Датасет кластеризации

<https://www.kaggle.com/datasets?tags=13304-Clustering>

Classify iris plants into three species in this classic dataset

**Iris dataset**



**Бесплатный вебинар по chatGPT "**Создай нейро-сотрудника с нуля, без опыта программирования"

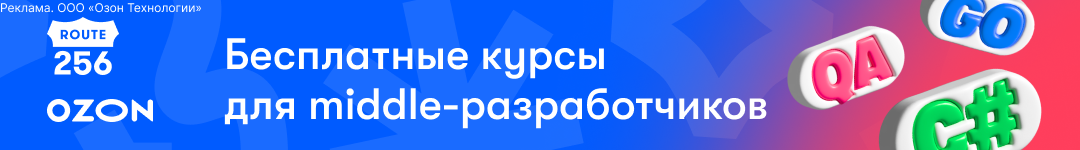
<https://neural-university.ru/lp_chatgpt_neurostaff_web?utm_source=yandex-danil&utm_medium=cpc&utm_campaign=87399880&utm_content=14121239967&utm_term=---autotargeting&roistat=direct107_context_14121239967_---autotargeting&roistat_referrer=yandex.ru&roistat_pos=none_0&yclid=10396120684581093375>

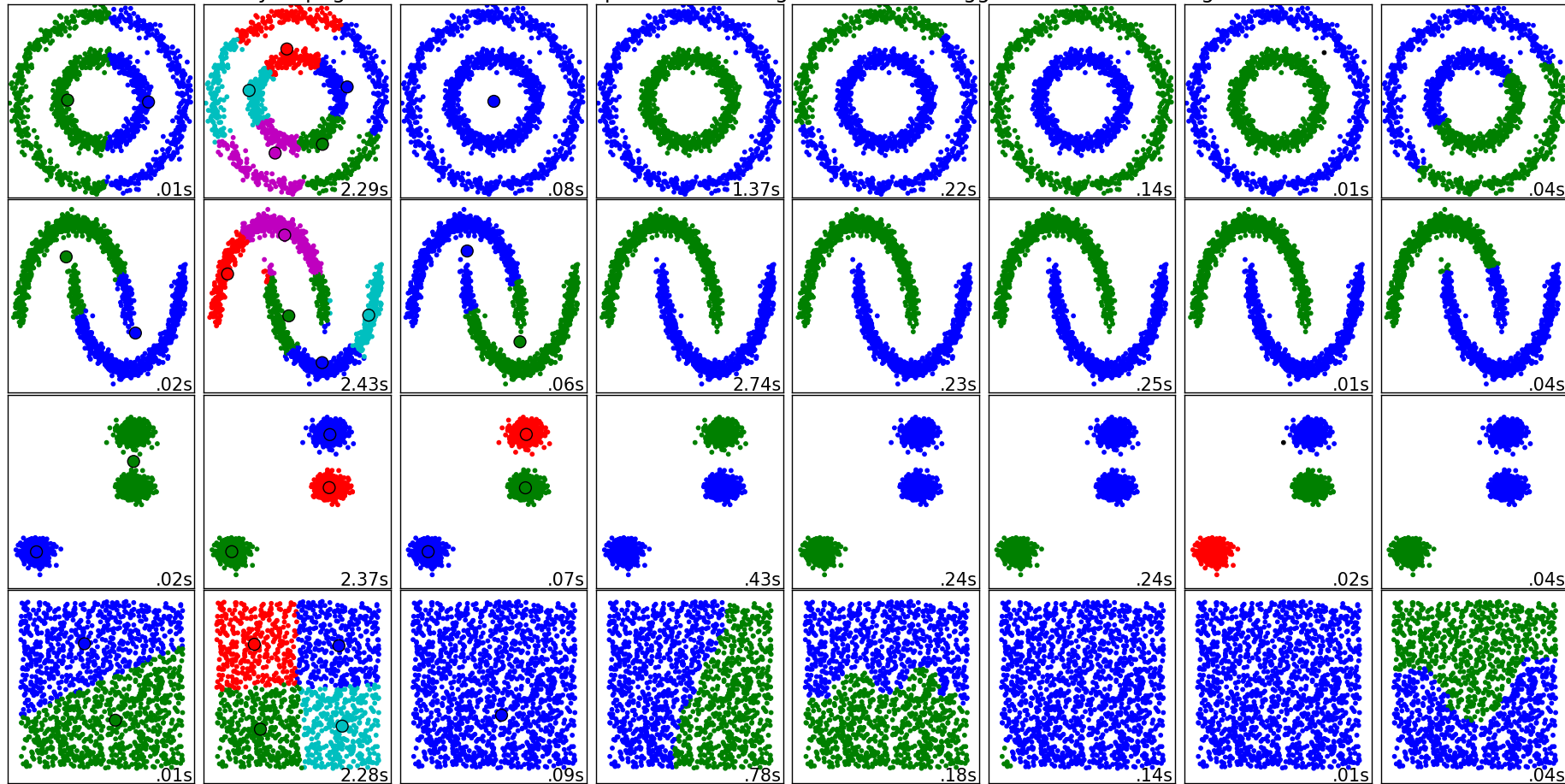
<https://proglib.io/p/unsupervised-ml-with-python?ysclid=ls7gmq8wln390642328>

# Обучение без учителя: 4 метода кластеризации данных на Python

Описаны четыре популярных метода обучения без учителя для кластеризации данных с соответствующими примерами программного кода на Python.

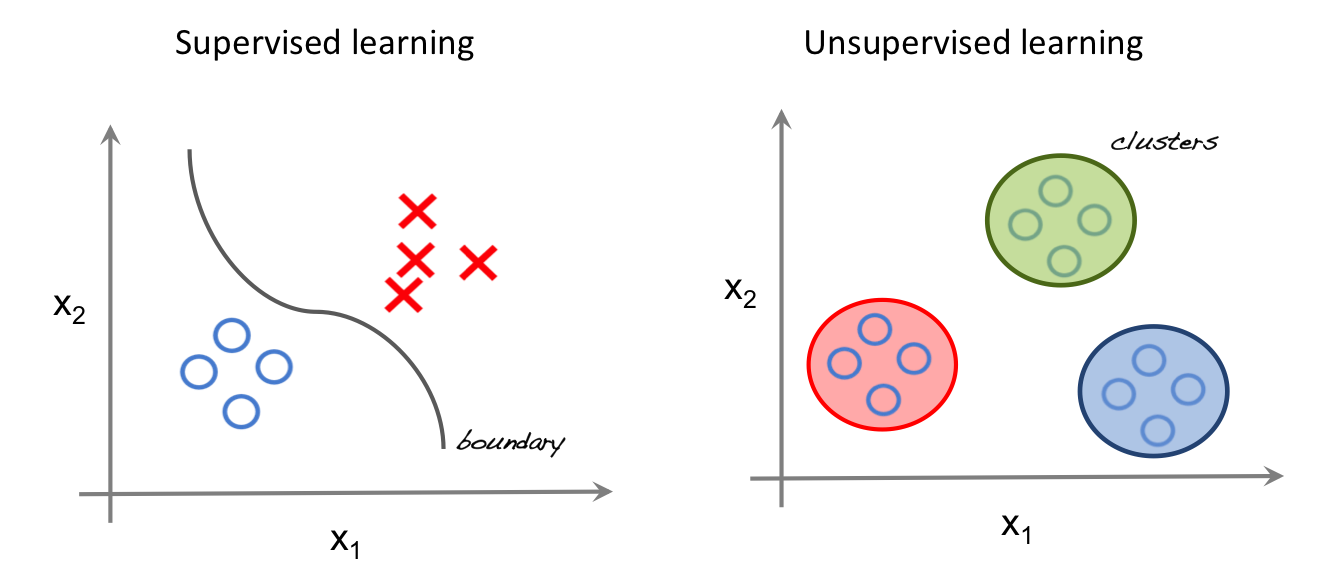
[1](https://proglib.io/p/unsupervised-ml-with-python?ysclid=ls7gmq8wln390642328#comments)

[](https://ads.adfox.ru/280242/clickURL?ad-session-id=8658391707048551840&hash=d331380285fefe64&rand=jzbquiz&rqs=AAAAAAAAAABnfr9lk0muBWFNpp574Snh&pr=evftbwi&p1=cdqsc&ytt=262783279038469&p5=scunr&ybv=0.958942&p2=ggzs&ylv=0.958942&pf=https%3A%2F%2Fs.ozon.ru%2FeWdDQdX)



# Обучение без учителя

Обучение без учителя (unsupervised learning, неконтролируемое обучение) – 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 контролируемом обучении (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.



На левой части изображения представлен пример контролируемого обучения: здесь для того, чтобы найти лучшую функцию, соответствующую представленным точкам, используется метод регрессии. В то же время при неконтролируемом обучении входные данные разделяются на основе представленных характеристик, а предсказание свойств основывается на том, какому кластеру принадлежит пример.

Методы кластеризации данных являются одним из наиболее популярных семейств машинного обучения без учителя. Рассмотрим некоторые из них подробнее.

# Важная терминология

* **Feature**(Особенности): входная переменная, используемая для создания прогнозов.
* **Predictions**(Прогнозы): выходные данные модели при наличии входного примера.
* **Example**(Пример): строка набора данных. Пример обычно содержит один или несколько объектов.
* **Label**(Метки): результат функции.

# Подготовка выборки для кластеризации данных

Для составления прогнозов воспользуемся классическим [набором данных ирисов Фишера](https://ru.wikipedia.org/wiki/Ирисы_Фишера). Датасет представляет набор из 150 записей с пятью атрибутами в следующем порядке: длина чашелистика (sepal length), ширина чашелистика (sepal width), длина лепестка (petal length), ширина лепестка (petal width) и класс, соответствующий одному из трех видов: Iris Setosa, Iris Versicolor или Iris Virginica, обозначенных соответственно 0, 1, 2. Наш алгоритм должен принимать четыре свойства одного конкретного цветка и предсказывать, к какому классу (виду ириса) он принадлежит. Имеющиеся в наборе данных метки можно использовать для оценки качества предсказания.

Для решения задач кластеризации данных в этой статье мы используем Python, библиотеку scikit-learn для загрузки и обработки набора данных и matplotlib для визуализации. Ниже представлен программный код для исследования исходного набора данных.

# Импортируем библиотеки

from sklearn import datasets

import matplotlib.pyplot as plt

# Загружаем набор данных

iris\_df = datasets.load\_iris()

# Методы, доступные для набора данных

print(dir(iris\_df))

# Признаки

print(iris\_df.feature\_names)

# Метки

print(iris\_df.target)

# Имена меток

print(iris\_df.target\_names)

# Разделение набора данных

x\_axis = iris\_df.data[:, 0] # Sepal Length

y\_axis = iris\_df.data[:, 1] # Sepal Width

# Построение

plt.xlabel(iris\_df.feature\_names[0])

plt.ylabel(iris\_df.feature\_names[1])

plt.scatter(x\_axis, y\_axis, c=iris\_df.target)

plt.show()

В результате запуска программы вы увидим следующие текст и изображение.

['DESCR', 'data', 'feature\_names', 'target', 'target\_names']

['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

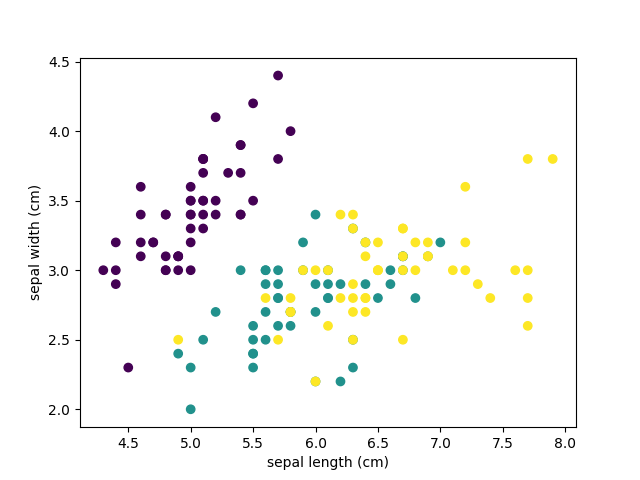
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2

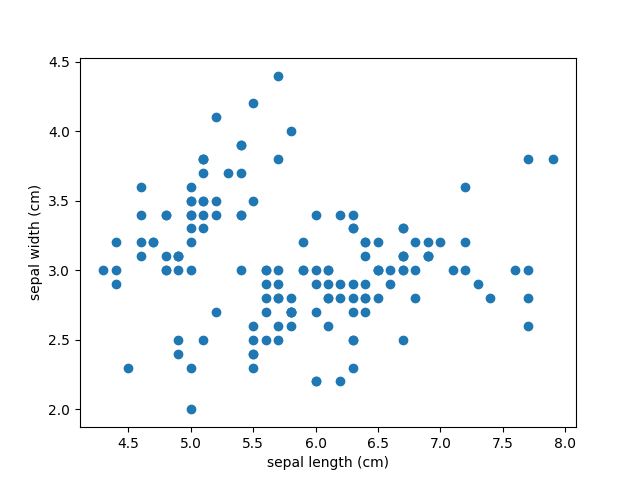
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

2 2]

['setosa' 'versicolor' 'virginica']



На диаграмме фиолетовым цветом обозначен вид Setosa, зеленым – Versicolor и желтым – Virginica. При построении были взяты лишь два признака. Вы можете проанализировать как разделяются классы при других комбинациях параметров.

Цель кластеризации данных состоит в том, чтобы выделить группы примеров с похожими чертами и определить соответствие примеров и кластеров. При этом исходно у нас нет примеров такого разбиения. Это аналогично тому, как если бы в приведенном наборе данных у нас не было меток, как на рисунке ниже.

Наша задача – используя все имеющиеся данные, предсказать соответствие объектов выборки их классам, сформировав таким образом кластеры.

# Метод k-средних

Наиболее популярным алгоритмом кластеризации данных является [метод k-средних](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_k-%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D1%85). Это итеративный алгоритм кластеризации, основанный на минимизации суммарных квадратичных отклонений точек кластеров от центроидов (средних координат) этих кластеров.

Первоначально выбирается желаемое количество кластеров. Поскольку нам известно, что в нашем наборе данных есть 3 класса, установим параметр модели **n\_clusters** равный трем.

Теперь случайным образом из входных данных выбираются три элемента выборки, в соответствие которым ставятся три кластера, в каждый из которых теперь включено по одной точке, каждая при этом является центроидом этого кластера.

Далее ищем ближайшего соседа текущего центроида. Добавляем точку к соответствующему кластеру и пересчитываем положение центроида с учетом координат новых точек.  Алгоритм заканчивает работу, когда координаты каждого центроида перестают меняться. Центроид каждого кластера в результате представляет собой набор значений признаков, описывающих усредненные параметры выделенных классов.

# Импортируем библиотеки

from sklearn import datasets

from sklearn.cluster import KMeans

# Загружаем набор данных

iris\_df = datasets.load\_iris()

# Описываем модель

model = KMeans(n\_clusters=3)

# Проводим моделирование

model.fit(iris\_df.data)

# Предсказание на единичном примере

predicted\_label = model.predict([[7.2, 3.5, 0.8, 1.6]])

# Предсказание на всем наборе данных

all\_predictions = model.predict(iris\_df.data)

# Выводим предсказания

print(predicted\_label)

print(all\_predictions)

Результат:

[1]

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2

2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2

2 0]

При выводе данных нужно понимать, что алгоритм не знает ничего о нумерации классов, и числа 0, 1, 2 – это лишь номера кластеров, определенных в результате работы алгоритма. Так как исходные точки выбираются случайным образом, вывод будет несколько меняться от одного запуска к другому.

Характерной особенностью набора данных ирисов Фишера является то, что один класс (Setosa) легко отделяется от двух остальных. Это заметно и в приведенном примере.

# Иерархическая кластеризация

[Иерархическая кластеризация](https://ru.wikipedia.org/wiki/%D0%98%D0%B5%D1%80%D0%B0%D1%80%D1%85%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%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%D1%8F), как следует из названия, 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.

Результат иерархической кластеризации может быть представлен с помощью дендрограммы. Рассмотрим этот тип кластеризации на примере данных для различных видов зерна.

# Импортируем библиотеки

from scipy.cluster.hierarchy import linkage, dendrogram

import matplotlib.pyplot as plt

import pandas as pd

# Создаем датафрейм

seeds\_df = pd.read\_csv(

"http://qps.ru/jNZUT")

# Исключаем информацию об образцах зерна, сохраняем для дальнейшего использования

varieties = list(seeds\_df.pop('grain\_variety'))

# Извлекаем измерения как массив NumPy

samples = seeds\_df.values

# Реализация иерархической кластеризации при помощи функции linkage

mergings = linkage(samples, method='complete')

# Строим дендрограмму, указав параметры удобные для отображения

dendrogram(mergings,

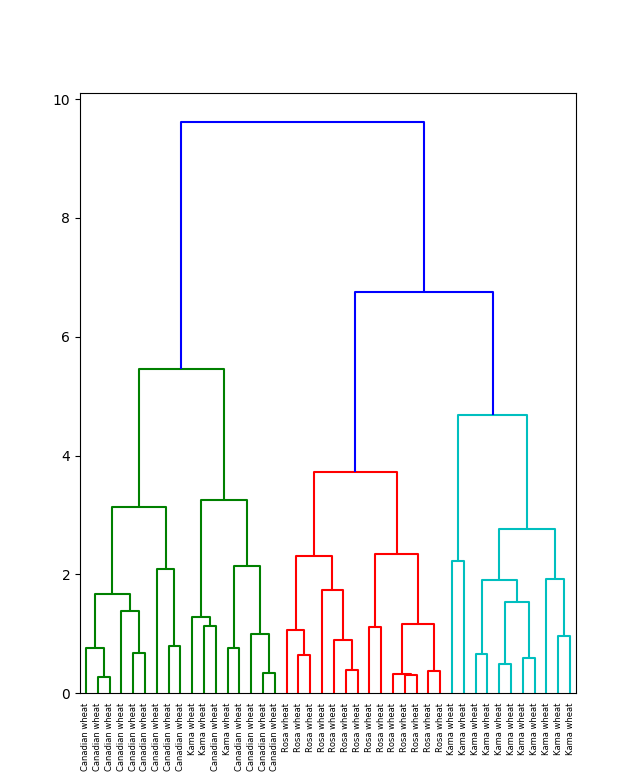
labels=varieties,

leaf\_rotation=90,

leaf\_font\_size=6,

)

plt.show()



Можно видеть, что в результате иерархической кластеризации данных естественным образом произошло разбиение на три кластера, обозначенных на рисунке различным цветом. При этом исходно число кластеров не задавалось.

# 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](https://ru.wikipedia.org/wiki/%D0%92%D1%80%D0%B5%D0%BC%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B0) 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.

# Понижение размерности с методом t-SNE

Метод t-SNE (t-distributed stochastic neighbor embedding) представляет собой один из методов обучения без учителя, используемых для визуализации, например, отображения пространства высокой размерности в двух- или трехмерное пространство. t-SNE расшифровывается как распределенное стохастическое соседнее вложение.

Метод моделирует каждый объект пространства высокой размерности в двух- или трехкоординатную точку таким образом, что близкие по характеристикам элементы данных в многомерном пространстве (например, датасете с большим числом столбцов) проецируются в соседние точки, а разнородные объекты с большей вероятностью моделируются точками, далеко отстоящими друг от друга. Математическое описание работы метода можно найти [здесь](https://habr.com/post/267041/).

Вернемся к примеру с ирисами и посмотрим, как произвести моделирование по этому методу при помощи библиотеки sklearn.

# Импорт библиотек

from sklearn import datasets

from sklearn.manifold import TSNE

import matplotlib.pyplot as plt

# Загрузка датасета

iris\_df = datasets.load\_iris()

# Определяем модель и скорость обучения

model = TSNE(learning\_rate=100)

# Обучаем модель

transformed = model.fit\_transform(iris\_df.data)

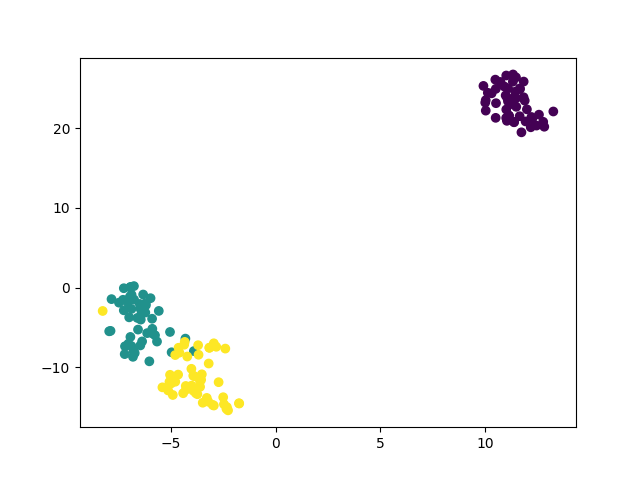
# Представляем результат в двумерных координатах

x\_axis = transformed[:, 0]

y\_axis = transformed[:, 1]

plt.scatter(x\_axis, y\_axis, c=iris\_df.target)

plt.show()



В этом случае каждый экземпляр представлен четырьмя координатами – таким образом, при отображении признаков на плоскость размерность пространства понижается с четырех до двух.

# Метод кластеризации на основе плотности DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise, плотностной алгоритм пространственной кластеризации с присутствием шума) – популярный алгоритм кластеризации, используемый в анализе данных в качестве одной из замен метода k-средних.

Метод не требует предварительных предположений о числе кластеров, но нужно настроить два других параметра: **eps** и **min\_samples**. Данные параметры – это соответственно максимальное расстояние между соседними точками и минимальное число точек в окрестности (количество соседей), когда можно говорить, что эти экземпляры данных образуют один кластер. В scikit-learn есть соответствующие значения параметров по умолчанию, но, как правило, их приходится настраивать самостоятельно.

# Импортируем библиотеки

from sklearn.datasets import load\_iris

import matplotlib.pyplot as plt

from sklearn.cluster import DBSCAN

from sklearn.decomposition import PCA

# Загружаем датасет

iris = load\_iris()

# Определяем модель

dbscan = DBSCAN()

# Обучаем

dbscan.fit(iris.data)

# Уменьшаем размерность при помощи метода главных компонент

pca = PCA(n\_components=2).fit(iris.data)

pca\_2d = pca.transform(iris.data)

# Строим в соответствии с тремя классами

for i in range(0, pca\_2d.shape[0]):

if dbscan.labels\_[i] == 0:

c1 = plt.scatter(pca\_2d[i, 0], pca\_2d[i, 1], c='r', marker='+')

elif dbscan.labels\_[i] == 1:

c2 = plt.scatter(pca\_2d[i, 0], pca\_2d[i, 1], c='g', marker='o')

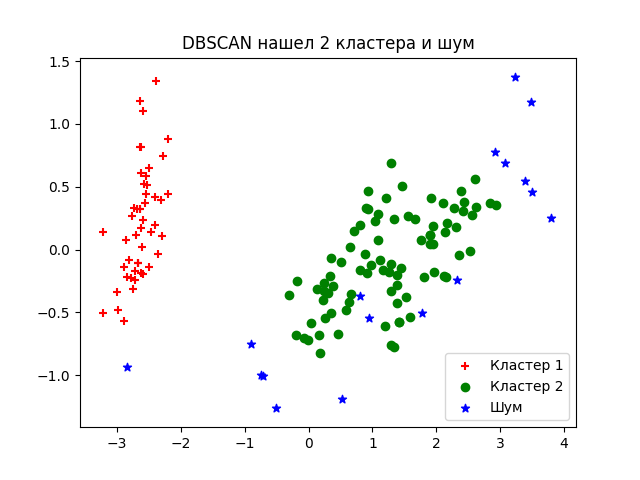
elif dbscan.labels\_[i] == -1:

c3 = plt.scatter(pca\_2d[i, 0], pca\_2d[i, 1], c='b', marker='\*')

plt.legend([c1, c2, c3], ['Кластер 1', 'Кластер 2', 'Шум'])

plt.title('DBSCAN нашел 2 кластера и шум')

plt.show()



Об устройстве алгоритма простыми словами и о математической подноготной можно прочитать [в этой статье](https://habr.com/post/322034/).

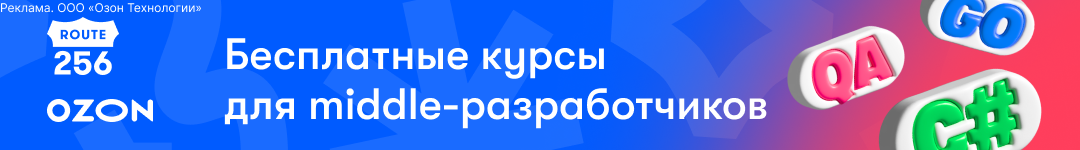
[Источник](https://towardsdatascience.com/unsupervised-learning-with-python-173c51dc7f03)

# Другие материалы по теме:

* [Актуальная математика: самый понятный курс по анализу данных](https://proglib.io/p/current-mathematics/)
* [11 must-have алгоритмов машинного обучения для Data Scientist](https://proglib.io/p/11-ml-algorithms/)
* [27 шпаргалок по машинному обучению и Python в 2017](https://proglib.io/p/ds-cheatsheets/)

[1](https://proglib.io/p/unsupervised-ml-with-python?ysclid=ls7gmq8wln390642328#comments)

[DS](https://proglib.io/tag/DS) [Python](https://proglib.io/tag/Python)

[](https://ads.adfox.ru/280242/clickURL?ad-session-id=8658391707048551840&duid=1707048553613744620&hash=30b6ab346fcf2570&sj=RPKADYiHl6vCMMFBrttN4O-3_PJ-OLwDTjP2rNQZqNBhgGqfWcnlT6rxV8sy&rand=bdzuzbt&rqs=AAAAAAAAAABrfr9lP9qsKBAs-u2QfBlv&pr=evftbwi&p1=cdqsd&ytt=262783279038469&p5=scunr&ybv=0.958942&p2=ggzs&ylv=0.958942&pf=https%3A%2F%2Fs.ozon.ru%2FeWdDQdX)

### [МЕРОПРИЯТИЯ](https://proglib.io/events)

### [Cinimex DATA meetup](https://proglib.io/events/cinimex-data-meetup-sinimeks-2024-01-31)

15 февраля [Санкт-Петербург](https://proglib.io/events?city=%D0%A1%D0%B0%D0%BD%D0%BA%D1%82-%D0%9F%D0%B5%D1%82%D0%B5%D1%80%D0%B1%D1%83%D1%80%D0%B3) [Онлайн](https://proglib.io/events?online=1) [Бесплатно](https://proglib.io/events?free=1)

[+ Показать еще](https://proglib.io/events)

## Комментарии

Начало формы



Конец формы

Популярные По порядку

[https://media.proglib.io/users/avatars/2021/05/01/5ac8246dc80d981d670708c45d0ac387.pngd.boltianski](https://proglib.io/u/d.boltianski)

05 августа 2022

Я правильно понял, что все представленные тут методы умеют кластеризовать исключительно нормативные данные?

Ответить

### ЛУЧШИЕ СТАТЬИ ПО ТЕМЕ

## [ООП на Python: концепции, принципы и примеры реализации](https://proglib.io/p/python-oop)

[Программирование на Python допускает различные методологии, но в его основе лежит объектный подход, поэтому работать в стиле ООП на Python очень просто.](https://proglib.io/p/python-oop)

## [Пишем свою нейросеть: пошаговое руководство](https://proglib.io/p/neural-nets-guide)

[Отличный гайд про нейросеть от теории к практике. Вы узнаете из каких элементов состоит ИНС, как она работает и как ее создать самому.](https://proglib.io/p/neural-nets-guide)

## [Программирование на Python: от новичка до профессионала](https://proglib.io/p/python-from-newbie-to-professional)

[Пошаговая инструкция для всех, кто хочет изучить программирование на Python (или программирование вообще), но не знает, куда сделать первый шаг.](https://proglib.io/p/python-from-newbie-to-professional)

* [О проекте](https://proglib.io/w/3fb4bf84)
* [Реклама](https://proglib.io/w/35def346)
* [Пользовательское соглашение](https://proglib.io/user-agreement)
* [Публичная оферта](https://proglib.io/offer)
* [Политика конфиденциальности](https://proglib.io/privacy-policy)
* [Контакты](https://proglib.io/contact)

Push-уведомления Темная тема

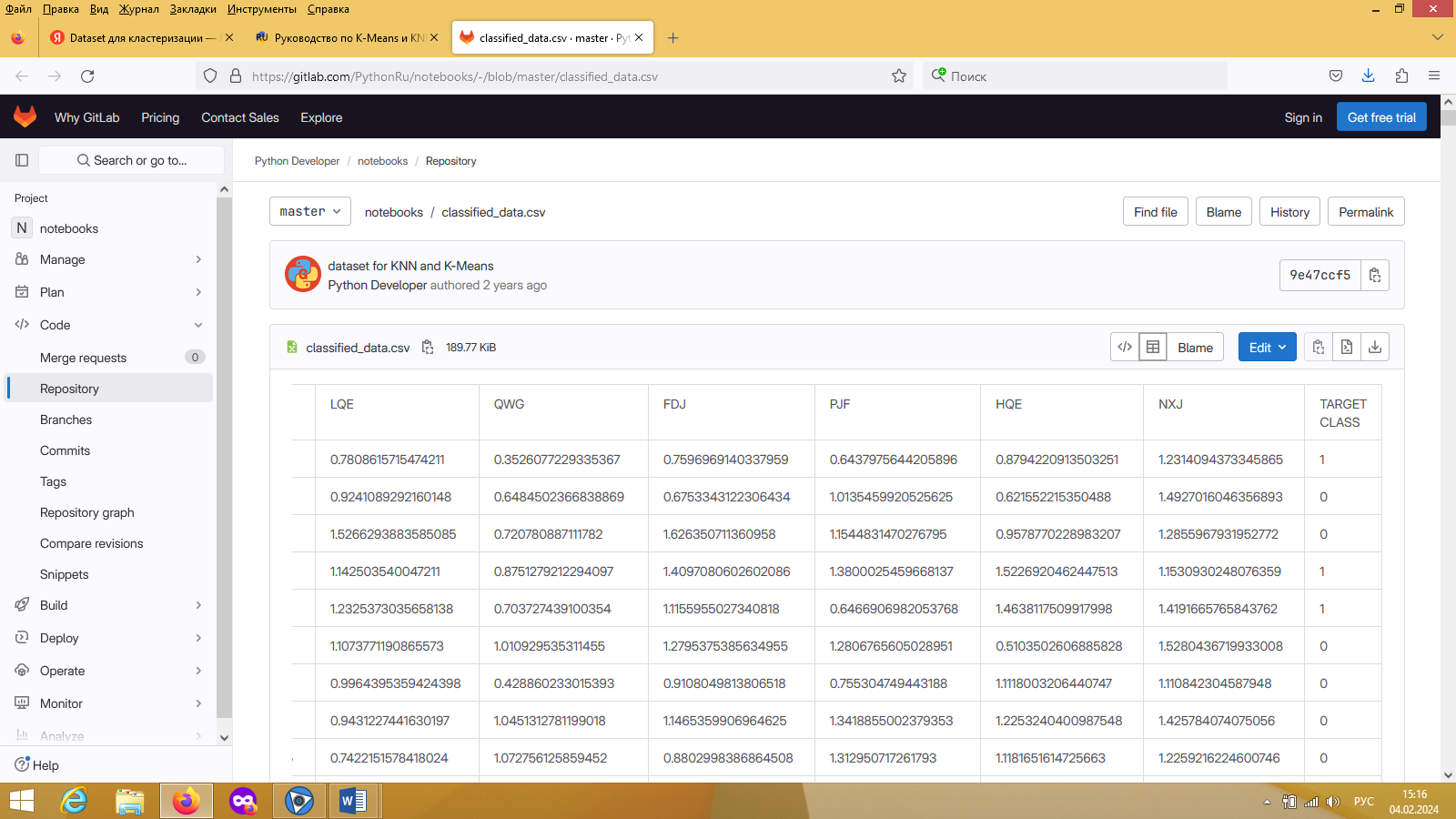
© 2024, Proglib. При копировании мате

<https://gitlab.com/PythonRu/notebooks/-/blob/master/classified_data.csv>

[dataset for KNN and K-Means](https://gitlab.com/PythonRu/notebooks/-/commit/9e47ccf5b1e0da135dc61dc90f716cae6c9ae110)

[Python Developer](https://gitlab.com/PythonRu) authored 2 years ago

**classified\_data.csv** 189.77 KiB



Одно из самых популярных приложений машинного обучения — решение задач классификации. Задачи классификации — это ситуации, когда у вас есть набор данных, и вы хотите классифицировать наблюдения из этого набора в определенную категорию.

Известный пример — спам-фильтр для электронной почты. Gmail использует методы машинного обучения с учителем, чтобы автоматически помещать электронные письма в папку для спама в зависимости от их содержания, темы и других характеристик.

Две модели машинного обучения выполняют большую часть работы, когда дело доходит до задач классификации:

* Метод K-ближайших соседей
* Метод К-средних

Из этого руководства вы узнаете, как применять алгоритмы K-ближайших соседей и K-средних в коде на Python.

## Модели K-ближайших соседей

Алгоритм K-ближайших соседей является одним из самых популярных среди ML-моделей для решения задач классификации.

Обычным упражнением для студентов, изучающих машинное обучение, является применение алгоритма K-ближайших соседей к датасету, категории которого неизвестны. Реальным примером такой ситуации может быть случай, когда вам нужно делать предсказания, используя ML-модели, обученные на секретных правительственных данных.

В этом руководстве вы изучите алгоритм машинного обучения K-ближайших соседей и напишите его реализацию на Python. Мы будем работать с анонимным набором данных, как в описанной выше ситуации.

### Используемый датасет

Первое, что вам нужно сделать, это скачать набор данных, который мы будем использовать в этом руководстве. Вы можете [скачать его на Gitlab](https://gitlab.com/PythonRu/notebooks/-/blob/master/classified_data.csv).

Далее вам нужно переместить загруженный файл с датасетом в рабочий каталог. После этого откройте Jupyter Notebook — теперь мы можем приступить к написанию кода на Python!

### Необходимые библиотеки

Чтобы написать алгоритм K-ближайших соседей, мы воспользуемся преимуществами многих Python-библиотек с открытым исходным кодом, включая NumPy, pandas и scikit-learn.

Начните работу, добавив следующие инструкции импорта:

Копировать

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

### Импорт датасета

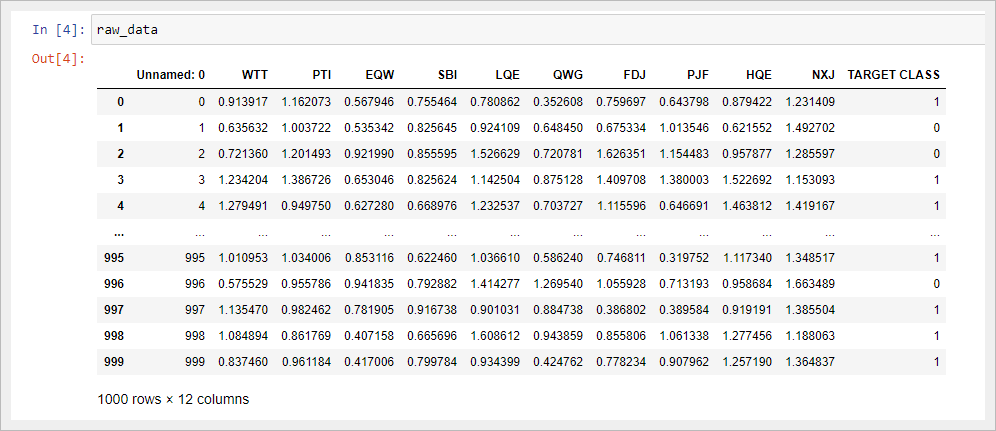
Следующий шаг — добавление файла classified\_data.csv в наш код на Python. Библиотека pandas позволяет довольно просто импортировать данные в DataFrame.

Поскольку датасет хранится в файле csv, мы будем использовать метод read\_csv:

Копировать

raw\_data = pd.read\_csv('classified\_data.csv')

Отобразив полученный DataFrame в Jupyter Notebook, вы увидите, что представляют собой наши данные:



Стоит заметить, что таблица начинается с безымянного столбца, значения которого равны номерам строк DataFrame. Мы можем исправить это, немного изменив команду, которая импортировала наш набор данных в скрипт Python:

Копировать

raw\_data = pd.read\_csv('classified\_data.csv', index\_col = 0)

Затем давайте посмотрим на показатели (признаки), содержащиеся в этом датасете. Вы можете вывести список имен столбцов с помощью следующей инструкции:

Копировать

raw\_data.columns

Получаем:

Index(['WTT', 'PTI', 'EQW', 'SBI', 'LQE', 'QWG', 'FDJ', 'PJF', 'HQE', 'NXJ',

'TARGET CLASS'],

dtype='object')

Поскольку этот набор содержит секретные данные, мы понятия не имеем, что означает любой из этих столбцов. На данный момент достаточно признать, что каждый столбец является числовым по своей природе и поэтому хорошо подходит для моделирования с помощью методов машинного обучения.

### Стандартизация датасета

Поскольку алгоритм K-ближайших соседей делает прогнозы относительно точки данных (семпла), используя наиболее близкие к ней наблюдения, существующий масштаб показателей в датасете имеет большое значение.

Из-за этого специалисты по машинному обучению обычно стандартизируют набор данных, что означает корректировку каждого значения x так, чтобы они находились примерно в одном диапазоне.

К счастью, библиотека scikit-learn позволяет сделать это без особых проблем.

Для начала нам нужно будет импортировать класс StandardScaler из scikit-learn. Для этого добавьте в свой скрипт Python следующую команду:

Копировать

from sklearn.preprocessing import StandardScaler

Этот класс во многом похож на классы LinearRegression и LogisticRegression, которые мы использовали ранее в этом курсе. Нам нужно создать экземпляр StandardScaler, а затем использовать этот объект для преобразования наших данных.

Во-первых, давайте создадим экземпляр класса StandardScaler с именем scaler следующей инструкцией:

Копировать

scaler = StandardScaler()

Теперь мы можем обучить scaler на нашем датасете, используя метод fit:

Копировать

scaler.fit(raw\_data.drop('TARGET CLASS', axis=1))

Теперь мы можем применить метод transform для стандартизации всех признаков, чтобы они имели примерно одинаковый масштаб. Мы сохраним преобразованные семплы в переменной scaled\_features:

Копировать

scaled\_features = scaler.transform(raw\_data.drop('TARGET CLASS', axis=1))

В качестве результата мы получили массив NumPy со всеми точками данных из датасета, но нам желательно преобразовать его в формат DataFrame библиотеки pandas.

К счастью, сделать это довольно легко. Мы просто обернем переменную scaled\_features в метод pd.DataFrame и назначим этот DataFrame новой переменной scaled\_data с соответствующим аргументом для указания имен столбцов:

Копировать

scaled\_data = pd.DataFrame(scaled\_features, columns = raw\_data.drop('TARGET CLASS', axis=1).columns)

Теперь, когда мы импортировали наш датасет и стандартизировали его показатели, мы готовы разделить этот набор данных на обучающую и тестовую выборки.

### Разделение датасета на обучающие и тестовые данные

Мы будем использовать функцию train\_test\_split библиотеки scikit-learn в сочетании с распаковкой списка для создания обучающих и тестовых датасетов из нашего набора секретных данных.

Во-первых, вам нужно импортировать train\_test\_split из модуля model\_validation библиотеки scikit-learn:

Копировать

from sklearn.model\_selection import train\_test\_split

Затем нам необходимо указать значения x и y, которые будут переданы в функцию train\_test\_split.

Значения x представляют собой DataFrame scaled\_data, который мы создали ранее. Значения y хранятся в столбце "TARGET CLASS" нашей исходной таблицы raw\_data.

Вы можете создать эти переменные следующим образом:

Копировать

x = scaled\_data

y = raw\_data['TARGET CLASS']

Затем вам нужно запустить функцию train\_test\_split, используя эти два аргумента и разумный test\_size. Мы будем использовать test\_size 30%, что дает следующие параметры функции:

Копировать

x\_training\_data, x\_test\_data, y\_training\_data, y\_test\_data = train\_test\_split(x, y, test\_size = 0.3)

Теперь, когда наш датасет разделен на данные для обучения и данные для тестирования, мы готовы приступить к обучению нашей модели!

### Обучение модели K-ближайших соседей

Начнем с импорта KNeighborsClassifier из scikit-learn:

Копировать

from sklearn.neighbors import KNeighborsClassifier

Затем давайте создадим экземпляр класса KNeighborsClassifier и назначим его переменной model.

Для этого требуется передать параметр n\_neighbors, который равен выбранному вами значению K алгоритма K-ближайших соседей. Для начала укажем n\_neighbors = 1:

Копировать

model = KNeighborsClassifier(n\_neighbors = 1)

Теперь мы можем обучить нашу модель, используя метод fit и переменные x\_training\_data и y\_training\_data:

Копировать

model.fit(x\_training\_data, y\_training\_data)

Теперь давайте сделаем несколько прогнозов с помощью полученной модели!

### Делаем предсказания с помощью алгоритма K-ближайших соседей

Способ получения прогнозов на основе алгоритма K-ближайших соседей такой же, как и у моделей линейной и логистической регрессий, построенных нами ранее в этом курсе: для предсказания достаточно вызвать метод predict, передав в него переменную x\_test\_data.

В частности, вот так вы можете делать предсказания и присваивать их переменной predictions:

Копировать

predictions = model.predict(x\_test\_data)

Давайте посмотрим, насколько точны наши прогнозы, в следующем разделе этого руководства.

### Оценка точности нашей модели

В руководстве по логистической регрессии мы видели, что scikit-learn поставляется со встроенными функциями, которые упрощают измерение эффективности классификационных моделей машинного обучения.

Для начала импортируем в наш отчет две функции classification\_report и confusion\_matrix:

Копировать

from sklearn.metrics import classification\_report

from sklearn.metrics import confusion\_matrix

Теперь давайте поработаем с каждой из них по очереди, начиная с classification\_report. С ее помощью вы можете создать отчет следующим образом:

Копировать

print(classification\_report(y\_test\_data, predictions))

Полученный вывод:

precision recall f1-score support

0 0.92 0.91 0.91 148

1 0.91 0.92 0.92 152

accuracy 0.91 300

macro avg 0.91 0.91 0.91 300

weighted avg 0.91 0.91 0.91 300

Точно так же вы можете сгенерировать матрицу ошибок:

Копировать

print(confusion\_matrix(y\_test\_data, predictions))

# Вывод:

# [[134 14]

# [ 12 140]]

Глядя на такие метрики производительности, похоже, что наша модель уже достаточно эффективна. Но ее еще можно улучшить.

В следующем разделе будет показано, как мы можем повлиять на работу модели K-ближайших соседей, выбрав более подходящее значение для K.

### Выбор оптимального значения для K с помощью метода «Локтя»

В этом разделе мы будем использовать метод «локтя», чтобы выбрать оптимальное значение K для нашего алгоритма K-ближайших соседей.

Метод локтя включает в себя итерацию по различным значениям K и выбор значения с наименьшей частотой ошибок при применении к нашим тестовым данным.

Для начала создадим пустой список error\_rates. Мы пройдемся по различным значениям K и добавим их частоту ошибок в этот список.

Копировать

error\_rates = []

Затем нам нужно создать цикл Python, который перебирает различные значения K, которые мы хотим протестировать, и на каждой итерации выполняет следующее:

* Создает новый экземпляр класса KNeighborsClassifier из scikit-learn.
* Тренирует эту модель, используя наши обучающие данные.
* Делает прогнозы на основе наших тестовых данных.
* Вычисляет долю неверных предсказаний (чем она ниже, тем точнее наша модель).

Реализация описанного цикла для значений K от 1 до 100:

Копировать

for i in np.arange(1, 101):

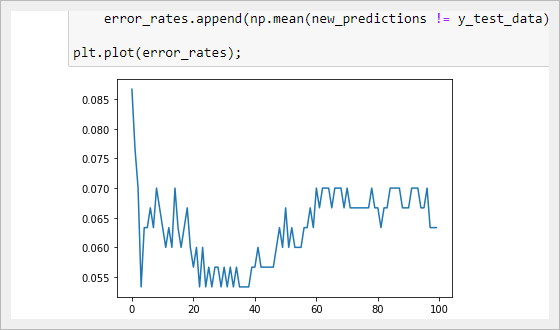
new\_model = KNeighborsClassifier(n\_neighbors = i)

new\_model.fit(x\_training\_data, y\_training\_data)

new\_predictions = new\_model.predict(x\_test\_data)

error\_rates.append(np.mean(new\_predictions != y\_test\_data))

Давайте визуализируем, как изменяется частота ошибок при различных K, используя matplotlib — plt.plot(error\_rates):



Как видно из графика, мы достигаем минимальной частоты ошибок при значении K, равном приблизительно 35. Это означает, что 35 является подходящим выбором для K, который сочетает в себе простоту и точность предсказаний.

Вы можете найти весь код в ноутбуке на GitLab:  
<https://gitlab.com/PythonRu/notebooks/-/blob/master/sklearn_kmeans_and_knn.ipynb>

## Модели кластеризации методом K-средних

Алгоритм кластеризации K-средних обычно является первой моделью машинного обучения без учителя, которую изучают студенты.

Он позволяет специалистам по машинному обучению создавать группы точек данных со схожими количественными характеристиками в датасете. Это полезно для решения таких задач, как формирование клиентских сегментов или определение городских районов с высоким уровнем преступности.

В этом разделе вы узнаете, как создать свой первый алгоритм кластеризации K-средних на Python.

### Используемый датасет

В этом руководстве мы будем использовать набор данных, созданный с помощью scikit-learn.

Давайте импортируем функцию make\_blobs из scikit-learn, чтобы сгенерировать необходимые данные. Откройте Jupyter Notebook и запустите свой скрипт Python со следующей инструкцией:

Копировать

from sklearn.datasets import make\_blobs

Теперь давайте воспользуемся функцией make\_blobs, чтобы получить фиктивные данные!

В частности, вот как вы можете создать набор данных из 200 семплов, который имеет 2 показателя и 4 кластерных центров. Стандартное отклонение для каждого кластера будет равно 1.8.

Копировать

raw\_data = make\_blobs(

n\_samples = 200,

n\_features = 2,

centers = 4,

cluster\_std = 1.8

)

Если вы выведите объект raw\_data, то заметите, что на самом деле он представляет собой кортеж Python. Первым его элементом является массив NumPy с 200 наблюдениями. Каждое наблюдение содержит 2 признака (как мы и указали в нашей функции make\_blobs).

Теперь, когда наши данные созданы, мы можем перейти к импорту других необходимых библиотек с открытым исходным кодом в наш скрипт Python.

### Импортируемые библиотеки

В этом руководстве будет использоваться ряд популярных библиотек Python с открытым исходным кодом, включая pandas, NumPy и matplotlib. Продолжим написание скрипта, добавив следующие импорты:

Копировать

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

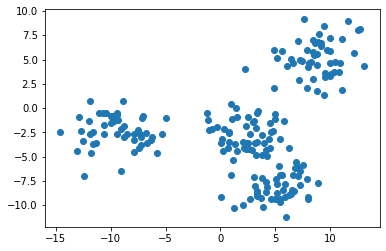
Первая группа библиотек в этом блоке кода предназначена для работы с большими наборами данных. Вторая группа предназначена для визуализации результатов.

Теперь перейдем к созданию визуального представления нашего датасета.

### Визуализация датасета

В функции make\_blobs мы указали, что в нашем наборе данных должно быть 4 кластерных центра. Лучший способ убедиться, что все действительно так, — это создать несколько простых точечных диаграмм.

Для этого мы воспользуемся функцией plt.scatter, передав в нее все значения из первого столбца нашего набора данных в качестве X и соответствующие значения из второго столбца в качестве Y:



Примечание: ваш датасет будет отличаться от моего, поскольку его данные сгенерированы случайным образом.

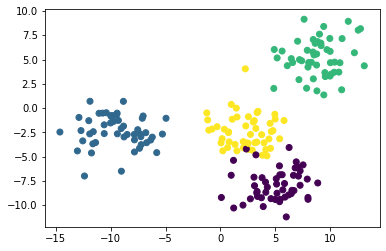
Представленное изображение, похоже, указывает на то, что в нашем датасете всего три кластера. Нам так кажется потому, что два кластера расположены очень близко друг к другу.

Чтобы исправить это, нужно сослаться на второй элемент кортежа raw\_data, представляющий собой массив NumPy: он содержит индекс кластера, которому принадлежит каждое наблюдение.

Если при построении мы будем использовать уникальный цвет для каждого кластера, то мы легко различим 4 группы наблюдений. Вот код для этого:

Копировать

plt.scatter(raw\_data[0][:,0], raw\_data[0][:,1], c=raw\_data[1]);



Теперь мы видим, что в нашем наборе данных есть четыре уникальных кластера. Давайте перейдем к построению нашей модели на основе метода K-средних на Python!

### Создание и обучение модели кластеризации K-средних

Для того, чтобы начать использовать метод K-средних, импортируем соответствующий класс из scikit-learn. Для этого добавьте в свой скрипт следующую команду:

Копировать

from sklearn.cluster import KMeans

Затем давайте создадим экземпляр класса KMeans с параметром n\_clusters=4 и присвоим его переменной model:

Копировать

model = KMeans(n\_clusters=4)

Теперь обучим нашу модель, вызвав на ней метод fit и передав первый элемент нашего кортежа raw\_data:

Копировать

model.fit(raw\_data[0])

В следующем разделе мы рассмотрим, как делать прогнозы с помощью модели кластеризации K-средних.

Прежде чем двигаться дальше, я хотел бы указать на одно различие, которое вы, возможно заметили, между процессом построения модели, используя метод K-средних (он является алгоритмом кластеризации без учителя), и алгоритмами машинного обучения с учителем, с которыми мы работали ранее в данном курсе.

Оно заключается в том, что нам не нужно разбивать набор данных на обучающую и тестовую выборки. Это важное различие, так как вам никогда не нужно разделять таким образом датасет при построении моделей машинного обучения без учителя!

### Применяем нашу модель кластеризации K-средних для получения предсказаний

Специалисты по машинному обучению обычно используют алгоритмы кластеризации, чтобы делать два типа прогнозов:

* К какому кластеру принадлежит каждая точка данных.
* Где находится центр каждого кластера.

Теперь, когда наша модель обучена, мы можем легко сгенерировать такие предсказания.

Во-первых, давайте предскажем, к какому кластеру принадлежит каждая точка данных. Для этого обратимся к атрибуту labels\_ из объекта model с помощью оператора точки:

Копировать

model.labels\_

Таким образом мы получаем массив NumPy с прогнозами для каждого семпла:

array([3, 2, 1, 1, 3, 2, 1, 0, 0, 0, 0, 0, 3, 2, 1, 2, 1, 3, 3, 3, 3, 1,

1, 1, 2, 2, 3, 1, 3, 2, 1, 0, 1, 3, 1, 1, 3, 2, 0, 1, 3, 2, 3, 3,

0, 3, 2, 2, 3, 0, 0, 0, 1, 1, 2, 1, 2, 0, 1, 2, 2, 1, 2, 3, 0, 3,

0, 2, 0, 0, 1, 1, 0, 3, 2, 3, 2, 0, 1, 2, 0, 2, 0, 3, 3, 0, 3, 3,

0, 3, 2, 3, 2, 1, 2, 1, 3, 3, 2, 2, 0, 2, 0, 2, 0, 2, 1, 0, 0, 2,

3, 2, 1, 2, 3, 0, 1, 1, 1, 3, 2, 2, 3, 3, 2, 1, 3, 0, 0, 3, 0, 1,

1, 3, 1, 0, 1, 1, 0, 3, 2, 0, 3, 0, 1, 2, 1, 2, 1, 2, 2, 3, 2, 1,

0, 2, 3, 3, 2, 0, 1, 3, 3, 2, 0, 0, 0, 3, 1, 2, 0, 2, 3, 3, 2, 2,

3, 1, 0, 1, 2, 3, 1, 3, 1, 1, 0, 2, 1, 0, 2, 1, 3, 1, 3, 3, 1, 3,

0, 3])

Чтобы узнать, где находится центр каждого кластера, аналогичным способом обратитесь к атрибуту cluster\_centers\_:

Копировать

model.cluster\_centers\_

Получаем двумерный массив NumPy, содержащий координаты центра каждого кластера. Он будет выглядеть так:

array([[ 5.2662658 , -8.20493969],

[-9.39837945, -2.36452588],

[ 8.78032251, 5.1722511 ],

[ 2.40247618, -2.78480268]])

### Визуализация точности предсказаний модели

Последнее, что мы сделаем в этом руководстве, — это визуализируем точность нашей модели. Для этого можно использовать следующий код:

Копировать

f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(10,6))

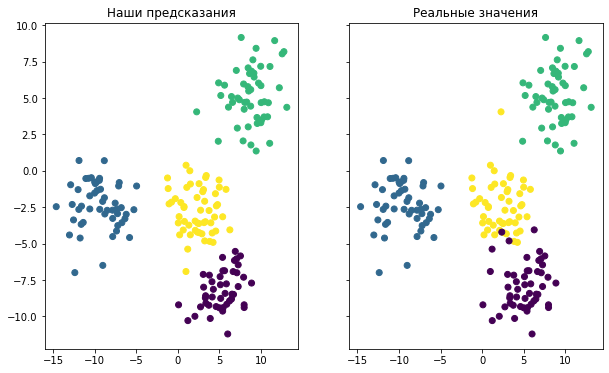
ax1.set\_title('Наши предсказания')

ax1.scatter(raw\_data[0][:,0], raw\_data[0][:,1],c=model.labels\_)

ax2.set\_title('Реальные значения')

ax2.scatter(raw\_data[0][:,0], raw\_data[0][:,1],c=raw\_data[1]);

Он генерирует две точечные диаграммы. Первая показывает кластеры, используя фактические метки из нашего датасета, а вторая основана на предсказаниях, сделанных нашей моделью. Вот как выглядит результат:



Хотя окраска двух графиков разная, вы можете видеть, что созданная модель довольно хорошо справилась с предсказанием кластеров в нашем наборе данных. Вы также можете заметить, что модель не идеальна: точки данных, расположенные на краях кластеров, в некоторых случаях классифицируются неверно.

И последнее, о чем следует упомянуть, говоря об оценке точности нашей модели. В этом примере мы знали, к какому кластеру принадлежит каждое наблюдение, потому что мы сами создали этот набор данных.

Такая ситуация встречается крайне редко. Метод К-средних обычно применяется, когда не известны ни количество кластеров, ни присущие им качества. Таким образом, специалисты по машинному обучению используют данный алгоритм, чтобы обнаружить закономерности в датасете, о которых они еще ничего не знают.

Вы можете найти весь код в ноутбуке на GitLab:  
<https://gitlab.com/PythonRu/notebooks/-/blob/master/sklearn_kmeans_and_knn.ipynb>

## Заключительные мысли

В этом руководстве вы научились создавать модели машинного обучения на Python, используя методы K-ближайших соседей и K-средних.

Вот краткое изложение того, что вы узнали о моделях K-ближайших соседей в Python:

* Как засекреченные данные являются распространенным инструментом для обучения студентов решению задач K-ближайших соседей.
* Почему важно стандартизировать набор данных при построении моделей K-ближайших соседей.
* Как разделить датасет на обучающую и тестирующую выборки с помощью функции train\_test\_split.
* Способ обучить вашу первую модель K-ближайших соседей и как получить сделанные ее прогнозы.
* Как оценить эффективность модели K-ближайших соседей.
* Метод локтя для выбора оптимального значения K в модели K-ближайших соседей.

А вот краткое изложение того, что вы узнали о моделях кластеризации K-средних в Python:

* Как сгенерировать фиктивные данные в scikit-learn с помощью функции make\_blobs.
* Как создать и обучить модель кластеризации K-средних.
* То, что ML-методы без учителя не требуют, чтобы вы разделяли датасет на данные для обучения и данные для тестирования.
* Как создать и обучить модель кластеризации K-средних с помощью scikit-learn.
* Как визуализировать эффективность алгоритма K-средних, если вы изначально владеете информацией о кластерах.
* ТЕГИ
* [Data Science](https://pythonru.com/tag/data-science)

[Facebook](https://www.facebook.com/sharer.php?u=https%3A%2F%2Fpythonru.com%2Furoki%2Fsklearn-kmeans-i-knn)

[Telegram](https://telegram.me/share/url?url=https://pythonru.com/uroki/sklearn-kmeans-i-knn&text=Алгоритмы%20K-ближайших%20соседей%20и%20K-средних%20на%20Python)

[Twitter](https://twitter.com/intent/tweet?text=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B+K-%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85+%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9+%D0%B8+K-%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D1%85+%D0%BD%D0%B0+Python&url=https%3A%2F%2Fpythonru.com%2Furoki%2Fsklearn-kmeans-i-knn&via=PythonRu)

[VK](https://vk.com/share.php?url=https://pythonru.com/uroki/sklearn-kmeans-i-knn)

Максим

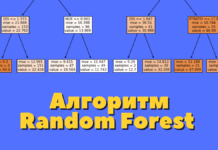
Я создал этот блог в 2018 году, чтобы распространять полезные учебные материалы, документации и уроки на русском. На сайте опубликовано множество статей по основам python и библиотекам, уроков для начинающих и примеров написания программ.

Мои контакты: [Почта](mailto:admin@pythonru.com)

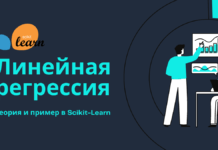
#### Статьи по теме

[](https://pythonru.com/uroki/docker-django)

### [Запуск Django-приложения в Docker контейнере](https://pythonru.com/uroki/docker-django)

[](https://pythonru.com/uroki/sklearn-random-forest)

### [Алгоритм классификации Random Forest на Python](https://pythonru.com/uroki/sklearn-random-forest)

[](https://pythonru.com/uroki/linear-regression-sklearn)

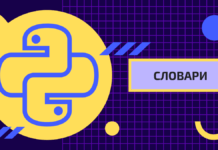
### [Полное руководство по линейной регрессии в Scikit-Learn](https://pythonru.com/uroki/linear-regression-sklearn)

[](https://pythonru.com/uroki/django-rest-api)

### [Создаем API блога на Django REST Framefork](https://pythonru.com/uroki/django-rest-api)

[](https://pythonru.com/uroki/django-static)

### [Настройка и подключение статических файлов в Django](https://pythonru.com/uroki/django-static)

[](https://pythonru.com/uroki/python-dlja-nachinajushhih/slovari)

### [8. Словари](https://pythonru.com/uroki/python-dlja-nachinajushhih/slovari)

#### Содержание

* [Модели K-ближайших соседей](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8-K-%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85-%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9)
  + [Используемый датасет](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D0%B5%D0%BC%D1%8B%D0%B9-%D0%B4%D0%B0%D1%82%D0%B0%D1%81%D0%B5%D1%82)
  + [Необходимые библиотеки](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%9D%D0%B5%D0%BE%D0%B1%D1%85%D0%BE%D0%B4%D0%B8%D0%BC%D1%8B%D0%B5-%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B8)
  + [Импорт датасета](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%98%D0%BC%D0%BF%D0%BE%D1%80%D1%82-%D0%B4%D0%B0%D1%82%D0%B0%D1%81%D0%B5%D1%82%D0%B0)
  + [Стандартизация датасета](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%B4%D0%B0%D1%82%D0%B0%D1%81%D0%B5%D1%82%D0%B0)
  + [Разделение датасета на обучающие и тестовые данные](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5-%D0%B4%D0%B0%D1%82%D0%B0%D1%81%D0%B5%D1%82%D0%B0-%D0%BD%D0%B0-%D0%BE%D0%B1%D1%83%D1%87%D0%B0%D1%8E%D1%89%D0%B8%D0%B5-%D0%B8-%D1%82%D0%B5%D1%81%D1%82%D0%BE%D0%B2%D1%8B%D0%B5-%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5)
  + [Обучение модели K-ближайших соседей](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8-K-%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85-%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9)
  + [Делаем предсказания с помощью алгоритма K-ближайших соседей](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%94%D0%B5%D0%BB%D0%B0%D0%B5%D0%BC-%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D0%BA%D0%B0%D0%B7%D0%B0%D0%BD%D0%B8%D1%8F-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D0%B0-K-%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85-%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9)
  + [Оценка точности нашей модели](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%9E%D1%86%D0%B5%D0%BD%D0%BA%D0%B0-%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D0%B8-%D0%BD%D0%B0%D1%88%D0%B5%D0%B9-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8)
  + [Выбор оптимального значения для K с помощью метода «Локтя»](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%92%D1%8B%D0%B1%D0%BE%D1%80-%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE-%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D1%8F-%D0%B4%D0%BB%D1%8F-K-%D1%81-%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D1%8C%D1%8E-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0-%C2%AB%D0%9B%D0%BE%D0%BA%D1%82%D1%8F%C2%BB)
* [Модели кластеризации методом K-средних](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8-%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-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BC-K-%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D1%85)
  + [Используемый датасет](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D0%B5%D0%BC%D1%8B%D0%B9-%D0%B4%D0%B0%D1%82%D0%B0%D1%81%D0%B5%D1%82-2)
  + [Импортируемые библиотеки](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%98%D0%BC%D0%BF%D0%BE%D1%80%D1%82%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B5-%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B8)
  + [Визуализация датасета](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%B4%D0%B0%D1%82%D0%B0%D1%81%D0%B5%D1%82%D0%B0)
  + [Создание и обучение модели кластеризации K-средних](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%A1%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D0%B8-%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8-%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-K-%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D1%85)
  + [Применяем нашу модель кластеризации K-средних для получения предсказаний](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%9F%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D1%8F%D0%B5%D0%BC-%D0%BD%D0%B0%D1%88%D1%83-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-%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-K-%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D1%85-%D0%B4%D0%BB%D1%8F-%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F-%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D0%BA%D0%B0%D0%B7%D0%B0%D0%BD%D0%B8%D0%B9)
  + [Визуализация точности предсказаний модели](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%92%D0%B8%D0%B7%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D0%B8-%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D0%BA%D0%B0%D0%B7%D0%B0%D0%BD%D0%B8%D0%B9-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8)
* [Заключительные мысли](https://pythonru.com/uroki/sklearn-kmeans-i-knn?ysclid=ls7grvkqh5898825945#%D0%97%D0%B0%D0%BA%D0%BB%D1%8E%D1%87%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5-%D0%BC%D1%8B%D1%81%D0%BB%D0%B8)

#### Новое в блоге

* [Нахождение делителей числа с помощью Python](https://pythonru.com/primery/nahozhdenie-delitelej-chisla-s-pomoshhju-python)
* [Лямбда-функции и анонимные функции в Python](https://pythonru.com/osnovy/ljambda-funkcii-i-anonimnye-funkcii-v-python)
* [Когда стоит использовать yield вместо return в Python](https://pythonru.com/osnovy/kogda-stoit-ispolzovat-yield-vmesto-return-v-python)
* [Как извлечь кубический корень в Python](https://pythonru.com/osnovy/kak-izvlech-kubicheskij-koren-v-python)
* [О проекте](https://pythonru.com/o-proekte)
* [Политика конфиденциальности](https://pythonru.com/politika-konfidencialnosti)
* [Правообладателям](https://pythonru.com/pravoobladateljam)
* [Контакты](https://pythonru.com/kontakty)
* [Хостинг⇲](https://pythonru.com/go/reg-ru)

© PythonRu 2018-2021 — Образовательный блог о Python

<https://data.world/datasets/clustering>

# There are 102 **clustering** datasets available on data.world.

People are adding new clustering datasets everyday to data.world. We have clustering datasets covering topics from social media, gaming and more. We hope you find the clustering data you're looking for to include in your next big project.

# There are 102 **clustering** datasets available on data.world.

People are adding new clustering datasets everyday to data.world. We have clustering datasets covering topics from social media, gaming and more. We hope you find the clustering data you're looking for to include in your next big project.

Задание для студента

<https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/kmeans-clustering-and-pca-on-wine-dataset/>

Кластеризация K-Means - это неконтролируемый алгоритм машинного обучения, который группирует немаркированный набор данных в разные кластеры. Целью статьи является изучение основ и работы кластеризации k-mean, а также реализации.

**Содержание**

* [Что такое кластеризация K-means?](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/k-means-clustering-introduction/#what-is-kmeans-clustering)
* [Какова цель кластеризации k-средних?](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/k-means-clustering-introduction/#what-is-the-objective-of-kmeans-clustering)
* [Как работает кластеризация k-средств?](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/k-means-clustering-introduction/#how-kmeans-clustering-works)
* [Реализация кластеризации K-средних в Python](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/k-means-clustering-introduction/#implementation-of-kmeans-clustering-in-python)

## Что такое кластеризация K-means?

[Неконтролируемое машинное обучение](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/supervised-unsupervised-learning/) - это процесс обучения компьютера использованию немаркированных, неклассифицированных данных и предоставления алгоритму возможности оперировать этими данными без надзора. Без какой-либо предварительной подготовки данных задача машины в этом случае заключается в организации несортированных данных в соответствии с параллелями, шаблонами и вариациями.

## Какова цель кластеризации k-средних?

Целью [кластеризации](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/clustering-in-machine-learning/) является разделение совокупности или набора точек данных на несколько групп, чтобы точки данных внутри каждой группы были более сопоставимы друг с другом и отличались от точек данных внутри других групп. По сути, это группировка вещей, основанная на том, насколько они похожи и отличаются друг от друга.

## Как работает кластеризация k-средств?

Нам предоставляется набор данных элементов с определенными характеристиками и значениями для этих функций (например, вектор). Задача состоит в том, чтобы разделить эти элементы на группы. Для достижения этой цели мы будем использовать алгоритм K-средних, алгоритм обучения без учителя. ‘K’ в названии алгоритма представляет количество групп / кластеров, на которые мы хотим классифицировать наши элементы.

(Это поможет, если вы будете представлять элементы как точки в n-мерном пространстве). Алгоритм распределит элементы по k группам или кластерам сходства. Чтобы вычислить это сходство, мы будем использовать евклидово расстояние в качестве измерения.

Алгоритм работает следующим образом:

1. Сначала мы случайным образом инициализируем k точек, называемых средними или центроидами кластера.
2. Мы классифицируем каждый элемент по ближайшему среднему значению и обновляем координаты среднего значения, которые являются средними значениями элементов, классифицированных в этом кластере на данный момент.
3. Мы повторяем процесс в течение заданного количества итераций, и в конце у нас есть наши кластеры.

Упомянутые выше "баллы” называются средними, потому что они представляют собой средние значения элементов, классифицированных в них. Чтобы инициализировать эти средства, у нас есть множество вариантов. Интуитивно понятный метод заключается в инициализации средств для случайных элементов в наборе данных. Другой метод заключается в инициализации средних значений случайными значениями между границами набора данных (если для объекта x, элементы имеют значения в [0,3], мы инициализируем средние значения значениями для x в [0,3]).

Приведенный выше алгоритм в псевдокоде выглядит следующим образом:

Initialize k means with random values  
--> For a given number of iterations:  
   
 --> Iterate through items:  
   
 --> Find the mean closest to the item by calculating   
 the euclidean distance of the item with each of the means  
   
 --> Assign item to mean  
   
 --> Update mean by shifting it to the average of the items in that cluster

## Реализация кластеризации K-средних в Python

### Пример 1

#### Импортируйте необходимые библиотеки

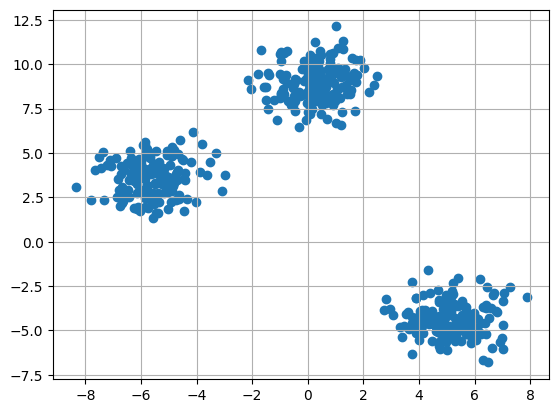
Мы импортируем Numpy для статистических вычислений, Matplotlib для построения графика и make\_blobs из sklearn.datasets.

|  |
| --- |
| import numpy as np  import matplotlib.pyplot as plt  from sklearn.datasets import make\_blobs |

#### Создайте пользовательский набор данных с помощью make\_blobs и нанесите его на график

|  |
| --- |
| X,y = make\_blobs(n\_samples = 500,n\_features = 2,centers = 3,random\_state = 23)    fig = plt.figure(0)  plt.grid(True)  plt.scatter(X[:,0],X[:,1])  plt.show() |

**Результат**:



Набор данных для кластеризации

#### Инициализируем случайные центроиды

Код инициализирует три кластера для кластеризации K-средств. Он задает случайное начальное значение и генерирует случайные центры кластеров в пределах указанного диапазона, а также создает пустой список точек для каждого кластера.

|  |
| --- |
| k = 3    clusters = {}  np.random.seed(23)    for idx in range(k):      center = 2\*(2\*np.random.random((X.shape[1],))-1)      points = []      cluster = {          'center' : center,          'points' : []      }        clusters[idx] = cluster    clusters |

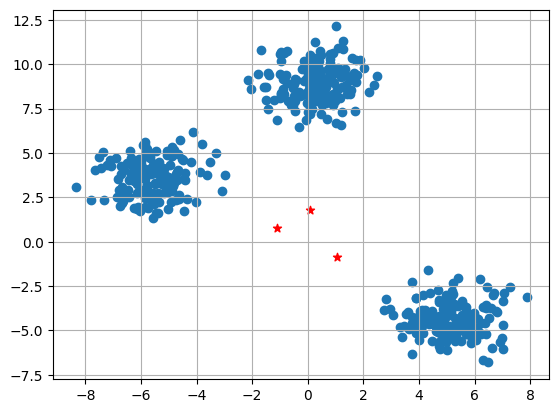
**Результат:**

{0: {'center': array([0.06919154, 1.78785042]), 'points': []},  
 1: {'center': array([ 1.06183904, -0.87041662]), 'points': []},  
 2: {'center': array([-1.11581855, 0.74488834]), 'points': []}}

#### Постройте центр случайной инициализации с точками данных

|  |
| --- |
| plt.scatter(X[:,0],X[:,1])  plt.grid(True)  for i in clusters:      center = clusters[i]['center']      plt.scatter(center[0],center[1],marker = '\*',c = 'red')  plt.show() |

**Результат**:



Точки данных со случайным центром

На графике отображается точечная диаграмма точек данных (X[:, 0], X [:, 1]) с линиями сетки. На ней также отмечены начальные центры скоплений (красные звездочки), созданные для кластеризации K-средних.

#### Определите евклидово расстояние

|  |
| --- |
| def distance(p1,p2):      return np.sqrt(np.sum((p1-p2)\*\*2)) |

#### Создайте функцию для назначения и обновления центра кластера

На этапе E точки данных назначаются ближайшему центру кластера, а на этапе M центры кластера обновляются на основе среднего значения назначенных точек при кластеризации K-средних.

|  |
| --- |
| #Implementing E step  def assign\_clusters(X, clusters):      for idx in range(X.shape[0]):          dist = []            curr\_x = X[idx]            for i in range(k):              dis = distance(curr\_x,clusters[i]['center'])              dist.append(dis)          curr\_cluster = np.argmin(dist)          clusters[curr\_cluster]['points'].append(curr\_x)      return clusters    #Implementing the M-Step  def update\_clusters(X, clusters):      for i in range(k):          points = np.array(clusters[i]['points'])          if points.shape[0] > 0:              new\_center = points.mean(axis =0)              clusters[i]['center'] = new\_center                clusters[i]['points'] = []      return clusters |

#### Шаг 7: Создайте функцию для прогнозирования кластера для точек данных

|  |
| --- |
| def pred\_cluster(X, clusters):      pred = []      for i in range(X.shape[0]):          dist = []          for j in range(k):              dist.append(distance(X[i],clusters[j]['center']))          pred.append(np.argmin(dist))      return pred |

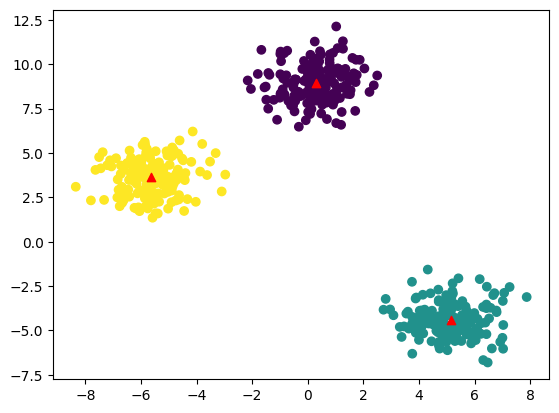
#### Назначайте, обновляйте и прогнозируйте центр кластера

|  |
| --- |
| clusters = assign\_clusters(X,clusters)  clusters = update\_clusters(X,clusters)  pred = pred\_cluster(X,clusters) |

#### Постройте точки данных с их прогнозируемым центром кластера

|  |
| --- |
| plt.scatter(X[:,0],X[:,1],c = pred)  for i in clusters:      center = clusters[i]['center']      plt.scatter(center[0],center[1],marker = '^',c = 'red')  plt.show() |

**Результат**:



K- означает кластеризацию

На графике показаны точки данных, окрашенные в соответствии с их предсказанными кластерами. Красные маркеры представляют обновленные центры кластеров после E-M шагов в алгоритме кластеризации K-средних.

### Пример 2

#### Импортируйте необходимые библиотеки

|  |
| --- |
| import pandas as pd  import numpy as np  import seaborn as sns  import matplotlib.pyplot as plt  import matplotlib.cm as cm  from sklearn.datasets import load\_iris  from sklearn.cluster import KMeans |

#### Загрузите набор данных

|  |
| --- |
| X, y = load\_iris(return\_X\_y=True) |

#### Метод локтя

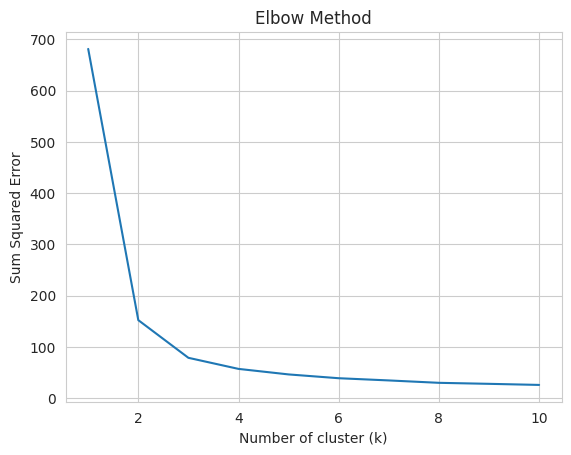
Поиск идеального количества групп для разделения данных является базовым этапом в любом неконтролируемом алгоритме. Одним из наиболее распространенных методов определения этого идеального значения k является локтевой подход.

|  |
| --- |
| #Find optimum number of cluster  sse = [] #SUM OF SQUARED ERROR  for k in range(1,11):      km = KMeans(n\_clusters=k, random\_state=2)      km.fit(X)      sse.append(km.inertia\_) |

#### Постройте коленный график, чтобы найти оптимальное количество кластеров

|  |
| --- |
| sns.set\_style("whitegrid")  g=sns.lineplot(x=range(1,11), y=sse)    g.set(xlabel ="Number of cluster (k)",        ylabel = "Sum Squared Error",        title ='Elbow Method')    plt.show() |

**Результат:**



Метод локтя

Из приведенного выше графика мы можем наблюдать, что при k = 2 и k = 3 ситуация, подобная локтевому сгибу. Итак, мы рассматриваем K = 3

#### Постройте модель кластеризации K Означает

|  |
| --- |
| kmeans = KMeans(n\_clusters = 3, random\_state = 2)  kmeans.fit(X) |

**Результат:**

KMeans  
KMeans(n\_clusters=3, random\_state=2)

#### Найдите центр кластера

|  |
| --- |
| kmeans.cluster\_centers\_ |

**Результат:**

array([[5.006 , 3.428 , 1.462 , 0.246 ],  
 [5.9016129 , 2.7483871 , 4.39354839, 1.43387097],  
 [6.85 , 3.07368421, 5.74210526, 2.07105263]])

#### Спрогнозируйте кластерную группу:

|  |
| --- |
| pred = kmeans.fit\_predict(X)  pred |

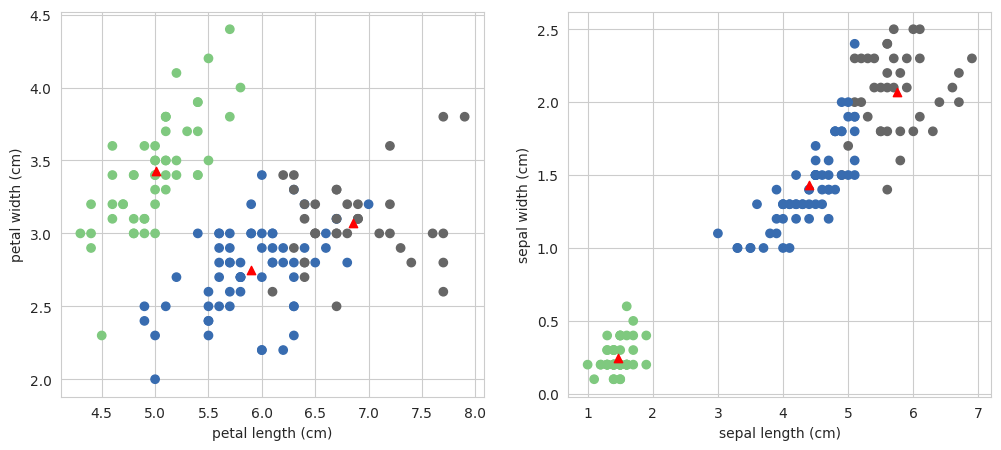
**Результат:**

array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 2, 2, 2, 1, 2, 2, 2,  
 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 2, 1, 2, 1, 2, 2, 1, 1, 2, 2, 2, 2,  
 2, 1, 2, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 1], dtype=int32)

#### Постройте центр кластера с точками данных

|  |
| --- |
| plt.figure(figsize=(12,5))  plt.subplot(1,2,1)  plt.scatter(X[:,0],X[:,1],c = pred, cmap=cm.Accent)  plt.grid(True)  for center in kmeans.cluster\_centers\_:      center = center[:2]      plt.scatter(center[0],center[1],marker = '^',c = 'red')  plt.xlabel("petal length (cm)")  plt.ylabel("petal width (cm)")    plt.subplot(1,2,2)  plt.scatter(X[:,2],X[:,3],c = pred, cmap=cm.Accent)  plt.grid(True)  for center in kmeans.cluster\_centers\_:      center = center[2:4]      plt.scatter(center[0],center[1],marker = '^',c = 'red')  plt.xlabel("sepal length (cm)")  plt.ylabel("sepal width (cm)")  plt.show() |

**Результат:**



K- означает кластеризацию

На подзаголовке слева отображается зависимость длины лепестка от ширины лепестка, точки данных окрашены кластерами, а красные маркеры обозначают K-средние центры кластеров. На подзаголовке справа аналогично показана длина чашелистика и ширина чашелистика.

## Заключение

В заключение, кластеризация K-средств - это мощный неконтролируемый алгоритм машинного обучения для группировки немаркированных наборов данных. Его цель - разделить данные на кластеры, сделав похожие точки данных частью одной группы. Алгоритм инициализирует центроиды кластера и итеративно присваивает точки данных ближайшему центроиду, обновляя центроиды на основе среднего значения точек в каждом кластере.

## Часто задаваемые вопросы (FAQs)

### 1. Что такое кластеризация k-средств для анализа данных?

K-средние - это метод секционирования, который разделяет набор данных на ‘k’ различных, неперекрывающихся подмножеств (кластеров) на основе сходства, направленный на минимизацию различий внутри каждого кластера.

### 2.Каков пример k-средств в реальной жизни?

Сегментация клиентов в маркетинге, где k-means группирует клиентов на основе покупательского поведения, позволяя компаниям адаптировать маркетинговые стратегии для различных сегментов.

### 3. К какому типу данных относится модель кластеризации k-средних?

K-средние хорошо работают с числовыми данными, где концепция расстояния между точками данных имеет смысл. Обычно она применяется к непрерывным переменным.

### 4.Is K-средние, используемые для прогнозирования?

K-means в основном используется для кластеризации похожих точек данных. Он не предсказывает метки для новых данных; он присваивает их существующим кластерам на основе сходства.

### 5.Какова цель кластеризации k-средств?

Цель состоит в том, чтобы разделить данные на "k" кластеров, минимизируя внутрикластерную дисперсию. Он стремится сформировать группы, в которых точки данных внутри каждого кластера больше похожи друг на друга, чем в других кластерах.

Не упустите свой шанс оседлать волну информационной революции! Каждая отрасль достигает новых высот, используя мощь данных. Оттачивайте свои навыки и станьте частью самого модного тренда 21 века.

Погрузитесь в будущее технологий - изучите [полную программу машинного обучения и обработки данных](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/courses/data-science-live?utm_source=geeksforgeeks&utm_medium=article_bottom_text&utm_campaign=courses) от GeeksforGeeks и будьте на шаг впереди.

**Последнее обновление :** 21 декабря 2023 г.

Понравилась статья

Сохранить статью

[Предыдущая страница](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/music-recommendation-system-using-machine-learning/?ref=lbp)

[Система рекомендаций по музыке с использованием машинного обучения](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/music-recommendation-system-using-machine-learning/?ref=lbp)

[Далее](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/?ref=lbp)

[Сегментация изображений с использованием K означает кластеризацию](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/?ref=lbp)

Поделитесь своими мыслями в комментариях

Похожие чтения

[Анализ тестовых данных с использованием кластеризации K-Means в Python](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/analysis-of-test-data-using-k-means-clustering-in-python/)

[ML | Определите оптимальное значение K в кластеризации K-средних](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/ml-determine-the-optimal-value-of-k-in-k-means-clustering/)

[ML | Мини-пакет K-означает алгоритм кластеризации](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/ml-mini-batch-k-means-clustering-algorithm/)

[Сжатие изображений с использованием кластеризации K-средств](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/image-compression-using-k-means-clustering/)

[K- означает кластеризацию в программировании на R](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/k-means-clustering-in-r-programming/)

[Разница между K означает и иерархической кластеризацией](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/difference-between-k-means-and-hierarchical-clustering/)

[Сегментация изображений с использованием K означает кластеризацию](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/)

[K- означает кластеризацию с помощью SciPy](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/k-means-clustering-with-scipy/)

[K означает кластеризацию с использованием Weka](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/k-means-clustering-using-weka/)

[Кластеризация текстовых документов с использованием K-средств в Scikit Learn](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/clustering-text-documents-using-k-means-in-scikit-learn/)

Полные руководства

[Ускоренный курс Python](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/python-crash-course/)

[Руководство по Python API: начало работы с API](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/python-api-tutorial-getting-started-with-apis/)

[Расширенные руководства по Python](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/advanced-python-tutorials/)

[Учебное пособие по автоматизации Python](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/python-automation/)

[OpenAI Python API - Полное руководство](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/openai-python-api/)



**GeeksforGeeks**

Теги статей :

* [Машинное обучение](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/category/data-science/machine-learning/)
* [Python](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/www.geeksforgeeks.org/category/programming-language/python/)

Практические теги :

* [Машинное обучение](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/practice.geeksforgeeks.org/explore?category=Machine)
* [python](https://translated.turbopages.org/proxy_u/en-ru.ru.cbb4aacf-65bf8788-6f27b08e-74722d776562/https/practice.geeksforgeeks.org/explore?category=python)

**Дополнительная информация**

# Кластеризация KMeans и PCA в наборе данных Wine

Читать

Курсы

Упражняться

[**Кластеризация K-средних:**](https://www.geeksforgeeks.org/k-means-clustering-introduction/) Кластеризация K-средних — это алгоритм обучения без учителя, который пытается кластеризовать данные на основе их сходства. Обучение без учителя означает, что результат невозможно предсказать, и алгоритм просто пытается найти закономерности в данных. В k означает кластеризацию, мы указываем количество кластеров, в которые мы хотим сгруппировать данные. Алгоритм случайным образом присваивает каждое наблюдение набору и находит центр тяжести каждого набора. Затем алгоритм выполняет два шага: переназначает точки данных кластеру, центроид которого находится ближе всего. Вычислите новый центроид каждого кластера. Эти два шага повторяются до тех пор, пока внутрикластерная вариация не сможет быть уменьшена дальше. Отклонение внутри кластера рассчитывается как сумма евклидова расстояния между точками данных и соответствующими центроидами кластера.

В этой статье мы кластеризуем наборы винных данных и визуализируем их после уменьшения размерности с помощью PCA.

### Импорт библиотек, необходимых для анализа набора данных.

Сначала мы импортируем некоторые полезные библиотеки [Python , такие как](https://www.geeksforgeeks.org/python-programming-language/) [Pandas](https://www.geeksforgeeks.org/pandas-tutorial/) , [Seaborn](https://www.geeksforgeeks.org/boxplot-using-seaborn-in-python/) , [Matplotlib](https://www.geeksforgeeks.org/pyplot-in-matplotlib/) и SKlearn, для выполнения сложных вычислительных задач.

|  |
| --- |
| import pandas as pd  import seaborn as sns  import matplotlib.pyplot as plt  from sklearn.preprocessing import StandardScaler  from sklearn.datasets import load\_wine  from sklearn.cluster import KMeans  from sklearn.decomposition import PCA |

### **Импорт набора данных:**

Эти данные являются результатом химического анализа вин, выращенных в одном и том же регионе Италии, но полученных из трех разных сортов. В результате анализа было установлено количество 13 компонентов, обнаруженных в каждом из трех видов вин.

|  |
| --- |
| df = load\_wine(as\_frame=True)  df = df.frame  df.head() |

**Выход :**

|  | **алкоголь** | **яблочная\_кислота** | **пепел** | **щелочность\_пепла** | **магний** | **всего\_фенолов** | **флавоноиды** | **нефлаваноидные\_фенолы** | **проантоцианы** | **цвет\_интенсивность** | **оттенок** | **od280/od315\_of\_разбавленных\_вин** | **пролин** | **цель** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 14.23 | 1,71 | 2.43 | 15,6 | 127,0 | 2,80 | 3.06 | 0,28 | 2.29 | 5,64 | 1.04 | 3,92 | 1065,0 | 0 |
| **1** | 13.20 | 1,78 | 2.14 | 11.2 | 100,0 | 2,65 | 2,76 | 0,26 | 1,28 | 4.38 | 1.05 | 3.40 | 1050,0 | 0 |
| **2** | 13.16 | 2.36 | 2,67 | 18,6 | 101,0 | 2,80 | 3.24 | 0,30 | 2,81 | 5,68 | 1.03 | 3.17 | 1185,0 | 0 |
| **3** | 14.37 | 1,95 | 2,50 | 16,8 | 113,0 | 3,85 | 3,49 | 0,24 | 2.18 | 7.80 | 0,86 | 3,45 | 1480,0 | 0 |
| **4** | 13.24 | 2,59 | 2,87 | 21,0 | 118,0 | 2,80 | 2,69 | 0,39 | 1,82 | 4.32 | 1.04 | 2,93 | 735,0 | 0 |

Потому что Мы занимаемся здесь обучением без присмотра. Поэтому мы удаляем целевой столбец Customer\_Segment из наших наборов данных.

|  |
| --- |
| df.drop('target', axis =1, inplace=True)    # Check the data informations  df.info() |

**Выход:**

<класс 'pandas.core.frame.DataFrame'>

RangeIndex: 178 записей, от 0 до 177.

Столбцы данных (всего 13 столбцов):

# Столбец, ненулевой счетчик Dtype

--- ------ -------------- -----

0 алкоголь 178 ненулевое число с плавающей запятой64

1 malic\_acid 178 ненулевое число с плавающей запятой64

2 золы 178 ненулевое число с плавающей запятой64

3 alcalinity\_of\_ash 178 ненулевое float64

4 магний 178 ненулевое число с плавающей запятой64

5 total\_phenols 178 ненулевое float64

6 флавоноидов 178 ненулевое float64

7 nonflavanoid\_phenols 178 ненулевое float64

8 проантоцианов 178 ненулевое float64

9 color\_intensity 178 ненулевое float64

10 оттенок 178 ненулевой float64

11 od280/od315\_of\_diluted\_wines 178 ненулевое float64

12 пролин 178 ненулевое float64

типы данных: float64(13)

использование памяти: 18,2 КБ

### Масштабирование данных:

Данные масштабируются с помощью StandardScaler, за исключением целевого столбца (Customer\_Segment), значения которого должны оставаться неизменными.

|  |
| --- |
| scaler =StandardScaler()    features =scaler.fit(df)  features =features.transform(df)    # Convert to pandas Dataframe  scaled\_df =pd.DataFrame(features,columns=df.columns)  # Print the scaled data  scaled\_df.head(2) |

**Выход** :

|  | **алкоголь** | **яблочная\_кислота** | **пепел** | **щелочность\_пепла** | **магний** | **всего\_фенолов** | **флавоноиды** | **нефлаваноидные\_фенолы** | **проантоцианы** | **цвет\_интенсивность** | **оттенок** | **od280/od315\_of\_разбавленных\_вин** | **пролин** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 1,518613 | -0,562250 | 0,232053 | -1,169593 | 1,913905 | 0,808997 | 1,034819 | -0,659563 | 1,224884 | 0,251717 | 0,362177 | 1,847920 | 1.013009 |
| **1** | 0,246290 | -0,499413 | -0,827996 | -2,490847 | 0,018145 | 0,568648 | 0,733629 | -0,820719 | -0,544721 | -0,293321 | 0,406051 | 1,113449 | 0,965242 |

В общем, для работы K-Means требуются немаркированные данные.

Итак, берем данные без меток для кластеризации K-средних.

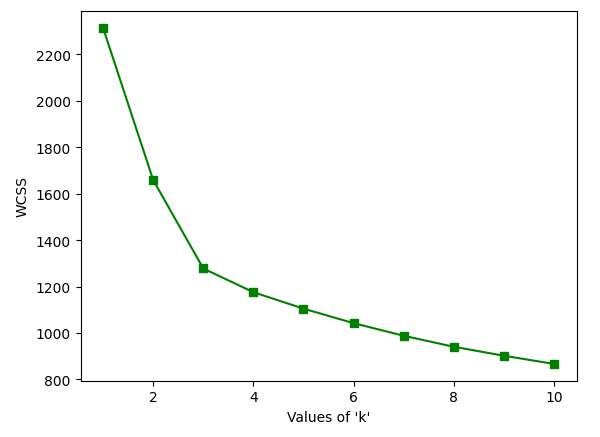
|  |
| --- |
| X=scaled\_df.values |

### Метод локтя

[Метод локтя](https://www.geeksforgeeks.org/determining-the-number-of-clusters-in-data-mining/) используется для определения количества кластеров.

|  |
| --- |
| wcss = {}  for i in range(1, 11):      kmeans = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)      kmeans.fit(X)      wcss[i] = kmeans.inertia\_    plt.plot(wcss.keys(), wcss.values(), 'gs-')  plt.xlabel("Values of 'k'")  plt.ylabel('WCSS')  plt.show() |

**Выход** :



Кривая локтя

Как мы видим из приведенного выше графика, при k=3 происходит поворот как локоть. Итак, мы можем сказать, что правильное количество кластеров для данных наборов данных равно 3.

### Реализация K-средних:

Давайте выполним кластеризацию K-Means для n\_clusters=3.

|  |
| --- |
| kmeans=KMeans(n\_clusters=3)  kmeans.fit(X) |

**Выход :**

KMeans(n\_clusters=3)

Для каждого кластера существуют значения центров кластеров в соответствии с количеством столбцов, присутствующих в данных.

|  |
| --- |
| kmeans.cluster\_centers\_ |

**Выход :**

массив([[ 0.83523208, -0.30380968, 0.36470604, -0.61019129, 0.5775868 ,

0,88523736, 0,97781956, -0,56208965, 0,58028658, 0,17106348,

0,47398365, 0,77924711, 1,12518529],

[-0,92607185, -0,39404154, -0,49451676, 0,17060184, -0,49171185,

-0,07598265, 0,02081257, -0,03353357, 0,0582655, -0,90191402,

0,46180361, 0,27076419, -0,75384618],

[ 0,16490746, 0,87154706, 0,18689833, 0,52436746, -0,07547277,

-0,97933029, -1,21524764, 0,72606354, -0,77970639, 0,94153874,

-1,16478865, -1,29241163, -0,40708796]])

labels\_ Индекс кластера, к которому принадлежит каждый образец.

|  |
| --- |
| kmeans.labels\_ |

**Выход :**

массив([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,

2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,

2, 2], dtype=int32)

### Примените метод уменьшения размерности PCA

[Анализ главных компонентов](https://www.geeksforgeeks.org/principal-component-analysis-with-python/) — это метод, который преобразует данные большого размера в данные меньшего размера, сохраняя при этом как можно больше информации.

* Он используется для интерпретации и визуализации данных.
* Количество переменных уменьшается, что упрощает дальнейший анализ.

Затем мы можем просмотреть компоненты PCA\_, то есть главные оси в пространстве признаков, которые представляют направления максимальной дисперсии в наборе данных. Эти компоненты отсортированы по объяснению\_дисперсии\_.

Минимизируйте набор данных с 15 объектов до 2 объектов с помощью анализа главных компонентов (PCA).

|  |
| --- |
| pca=PCA(n\_components=2)    reduced\_X=pd.DataFrame(data=pca.fit\_transform(X),columns=['PCA1','PCA2'])    #Reduced Features  reduced\_X.head() |

**Выход :**

|  | **PCA1** | **PCA2** |
| --- | --- | --- |
| **0** | 3,316751 | -1,443463 |
| **1** | 2,209465 | 0,333393 |
| **2** | 2,516740 | -1,031151 |
| **3** | 3,757066 | -2,756372 |
| **4** | 1,008908 | -0,869831 |

#### Редукционные центры:

Сокращение кластерных центров с помощью PCA.

|  |
| --- |
| centers=pca.transform(kmeans.cluster\_centers\_)    # reduced centers  centers |

**Выход :**

массив([[ 2.2761936 , -0.93205403],

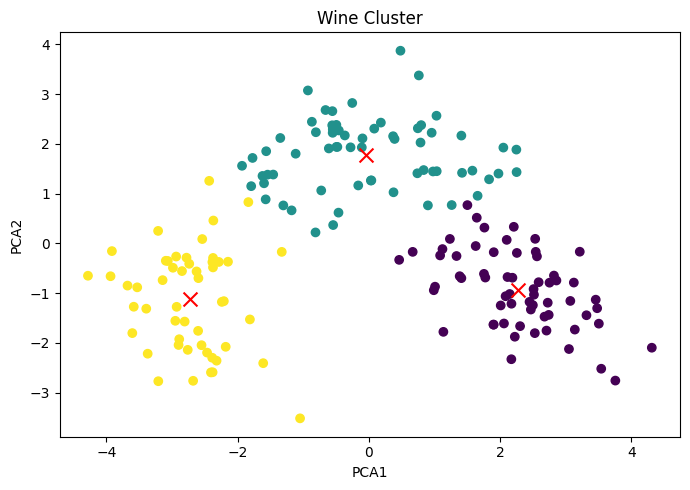
[-0,03695661, 1,77223945],

[-2.72003575, -1.12565126]])

Представьте кластерный график на основе PCA1 и PCA2. Дифференцируйте кластеры, передавая параметр цвета как c=kmeans.labels\_.

|  |
| --- |
| plt.figure(figsize=(7,5))    # Scatter plot  plt.scatter(reduced\_X['PCA1'],reduced\_X['PCA2'],c=kmeans.labels\_)  plt.scatter(centers[:,0],centers[:,1],marker='x',s=100,c='red')  plt.xlabel('PCA1')  plt.ylabel('PCA2')  plt.title('Wine Cluster')  plt.tight\_layout() |

**Выход :**



Винный кластер

### Влияние PCA1 и PCA2 на кластеры:

Если мы действительно хотим уменьшить размер набора данных, лучшее количество главных компонентов будет намного меньше, чем количество переменных в исходном наборе данных.

|  |
| --- |
| pca.components\_ |

**Выход :**

массив([[ 0.1443294 , -0.24518758, -0.00205106, -0.23932041, 0.14199204,

0,39466085, 0,4229343, -0,2985331, 0,31342949, -0,0886167,

0,29671456, 0,37616741, 0,28675223],

[-0,48365155, -0,22493093, -0,31606881, 0,0105905, -0,299634,

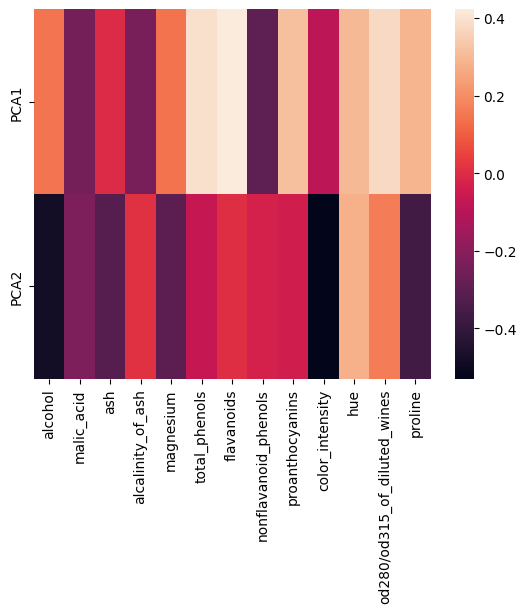
-0.06503951, 0.00335981, -0.02877949, -0.03930172, -0.52999567,

0,27923515, 0,16449619, -0,36490283]])

Представление эффектов воздействия на компонентах PCA.

|  |
| --- |
| component\_df=pd.DataFrame(pca.components\_,index=['PCA1',"PCA2"],columns=df.columns)  # Heat map  sns.heatmap(component\_df)  plt.show() |

**Выход :**



Влияние каждой функции на PCA 1 и PCA2

Не упустите свой шанс оседлать волну информационной революции! Каждая отрасль достигает новых высот, используя возможности данных. Оттачивайте свои навыки и станьте частью самой горячей тенденции 21 века.

Погрузитесь в будущее технологий — изучите [программу Complete Machine Learning and Data Science](https://www.geeksforgeeks.org/courses/data-science-live?utm_source=geeksforgeeks&utm_medium=article_bottom_text&utm_campaign=courses) от GeeksforGeeks и будьте на шаг впереди.

[Три 90 Challenge завершатся 5 февраля! Последний шанс получить возврат 90%, пройдя курс 90% за 90 дней. Изучите предложение сейчас.](https://www.geeksforgeeks.org/courses?itm_source=geeksforgeeks&itm_medium=article&itm_campaign=three90)

**Последнее обновление:** 27 марта 2023 г.

[Предыдущий](https://www.geeksforgeeks.org/iso-probability-lines-for-gaussian-processes-classification-gpc-in-scikit-learn/?ref=ml_lbp)

[Линии изовероятности для классификации гауссовских процессов (GPC) в Scikit Learn](https://www.geeksforgeeks.org/iso-probability-lines-for-gaussian-processes-classification-gpc-in-scikit-learn/?ref=ml_lbp)

[Следующий](https://www.geeksforgeeks.org/call-a-function-by-a-string-name-python/)

[Вызов функции по имени строки – Python](https://www.geeksforgeeks.org/call-a-function-by-a-string-name-python/)

Поделитесь своими мыслями в комментариях

Похожие чтения

[Сравнение 2D-проекции LDA и PCA набора данных Iris в Scikit Learn](https://www.geeksforgeeks.org/comparison-of-lda-and-pca-2d-projection-of-iris-dataset-in-scikit-learn/)

[Разница между Dataset.from\_tensors и Dataset.from\_tensor\_slices](https://www.geeksforgeeks.org/difference-between-dataset-from_tensors-and-dataset-from_tensor_slices/)

[Метод локтя для оптимального значения k в KMeans](https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/)

[Анализ дерева решений и кластеризации K-средних с использованием набора данных Iris](https://www.geeksforgeeks.org/analyzing-decision-tree-and-k-means-clustering-using-iris-dataset/)

[Разница между кластеризацией CURE и кластеризацией DBSCAN](https://www.geeksforgeeks.org/difference-between-cure-clustering-and-dbscan-clustering/)

[Разница между агломеративной кластеризацией и разделительной кластеризацией](https://www.geeksforgeeks.org/difference-between-agglomerative-clustering-and-divisive-clustering/)

[Кластеризация DBSCAN в машинном обучении | Кластеризация на основе плотности](https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/)

[Прогнозирование типа вина с использованием глубокого обучения](https://www.geeksforgeeks.org/prediction-of-wine-type-using-deep-learning/)

[Прогноз качества вина — машинное обучение](https://www.geeksforgeeks.org/wine-quality-prediction-machine-learning/)

[Выбор модели с помощью вероятностного PCA и факторного анализа (FA) в Scikit Learn](https://www.geeksforgeeks.org/model-selection-with-probabilistic-pca-and-factor-analysis-fa-in-scikit-learn/)

Полные руководства

[Учебное пособие по искусственному интеллекту с Python](https://www.geeksforgeeks.org/python-ai/)

[Лучшие проекты анализа данных с исходным кодом [2024]](https://www.geeksforgeeks.org/data-analyst-projects/)

[Ускоренный курс Python](https://www.geeksforgeeks.org/python-crash-course/)

[Учебное пособие по Python API: начало работы с API](https://www.geeksforgeeks.org/python-api-tutorial-getting-started-with-apis/)

[Продвинутые руководства по Python](https://www.geeksforgeeks.org/advanced-python-tutorials/)

[[](https://auth.geeksforgeeks.org/user/nishantd369/articles?utm_source=geeksforgeeks&utm_medium=article_author&utm_campaign=auth_user)](https://auth.geeksforgeeks.org/user/nishantd369/articles?utm_source=geeksforgeeks&utm_medium=article_author&utm_campaign=auth_user)

[**нишантд369**](https://auth.geeksforgeeks.org/user/nishantd369/articles?utm_source=geeksforgeeks&utm_medium=article_author&utm_campaign=auth_user)

Тэги статьи :

* [Технический сценарист 2022](https://www.geeksforgeeks.org/tag/technical-scripter-2022/)
* [Наука о данных](https://www.geeksforgeeks.org/category/data-science/)
* [Машинное обучение](https://www.geeksforgeeks.org/category/data-science/machine-learning/)
* [Питон](https://www.geeksforgeeks.org/category/programming-language/python/)
* [Технический сценарист](https://www.geeksforgeeks.org/category/technical-scripter/)

Теги практики:

* [Машинное обучение](https://practice.geeksforgeeks.org/explore?category=Machine)
* [питон](https://practice.geeksforgeeks.org/explore?category=python)

**Дополнительная информация**