Лекция 12. Кластерный анализ в БигДата

1. Кластерный анализ (на примере сегментации потребителей)

Мы знаем, что Земля – это одна из 8 планет, которые вращаются вокруг Солнца. Солнце – это всего лишь звезда среди порядка 200 миллиардов звезд в галактике Млечный Путь. Очень тяжело осознать это число. Зная это, можно сделать предположение о количестве звезд во вселенной – приблизительно 4*10^22. Мы можем видеть около миллиона звезд на небе, хотя это всего лишь малая часть от всего фактического количества звезд. Итак, у нас появилось два вопроса:

1. Что такое галактика?
2. И какая связь между галактиками и темой (кластерный анализ)

Галактика – это скопление звезд, газа, пыли, планет и межзвездных облаков. Обычно галактики напоминают спиральную или эллиптическую фигуру. В пространстве галактики отделены друг от друга. Огромные черные дыры чаще всего являются центрами большинства галактик.

Как мы будем обсуждать в следующем разделе, есть много общего между галактиками и кластерным анализом. Галактики существуют в трехмерном пространстве, кластерный анализ – это многомерный анализ, проводимый в n-мерном пространстве.

Заметка: *Черная дыра – это центр галактики. Мы будем использовать похожую идею в отношении центроидов для кластерного анализа.*

Кластерный анализ

Предположим, вы глава отдела по маркетингу и взаимодействию с потребителями в телекоммуникационной компании. Вы понимаете, что все потребители разные, и что вам необходимы различные стратегии для привлечения различных потребителей. Вы оцените мощь такого инструмента как сегментация клиентов для оптимизации затрат. Для того, чтобы освежить ваши знания кластерного анализа, рассмотрим следующий пример, иллюстрирующий 8 потребителей и среднюю продолжительность их разговоров (локальных и международных). Данные представлены в таблице 1.

Таблица 1. Данные по потребителям

№ клиента	Сред.прод.звонка	Сред.прод.межд.звонка
1	2	2
2	1	2
3	1	3
4	3	2
5	4	5
6	4	4
7	5	5
8	6	5

Для лучшего восприятия нарисуем график, где по оси x будет откладываться средняя продолжительность международных разговоров, а по оси y — средняя продолжительность локальных разговоров (рис. 1).

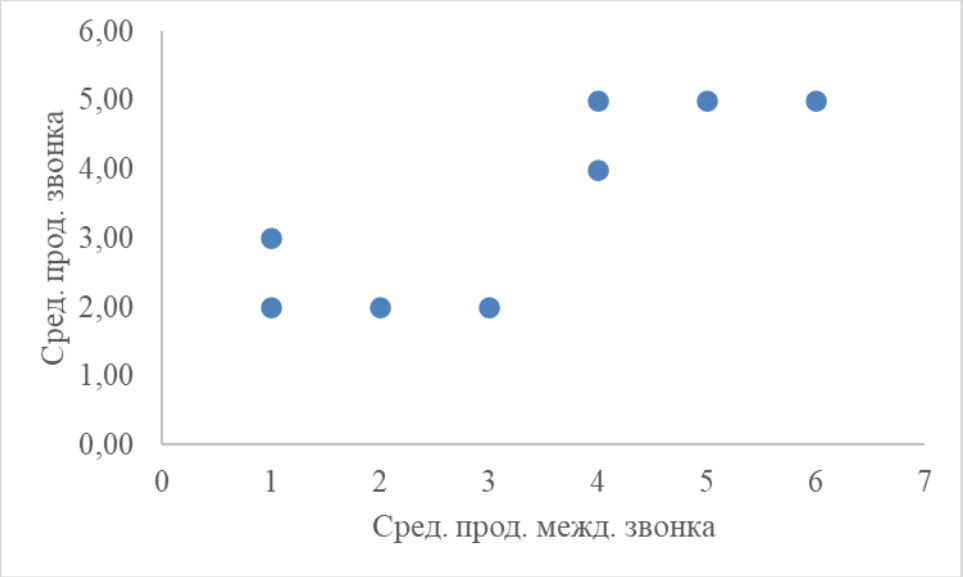

Рисунок 1. График средней продолжительности разговоров

Заметка: *Это похоже на анализ расположения звезд на ночном небе (здесь звезды заменены потребителями). В дополнение, вместо трехмерного пространства у нас двумерное, заданное продолжительностью локальных и международных разговоров, в качестве осей x и y.*

Сейчас, разговаривая в терминах галактик, задача формулируется так – найти положение черных дыр; в кластерном анализе они называются центроидами. Для обнаружения центроидов мы начнем с того, что возьмем произвольные точки в качестве положения центроидов.

Евклидово расстояние для нахождения Центроидов для Кластеров

В нашем случае два центроида (C1 и C2) мы произвольным образом поместим в точки с координатами (1, 1) и (3, 4). Почему мы выбрали именно эти два центроида? Визуальное отображение точек на графике показывает нам, что есть два кластера, которые мы будем анализировать. Однако, впоследствии мы увидим, что ответ на этот вопрос будет не таким уж простым для большого набора данных. Далее, мы измерим расстояние между центроидами (C1 и C2) и всеми точками на графике использую формулу Евклида для нахождения расстояния между двумя точками.

$$Distance = \sqrt{\left(X_{centroid\ C_1} - X_i\right)^2 + \left(Y_{centroid\ C_1} - Y_i\right)^2}$$

Примечание: Расстояние может быть вычислено и по другим формулам, например,

1. квадрат евклидова расстояния – для придания веса более отдаленным друг от друга объектам
2. манхэттенское расстояние – для уменьшения влияния выбросов
3. степенное расстояние – для увеличения/уменьшения влияния по конкретным координатам
4. процент несогласия – для категориальных данных

5. и др.

Колонка 3 и 4 (расстояние между C1 и C2) и есть расстояние, вычисленное по этой формуле. Например, для первого потребителя

$$Distance\ from\ C_1 = \sqrt{(1-2)^2 + (1-2)^2} = \sqrt{2} = 1.41$$

Принадлежность к центроидам (последняя колонка) вычисляется по принципу близости к центроидам (C1 и C2). Первый потребитель ближе к центроиду №1 (1.41 по сравнению с 2.24), следовательно, принадлежит к кластеру с центроидом C1 (табл. 2).

Таблица 2. Данные по потребителям с принадлежностью к кластерам.

Сред.прод. звонка	Сред.прод. межд.звонка	Расстояние от C1	Расстояние от C2	Принадлежность к кластеру
2	2	1.41	2.24	C1
1	2	1.00	2.83	C1
1	3	2.00	2.24	C1
3	2	2.24	2.00	C2
4	5	5.00	1.41	C2
4	4	4.24	1.00	C2
5	5	5.66	2.24	C2
6	5	6.40	3.16	C2

Ниже график, иллюстрирующий центроиды C1 и C2 (изображенные в виде голубого и оранжевого ромбика). Потребители изображены цветом соответствующего центроида, к кластеру которого они были отнесены (рис. 2).

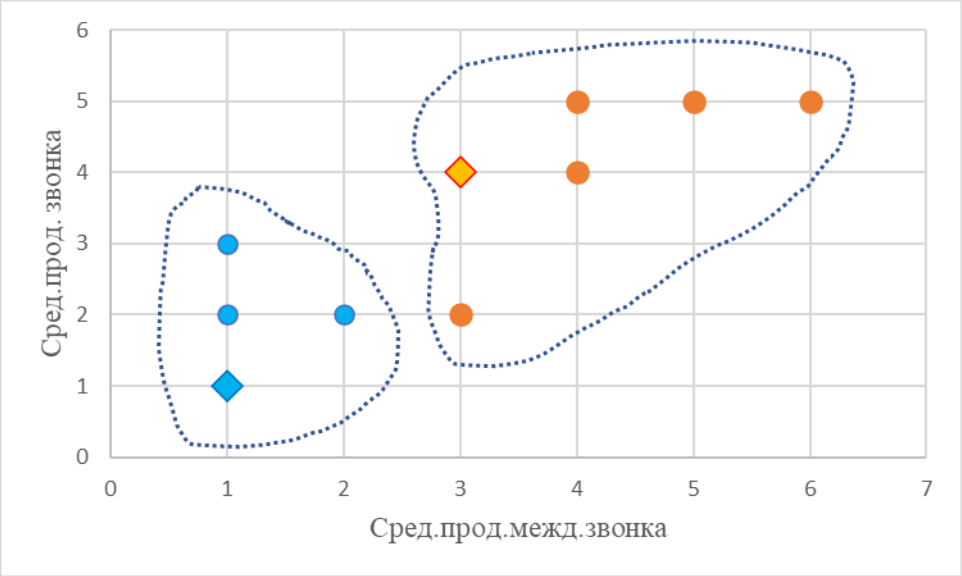

Рисунок 2. Итерация 1

Так как мы произвольным образом выбрали центроиды, вторым шагом мы сделать этот выбор итеративным. Новая позиция центроидов выбирается как средняя для точек соответствующего кластера. Так, например, для первого центроида (это потребители 1, 2 и 3). Следовательно, новая координата x для центроида C1 это средняя координат x

этих потребителей (2+1+1)/3 = 1.33. Мы получим новые координаты для C1 (1.33, 2.33) и C2 (4.4, 4.2). Новый график на рис. 3.

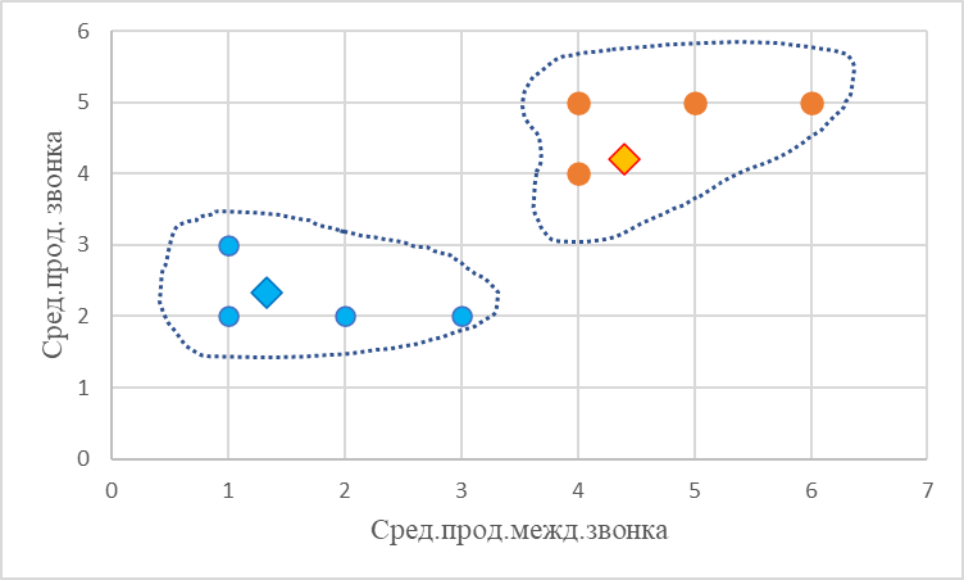

Рисунок 3. Итерация 2

В конце концов, мы поместим центроиды в центр соответствующего кластера (рис. 4).

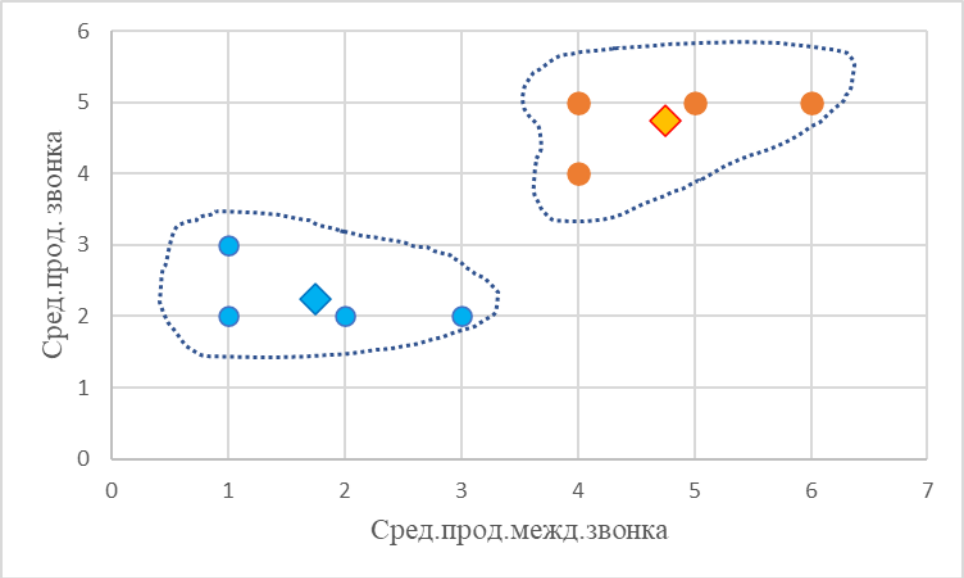

Рисунок 4. Последняя итерация

Позиции наших черных дыр (центров кластеров) в нашем примере C1 (1.75, 2.25) и C2(4.75, 4.75). Два кластера выше подобны двум галактикам, разделенным в пространстве друг от друга.

Итак, рассмотрим примеры дальше. Пусть перед нами стоит задача по сегментации потребителей по двум параметрам: возраст и доход. Предположим, что у нас есть 2 потребителя с возрастом 37 и 44 лет и доходом в $90,000 и $62,000

соответственно. Если мы хотим измерить Евклидово расстояние между точками (37, 90000) и (44, 62000), мы увидим, что в данном случае переменная доход «доминирует» над переменной возраст и ее изменение сильно сказывается на расстоянии. Нам необходима какая-нибудь стратегия для решения данной проблемы, иначе наш анализ даст неверный результат. Решение данной проблемы — это приведение наших значений к сравнимым шкалам. Нормализация – вот решение нашей проблемы.

Нормализация данных

Существует много подходов для нормализации данных. Например, нормализация минимума-максимума. Для данной нормализации используется следующая формула

$$X^* = \frac{X - \min(X)}{\max(X) - \min(X)}$$

в данном случае X* — это нормализованное значение, min и max – минимальная и максимальная координата по всему множеству X (Примечание, данная формула располагает все координаты на отрезке [0;1]) Рассмотрим наш пример, пусть максимальный доход $130000, а минимальный — $45 000. Нормализованное значение дохода для потребителя A равно

$$Y^* = \frac{(90\,000 - 45\,000)}{(130\,000 - 45\,000)} = 0.529$$

Мы сделаем это упражнение для всех точек для каждых переменных (координат). Доход для второго потребителя (62 000) станет 0.2 после процедуры нормализации. Дополнительно, пусть минимальный и максимальный возрасты 23 и 58 соответственно. После нормализации возрасты двух наших потребителей составит 0.4 и 0.6.

Легко увидеть, что теперь все наши данные расположены между значениями 0 и 1. Следовательно, у нас теперь есть нормализованные наборы данных в сравнимых шкалах.

Запомните, перед процедурой кластерного анализа необходимо произвести нормализацию.

2. 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 рисунке 5,а представлен пример контролируемого обучения: здесь для того, чтобы найти лучшую функцию, соответствующую представленным точкам, используется метод регрессии. В то же время при неконтролируемом обучении (рис. 5,б) входные данные разделяются на основе представленных характеристик, а предсказание свойств основывается на том, какому кластеру принадлежит пример.

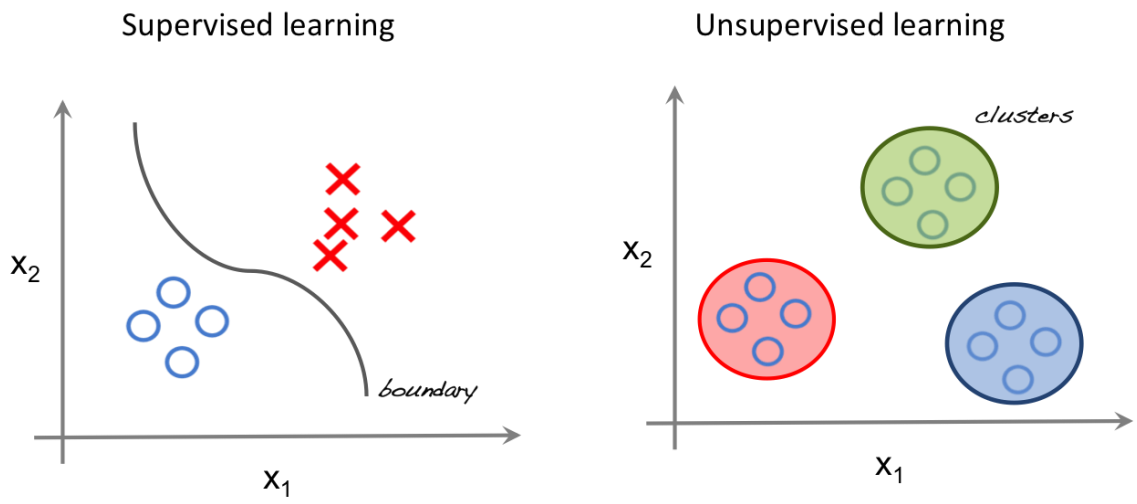

Рисунок 5. а) обучение с учителем, б) обучение без учителя

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. 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 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 2]
['setosa' 'versicolor' 'virginica']
```

На рисунке 6 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.

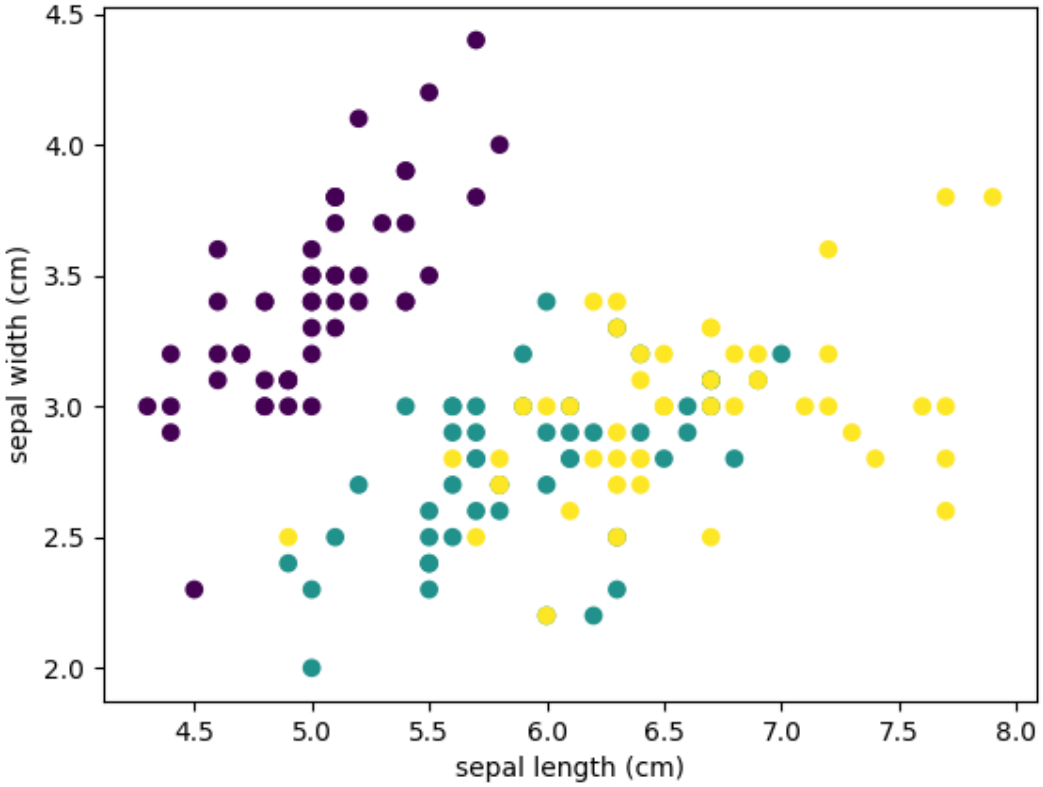

Рисунок 6. Диаграмма видов ириса

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 7.

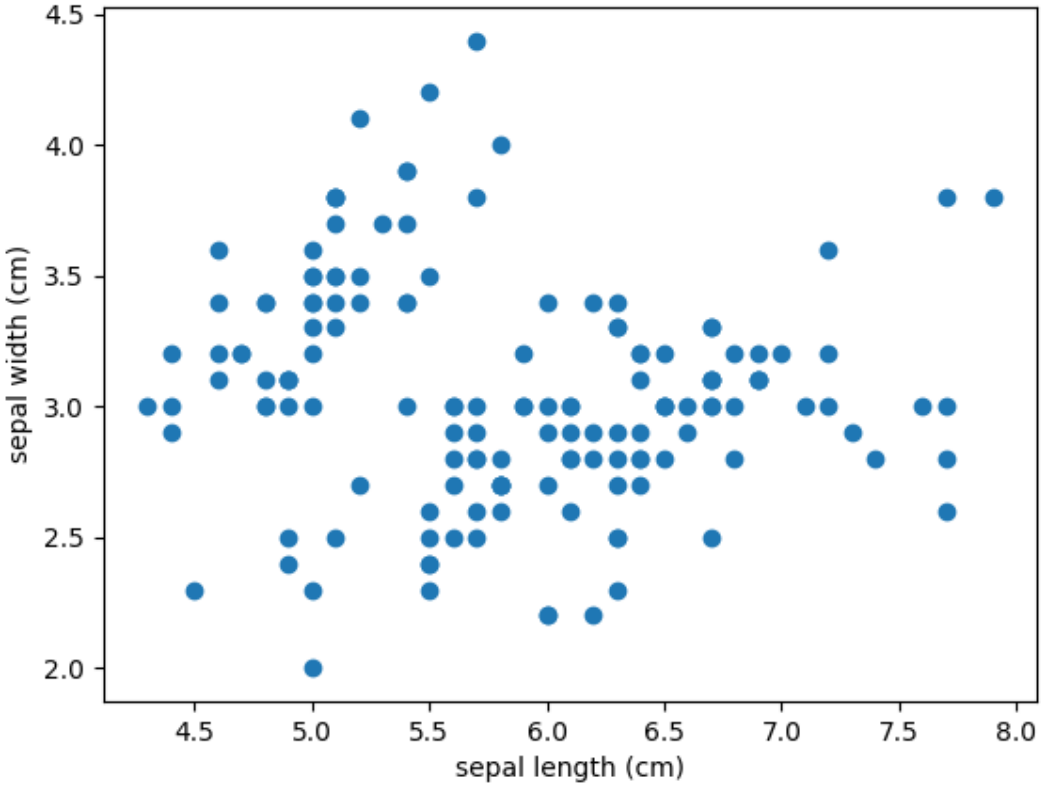

Рисунок 7. Диаграмма видов ириса без цветовой дифференциации

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

3. Метод k-средних

Наиболее популярным алгоритмом кластеризации данных является метод k-средних. 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 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.

4. Иерархическая кластеризация

Иерархическая кластеризация, 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 (рис. 8). При этом исходно число кластеров не задавалось.

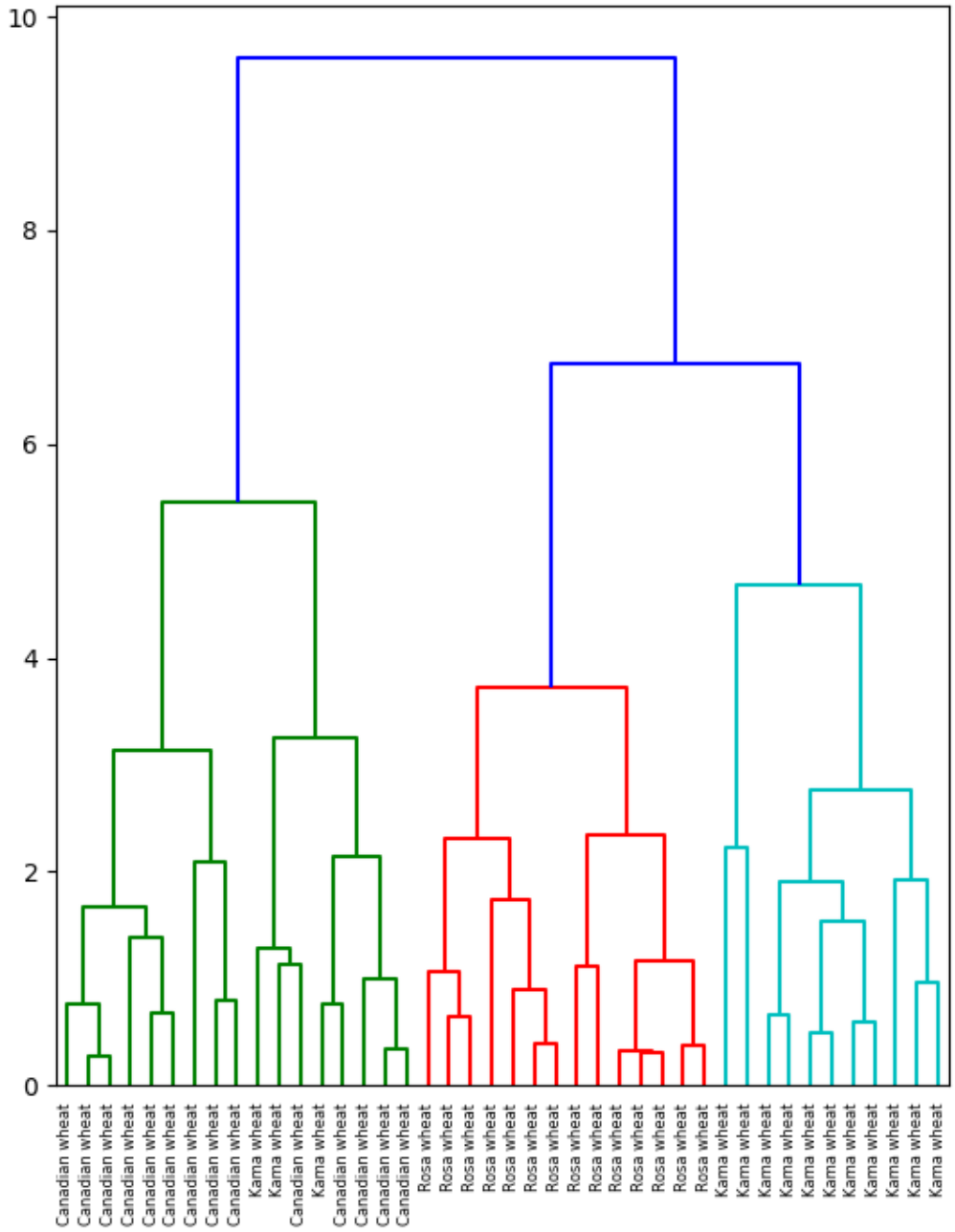

Рисунок 8. Дендрограмма

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 линейна для метода k-средних (O(n)) и квадратична для метода иерархической кластеризации ($O(n^2)$)

- 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.

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 (рис. 9).

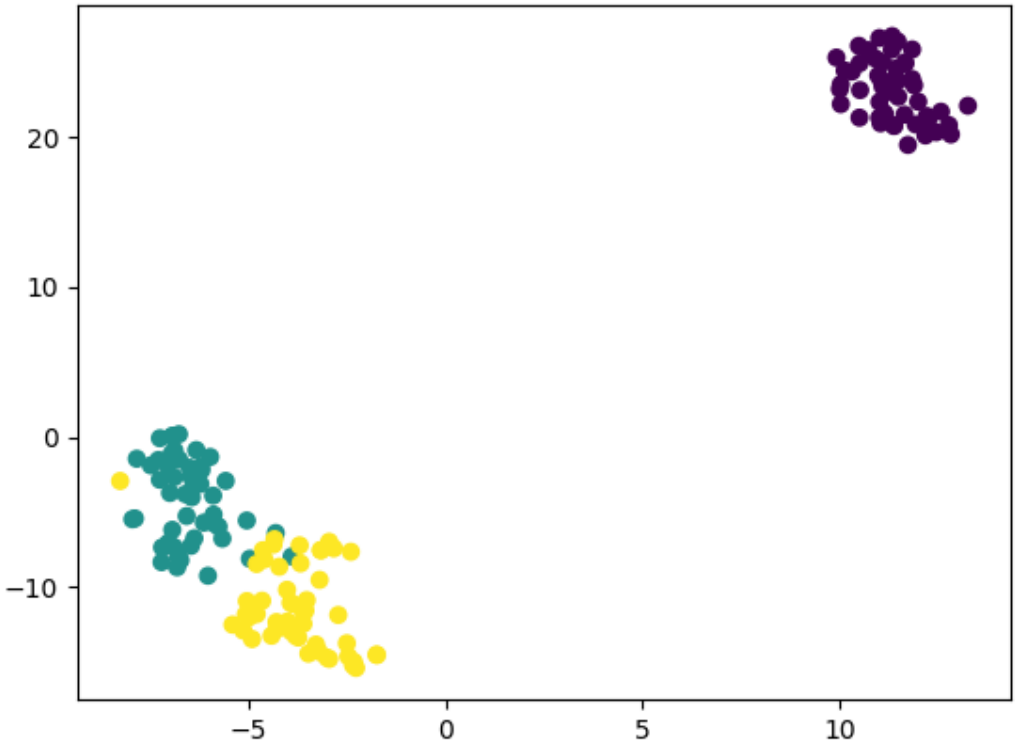

Рисунок 9. Классы ириса на точечной диаграмме

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()
```

В результате получаем график, как на рис. 10.

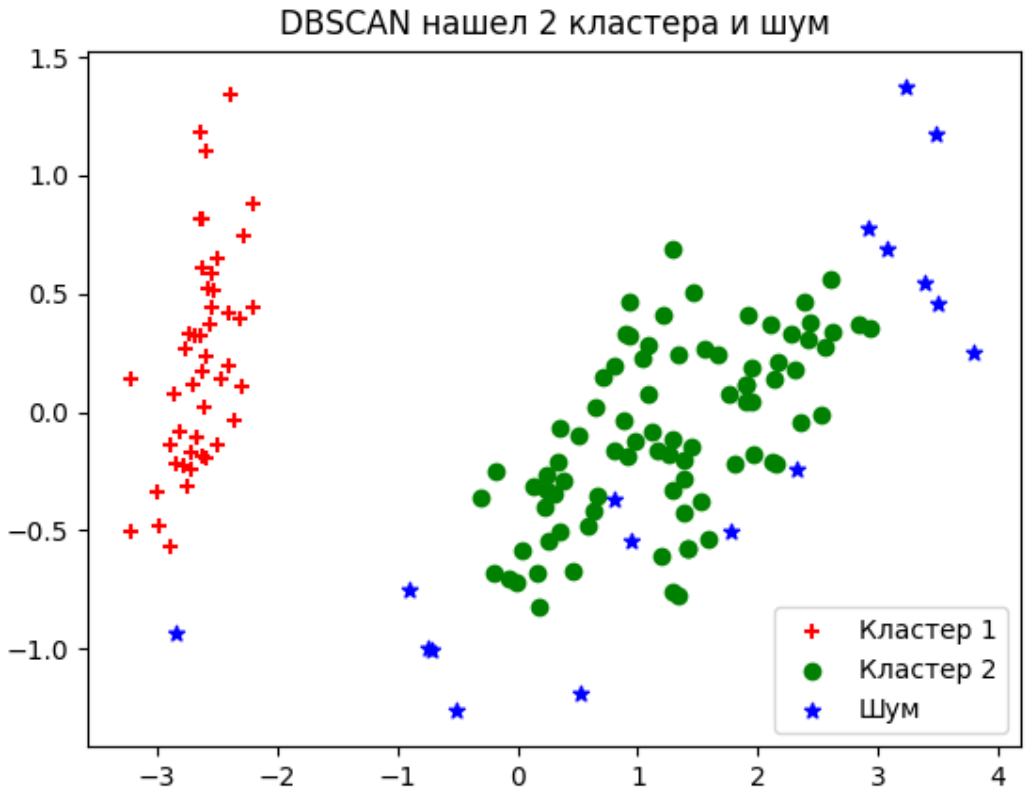

Рисунок 10. 2 кластера, полученные методом DBSCAN

Список использованных источников:

1. Big Data, Data Mining, and Machine Learning: Value Creation for Business Leaders and Practitioners (Wiley and SAS Business Series) 1st Edition.

2. Кластерный анализ (на примере сегментации потребителей) часть 1. URL: https://habr.com/ru/post/228477/ (Дата обращения: 22.09.2020)

3. Обучение без учителя: 4 метода кластеризации данных на Python. URL: https://proglib.io/p/unsupervised-ml-with-python (Дата обращения: 22.09.2020)