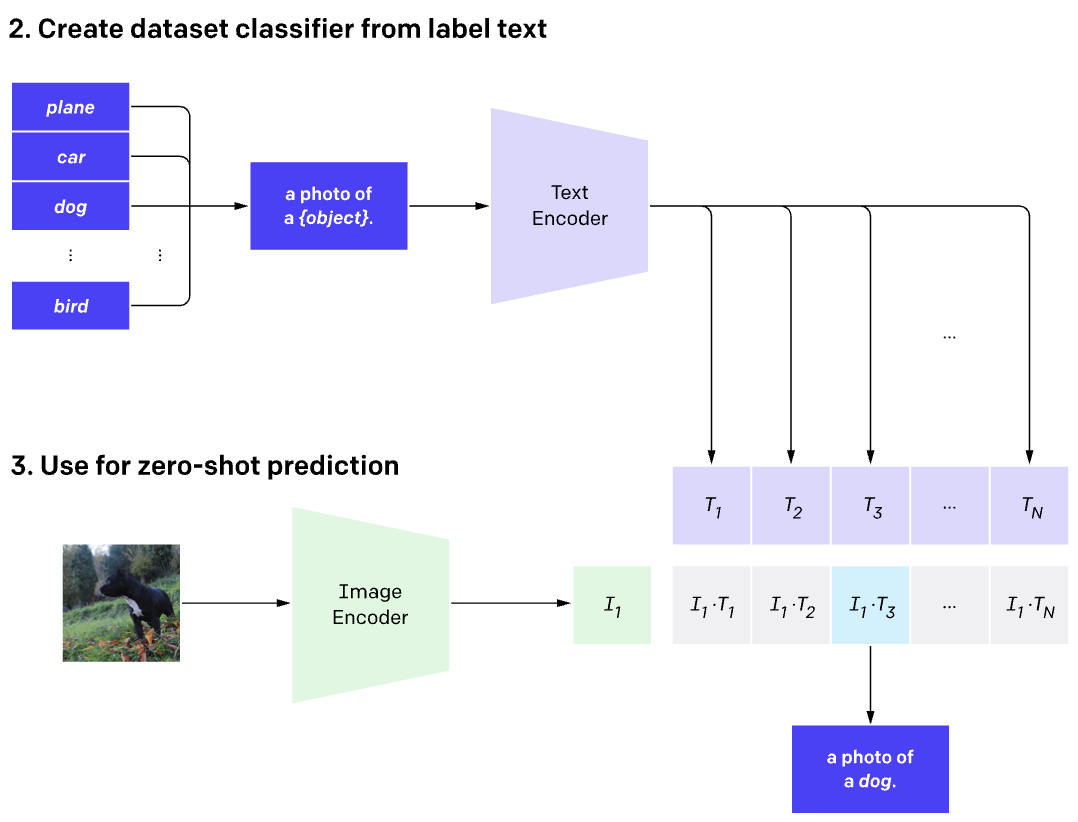
Нейросеть CLIP обучена на 400 миллионов пар изображений и текста, каждая из которых подается на вход нейросети и объединяется с другими парами в батч. Затем сеть обучается предсказывать, какие из пар картинок в батче действительно схожи друг с другом. Тем самым векторные представления в паре текста и изображения сближаются во время обучения.

Обученная модель позволяет одновременно получать эмбединги по произвольному тексту и изображению — то есть с ней можно сравнивать текст и картинки в едином пространстве. Фактически это позволяет комбинировать категории и классифицировать изображения по более сложным текстовым описаниям.Приведу пример. Раньше, используя обучение с «учителем», модели отличали в основном лишь односложные и крупные категории вроде «cat» или «dog». А сейчас могут отличать лежащих котов от прыгающих и не только. Можно отличать изображения по стилю, наличию известных личностей на кадре, количеству предметов — многие подобные комбинации работают без дополнительного обучения уже из коробки.

Работает это так: в нейросеть подаётся изображение и текст, а она возвращает векторы изображения и текста. Затем можно посчитать косинусное расстояние и понять, насколько данный текст похож на изображение. В задаче классификации изображений можно выбирать класс по наибольшей близости векторов картинки и текстового описания класса:

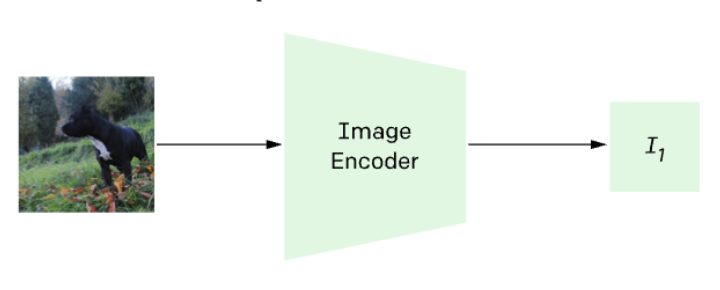
Источник: блог OpenAI

CLIP имеет кодировщик изображений (Image Encoder) и кодировщик текста (Text Encoder), которые преобразуют данные в векторное пространство и предсказывают, какие изображения с какими текстами были соединены.

Источник: блог OpenAI

Для кластеризации изображений в iFunny мы не используем тексты, но используем Image Encoder, который на выходе даёт высокосодержательные векторы, описывающие изображение в многомерном пространстве признаков.

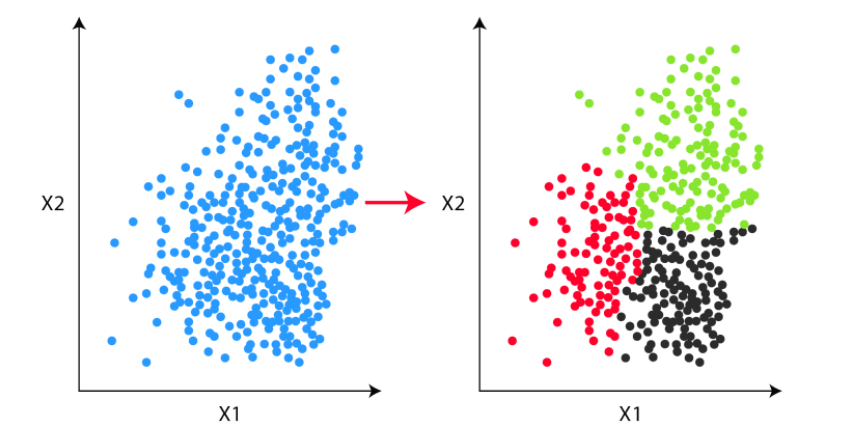
По сути, берём только эту часть из CLIP:

Источник: блог OpenAI

### Выполняем кластеризацию изображений

Нам нужно собрать вектор I(x) для каждого изображения в датасете и затем построить кластеризации этих векторов.

В общих чертах кластеризацию можно изобразить так:



Есть набор точек/векторов в неком пространстве, который нужно разделить на n кластеров. В нашем случае берём векторы из картиночной модели CLIP. На картинке простой пример в двумерном пространстве, но модель из CLIP отдает вектор длиной 512 признаков — а это довольно много для обычных алгоритмов кластеризации.

В таких случаях стоит воспользоваться алгоритмами понижения размерности — например, PCA, TSNE или UMAP. Для кластеризации мы используем библиотеку [HDBSCAN](https://link.springer.com/chapter/10.1007%2F978-3-642-37456-2_14), основанную на базе одноименного алгоритма.

Не буду приводить весь код обучения, так как он специфичен под наши задачи, но опишу необходимые шаги, если захотите повторить кластеризацию на своих данных.

1. После сбора датасета, нужно собрать вектор из модели CLIP по каждой картинке. Для этого устанавливаем библиотеку CLIP и используем только функцию encode\_image. Предварительно готовим изображения в torch.Tensor с помощью Pillow, как это делается в самой библиотеке CLIP.

image = prepare\_pil\_image(image, transform).to(device)

image\_features = self.model.encode\_image(image).cpu().numpy()

На выходе получится массив размера [n, 512], где n — количество изображений в датасете, а 512 — число признаков для каждого изображения из модели CLIP. Затем делим полученный массив на набор для обучения (train\_embeddings) и для теста (test\_embeddings).

2. Далее используем библиотеку [umap-learn](https://pypi.org/project/umap-learn/). С её помощью учим и сокращаем размерность до 2-5 признаков. Гиперпараметры приведены для примера, скорее всего для вашего датасета лучший результат будет при других значениях.

dimension\_model = umap.UMAP(n\_neighbors=70,

n\_epochs=300,

min\_dist=0.03,

n\_components=5,

random\_state=35)

train\_clusterable\_embedding = dimension\_model.fit\_transform(train\_embeddings)

3. Затем на уменьшенных векторах строим кластеризацию. Для этого создаем модель HDBSCAN, обучаем и собираем информацию о том, какая картинка какому кластеру соответствует и с какой вероятностью.

cluster\_model = hdbscan.HDBSCAN(

min\_cluster\_size=1000,

alpha=2.,

cluster\_selection\_method="leaf",

prediction\_data=True

)

cluster\_model.fit\_predict(train\_clusterable\_embedding)

train\_labels, train\_probabilities = hdbscan.approximate\_predict(cluster\_model, train\_clusterable\_embedding)

test\_labels, test\_probabilities = hdbscan.approximate\_predict(cluster\_model, test\_clusterable\_embeddings)

На тестовой выборке (test\_clusterable\_embeddings) можно оценить качество кластеризации на новых данных.

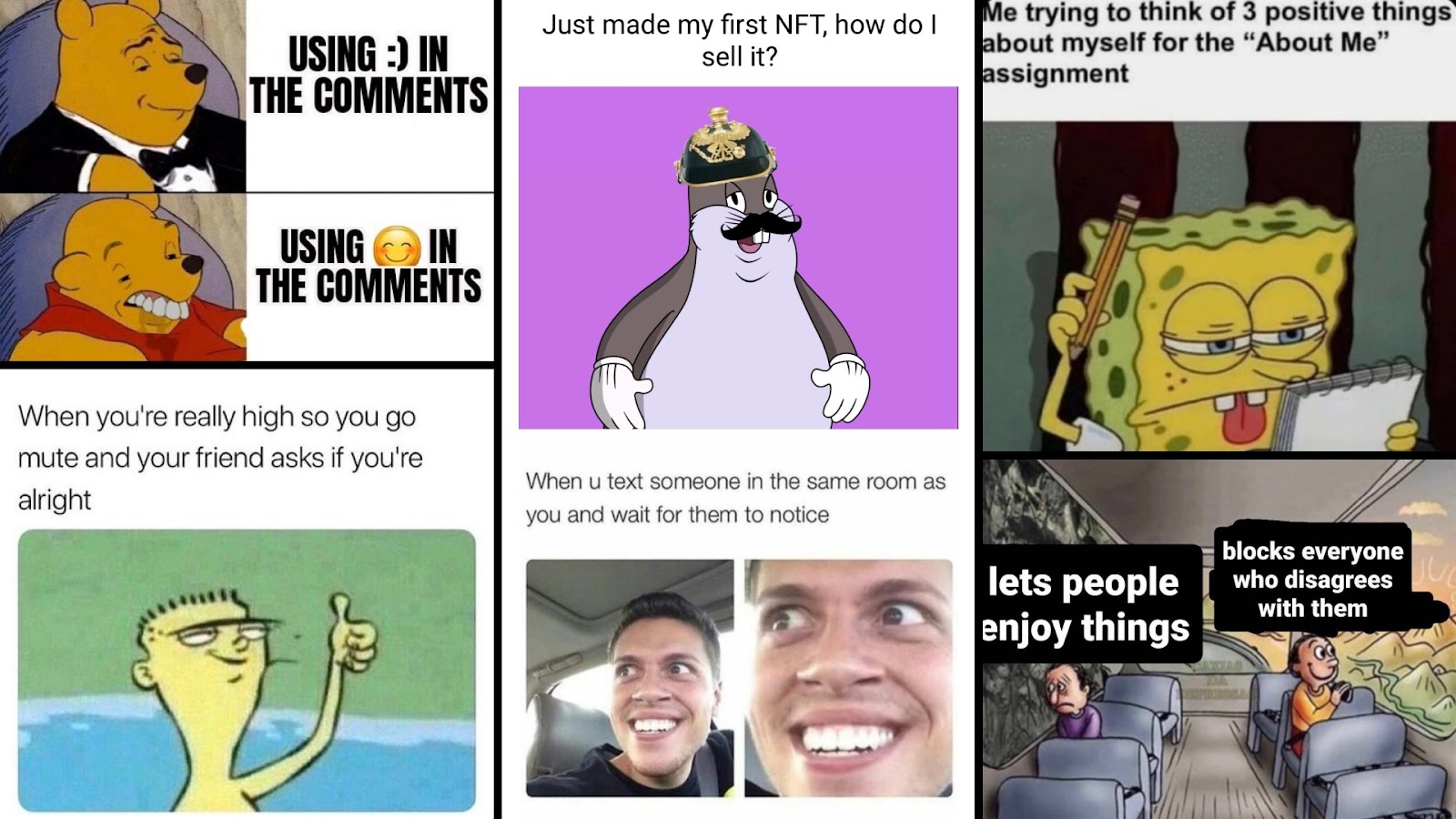
### Результаты

Итого у нас был на руках датасет на 150 000 изображений после удаления дубликатов. По времени вышло примерно так:

* Обучение umap и HDBSCAN — около 2 часов.
* Кластеризация на проде — 1-2 секунды на 1 изображение.

Теперь посмотрим на получившиеся в наших экспериментах кластеры. Получились весьма интересные и показательные категории:

Мемы:



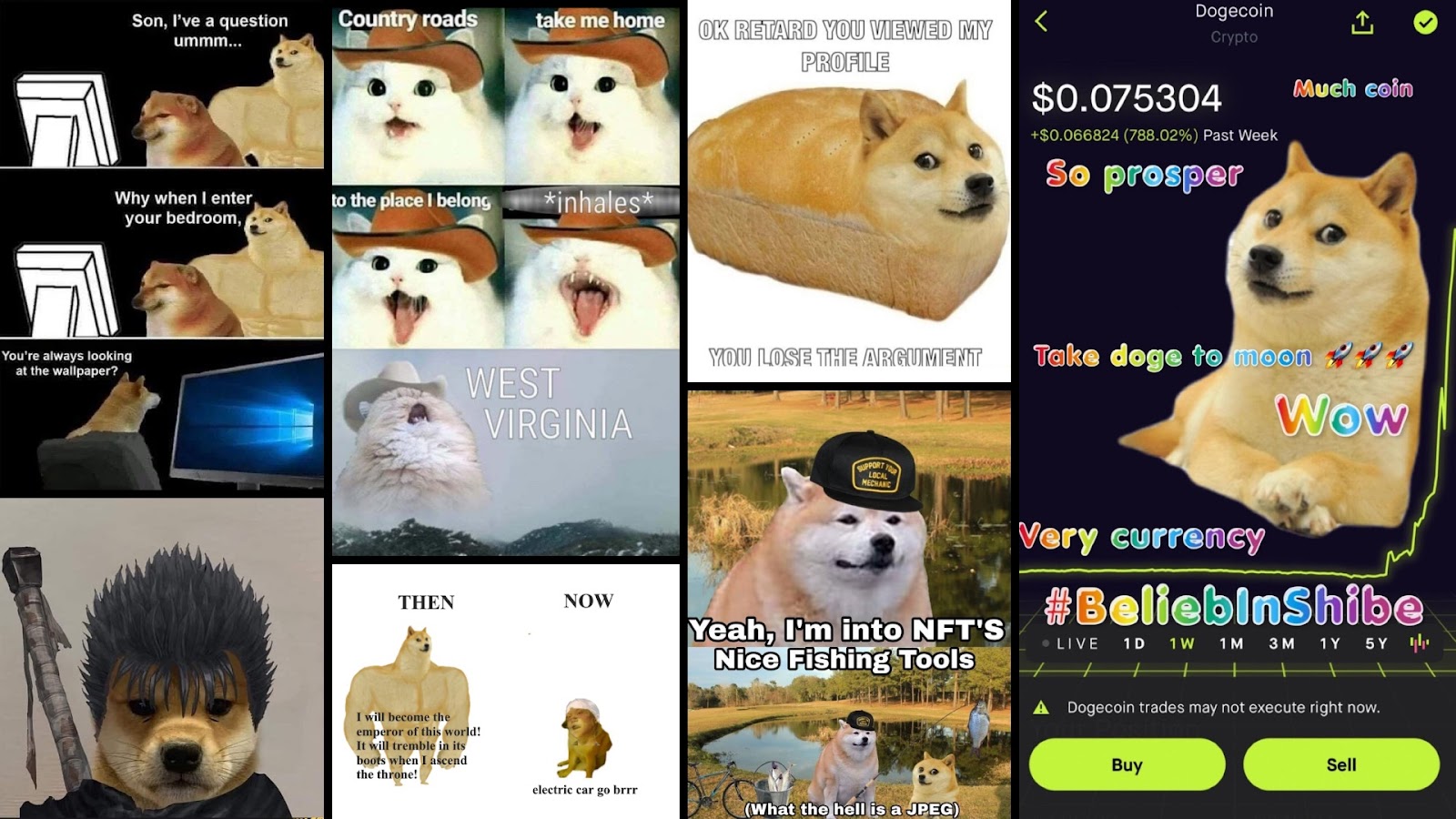
Еда:



Мотивационные картинки:



Также в виде отдельных кластеров выделились: животные, политика, 4chan, аниме, скриншоты твиттера, скриншоты приложения iFunny, шутки про США и другие страны, селфи.Из интересного — кластеризатор выделил в отдельную категорию мемы с Doge, знаменитой собакой породы сиба-ину:



### Минусы подхода

1. Тяжело получить качественное разделение на кластеры. Мы экспериментировали с параметрами UMAP и HDBSCAN, но все равно ошибки остаются, и часть сложного контента попадает в кластер «другое».
2. Количество и логика кластеров может меняться от случайных параметров. Но если вы точно знаете, на какие классы хотите разделить изображения, то возможно лучшим решением будет разметить изображения и применить fine-tuning к той же модели CLIP.
3. Модели требуют времени и памяти на инференсе. Для ускорения в продакшене можно использовать GPU.

### Где использовать подход

Но есть и плюсы. Вот некоторые примеры, где можно применять такой подход кластеризации контента:

1. Для исследование контента. Можно брать контент только одной категории (или от одной группы юзеров) и смотреть, на какие кластеры его стоит разделить. Так можно лучше понять свою аудиторию.
2. Для получения предварительных классов для дальнейшей разметки. Например, если вам нужны только животные, то можно брать их из кластера «животные», но удалять на стадии разметки ошибки кластеризации. Это ускорит время разметки.
3. Для получения признаков для других моделей, например, рекомендаций. Даже с ошибками в кластеризации разделение по большей части выглядит логичным и точным. Таким образом, можно улучшать метрики качества рекомендации контента в ленте.
4. Для быстрых экспериментов с одной категорией контента. Можно взять, скажем, кластер с едой и сделать кнопку поиска только по фотографиям еды.

### Заключение

Модель CLIP отдает качественные признаки изображений, по которым можно получить кластеризацию, действительно отражающую логику и категории внутри вашего датасета. Сочетанием UMAP и HDBSCAN можно легко добиться среднего количества качественных кластеров (10-30) почти без пересечений.

Результаты нас в целом устроили. Эту кластеризацию можно делать без разметки, а затем полученные кластеры использовать в других задачах.

Автор: Олег Соколов

[*Источник*](https://habr.com/ru/post/645191/?utm_source=habrahabr&utm_medium=rss&utm_campaign=645191)

[](https://www.reg.ru/?rlink=reflink-717)

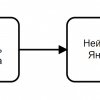
[Версия для печати](https://www.pvsm.ru/python/371244/print/) [Версия для печати](https://www.pvsm.ru/python/371244/print/)

Поделиться…

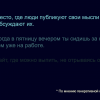
### Рекомендованный контент

[Генерация изображений с помощью echo-printf в 5 строчках кода без библиотек и заголовков](https://www.pvsm.ru/linux/363628)

[Рисуем вместе с CLIP Guided Diffusion HQ](https://www.pvsm.ru/obrabotka-izobrazhenij/369404)

[Нейросети в борьбе с идиотизмом заградительных мер в политике](https://www.pvsm.ru/analiz-i-proektirovanie-sistem/369126)

[Отзывы на «Маркете» стали писать нейросети Яндекса](https://www.pvsm.ru/torgovlya/364418)

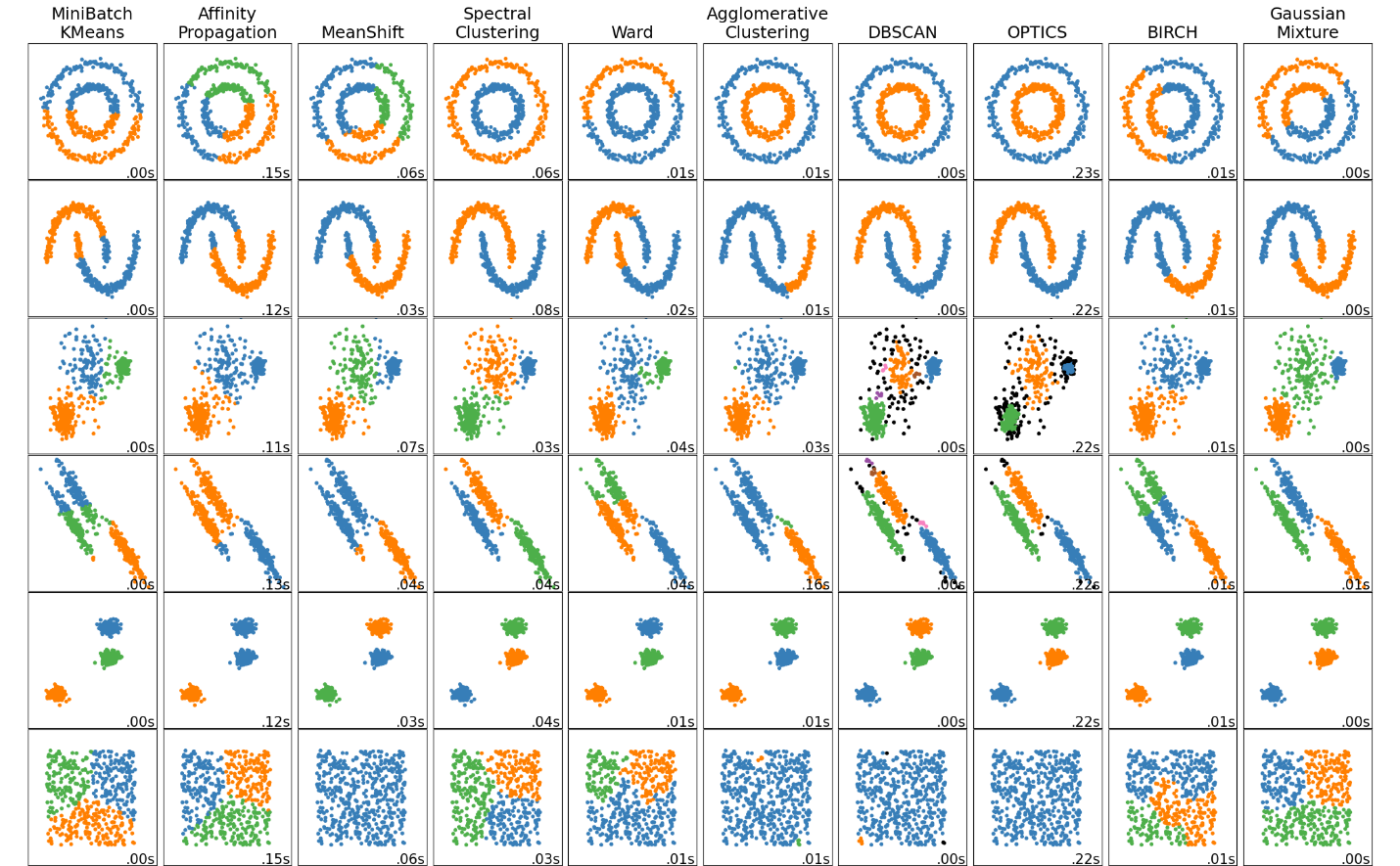
[Как Яндекс применил генеративные нейросети для поиска ответов](https://www.pvsm.ru/yandeks/365114)

[Нейросети для Natural Language Inference: логические умозаключения на русском языке](https://www.pvsm.ru/python/368618)

<https://www.pvsm.ru/python/371244>

1. <https://dzen.ru/list/gadgets/kak-sdelat-klasterizatciiu-dataseta-python>

<https://dzen.ru/video/watch/65464d6186d396253571df9c?rid=163594881.172.1707053394118.46843&referrer_clid=1400&t=3>



##### [Пример: кластеризация K-средних](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_cluster_iris)

[Пример: Сегментация изображения греческих монет по регионам](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_coin_segmentation) [Пример: Демонстрация структурированной иерархической кластеризации Уорда на изображении монет.](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_coin_ward_segmentation) [Пример: квантование цвета с использованием K-средних](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_color_quantization) [Пример: Демонстрация алгоритма кластеризации DBSCAN.](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_dbscan) [Пример: Онлайн-обучение словарю частей лица](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_dict_face_patches) [Пример: агломерация объектов](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_digits_agglomeration) [Пример: различные агломеративные кластеры при двумерном внедрении цифр.](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_digits_linkage) [Пример: Пример векторного квантования](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_face_compress) [Пример: агломерация признаков или одномерный выбор](https://runebook.dev/ru/docs/scikit_learn/auto_examples/cluster/plot_feature_agglomeration_vs_univariate_selection)