Лабораторная работа № 3
Кластерный анализ методом k-средних

Цель работы: научиться импортировать данные, проводить предобработку данных, применять к данным метод кластерного анализа (k-средних) и интерпретировать полученные на основе выделения кластеров знания с точки зрения бизнес-контекста.

Формируемые знания, умения и навыки: изучить, как реализуттся метод k-средних на языке Python, получить навыки выделения бизнес-контекста из результатов кластерного анализа

Необходимо:

1. С сайта https://www.kaggle.com/ импортировать один из наборов данных.
2. Используя возможности языка Python, провести кластеризацию данных методом k-средних, выдвинуть гипотезы о бизнес-контексте результатов анализа.

Бизнес-контекст

Опишите выбранный набор данных: контекст данных, какие переменные есть в наборе, какое количество кластеров было выбрано для данных, в чем отличия полученных кластеров друг от друга, как можно использовать в бизнесе полученное разбиение на кластеры.

Теоретическая часть

«**Кластерный анализ**» — это общее название множества вычислительных процедур, используемых при создании классификации. В результате работы с процедурами образуются «**кластеры**» или группы очень похожих объектов. Более точно, **кластерный метод** - это многомерная статистическая процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в сравнительно однородные группы.

Различные приложения кластерного анализа можно свести к четырем основным задачам:

1) разработка типологии или классификации;
2) исследование полезных концептуальных схем группирования объектов;
3) порождение гипотез на основе исследования данных;
4) проверка гипотез или исследования для определения, действительно ли типы (группы), выделенные тем или иным способом, присутствуют в имеющихся данных.

Техника кластеризации применяется в самых разнообразных областях. Например, в области медицины - кластеризация заболеваний, лечения заболеваний или симптомов заболеваний приводит к широко используемым таксономиям. Известны широкие применения кластерного анализа в маркетинговых исследованиях. В области защиты информации методы кластерного анализа применяются для анализа лог файлов серверов за большие промежутки времени и т.д. В общем, всякий раз, когда необходимо классифицировать «горы» информации к пригодным для дальнейшей обработки группам, кластерный анализ оказывается весьма полезным и эффективным.

Кластерный анализ – это метод группировки экспериментальных данных в классы. Наблюдения, попавшие в один класс, в некотором смысле ближе друг к другу, чем к наблюдениям из других классов.

Решение задачи кластеризации принципиально неоднозначно:

1. не существует однозначно наилучшего критерия качества кластеризации;
2. число кластеров, как правило, неизвестно заранее и устанавливается в соответствии с некоторым субъективным критерием;
3. результат кластеризации существенно зависит от метрики, выбор которой, как правило, также субъективен и определяется экспертом.

Задачей кластерного анализа является организация наблюдаемых данных в наглядные структуры. Для решения данной задачи в кластерном анализе используются следующие методы:

- Иерархические агломеративные методы или древовидная кластеризация (Joining /tree clustering);
- Метод К средних (K - means clustering);
- Двухвходовое объединение (Two-way joining).

В данной лабораторной работе будет рассматриваться **метод К средних**.

Метод k-средних

Материалы: [1], [2], [3, с. 289–298], [4, с. 82-85].

Кластеризация позволяет структурировать информацию в содержательные подгруппы (кластеры), не имея предварительных сведений о количестве таких групп и о значениях характеристик представителей кластеров (содержательных связях между данными кластера).

Кластеризацию относят к методам разведочного анализа данных: неразмеченные данные разносятся по различным кластерам на основе подобия характеризующих их признаков.

Один из самых популярных алгоритмов кластеризации – **метод k-средних**.

Алгоритм k-средних – кластеризация на основе **прототипов**. Это означает, что каждый кластер представлен прототипом, который может быть либо **центроидом** (средним) подобных точек с непрерывными признаками, либо **медоидом** (наиболее представительной или наиболее часто встречающейся точкой) в случае категориальных признаков (или иначе факторов, например, пол, наличие вредной привычки и т.п.).

Метод k-средних хорошо справляется с идентификацией кластеров сферической формы. Однако этот метод требует изначально задать количество кластеров. При неудачном выборе числа кластеров или неподходящем задании начального положения центроидов кластеров результат кластеризации может быть сомнительным. Кроме того, этот метод чувствителен к качеству данных: выбросы, аномальные наблюдения, шум в данных могут также ухудшить итоговый результат. С помощью ресурса, визуализирующего метод k-средних [5], можно проиллюстрировать первую проблему с неудачным начальным выбором кластеров.

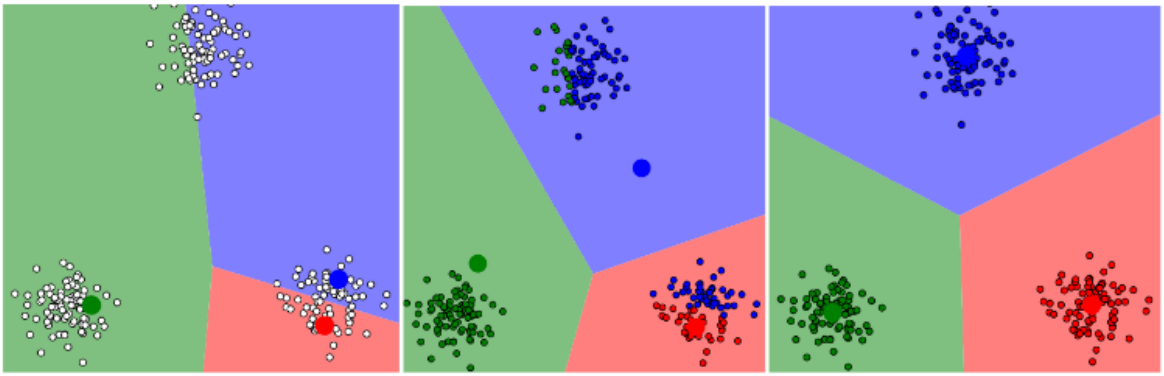


Рис. 1.1. Визуализация метода k-средних

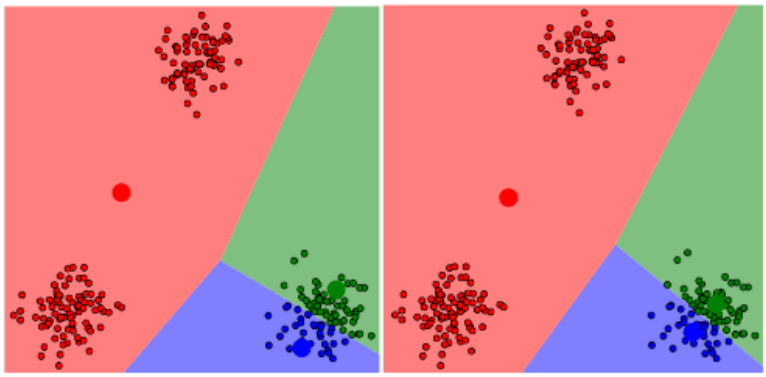


Рис. 1.2. Визуализация метода k-средних

В алгоритме k-средних последовательно реализуются следующие **шаги**:

1 шаг: выбрать количество кластеров (произвольно или опираясь на какие-то априорные предположения);

2 шаг: в пространстве данных расположить произвольным образом центроиды (центры кластеров);

3 шаг: определить для каждой точки набора данных, к какому центроиду она ближе;

4 шаг: для получившихся кластеров найти новое положение центроида (как точки, суммарное расстояние от которой до всех точек кластера – наименьшее).

Шаги 3 и 4 повторяются либо фиксированное количество раз, либо до момента, когда смещение новых центров кластеров относительно центров кластеров на предыдущем шаге будет меньше некоторого значения (т.е. центры кластеров стабилизируются).

Для кластеризации объектов с непрерывными признаками обычно используют евклидово расстояние, т.е. алгоритм минимизирует суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} (x_i^{(j)} - c_j)^2,$$

где $x_i^{(j)}$ - наблюдение с номером i, которое отнесено к кластеру j, а c_j – центроид кластера j, k – число кластеров, n – число наблюдений.

В классическом алгоритме k-средних для размещения начальных центроидов используется случайное число (random seed), что иногда дает плохие результаты кластеризации, а в некоторых случаях – замедление сходимости.

Есть модификация метода k-средних, которая обязывает выбирать в качестве центроида не произвольную точку, а одно из наблюдений данных.

Более достоверные результаты кластеризации дает реализация метода k-средних **k-means++**, когда в качестве центроидов выбираются точки наблюдений, далеко отстоящие друг от друга.

Так как метод k-средних находит локальный, а не глобальный минимум для внутрикластерных расстояний, то результаты будут зависеть от начального разбиения кластеров.

Важно запускать алгоритм **несколько раз**, используя разные конфигурации его реализации, и выбрать наилучшую. Также важным параметром качества кластеризации является интерпретируемость конечного результата. Если кластеризация проводится в данных бизнес-кейса, то необходимо, чтобы результаты кластеризации помогали достижению бизнес-целей: выявляли сходство объектов (общие шаблоны поведения) по тем параметрам, которые позволят управлять поведением объектов в выгодном для бизнеса направлении. Так как кластеризация дает неустойчивые решения, то, если объем данных позволяет, можно выполнять кластеризацию на разных частях данных, чтобы получить представление об устойчивости полученных кластеров.

Так как данные не размечены, то количество кластеров, на которые должен быть разделен исходный набор, изначально неизвестно.

При кластеризации методом k-средних количество кластеров чаще всего оценивают с помощью **«метода локтя» (Elbow method).** Это интуитивная, довольно грубая эвристика. На графике откладывается некоторая величина, характеризующая качество кластеризации, например, внутрикластерная сумма расстояний J для разного количества кластеров (в статистических пакетах для реализации метода k-средних нужная величина – это искажение или инерция, например в методе KMeans на языке Python – это значение атрибута **inertia_**).

Оптимальное количество кластеров соответствует значению k, после которого величина J перестает резко падать. Понятно, что лучший вариант J = 0 достигается при количестве кластеров, совпадающих с числом наблюдений. Но в бизнесе обычно нужны стабильные кластеры с большим количеством наблюдений, для которых можно выявить полезные закономерности в поведении, расходах и т.п.

Если в данных есть разметка (разбиение на классы), то на основе метода k-средних можно построить простую реализацию машинного обучения с метриками качества обучения.

На тренировочной части данных происходит разбиение на кластеры, а на тестовых данных определяется, насколько хорошо машина распознает в новых данных (тестовая часть, которая не участвовала в обучении) выявленные при обучении кластеры. Важно заметить, что такое машинное обучение – это уже обучение с учителем. Хотя мы не предъявляем разметку данных (классы или категории данных) при обучении, но именно на основе разметки происходит оценка качества обучения. Машинное обучение на неразмеченных данных с использованием метода k-средних также реализуемо, но в этом случае сложнее оценить качество.

Для описания метрик качества машинного обучения построим табл. 1.1

Таблица 1.1

Анализ соответствия кластеров и классов размеченных данных

		Класс	
		1	0
Кластер	1	True Positive (TP)	False Positive (FP)
	0	False Negative (FN)	True Negative (TN)

Очевидно, что при кластеризации в случае двух классов воз- можны ошибки двух видов: False Positive (FP) – неверно распознан объект класса с меткой 0 и False Negative (FN) – неверно распознан объект класса с меткой 1. Метрики, оценивающие качество проведенной кластеризации, следующие [8, 9].

Accuracy – это метрика, которая показывает долю правильных ответов модели. Ее значение равно отношению числа правильных ответов, которые дала модель, к числу всех объектов. Эта метрика зависит от соотношения классов объектов в выборке и используется нечасто.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Для оценки качества распознавания каждого класса вводятся метрики **precision (точность)** и **recall (полнота)**.

Precision (точность) показывает долю объектов, названных классификатором положительными и при этом действительно являющимися положительными.

$$precision = \frac{TP}{TP + FP}$$

Recall (полнота) показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

$$recall = \frac{TP}{TP + FN}$$

F-мера – среднее гармоническое precision и recall. F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

$$F = \frac{precision \cdot recall}{precision + recall}$$

Для получения кластеризации используем реализацию метода **k-means** в **SciPy**.

SciPy - это открытая библиотека высококачественных научных инструментов для языка программирования Python.

Метод k-средних в Python

В Python для реализации метода k-средних необходимо импортировать из класса **sklearn.cluster** (библиотека scikit-learn) класс KMeans [16].

```
from sklearn.cluster import KMeans
```

Параметры класса KMeans()

n_clusters	Количество кластеров
init	Метод определения начального положения центроидов
max_iter	Максимальное количество итераций

Атрибуты класса KMeans()

cluster_centers_	Массив кластерных центров
inertia_	Сумма квадратов расстояний выборок до их ближайших центров скоплений

Методы класса KMeans()

fit(X)	Выполняет кластеризацию методом k-средних
fit_predict(X)	Находит значение кластера для каждого наблюдения данных X

Подключение библиотек.

```
# Импортируем библиотеку inspect
import inspect
```

После подключения, используя имя модуля, можно получить доступ к его функциям: <имя модуля>.<имя функции>().

```
#возвращаем аргументы функции KMeans() с помощью signature() из inspect
inspect.signature(KMeans)
```

Можно подключать библиотеку, сразу задавая сокращенное имя для обращения к функциям библиотеки.

```
import pandas as pd
# Загружаем набор данных
iris_df = pd.read_csv('iris.csv')
```

Можно импортировать имена функций из модуля непосредственно в текущее пространство имен:

```
import matplotlib.pyplot as plt
```

Или импортировать конкретные функции из модуля.

```
from sklearn import datasets
# Загружаем встроенный набор данных
iris_df = datasets.load_iris()
```

Для выяснения имён, определенных в модуле, функции, переменной можно использовать встроенную функцию dir(). Она возвращает отсортированный список строк

```
In [125]: print(dir(KMeans))

['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq
__', '__format__', '__ge__', '__getattribute__', '__getstate__', '__
gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt_
_', '__module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__
', '__repr__', '__setattr__', '__setstate__', '__sizeof__', '__str__
', '__subclasshook__', '__weakref__', '_check_fit_data', '_check_tes
t_data', '_estimator_type', '_get_param_names', '_transform', 'fit',
'fit_predict', 'fit_transform', 'get_params', 'predict', 'score', 's
et_params', 'transform']
```

Для построения графиков в Python рекомендуется использовать библиотеку **matplotlib** [11] и **seaborn** [12], для работы с многомерными массивами **NumPy** [13], для работы с данными (чтения файлов, работы с пропусками и т.п.) библиотеку **pandas** [14], для реализации алгоритмов машинного обучения без учителя библиотеку **scikit-learn (sclearn)** [15], основанную на других библиотеках Python: NumPy, SciPy и matplotlib

Пример 1. Набор данных ирисы

Первый набор данных – набор данных ирисов Фишера.

Первый набор данных – классика всех обучающих курсов наряду с данными о пассажирах Титаника. Это набор данных ирисов Фишера (входит в пакет base имя dataset – iris), содержит 150 наблюдений.

Описание переменных набора:

длина чашелистика (sepal length); ширина чашелистика (sepal width);

длина лепестка (petal length);

ширина лепестка (petal width);

класс (Species), соответствующий одному из трех видов: Iris Setosa, Iris Versicolor или Iris Virginica.

1. Загрузим файл с данными iris.csv.

```
In [126]: # Импортируем библиотеки
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
%matplotlib inline
import pandas as pd

# Загружаем набор данных
iris_df = pd.read_csv('iris.csv')
# Выводим на просмотр первые строки набора
iris_df.head()
```

Out[126]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa

2. Создаем новый набор данных без переменной, определяющей вид ириса.

```
In [127]: # Копируем первые четыре столбца в новую переменную
iris_df1 = pd.DataFrame(iris_df[['Sepal.Length','Sepal.Width', 'Petal.Length', 'Petal.Width']])
iris_df1.head()
```

Out[127]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	5.1	3.5	1.4	0.2
2	4.9	3.0	1.4	0.2
3	4.7	3.2	1.3	0.2
4	4.6	3.1	1.5	0.2
5	5.0	3.6	1.4	0.2

3. Применим метод k-средних к данным набора ирисы.

```
In [129]: # Описываем модель
model = KMeans(n_clusters=3)

# Проводим моделирование
model.fit(iris_df1)

# Добавим в исходный набор новый столбец с номером кластера наблюдения
iris_df1['cluster'] = model.fit_predict(iris_df1)
iris_df1.head()
```

Out[129]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	cluster
1	5.1	3.5	1.4	0.2	0
2	4.9	3.0	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	4.6	3.1	1.5	0.2	0
5	5.0	3.6	1.4	0.2	0

4. Добавим новые столбцы в набор данных: номер кластера и факторную переменную на основе данных о виде ириса. В Python нумерация кластеров и преобразование категориальной переменной в фактор начинается с нуля.

```
In [130]: # Добавим в исходный набор новый столбец, преобразовав категориальную переменную в фактор
iris_df1['species'] = pd.factorize(iris_df['Species'])[0]

iris_df1.head()
```

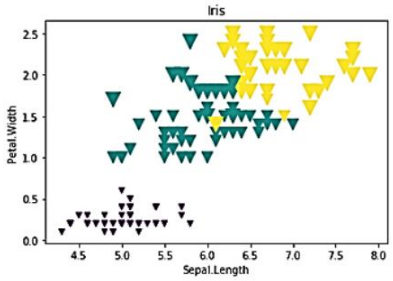
Out[130]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	cluster	species
1	5.1	3.5	1.4	0.2	0	0
2	4.9	3.0	1.4	0.2	0	0
3	4.7	3.2	1.3	0.2	0	0
4	4.6	3.1	1.5	0.2	0	0
5	5.0	3.6	1.4	0.2	0	0

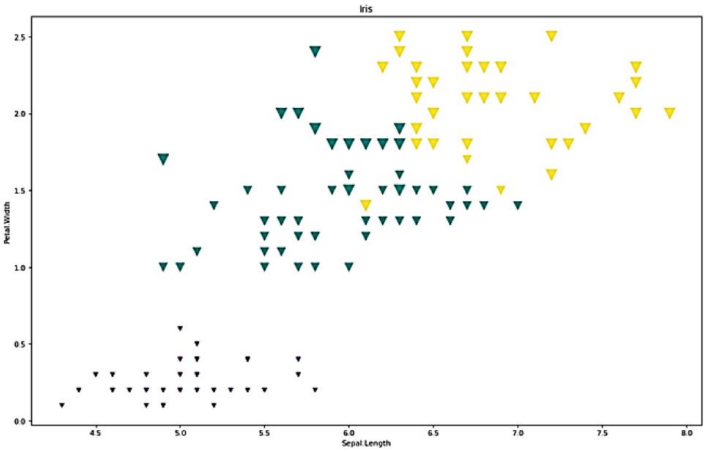
5. Построим график по переменным Sepal.Length и Petal.Width, переменная cluster задает цвет (color), а переменная species – размер (используется формула, так как нулевое значение не позволит отобразить маркер).

```
In [131]: #Строим график
plt.scatter(iris_df1['Sepal.Length'], iris_df1['Petal.Width'], c = iris_df1['cluster'],
            s = 30*(3*iris_df1['species']+1), marker = "v")
plt.title('Iris')
plt.xlabel('Sepal.Length')
plt.ylabel('Petal.Width')
```

```
Out[131]: Text(0,0.5,'Petal.Width')
```



6. Для повышения читабельности графика настроим параметры отображения, добавив дополнительную строчку в начало фрагмента кода, строящего график. plt.figure(figsize=(16, 10), dpi= 80, facecolor='w', edgecolor='k)



7. Составим таблицу сопряженности переменных cluster и species для анализа верности определения классов растений при разбиении на кластеры с помощью кластерного анализа (значения NaN – это нулевые значения).

```
In [132]: #Составим по данным класса и кластера сводную таблицу
iris_df1.groupby(['cluster', 'species'])['cluster'].count().unstack()
```

```
Out[132]:
```

species	0	1	2
cluster			
0	50.0	NaN	NaN
1	NaN	48.0	14.0
2	NaN	2.0	36.0

Пример 2. Набор данных медицинское страхование

Второй набор данных – набор данных медицинское страхование.

Рассмотрим теперь более корректный случай, когда изначально разметки у данных по классам нет, и мы не знаем, какое количество кластеров нужно выбрать.

Данные, которые мы используем – расходы на медицинское обслуживание тех, кто имеет медицинскую страховку: https://www.kaggle.com/mirichoi0218/insurance .

Описание переменных набора:

age: возраст основного бенефициара;

sex: пол застрахованного;

bmi: индекс массы тела;

children: число детей, охваченных медицинским страхованием / число иждивенцев;

smoker: курит ли застрахованный;

region: жилой район получателя в США, Северо-Восток, Юго-Восток, Юго-Запад, Северо-Запад.

charges: индивидуальные медицинские расходы, оплачиваемые страховкой.

1. Загрузим файл с данными **insurance.csv**.

```
In [56]: # Импортируем библиотеки
         from sklearn.cluster import KMeans
         import matplotlib.pyplot as plt
         %matplotlib inline
         import pandas as pd
         import numpy as np
         import seaborn as sns

         # Загружаем набор данных
         insurance_df = pd.read_csv('insurance.csv')
         insurance_df.head()
```

Out[56]:

	age	sex	bmi	children	smoker	region	charges
0	19	female	27.900	0	yes	southwest	16884.92400
1	18	male	33.770	1	no	southeast	1725.55230
2	28	male	33.000	3	no	southeast	4449.46200
3	33	male	22.705	0	no	northwest	21984.47061
4	32	male	28.880	0	no	northwest	3866.85520

2. Преобразуем категориальные переменные в факторы, так как модель на основе класса KMeans() применяется только к числовым данным.

```
In [62]: insurance_nu = insurance_df
         # Преобразуем категориальные переменные в факторы
         insurance_nu['sex'] = pd.factorize(insurance_nu['sex'])[0]
         insurance_nu['smoker'] = pd.factorize(insurance_nu['smoker'])[0]
         insurance_nu['region'] = pd.factorize(insurance_nu['region'])[0]
         # Создадим дубликат преобразованного набора
         insurance_nu1 = insurance_nu
         insurance_nu.head()
```

Out[62]:

	age	sex	bmi	children	smoker	region	charges
0	19	0	27.900	0	0	0	16884.92400
1	18	1	33.770	1	1	1	1725.55230
2	28	1	33.000	3	1	1	4449.46200
3	33	1	22.705	0	1	2	21984.47061
4	32	1	28.880	0	1	2	3866.85520

3. Выполним кластеризацию

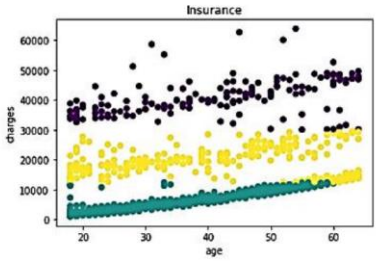
```
In [63]: # Описываем модель
         model = KMeans(n_clusters=3)

         # Проводим моделирование
         model.fit(insurance_nu)

         # Добавляем в исходный набор новый столбец с номером кластера наблюдения
         insurance_nu['cluster'] = model.fit_predict(insurance_nu)
         insurance_nu.head()
```

Out[63]:

	age	sex	bmi	children	smoker	region	charges	cluster
0	19	0	27.900	0	0	0	16884.92400	2
1	18	1	33.770	1	1	1	1725.55230	1
2	28	1	33.000	3	1	1	4449.46200	1
3	33	1	22.705	0	1	2	21984.47061	2
4	32	1	28.880	0	1	2	3866.85520	1

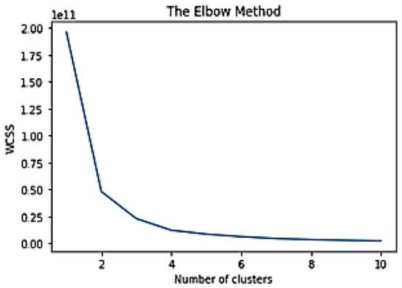
4. Построим график зависимости медицинских расходов от возраста для разных кластеров

```
In [64]: #Строим график
         plt.scatter(insurance_nu['age'], insurance_nu['charges'], c = insurance_nu['cluster'])
         plt.title('Insurance')
         plt.xlabel('age')
         plt.ylabel('charges')

Out[64]: Text(0,0.5,'charges')
```



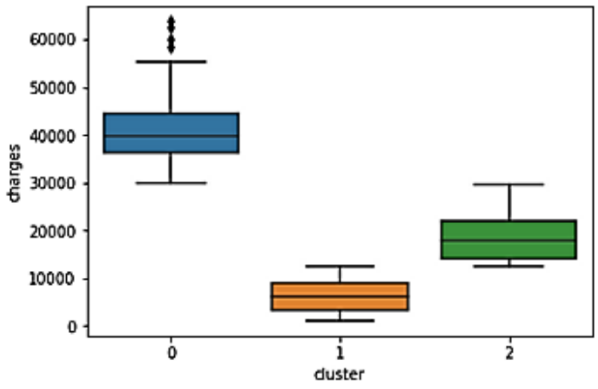
5. Выясним наилучшее количество кластеров с помощью «метода локтя». Как видно, оптимальное число кластеров для набора данных 3 или 4

```
In [65]: # Инициализируем переменную, в которую будем записывать kmeans.inertia_
         wcss = []
         # В цикле проведем кластерный анализ для различного числа кластеров,
         # присоединяя к массиву wcss соответствующие значения внутрикластерной суммы квадратов
         for i in range(1, 11):
             kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
             kmeans.fit(insurance_nu1)
             wcss.append(kmeans.inertia_)
         # Построим график по значениям wcss
         plt.plot(range(1, 11), wcss)
         plt.title('The Elbow Method')
         plt.xlabel('Number of clusters')
         plt.ylabel('WCSS')

Out[65]: Text(0,0.5,'WCSS')
```



6. Визуализируем различие в медицинских расходах для разных кластеров с помощью графика «ящик с усами»

```
Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x189f6762cc0>
```



7. Составим таблицу сопряженности для переменных smoker и cluster (анализ совместного распределения курящих и некурящих застрахованных по кластерам). Значения NaN – это нулевые значения

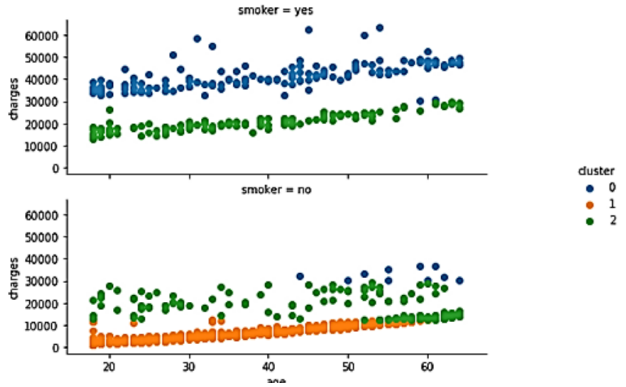
```
In [87]: #Составим по данным о наличии вредной привычки у застрахованного и кластера сводную таблицу
insurance_df['cluster'] = insurance_nu['cluster']
insurance_df.groupby(['cluster', 'smoker'])['cluster'].count().unstack()
```

Out[87]:

cluster	smoker no	yes
0	10.0	152.0
1	870.0	NaN
2	184.0	122.0

8. Визуализируем различие в медицинских расходах в зависимости от возраста для разных кластеров отдельно для курящих и некурящих застрахованных.

```
In [88]: # Построим визуализацию разброса расходов на медицинское обслуживание
# в зависимости от возраста отдельно для некурящих и для курящих smoker = 0
g = sns.FacetGrid(insurance_df, row="smoker", hue='cluster')
g = g.map(plt.scatter, "age", "charges").add_legend()
# Зададим размер отображения графиков
g.fig.set_size_inches(10,5)
```



Пример 3. Набор данных sms

Третий набор данных – набор данных sms.

Это данные, полученные на основе методов ОЕЯ – обработки естественного языка. Подготовленные данные возьмем из открытого репозитория на github https://github.com/stedy/Machine-Learning-with-R-datasets Набор данных **snsdata.csv**.

Описание переменных набора:

gradyear – год выпуска;

gender – пол;

age – возраст;

friends – число друзей в социальной сети;

basketball, football, soccer, softball, volleyball, swimming, cheerleading, baseball, tennis, sports, cute, sex, sexy, hot, kissed, dance, band, marching, music, rock, god, church, jesus, bible, hair, dress, blonde, mall, shopping, clothes, hollister , abercrombiedie, death, drunk, drugs (переменные с 5 по 40) – частота встречаемости соответствующих слов в sms выпускника.

С точки зрения машинного обучения исходный текст сообщений бесполезен, его нужно преобразовать, чтобы использовать, например, алгоритмы кластеризации.

Для преобразования текста сообщений в форму пригодную для машинного обучения, можно использовать вариант редакционного расстояния – набор слов. В этом подходе порядок слов полностью игнорируется, а в основу кладутся просто счетчики слов. Этот процесс называется **векторизацией**. Векторы получаются очень большими, так как в каждый вектор входит столько элементов, сколько есть слов во всем наборе данных.

Данные sms были предварительно обработаны: вначале были отобраны слова, которые встречались в сообщениях максимально часто, из них были выбраны слова,

характеризующие пять групп интересов подростков: религия, романтические отношения, мода, внеклассные занятия (хобби) и асоциальные интересы. Таких слов индикаторов интересов оказалось 36.

Переменная, соответствующая конкретному слову, показывает, как часто оно встречается в сообщениях определенного выпускника.

Данные обезличенные, в них есть пропуски и выбросы.

1. Загрузим файл с данными **snsdata.csv.**

In [13]:
```
# Импортируем библиотеки
from sklearn.cluster import KMeans
import pandas as pd

# Загружаем набор данных
sms_df = pd.read_csv('snsdata.csv')
# Копируем столбцы с частотой слов в отдельный набор
interests = pd.DataFrame(sms_df.iloc[:, 4:39])
interests.head()
```

Out[13]:

	basketball	football	soccer	softball	volleyball	swimming	cheerleading	baseball	tennis	sports	...	dress	blonde	mall	shopping	clothes	holl
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
1	0	1	0	0	0	0	0	0	0	0	...	4	0	1	0	0	
2	0	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	2	0	

5 rows × 35 columns

2. Предварительно подготовим данные для анализа. Единообразие данных, которые используются для кластерного анализа, позволяет улучшить его результаты. Проведем стандартизацию данных (в стандартизованных данных среднее значение равно нулю, стандартное отклонение 1)

In [51]:
```
# Стандартизация данных
from sklearn.preprocessing import StandardScaler
colnames_int = list(interests) # массив из имен столбцов
scaler = StandardScaler()
interests_sc = pd.DataFrame(scaler.fit_transform(interests), columns= colnames_int)
interests_sc.head()
```

Out[51]:

	basketball	football	soccer	softball	volleyball	swimming	cheerleading	baseball	tennis	sports	...	dress	blonde	mall
0	-0.332217	-0.357697	-0.242874	-0.217928	-0.22367	-0.259971	-0.207327	-0.201131	-0.168939	-0.297123	...	-0.246906	-0.050937	-0.369915
1	-0.332217	1.060049	-0.242874	-0.217928	-0.22367	-0.259971	-0.207327	-0.201131	-0.168939	-0.297123	...	8.653277	-0.050937	1.067392
2	-0.332217	1.060049	-0.242874	-0.217928	-0.22367	-0.259971	-0.207327	-0.201131	-0.168939	-0.297123	...	-0.246906	-0.050937	-0.369915
3	-0.332217	-0.357697	-0.242874	-0.217928	-0.22367	-0.259971	-0.207327	-0.201131	-0.168939	-0.297123	...	-0.246906	-0.050937	-0.369915
4	-0.332217	-0.357697	-0.242874	-0.217928	-0.22367	-0.259971	-0.207327	-0.201131	-0.168939	-0.297123	...	-0.246906	-0.050937	-0.369915

5 rows × 35 columns

3. Проведем кластеризацию методом k-средних с числом кластеров равным 5

```
In [52]: # Описываем модель
model = KMeans(n_clusters=5)

# Проводим моделирование
model.fit(interests_sc)
# Добавляем в исходный набор новый столбец с номером кластера наблюдения
sms_df['cluster'] = model.fit_predict(interests_sc)
sms_df.head()
```

Out[52]:

	gradyear	gender	age	friends	basketball	football	soccer	softball	volleyball	swimming	...	mall	shopping	clothes
0	2006	M	18.982	7	0	0	0	0	0	0	...	0	0	0
1	2006	F	18.801	0	0	1	0	0	0	0	...	1	0	0
2	2006	M	18.335	69	0	1	0	0	0	0	...	0	0	0
3	2006	F	18.875	0	0	0	0	0	0	0	...	0	0	0
4	2006	NaN	18.995	10	0	0	0	0	0	0	...	0	2	0

5 rows × 41 columns

4. Частоту использования в смс отдельных слов для кластера наиболее корректно характеризует центроид соответствующего кластера

```
In [53]: # Выведем на просмотр информацию о центрах кластеров
cluster_centers = model.cluster_centers_

cluster_centers
```

```
Out[53]: array([[ 1.58325356e-01,   2.27893455e-01,   1.03914141e-01,
                  7.28719462e-02,   1.87558332e-01,   2.53483344e-01,
                  3.84416117e-01,   2.91421521e-02,   1.32068452e-01,
                  9.87792448e-02,   3.74741640e-01,   2.54442852e-02,
                  1.16086881e-01,   4.02018409e-01,   3.99933608e-02,
                  2.21838772e-01,  -9.08674343e-02,  -9.35957638e-02,
                  1.36251154e-01,   5.61099773e-02,   3.17964852e-02,
                  4.25601835e-03,   1.78080973e-02,  -4.27455465e-02,
                  4.47766983e-01,   1.52787660e-01,   6.21670739e-02,
                  6.20083424e-01,   7.81033998e-01,   5.79090603e-01,
                  4.13730941e+00,   3.98350903e+00,   5.51403727e-02,
                  1.18296549e-01,   5.19369814e-02],
```

Массив значений центроидов построенных кластеров (первая строка массива)

```
[ -1.65956198e-01,  -1.62716858e-01,  -9.01478905e-02,
  -1.13313569e-01,  -1.16142800e-01,  -1.06937332e-01,
  -1.14323459e-01,  -1.08483492e-01,  -5.32974087e-02,
  -1.30724866e-01,  -1.87726342e-01,  -9.61579560e-02,
  -9.07775272e-02,  -1.37458098e-01,  -1.36726193e-01,
  -1.58812828e-01,  -7.58207116e-02,  -3.77010459e-02,
  -1.49894235e-01,  -1.21683418e-01,  -1.20993141e-01,
  -1.59284053e-01,  -8.76107834e-02,  -6.99319349e-02,
  -2.00636484e-01,  -1.44024175e-01,  -2.91261135e-02,
  -1.87449580e-01,  -2.33185072e-01,  -1.89063072e-01,
  -1.55846874e-01,  -1.48613943e-01,  -9.40689834e-02,
  -8.22795712e-02,  -8.37584393e-02],
[ -3.32217264e-01,   2.47779503e+00,  -2.42874104e-01,
  -2.17927819e-01,  -2.23669630e-01,   1.67433426e+00,
  -2.07327097e-01,  -2.01130618e-01,  -1.68938902e-01,
  -2.97123387e-01,   8.43855726e-01,   1.93955434e+01,
   1.62589738e+00,  -2.64224954e-01,  -2.02619370e-01,
   4.94457044e-01,   6.26046293e-01,  -1.41421093e-01,
  -5.89161280e-01,   1.05039699e+00,   7.09847711e+00,
  -2.97556823e-01,   1.52644659e+00,  -1.04247248e-01,
   1.05446933e+01,   4.20318551e+00,   1.68307291e+02,
  -3.69914728e-01,  -4.87314241e-01,   1.80161279e+00,
  -2.01476270e-01,  -1.83031743e-01,   9.31280745e+00,
   1.34750994e+01,   1.48127445e+01],
```

Массив значений центроидов построенных кластеров (вторая и третья строки массива)

```
[  4.92555591e-01,   4.73371846e-01,   2.76613032e-01,
   3.54871362e-01,   3.61177715e-01,   2.99562034e-01,
   3.19751283e-01,   3.25202945e-01,   1.49093592e-01,
   3.17616312e-01,   5.17744471e-01,  -7.82746265e-03,
   2.08171122e-01,   3.72783590e-01,  -5.26508948e-02,
   4.37532715e-01,   1.72925452e-01,   1.11867830e-01,
   3.04585247e-01,   2.08097134e-01,   3.53795846e-01,
   5.36652190e-01,   2.93113324e-01,   2.35499704e-01,
   2.48517114e-01,   3.93890227e-01,   3.34635736e-02,
   4.85913956e-01,   6.61812947e-01,   3.79825804e-01,
  -5.98421807e-02,  -7.80725912e-02,   6.54981948e-02,
   1.26918617e-01,   2.45162771e-02],
[  4.46492601e-01,   4.30962316e-01,   1.64204362e-01,
   2.12225218e-01,   1.28099397e-01,   2.63380304e-01,
   1.79849353e-01,   3.40288086e-01,   1.28341311e-01,
   8.53883631e-01,   5.86250064e-01,   2.29025557e+00,
   6.30112806e-01,   3.52413779e-01,   3.56474849e+00,
   5.89411385e-01,   7.11174015e-01,   2.24247718e-01,
   1.36215601e+00,   1.42148237e+00,   4.32724270e-01,
   1.34557566e-01,   7.15369046e-02,   1.02004917e-01,
   2.64027516e+00,   5.95914438e-01,   2.23208102e-01,
   5.64956570e-01,   3.04527326e-01,   1.35967398e+00,
   1.91758990e-01,   2.89278267e-01,   1.72585296e+00,
   9.64954616e-01,   1.75635859e+00]])
```

Массив значений центроидов построенных кластеров (четвертая и пятая строки массива)

5. Запишем во фрейм данных **interests_1** значения центроида первого кластера и соответствующие им слова, проведем упорядочение по частоте использования слов в sms первого кластера.

```
In [55]: # Создадим набор данных по первому кластеру
         # Первый столбец - частота упоминания слов в первом кластере
         interests_1 = pd.DataFrame (cluster_centers[0])
         interests_1.rename(columns={0: 'freq'}, inplace=True)
         # Второй столбец - слова
         interests_1['word'] = colnames_int
         # Проведем упорядочение набора по столбцу freq по убыванию ascending=False
         interests_1.sort_values(['freq'], ascending=False).head(10)
```

Out[55]:

	freq	word
30	4.137309	hollister
31	3.983509	abercrombie
28	0.781034	shopping
27	0.620083	mall
29	0.579091	clothes
24	0.447767	hair
13	0.402018	hot
6	0.384416	cheerleading
10	0.374742	cute
5	0.253483	swimming

6. Аналогично можно построить фреймы данных, анализирующих частоту встречаемости слов в sms, для остальных кластеров.

Машинное обучение на основе метода k-средних

В заключении раздела о методе k-средних на языке Python проведем обучение по этому методу на данных **iris.csv**.

1. Проведем загрузку данных и подключим библиотеки для кластеризации методом k-средних.

```
[25] from sklearn.cluster import KMeans
     import pandas as pd

     # Загружаем набор данных
     iris_df = pd.read_csv('iris.csv')
     # Выводим на просмотр первые строки набора
     iris_df.head()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa

2. Преобразуем переменную Species в фактор.

```
[52] # Копируем первые четыре столбца в новую переменную
     iris_df1 = pd.DataFrame(iris_df[['Sepal.Length','Sepal.Width', 'Petal.Length', 'Petal.Width']])
     # Добавляем переменную класс как фактор
     iris_df1['species'] = pd.factorize(iris_df['Species'])[0]
     iris_df1.head()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	species
1	5.1	3.5	1.4	0.2	0
2	4.9	3.0	1.4	0.2	0
3	4.7	3.2	1.3	0.2	0
4	4.6	3.1	1.5	0.2	0
5	5.0	3.6	1.4	0.2	0

3. Проверим версию библиотеки sklearn (должна быть не ранее версии 0.21.1).

```
[66] import sklearn
     sklearn.__version__
```

```
'0.21.3'
```

4. Подключим из библиотеки **sklearn** необходимые методы. Исходные данные разбиваются на две части: тренировочную (в нашем случае 70% набора) и тестовую (оставшиеся 30%). Алгоритму машинного обучения предъявляются только тренировочные данные. На тестовых данных контролируется качество обучения. Механизм машинного обучения подразумевает, что данные размеченные, т.е. данные, на которых можно научиться предсказывать отклик, и для этих данных должны быть известны значения отклика. Для ирисов по первым четырем переменным, характеризующим цветки ириса, предсказывается класс ириса (species).

```
[101] from sklearn.model_selection import cross_validate
      from sklearn.model_selection import train_test_split
      from sklearn.metrics import accuracy_score
      from sklearn.metrics import classification_report
      # Разбиваем исходное множество данных на тренировочную
      # и тестовую выборки (в соотношении 7/3) и соответствующие им верные значения классов
      train_data, test_data, train_labels, test_labels = train_test_split(
          iris_df1[['Sepal.Length','Sepal.Width','Petal.Length','Petal.Width']],
          iris_df1[['species']], test_size = 0.3, random_state = 69)
      model = KMeans(n_clusters=3)
      # Проводим кластеризацию на тренировочной выборке
      model.fit(train_data)
      # Определяем кластеры по построенной модели на тестовых данных
      model_predictions = model.predict(test_data)
      # С помощью метрики accuracy сравниваем предсказанные значения
      # с исходной классификацией тестовой выборки
      accuracy_score(test_labels, model_predictions)
```

```
0.8444444444444444
```

5. Отчет по качеству построенной модели можно проинтерпретировать с использованием метрик качества (смотри теоретический раздел)

```
[102] print(classification_report(test_labels, model_predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.67      0.83      0.74        12
           2       0.86      0.71      0.77        17

    accuracy                           0.84        45
   macro avg       0.84      0.85      0.84        45
weighted avg       0.86      0.84      0.85        45
```

Задания к лабораторной работе

1) Сформировать структуру данных для анализа объекта исследования и заполнить ее.

2) Разработать методику и программу для сегментации объекта исследования.

3) Провести сегментацию объекта исследования по количественным признакам на основе выбранного метода кластеризации внутри каждого кластера.

4) Провести пространственную (по странам, регионам, городам, предметным областям, корпорациям...) сегментацию объекта исследования.

5) Провести временную сегментацию на основе тенденций «рост», «падение», «стабильность».

6) Сформулировать выводы и объяснить результаты.

Варианты заданий:

1. Кластеризация рынка технологий Big Data.
2. Кластеризация языков программирования.
3. Кластеризация рынка IT профессий.
4. Кластеризация рынка рекламных технологий (ТВ, радио, Интернет).
5. Кластеризация рынка IT продуктов.
6. Кластеризация рынка IT технологий разработки ПО.

7. Кластеризация рынка e-learning в области ИТ.
8. Кластеризация рынка распределенных систем.
9. Кластеризация рынка IoT.
10. Кластеризация регионов по индексам развития информационного общества.
11. Кластеризация средств визуального моделирования.
12. Кластеризация технических текстов при разработке ПО.
13. Кластеризация программных проектов по метрикам качества.
14. Кластеризация абитуриентов ИТ-направлений УВО.
15. Кластеризация покупателей по типу поведения.
16. Кластеризация ИТ-разработчиков по типу поведения.
17. Кластеризация пользователей ИТ-продуктов.
18. Кластеризация отзывов на ИТ-продукт.
19. Своя тема.

Контрольные вопросы

1. В чем состоит цель кластеризации? Приведите формальную постановку задачи кластеризации.
2. Приведите перечень и особенности методов кластеризации.
3. Какие метрики применяют в кластеризации?
4. Приведите математические характеристики кластеров и меры качества результатов кластеризации.
5. Охарактеризуйте этапы кластерного анализа выбранного метода.
6. Как импортируются данные в формате .csv?
7. Какие параметры, аргументы и методы у класса KMeans?
8. Как получить доступ к данным о кластерах наблюдений?
9. Как получить доступ к данным о центроидах?
10. Какие преобразования исходных данных выполнялись и почему?
11. Какие функции использовались для визуализации полученных результатов?

Список источников информации:

1. Открытый курс машинного обучения. Тема 7. Обучение без учителя: PCA и кластеризация / Open Data Science – URL: https://habr.com/ru/company/ods/blog/325654/
2. Факторный, дискриминантный и кластерный анализ: Пер. с англ./Дж.-О. Ким, Ч. У. Мьюллер, У. Р. Клекка и др.; Под ред. И. С. Енюкова. — М.: Финансы и статистика, 1989. - 215 с.
3. Рашка, С. Python и машинное обучение: крайне необходимое пособие по новейшей предсказательной аналитике, обязательное для более глубокого понимания методологии машинного обучения / С. Рашка ; пер. с англ. А. В. Логунова. – Москва : ДМК Пресс, 2017. – 418 с. – URL: http://znanium.com/catalog/product/1027758
4. Коэльо, Л. П. Построение систем машинного обучения на языке Python / Л. П. Коэльо, В. Ричарт ; пер. с англ. А. А. Слинкина. – 2-е изд. – Москва : ДМК Пресс, 2016. – 302 с. – URL: http://znanium.com/catalog/product/1027824

5. Visualizing K-Means Clustering. – URL: https://www. naftaliharris.com/blog/visualizing-k-means-clustering/
6. A Tutorial on Clustering Algorithms. – URL: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html
7. K Means. – URL: http://stanford.edu/~cpiech/cs221/handouts/kmeans.html
8. Машинное обучение: от Ирисов до Телекома. – URL: https:// habr.com/ru/company/billing/blog/334738/
9. Метрики в задачах машинного обучения – URL: https:// habr.com/ru/company/ods/blog/328372/
10. Рындина, С. В. Бизнес-аналитика: визуализация данных / С. В. Рындина. – Пенза : Изд-во ПГУ, 2018. –70 с.
11. Описание библиотеки matplotlib. – URL: https://matplotlib.org/
12. Описание библиотеки seaborn. – URL: https://seaborn.pydata. org/
13. Описание библиотеки NumPy. – URL: https://numpy.org/
14. Описание библиотеки pandas. – URL: https://pandas.pydata.org/
15. Описание библиотеки scikit-learn. – URL: https://scikit-learn. org/stable/index.html
16. Описание реализации метода k-средних на языке Python: sklearn.cluster.KMeans – URL: https://scikit-learn.org/stable/modules/ generated/sklearn.cluster.KMeans