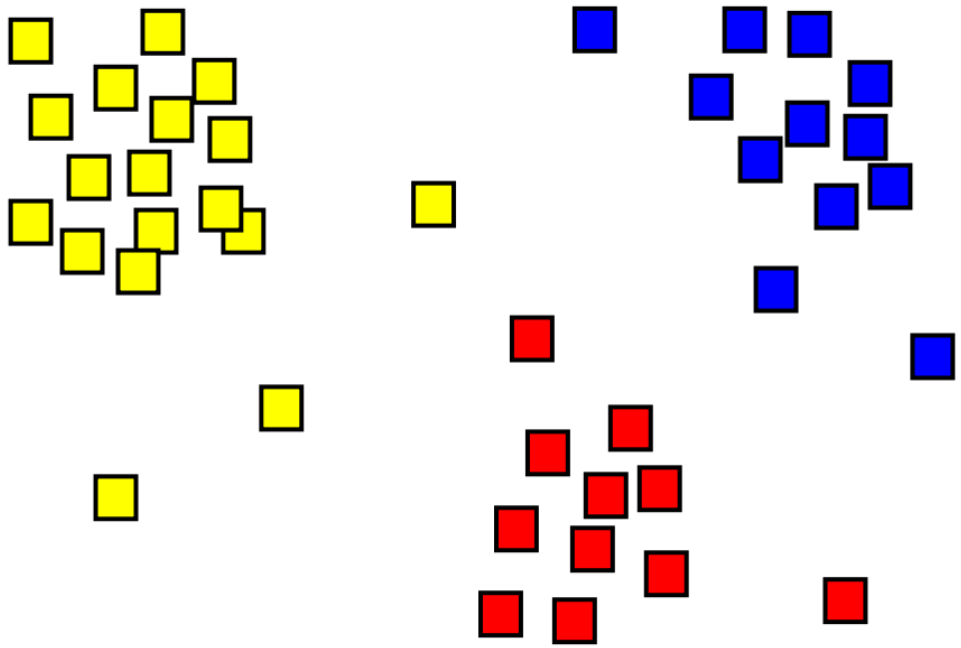
Лабораторная работа № 5.2

Кластеризация методом *k*-средних



Метод *k*-means (или *k*-средних)

Алгоритм *k*-средних – итерационная процедура, в которой выполняются следующие шаги:

1. Выбирается число кластеров *k*
2. Из исходного множества данных случайным образом выбираются *k* наблюдений – *начальные центры* кластеров
3. Для каждого наблюдения исходного множества определяется ближайший к ней центр кластера. Наблюдения, ближайшие к центру, образуют *начальные кластеры*
4. Вычисляются *центроиды* – центры тяжести кластеров. Каждый центроид – это вектор, элементы которого представляют собой средние значения соответствующих признаков, вычисленные по всем наблюдениям в кластере
5. Центр кластера смещается в его центроид и центроид становится центром нового кластера
6. 3-й и 4-й шаги итеративно повторяются. На каждой итерации происходит изменение границ кластеров и смещение их центров. В результате минимизируется расстояние между элементами внутри кластеров, и увеличиваются межкластерные расстояния

Остановка алгоритма происходит тогда, когда границы кластеров и расположения центроидов перестают изменяться от итерации к итерации, т.е. на каждой итерации в каждом кластере остается один и тот же набор наблюдений.

Достоинства и недостатки *k*-means

- Достоинства метода:

1) Быстрый,
2) Высокое качество кластеризации,
3) Большое множество модификаций.

- Недостатки:

1) Количество кластеров является параметром алгоритма,
2) Только евклидово расстояние,
3) Чувствителен к начальным условиям,
4) Чувствителен к выбросам и шумам.

Практическое задание

В файле mobile.xlsx находятся данные об абонентах телекоммуникационной компании.

Целью исследования является сегментация абонентской базы для разработки новых тарифов и проведения маркетинговых акций.

Примените метод *k*-средних. Используя метод «локтя» определите оптимальное количество кластеров. Визуализируйте полученный результат. Как распределились наблюдения по кластерам? Проанализируйте полученный результат.

Загружаем данные

```
import pandas as pd
df = pd.read_excel('mobile.xlsx')
df.head()
```

	Количество SMS за месяц	Количество звонков	Среднемесячный расход
0	56	82	121.54
1	1	221	287.51
2	36	68	113.70
3	23	96	410.23
4	29	139	537.60

Кластеризация методом k-means

```
# Импортируем k-means из библиотеки Scikit-Learn
from sklearn.cluster import KMeans

# Создаем экземпляр класса k-means
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, random_state=0)

# Обучаем агоритм на данных с помощью метода fit
kmeans.fit(df)

# Записываем номера кластеров в новый столбец
df['Кластер'] = kmeans.labels_
df.head()
```

	Количество SMS за месяц	Количество звонков	Среднемесячный расхо
0	56	82	121.5
1	1	221	287.5
2	36	68	113.7
3	23	96	410.2
4	29	139	537.6

Scikit-Learn – библиотека для машинного обучения на Python. С помощью Scikit-Learn реализовать различные алгоритмы классификации, ре грессии и кластеризации.

Смотри документацию sklearn.cluster.Kmeans

Параметры:

n_clusters – количество кластеров,

init – способ инициализации центроидов (по умолчанию ‘k-means++’),

n_init – количество запусков алгоритма со случайной инициализацией центроидов (по умолчанию n_init = 10),

random_state – определяет случайность инициализации центроида

и др.

! random_state устанавливаем 0, чтобы у всех получился одинаковый результат!

Атрибуты:

cluster_centers_ – координаты центров кластеров,

labels_ – возвращает метки кластеров,

inertia_ – сумма квадратов расстояний наблюдений до ближайшего центра кластера.

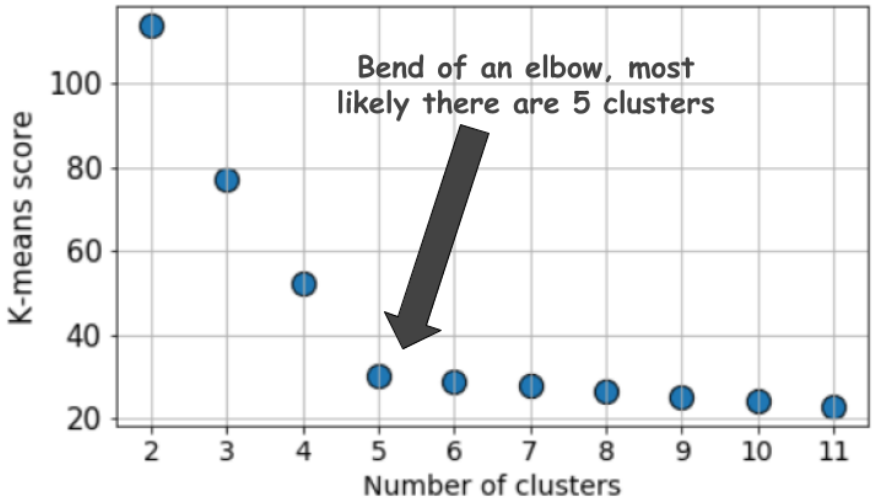
Метод «локтя»

При кластеризации методом *k*-средних количество кластеров чаще всего оценивают с помощью метода «локтя».

Метод «локтя» заключается в циклическом запуске алгоритма с различным (последовательно увеличивающимся) количеством кластеров.

Полученный результат отображается на графике: по оси Ox откладывается количество кластеров, а по оси Oy критерий качества алгоритма *k*-means (в нашем случае такую меру позволяет получить атрибут inertia_).

По графику на рисунке можно видеть, что в какой-то момент разрыв между точками резко уменьшается. Это и есть «локоть». А количество кластеров, соответствующих «локтю» принято считать оптимальным.



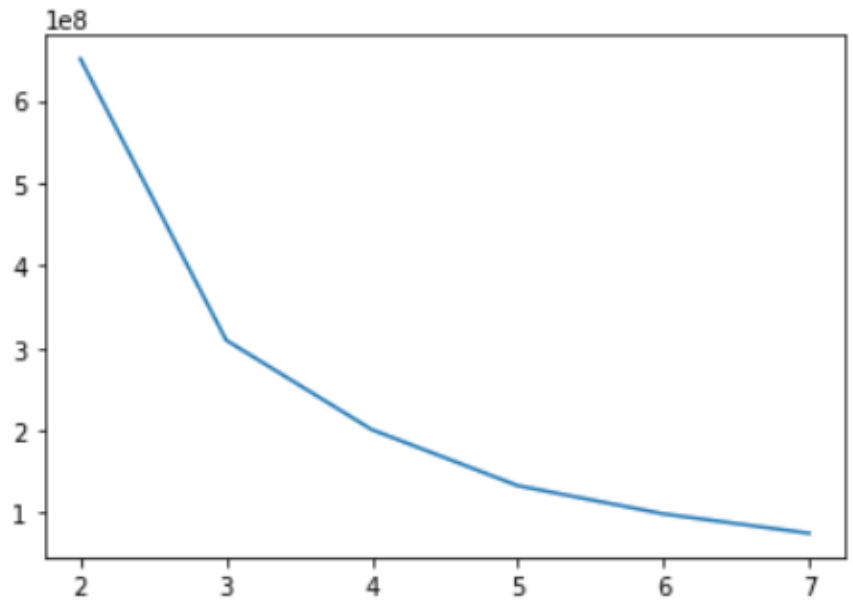
Недостаток метода: локоть не всегда может быть однозначно идентифицирован.

Метод локтя

```
# Создадим список, в который будем записывать значение критерия качества k-means
criterion = []
# Количество кластеров будем изменять от 2 до 8
for k in range(2,8):
    kmeans = KMeans(n_clusters = k)
    kmeans.fit(df)
    criterion.append(kmeans.inertia_)

import matplotlib.pyplot as plt
plt.plot(range(2,8), criterion)
```

```
[<matplotlib.lines.Line2D at 0x1c1c3738860>]
```



Визуализация кластеров

```
from mpl_toolkits.mplot3d import Axes3D
import numpy as np

fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(111, projection='3d')
x = np.array(df['Количество SMS за месяц'])
y = np.array(df['Количество звонков'])
z = np.array(df['Среднемесячный расход'])

ax.scatter(x,y,z, c=df['Кластер'])

plt.show()
```