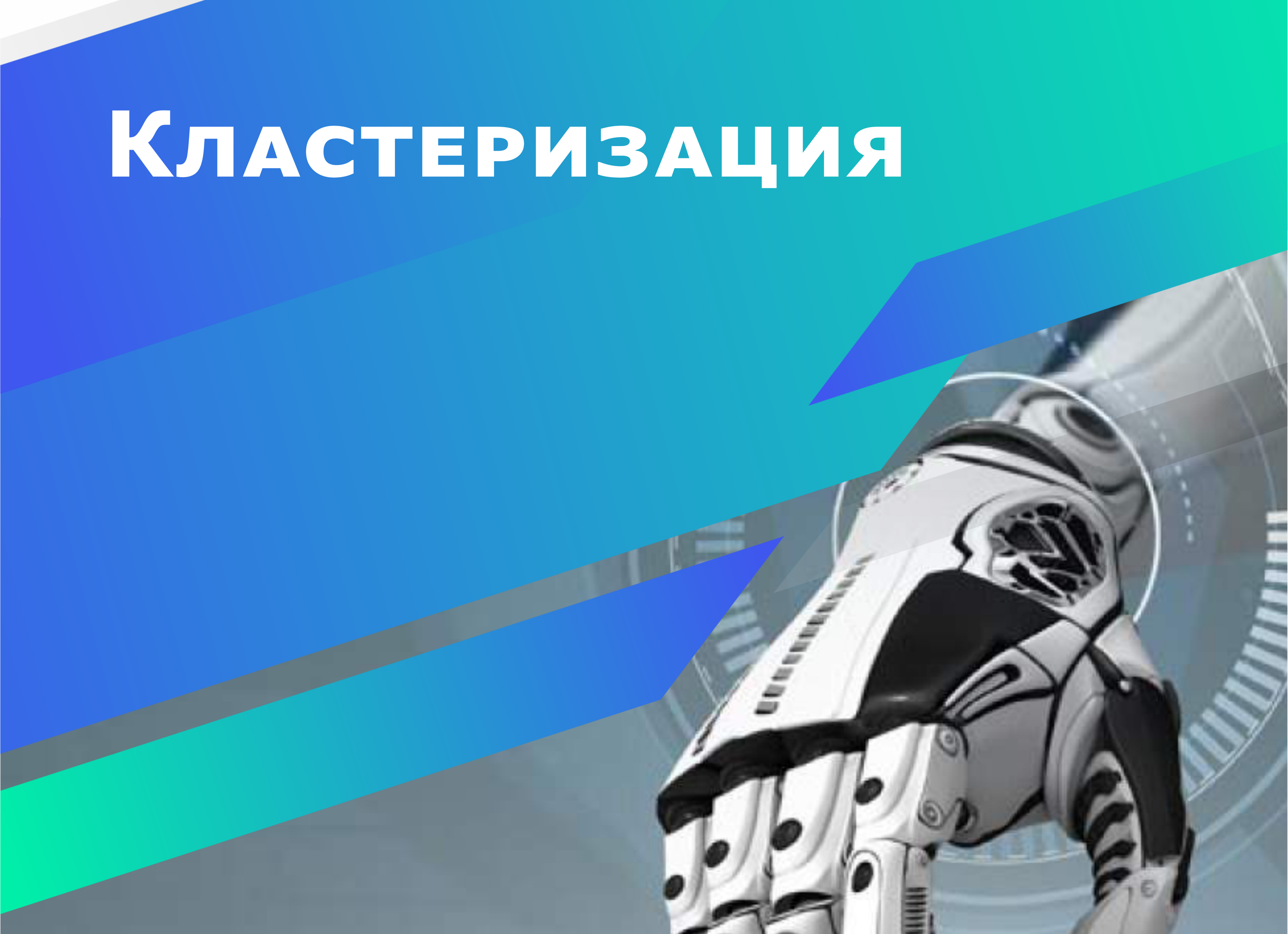
УНИВЕРСИТЕТ
искусственного
интеллекта

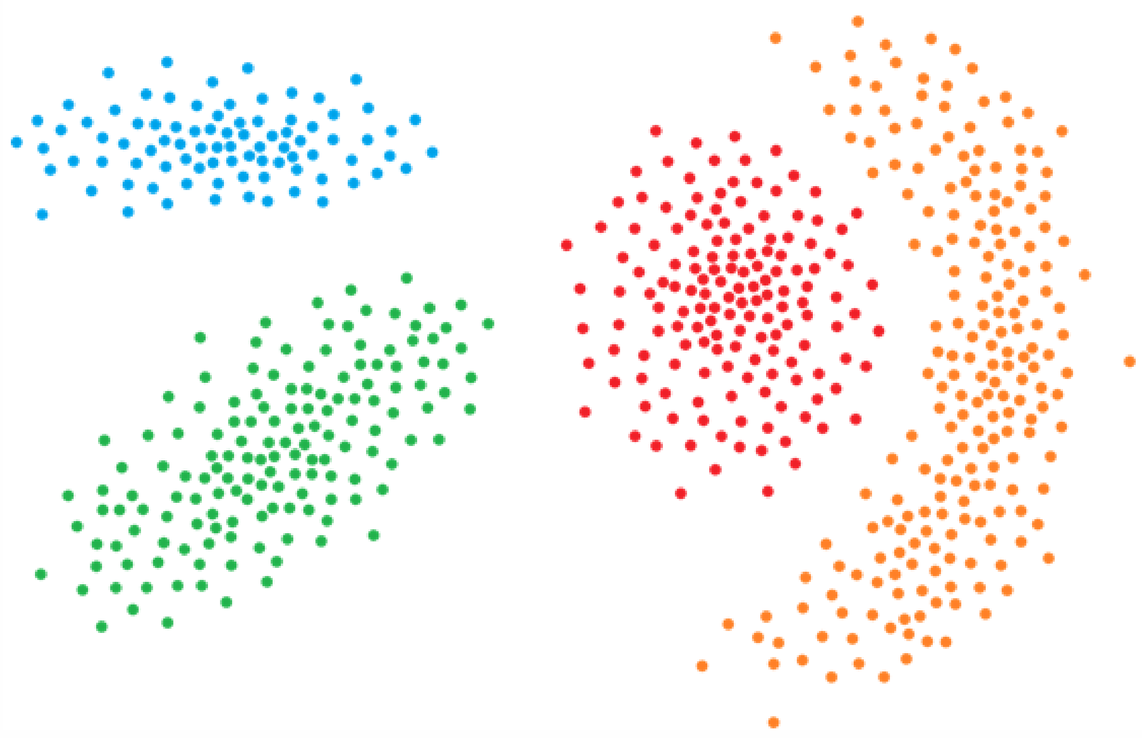
Кластеризация



Кластеризация

Кластеризация (англ. *cluster analysis*) — задача группировки множества объектов на подмножества (**кластеры**) таким образом, чтобы объекты из одного кластера были более похожи друг на друга, чем на объекты из других кластеров по какому-либо критерию.

Задача кластеризации относится к классу задач обучения без учителя.



Примеры применения кластеризации:

- кластеризация потребительской корзины
- кластеризация базы рассылки
- кластеризация фрагментов изображений
- кластеризация фотографий
- кластеризация фрагментов текста

Анализировать кластеры можно по следующим критериям:

1. Центр класса.
2. Среднее отклонение от центра.
3. Гистограмма отклонения от центра. Дает представление о разбросе кластеров.
4. Расстояние до других центров классов. Показывает, насколько текущий кластер обособлен от других.
5. Гистограмма расстояния до точек других классов.
6. Минимальная, средняя и максимальная близость чужих точек.

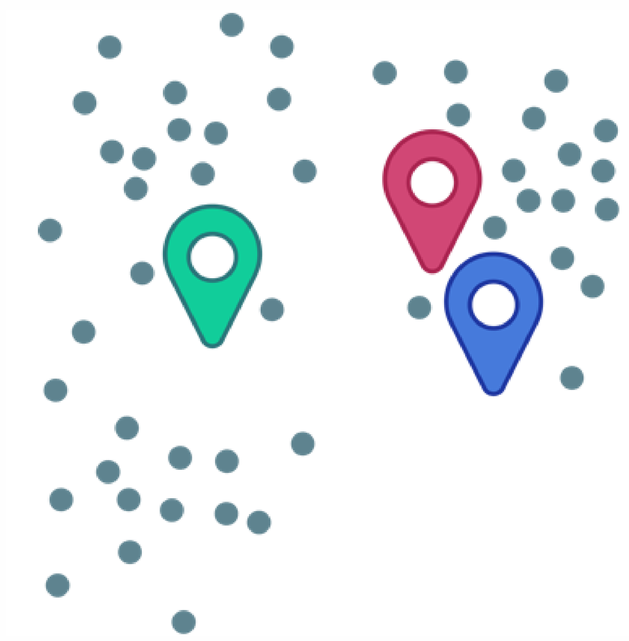
Основная метрика — это расстояние между элементами кластеров.

Количество кластеров изначально задается самостоятельно, а потом определяется оптимальное количество.

Кластеризация

Алгоритм k-mean (к-средних)

Метод *k*-средних – это специальный алгоритм кластеризации, подразумевающий, что у нас есть массив данных, которые мы хотим сгруппировать в кластеры, а точнее – в *k* кластеры.



В алгоритме метода *k*-средних есть два основных этапа. Сначала мы выбираем *k* разных центров кластеров, как правило, это просто случайные точки в наборе данных. Например, возьмем 3. Затем мы переходим к нашему основному циклу, который также состоит из двух этапов. Первый – это выбор, к какому из кластеров принадлежит каждая точка из *X*. Для этого мы берём каждый пример и выбираем кластер, чей центр ближе всего. Не забывайте, что вначале мы выбираем центры случайным образом.

Второй этап – заново вычислить каждый центр кластера, основываясь на множестве точек, которые к нему приписаны. Для этого берутся все соответствующие примеры и вычисляется их среднее значение, отсюда и название метода – метод *k*-средних.

Кластеризация

Всё это делается до тех пор, пока алгоритм не сойдется, то есть пока не прекратится изменение в распределении точек по кластерам или в координатах центров кластеров. Как правило, это происходит очень быстро: в районе от 5 до 15 проходов цикла. Это сильно отличается от градиентного спуска в глубоком обучении, где могут пройти тысячи итераций, пока не произойдёт схождение.

Реализация кластеризации k-mean в python из библиотеки sklearn

```
from sklearn.cluster import KMeans # Импортируем библиотеку KMeans
clustersCount = 10                 # Задаем количество кластеров
kmean = KMeans(clustersCount) # Создаем объект KMeans с указанным количеством кластеров
kmean.fit(xTrain01Scaled)      # Производим кластеризацию данных xTrain01Scaled
labels = kmean.labels_         # Получаем метки кластеров
```

Данные `xTrain01Scaled` это предобработанные распаршенные данные (числовые), которые несут информацию о базе резюме с hh.ru. После проведения кластеризации этой базы, каждое резюме (каждая строка базы) будет определено к какому-то из 10 (заданных нами) классов.

```
print(labels[:20])                 # Выводим первые 20 меток
```

```
[1 4 5 8 2 0 4 4 4 8 6 2 5 0 9 8 2 7 2 5]
```

Кластеризация

Первое резюме относится к 1 классу, второе к 4, третье к 5, четвертое к 8 и т. д.

При помощи специально написанной функции мы можем вывести информацию по каждому сформированному классу. Функция выглядит следующим образом:

```
'''
  Функция печати информации о кластере
    Входные данные:
      - x - набор данных
'''
def printCluster(x):
  print("Размер кластера:", x.shape[0]) # Выведем количество элементов в кластере
  mX = np.mean(x, axis=0)  # Считаем среднее значение по кластеру
  minX = np.min(x, axis=0) # Находим минимальное значение в кластере
  maxX = np.max(x, axis=0) # Находим максимальное значение в кластере
  stdX = np.std(x, axis=0) # Находим стандартное отклонение элементов кластера

  # Отображаем полученную информацию по указанному кластеру
  print("Пол: ", round(100*mX[0]), "% мужчины", sep="")
  print("Возраст: ", round(mX[1]), ", разброс: ", round(stdX[1],1), sep="")
  print("Опыт работы: ", round(mX[18]/12,1), ", разброс: ", round(stdX[18]/12,1), sep="")
  print("Зарплата: ", round(mX[19]), ", разброс: ", round(stdX[19],0), sep="")
```

Кластеризация

```
  print("\nТерриториально")
  print("Москва: ", round(100*mX[2]), "%", sep="")
  print("Санкт-Петербург: ", round(100*mX[3]), "%",
sep="")
  print("Города миллионники: ", round(100*mX[4]), "%",
sep="")
  print("Другие города: ", round(100*mX[5]), "%", sep="")

  print("\nТип занятости")
  print("Стажировка: ",  round(100*mX[6]), "%", sep="")
  print("Частичная занятость: ",  round(100*mX[7]), "%",
sep="")
  print("Проектная работа: ",  round(100*mX[8]), "%",
sep="")
  print("Полная занятость: ",  round(100*mX[9]), "%",
sep="")

  print("\nГрафик")
  print("Гибкий график: ",  round(100*mX[10]), "%",
sep="")
  print("Полный день: ",  round(100*mX[11]), "%", sep="")
  print("Сменный график: ",  round(100*mX[12]), "%",
sep="")
  print("Удаленная работа: ",  round(100*mX[13]), "%",
sep="")

  print("\nОбразование")
  print("Высшее: ",  round(100*mX[14]), "%", sep="")
  print("Среднее специальное: ",  round(100*mX[15]), "%",
sep="")
  print("Неоконченное высшее: ",  round(100*mX[16]), "%",
sep="")
  print("Среднее образование: ",  round(100*mX[17]), "%",
sep="")
```

Кластеризация

Если применить эту функцию к какому-нибудь из 10-ти наших классов, то получим следующую информацию:

```
clusterNumber = 5     # Укажем номер кластера
printCluster(xTrain01[labels==clusterNumber,:]) # Выведем
информацию о кластере с указанным номером
-----------------------------------------------------------------
Размер кластера: 691
Пол: 80.0% мужчины
Возраст: 31.0, разброс: 9.7
Опыт работы: 6.8, разброс: 6.1
Зарплата: 39154.0, разброс: 20703.0

Территориально
Москва: 21.0%
Санкт-Петербург: 5.0%
Города миллионники: 19.0%
Другие города: 55.0%

Тип занятости
Стажировка: 1.0%
Частичная занятость: 13.0%
Проектная работа: 2.0%
Полная занятость: 100.0%

График
Гибкий график: 18.0%
Полный день: 94.0%
Сменный график: 26.0%
Удаленная работа: 16.0%

Образование
Высшее: 0.0%
Среднее специальное: 100.0%
Неоконченное высшее: 0.0%
Среднее образование: 0.0%
```

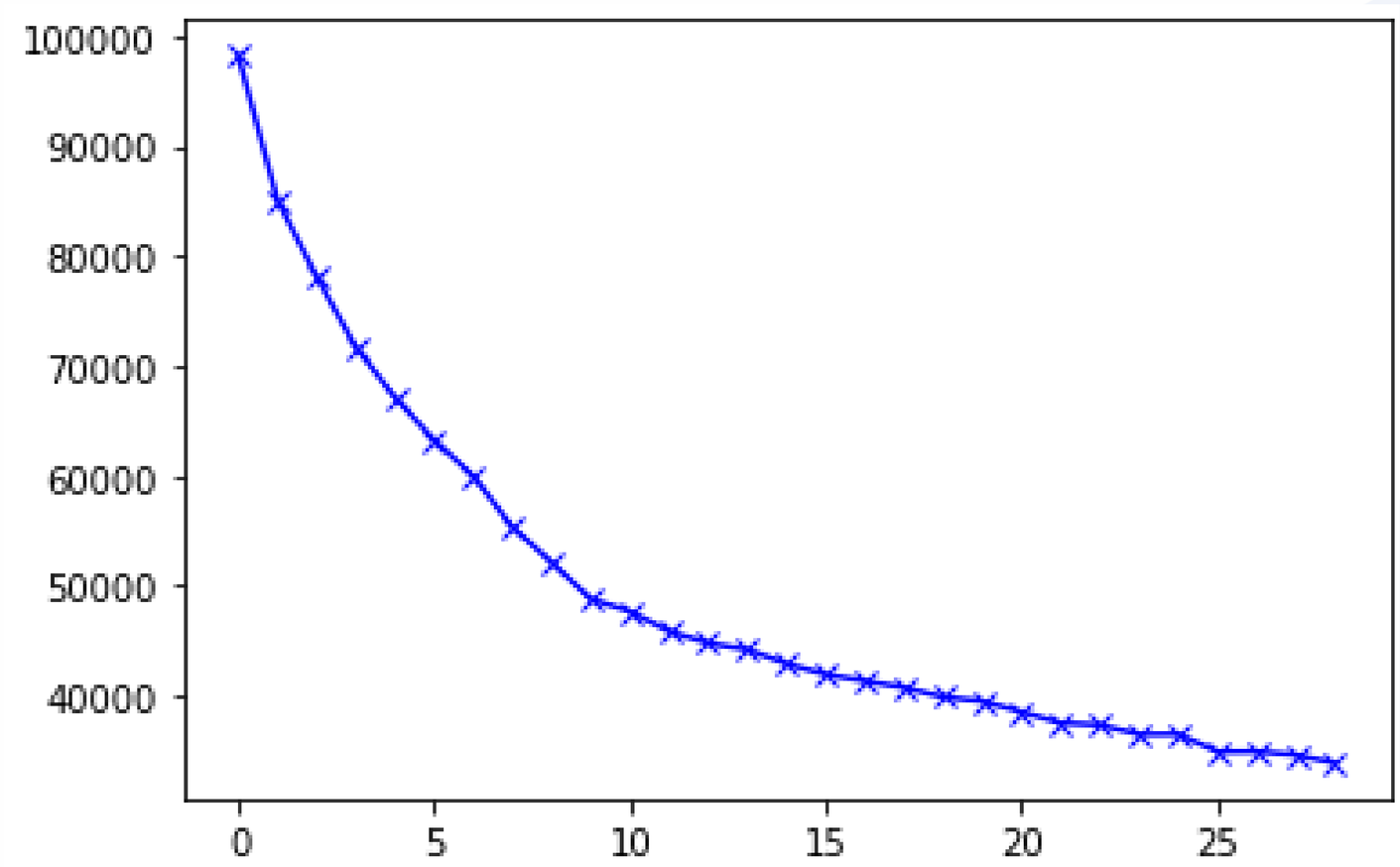
Таким образом, мы можем получить информацию о каждом сформированном кластере и сделать выводы об увеличении или уменьшении их количества.

А как же определить, сколько кластеров будет оптимально?

Кластеризация

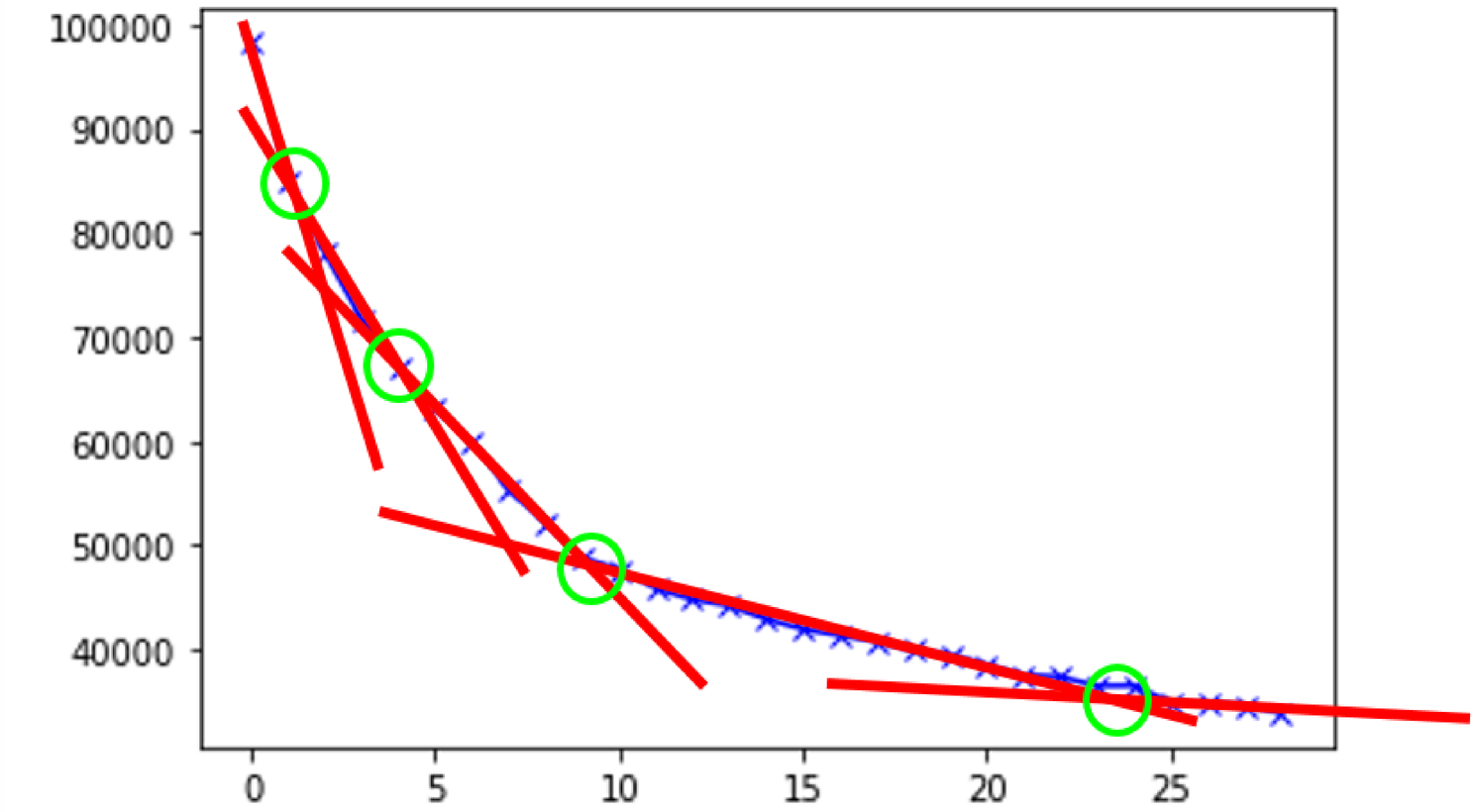
Это можно сделать, если визуализировать инерцию (среднее расстояние от точек до центров кластеров). Построим график.

```
n_clusters = 30                          # Зададим количество
кластеров
cost = []                                # Создаем пустой список
for i in range(1,n_clusters): # Пробегаем по списку от 1
до n_clusters
  kmean = KMeans(i)                      # Создаем объект KMeans с
i-классами
  kmean.fit(xTrain01Scaled)    # Проводим класстризацию
xTrain01Scaled
  cost.append(kmean.inertia_) # Добавляем в cost элемент
kmean.inertia_

# inertia  - это среднее расстояние от точек до центров их
классов
# Может использоваться для подбора оптимального количества
классов

# Отобразим значения списка cost на графике
plt.plot(cost, 'bx-')
plt.show()
```



Кластеризация

По изменению наклона графика можно понять, на какое количество кластеров оптимально разделить эти данные.



Т. е. там, где есть явное преломление графика, имеет место определение нового кластера. Судя по этому графику, можно разделить все данные примерно на 23 кластера.

Эти графики строятся не быстро (иногда несколько часов), и это нормально.

Таким образом можно делать кластеризацию разных данных (числовых, текстовых, картинок) с подробным описанием каждого кластера.