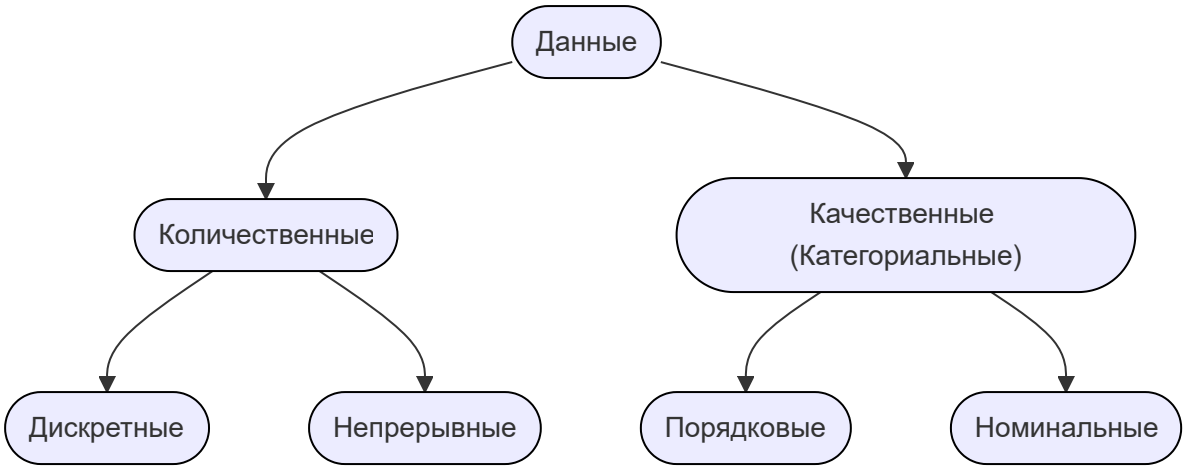
Коллоквиум 1

Тема 1. База

Типы данных



Дискретные

Целые значения.
Обычно результат счёта.

Непрерывные

float / диапазон.
Обычно результат измерения

Порядковые

Их нельзя складывать, но можно упорядочить.
S < M < L < XL < XXL
(Label encoding)

Номинальные

Их нельзя складывать и сравнивать
Бензин $\not<$ Дизель
(OneHot encoding)

Понятие однородности / неоднородности данных

Бытовое определение

Однородные данные

- Имеют одинаковую природу
  Сигнал, звук, изображение, изменение по времени какой-либо величины и т.д.

Неоднородные

- Смесь типов данных (числовые + категориальные)
- Данные из разных источников (разные страны)
- Данные с разным масштабом (сотни тысяч и десятки единиц)

Определение в ML

Однородные данные

- Каждая строка — независимый объект
- Порядок строк не важен (перемешивание не меняет свойства данных)
- Нет временных или пространственных зависимостей
- Все данные из одного распределения

Неоднородные

- Временные зависимости (порядок дней)
- Пространственные зависимости (изображение)
- Групповые зависимости (измерения внутри одного пациента зависимы)
- Разные распределение (данные собраны в разных условиях)

Как бороться с неоднородностью?

- **Для временных рядов:** Использовать специальные методы валидации (временные разбиения)
- **Для пространственных данных:** Учитывать пространственные autocorrelation
- **Для групповых данных:** Использовать групповую валидацию (GroupKFold)
- **Для разных распределений:** Техники domain adaptation (Перенос знаний с модели на модель)

Методы предобработки данных

Missing data

Пропущенные значения

- **Удаление**
  - Потеря информации
- **Заполнение**
  - **Числовые:** Среднее, медиана, мода
  - **Категориальные:** мода или категория *Unknown*

Encoding

Перевод категориальных данных в количественные

- **Label Ecoding**
  - М, Ж = 0, 1
    - Кодируем каждый признак целым числом
  - Модель может решить, что между числами есть порядок
- **One-Hot encoding**
  - М, Ж = [1, 0], [0, 1]
    - Каждая координата отвечает за конкретное значение. В данном случае первый столбец - is_M, второй - $is\_Ж$
  - Проклятие размерности
    - При большом количестве категорий матрица становится разреженной и большой, увеличивается риск переобучения

Масштабирование и нормализация

Уменьшение масштаба данных.

- **Стандартизирование**
  - $\frac{x - \mu}{\sigma}$
  - Сведение к среднему $\mu = 0$ и стандартному отклонению $\sigma = 1$
- **Min-max**
  - $\frac{x - \min}{\max - \min}$
  - Приведение значений к диапазону [0, 1]
  - *Сохраняет исходное распределение*

Преобразование признаков

Создание новых признаков из существующих для лучшего описания закономерностей.

- **Полиномиальные признаки**
  - Для учёта нелинейных зависимостей
  - $x^2,\ x^3,\ x_1 \cdot x_2$
- **Дискретизация**
  - Перевод непрерывного признака в категориальный / интервальный.
  - возраст → [0-18, 19-65, 66+]
- **Извлечение признаков**
  - **Из дат:** день недели, месяц, является ли выходным
  - **Из текста:** длина, наличие ключевых слов

- **Из взаимодействия признаков:** вычисление значений нового признака из нескольких других по формулам (цена, делённая на площадь, разность между выручкой и расходами и т. д.).

Работа с выбросами

Убирание значений, сильно отклоняющихся от остальных наблюдений.

- **Ящик с усами**
  - Берём 25 и 75 перцентиль; отнимаем / прибавляем $1.5 \cdot \mathrm{IQR}$ (интерквартильный размах) (a. k. a. делаем усы); Смотрим, что будет вне границ
    - $\begin{cases} x < Q_1 - 1.5 \cdot \mathrm{IQR} \\ x > Q_3 + 1.5 \cdot \mathrm{IQR} \end{cases}$
- **Z-score** метод (для нормально распределённого признака)
  - Признак приводят к стандартному нормальному распределению ($Z = \frac{X - \mu}{\sigma}$);
  - Определяют порог ($threshold$) (обычно 99.7% квантиль, т. е. $threshold = 3$);
  - Все значения, выходящие за заданный порог, считаются выбросами
    - $\begin{cases} Z > threshold \\ Z < -threshold \end{cases}$

Можем удалить, заменить на предельное значение или же как-то их преобразовать.

Типы задач, решаемых ИИ

С учителем

Есть размеченные данные. Пытаемся предсказать правильный ответ

Формальная постановка

D - множество пар $(x_i, y_i),$ где $x_i \in X, y_i \in Y$,

$L(Y, Y) \to \mathbb{R}$ - функция ошибки,

$M(\xi, X) \to Y$ - модель, ξ - вектор параметров модели.

Тогда задача обучения с учителем - задача поиска вектора ξ^*, такого что:

$$L(Y, M(\xi^*, X)) = L(Y, \hat{Y}) \to \min$$

Задачи:

- **Регрессия**
- **Классификация**
  - Бинарная, многоклассовая
- **Сегментация**

    - Разделение изображения на смысловые области
- **Ранжирование**
    - Упорядочивание объектов

Без учителя

Нет размеченных данных. Ищем скрытые структуры

Формальная постановка задачи кластеризации

X - множество объектов.

Пусть функция $\underline{L} : (X_1, X_2, \ldots, X_n) \to \mathbb{R}$, $(\overline{L} : (X_1, X_2, \ldots, X_n) \to \mathbb{R})$ - внутренняя метрика (внешняя метрика).

$M(\xi, X) \to (X_1, X_2, \ldots, X_n)$ - модель, ξ - вектор параметров модели.

Тогда задача кластеризации - поиск ξ, такого что

$$X_1, X_2, \ldots, X_n : \underline{L}(X_1, X_2, \ldots, X_n) \to \min$$

или

$$X_1, X_2, \ldots, X_n : \overline{L}(X_1, X_2, \ldots, X_n) \to \max$$

Задачи:

- **Кластеризация**
    - Группировка похожих объектов
- **Понижение размерности**
    - Сокращение числа признаков
- **Детекция аномалий**
    - Поиск выбросов
- **Генерация**
    - Создание новых данных (GAN)

С частичным контролем

Мало размеченных данных + много неразмеченных

- Обычно небольшое количество размеченных данных и большое количество неразмеченных.
- Неразмеченные данные помогают понять структуру пространства, а размеченные - определить границы классов

Обучение с подкреплением

Агент учится взаимодействовать со средой и получает "награды" за правильные действия

У среды есть множество всевозможных состояний S.
У агента - множество возможных действий A.
Функция награды за выбранное действие a в состоянии s, из-за которого среда перешла в состояние s'.

Обучение с подкреплением - задача получения агентом максимальной награды.

Формальная постановка задач

1. **Какие данные есть / могут быть получены?**
2. **Какой тип величины мы можем прогнозировать на основе данных?**
    1. Определение вида задачи
3. **Какой характер данных и зависимостей в них?**
    1. Структура данных, зависимости и их характер
4. **Какие у нас есть ресурсы (технические и человекоресурсы)?**
    1. Время обучения, память, GPU
    2. Эксперты для разметки данных и компетенции команды в ML
    3. Требования к точности и интерпретируемости
    4. Стоимость ошибки (FP, FN)
        1. Лучше, чтобы мы не дали кредит, чем дали и потеряли его.

Пример формальной постановки

Задача: Прогноз оттока клиентов банка

1. **Данные**:
    - 100K клиентов, 50 признаков (возраст, баланс, количество операций)
    - Есть пропуски в данных о доходе
2. **Целевая переменная**:
    - Бинарная: ушел/не ушел (классификация)
3. **Характер данных**:
    - Табличные данные, временные ряды операций
    - Наблюдения независимы (i.i.d.)
    - Признаки: числовые + категориальные
4. **Ресурсы**:
    - Сервер с 16ГБ RAM
    - Модель должна давать ответ < 1 секунды
    - Важна интерпретируемость (чтобы понимать причины оттока)

Тема 2. Регрессия

https://education.yandex.ru/handbook/ml/article/linear-models

Постановка регрессии как задачи оптимизации

Пусть дан датасет (X, y), где $y \in \mathbb{R}^N$ - вектор целевой переменной, а $X \in \mathbb{R}^{N \times M}$ - матрица M признаков.

Задача состоит в подборе линейной функции, "наилучшим образом" моделирующую линейную зависимость ($y = kx + b$) значения таргета y_i от фичи x_i. Тогда искомая функция будет иметь вид:

$$f(w, x_i) = (w, x_i) = w_0 + w_1 x_{i1} + w_2 x_{i2} + \cdots + w_M x_{iM} = \hat{y}_i$$

Формулировку "наилучшим образом" в данном контексте можно выразить например с помощью Евклидовой нормы:

$$L(f, X, y) = \frac{1}{N} \|y - f(w, X)\|^2 = \frac{1}{N} \sum_{k=1}^{N} (y_k - (w, x_k))^2$$

Линейная модель тем "лучше" моделирует зависимость чем меньше значение функционала L, а значит задача сводится нахождении вектора весов w^*, доставляющего минимум функционалу:

$$L_f(w) = \|y - f(w, X)\|^2 \to \min_w$$

Метрики и функции ошибки задач регрессии

Функция ошибки - функционал, используемый во время процесса оптимизации параметров
Метрика - функция, оценивающая результат, предсказанный моделью

Функции ошибок

1) MSE (Mean Squared Error)

$$\text{MSE}(y, \hat{y}) = \frac{1}{N} \sum_{k=1}^{N} (y_k - \hat{y_k})^2$$

2) MAE (Mean Absolute Error)

$$\text{MAE}(y, \hat{y}) = \frac{1}{N} \sum_{k=1}^{N} |y_k - \hat{y_k}|$$

Метрики качества

Довольно часто MSE и MAE используют как метрики. Однако на их базе существуют еще несколько метрик:

1) RMSE
Квадратный корень MSE

2) Коэффициент детерминации

$$R^2 = 1 - \frac{\sum\limits_{k=1}^{N}(y_k - \hat{y_k})^2}{\sum\limits_{k=1}^{N}(y_k - \overline{y})^2}$$

где $\overline{y}$ - среднее обучающей выборки (наилучшее константное предсказание с точки зрения MSE)

3) MAPE (Mean Absolute Percentage Error)

$$\mathrm{MAPE}(y, \hat{y}) = \frac{1}{N}\sum_{k=1}^{N}\frac{|y_k - \hat{y_k}|}{|y_k|}$$

Интерпретируемость и применимость метрик

Интерпретируемость модели - возможность объяснить её результаты заказчику

1) MSE
Интерпретируемость: метрика не ограничена сверху и возвращает значение на порядок большее, чем в данных, из-за чего её сложнее интерпретировать

Применимость: когда выбросов мало и их нужно сильно штрафовать

2) RMSE
Интерпретируемость: метрика теперь имеет тот же порядок, что и у данных, потому проще оценить разброс

Применимость: аналогична MSE

3) MAE
Интерпретируемость: средняя абсолютная ошибка

Применимость: когда выбросов много (метрика менее им подвержена, производная - константа).

4) R^2
Интерпретируемость: показывает какую долю дисперсии модель смогла предсказать
$R^2 = 1$ - идеальная модель
$R^2 = 0$ - предсказывает не лучше константного среднего
$R^2 < 0$ - предсказывает хуже чем просто среднее

Применимость: когда хотим сравнить модели на одном и том же наборе данных, желательно вместе с другими метриками.

5) MAPE

Интерпретируемость: средняя ошибка в процентах

Применимость: метрика сильнее штрафует отрицательные величины, данные не должны содержать нули

Вывод аналитического решения задачи линейной регрессии в векторной форме

при дифференцировании первое слагаемое можно сжато

$$X_{n\times k} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ x_{21} & x_{22} & \dots & x_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{k2} & \dots & x_{nk} \end{pmatrix}$$

$$\omega = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_k \end{pmatrix} \quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$$

$$\hat{y} = X\omega$$

Функция потерь

$$L = (X\omega - y)^2 \to \min$$

Преобразуем функцию потерь

$$L = (X\omega - y)^T(X\omega - y) = (X\omega)^T X\omega - (X\omega)^T y - y^T X\omega + y^T y$$

Используя свойства векторной алгебры получаем тождества

1. $y^T X\omega = (X\omega)^T y = \omega^T X^T y$
2. $(X\omega)^T X\omega = \omega^T X^T X\omega$

С учетом этих свойств, функция потерь принимает вид

$$L = \omega^T X^T X\omega - 2\omega^T X^T y + y^T y$$

Перейдем к решению задачи минимизации. Для этого продифференцируем функцию потерь по вектору весов

$$\frac{dL}{d\omega} = \frac{d}{d\omega}\omega^T X^T X\omega - 2\frac{d}{d\omega}\omega^T X^T y + \frac{d}{d\omega}y^T y = L_1 - 2L_2 + L_3$$

$$w^T X^T X w = w^T A w = (w_1, \dots, w_k) \begin{pmatrix} a_{11} & \dots & a_{1k} \\ \vdots & \ddots & \vdots \\ a_{k1} & \dots & a_{kk} \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_k \end{pmatrix} =$$

$$= (a_{11}w_1 + \ldots + a_{k1}w_k,\ \ldots,\ a_{1k}w_1 + \ldots + a_{kk}w_k) \begin{pmatrix} w_1 \\ \vdots \\ w_k \end{pmatrix} =$$

$$= w_1(a_{11}w_1 + \ldots + a_{k1}w_k) + \ldots + w_k(a_{1k}w_1 + \ldots + a_{kk}w_k) = \sum_{i=1}^{k} a_{ii}w_i^2 + 2\sum_{i=1}^{k-1}\sum_{j=i+1}^{k} a_{ij}w_i w_j$$

$$\frac{d}{dw_2}\begin{pmatrix} a_{11}w_1^2 & a_{12}w_1w_2 & \ldots & a_{1k}w_1w_k \\ a_{21}w_2w_1 & a_{22}w_2^2 & \ldots & a_{2k}w_2w_k \\ \vdots & \vdots & \ddots & \vdots \\ a_{k1}w_kw_1 & a_{k2}w_kw_2 & \ldots & a_{kk}w_k^2 \end{pmatrix} = 2\sum_{i=1}^{k} a_{2i} \cdot w_i$$

Получили квадратичную форму с матрицей A.
С учетом этого получаем

$$L_1 = \begin{pmatrix} 2\sum\limits_{i=1}^{k} a_{i1}w_i \\ 2\sum\limits_{i=1}^{k} a_{i2}w_i \\ \vdots \\ 2\sum\limits_{i=1}^{k} a_{ik}w_i \end{pmatrix} = 2A\omega = 2X^TX\omega$$

$$w^TX^Ty = (w_1, \ldots, w_k)\begin{pmatrix} x_{11}y_1 + \ldots + x_{n1}y_n \\ \vdots \\ x_{1k}y_1 + \ldots + x_{nk}y_n \end{pmatrix} =$$

$$= w_1(x_{11}y_1 + \ldots + x_{n1}y_n) + \ldots + w_k(x_{1k}y_1 + \ldots + x_{nk}y_n)$$

Таким образом

$$L_2 = \begin{pmatrix} x_{11}y_1 + \ldots + x_{n1}y_n \\ \vdots \\ x_{1k}y_1 + \ldots + x_{nk}y_n \end{pmatrix} = X^Ty$$

$$L_3 = 0$$

Тривиально (Очевидно)

В конечном счете имеем условие экстремума

$$\frac{dL}{d\omega} = 2X^TX\omega - 2X^Ty = 0$$

Находим оптимальный вектор весов

$$2X^TX\omega - 2X^Ty = 0$$

$$X^TX\omega = X^Ty$$

$$\omega = (X^TX)^{-1}X^Ty$$

Модели применяемые для решения задачи регрессии

1. Линейная регрессия
2. Полиномиальная регрессия
3. Дерево решений
   - Разбиваем пространство признаков на области
4. Случайный лес
   - Ансамбль из деревьев решений (*бэггинг*)
5. Градиентный бустинг
   - Последовательное построение ансамбля, где каждое новое дерево учится на ошибках предыдущих.
6. Нейронная сеть (без функции активации)
   - Линейные слои без нелинейных активаций эквивалентны линейной регрессии.

Внесение нелинейности в линейные модели. Случаи использования

Feature Engineering

Идея заключается в интерпретации нелинейного слагаемого как самостоятельной фичи, например:

$$f(w, x) = w_0 + w_1x_1 + w_2x_2 + w_3\ln(x_1) + w_4x_2^4$$

Мы по приколу ввели несколько нелинейных зависимостей в виде логарифма и четвертой степени, однако относительно весов модель как была линейной, так и осталась.

Увлекаться этим тоже не стоит, ибо есть ~~и~~риски потерять смысл фичи, а плодить юзлесс хуйню дело не благотворное.

Из полезного сюда же можно отнести преобразование периодических фич на окружность: условно у нас есть время суток, день года и т. д., имеет смысл вытащить значения синуса и косинуса для них. Так модели будет проще воспринимать эту фичу (в частности не будет путаницы между 0 и 24-м часами в сутках).

В целом применять стоит, если зависимость видна явно.

Kernel trick

Идея заключается в переходе к более высокой размерности с помощью некоторого ядра так, чтобы в новом пространстве данные стали линейно разделимыми.

Ядро (Kernel) — это функция, которая для любых двух точек `a` и `b` в исходном пространстве возвращает скаляр, равный их скалярному произведению в некотором пространстве признаков `ϕ`: `K(a, b) = ⟨ϕ(a), ϕ(b)⟩`.

Имеет смысл использовать, если зависимость сложна и на глаз сказать какая она нельзя.

Доп вопросы

Достаточное условие экстремума

Пусть у нас есть функция $f : \mathbb{R}^n \mapsto \mathbb{R}$ дважды непрерывно дифференцируемая, x_0 - стационарная точка, т. е. $\nabla f(x_0) = \left(\frac{\partial f}{\partial x_1}(x_0) \;\; \frac{\partial f}{\partial x_2}(x_0) \dots \frac{\partial f}{\partial x_n}(x_0)\right) = \vec{0}$.

$H(x)$ - матрица Гессе (матрица вторых производных):

$$H(x) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \frac{\partial^2 f}{\partial x_1 x_2}(x) & \frac{\partial^2 f}{\partial x_2^3}(x) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) \\ \dots & \dots & \dots & \dots \\ \frac{\partial^2 f}{\partial x_1 x_n}(x) & \frac{\partial^2 f}{\partial x_2 x_n}(x) & \dots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{pmatrix}$$

Тогда точка x_0 - точка минимума, если матрица Гессе в этой точке положительно определённая, и точка максимума, если отрицательно определённая.

По критерию Сильвестра, матрица квадратичной формы положительно определена, если все угловые миноры положительны:

$$\Delta_1 = \frac{\partial^2 f}{\partial x_1^2}(x_0) > 0, \Delta_2 = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2}(x_0) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x_0) \\ \frac{\partial^2 f}{\partial x_1 x_2}(x_0) & \frac{\partial^2 f}{\partial x_2^3}(x_0) \end{vmatrix} > 0, \dots, \Delta_n = \det H(x_0) > 0$$

и отрицательно определена, если знаки угловых миноров **чередуются, начиная с отрицательного**:

$$\Delta_1 = \frac{\partial^2 f}{\partial x_1^2}(x_0) < 0, \Delta_2 = \begin{vmatrix} \frac{\partial^2 f}{\partial x_1^2}(x_0) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x_0) \\ \frac{\partial^2 f}{\partial x_1 x_2}(x_0) & \frac{\partial^2 f}{\partial x_2^3}(x_0) \end{vmatrix} > 0, \dots, \Delta_n = \det H(x_0) > 0 \text{ (если n - чётное)}$$

Тема 3. Классификация

- **TP (True Positive)**: верно предсказанные положительные классы
- **TN (True Negative)**: верно предсказанные отрицательные классы

- **FP (False Positive)**: ложно-положительные (ошибка I рода)
- **FN (False Negative)**: ложно-отрицательные (ошибка II рода)

Метрики (для бинарной и мультиклассификации)

Бинарная

Матрица ошибок

	Predicted negative	**Predicted positive**
Actual negative	TN	FP
Actial positive	FN	TP

Accuracy

Доля верных предсказаний.

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

Не подходит, когда имеется дисбаланс классов или разные вероятности предсказания для каждого класса.

Precision

Доля верно предсказанных объектов класса 1 от всех объектов, предсказанных как 1-ый класс. (учитывает ошибку 1-ого рода)
Насколько точно мы способны предсказывать 1-й класс.

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

Recall

Доля верно предсказанных объектов класса 1 от всех объектов являющихся 1-м классом (учитывает ошибку второго рода).
Полнота предсказаний 1-го класса.

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

F_1-score

Метрика связывающая точность и полноту, также называемая их средним.

$$\frac{2 \cdot Recall \cdot Precision}{Recall + Precision}$$

В идеальной ситуации стремимся получить $Recall$ = $Precision = 1 \rightarrow F_1 = 1$

Обобщённая формула, если $Precision$ или $Recall$ важнее:

$$F_\beta = (1 + \beta^2)\frac{Recall \cdot Precision}{Recall + \beta^2 Precision}$$

При $\beta > 1$ важнее $Recall$, при $0 < \beta < 1$ важнее $Precision$.

**Почему в формуле используется гармоническое среднее $Precision$ и $Recall$, а не арифметическое?*

Гармоническое среднее гораздо чувствительнее к малым значениям, чем арифметическое.

Пример: имеется сильный дисбаланс классов, и модель ошибается в предсказании редкого положительного класса, потому $Precision$ $1, Recall = 0.01$.
Среднее арифметическое: $\frac{1}{2}(Precision + Recall) = 0.5005$
Среднее гармоническое: $2\frac{Recall \cdot Precision}{Recall + Precision} \approx 0.0198$
Среднее арифметическое подвержено влиянию больших значений, а среднее гармоническое - малых, потому такая оценка более справедливая.

Мультиклассовая

- **Macro-average** - равный вес каждому классу
  - Все классы равнозначны
  - Маленькие классы имеют такой же вес, как и большие, следовательно метрика меньше подвержена дисбалансу классов
- **Micro-average** - вес пропорционально размеру класса

Accuracy

Доля верных предсказаний модели (micro-average)
$\frac{\sum(\hat{y}_i = y_i)}{|S|}$

Precision

$$Precision_{macro} = \frac{Precision_A + Precision_B + \dots Precision_N}{N}$$

$$Precision_{micro} = \frac{TP_A + TP_B + \dots + TP_N}{TP_A + FP_A + TP_B + FP_B + \dots + TP_N + FP_N}$$

Recall

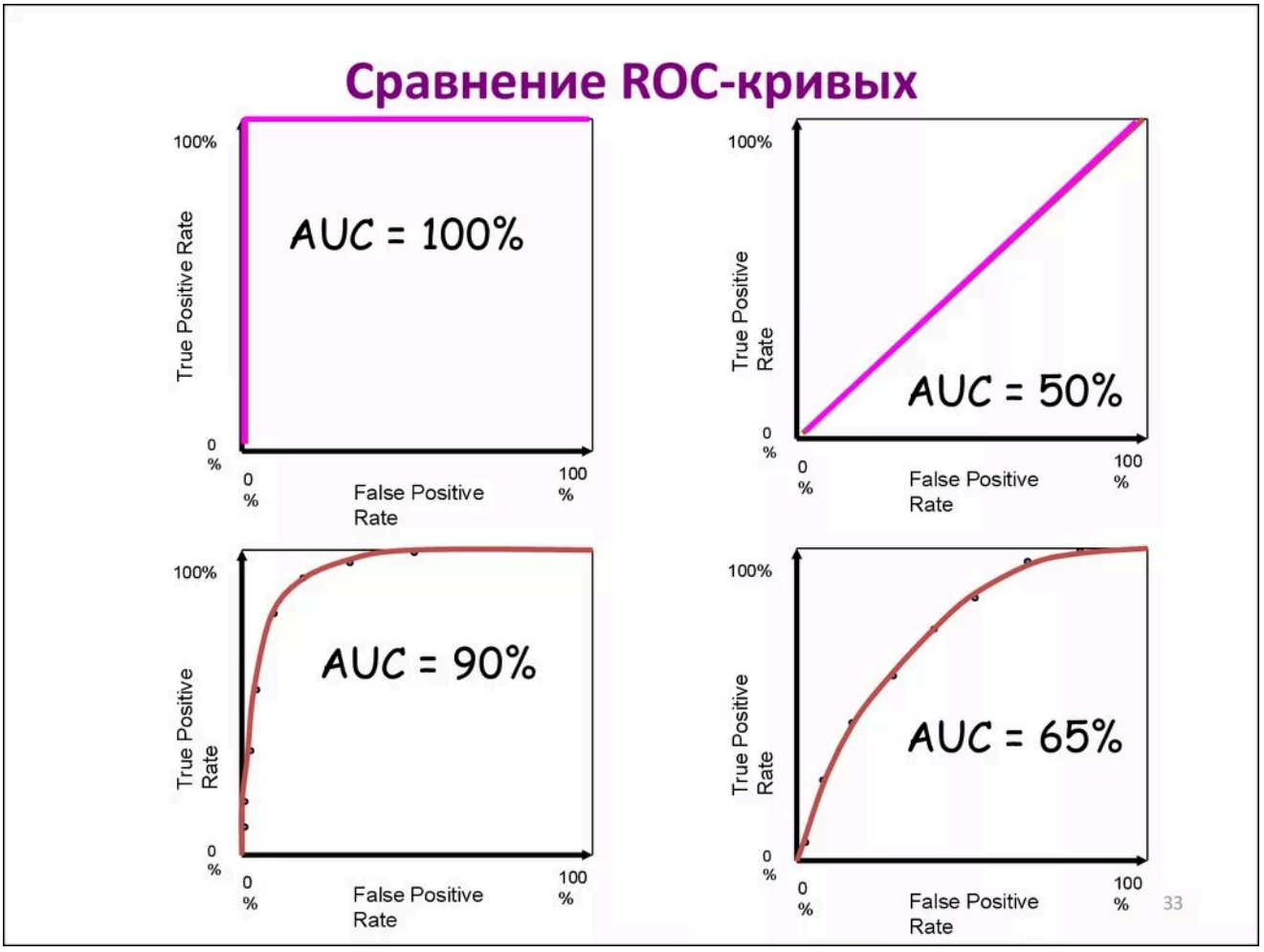
$$Recall_{macro} = \frac{Recall_A + Recall_B + \dots Recall_N}{N}$$

F1 score

$$F_1(class = a) = 2\frac{Precision(class = a) \times Recall(class = a)}{Precision(class = a) + Recall(class = a)}$$

ROC AUC

Идея



Reciever Operating Characteristic - кривая ошибок. Показывает компромисс между TP rate и FP rate

Area Under the Curve - площадь под кривой. Оценка качества бинарного классификатора.

Построение

1. **Получаем вероятности** положительного класса от модели
2. **Сортируем объекты** по убыванию вероятности
3. **Последовательно меняем порог** от 1.0 до 0.0
    1. $p(class = positive) \overset{?}{>} \text{border}$
4. **Для каждого порога** вычисляем:
    - **TPR** = Recall = $\frac{TP}{TP + FN}$

- **FPR** = $\frac{FP}{FP + TN}$

5. **Строим кривую**: FPR (X-axis) vs TPR (Y-axis)

ROC-AUC равен вероятности того, что случайно выбранный положительный объект (из класса "1") будет ранжирован моделью выше (ему будет присвоена более высокая вероятность принадлежности к положительному классу), чем случайно выбранный отрицательный объект (из класса "0").

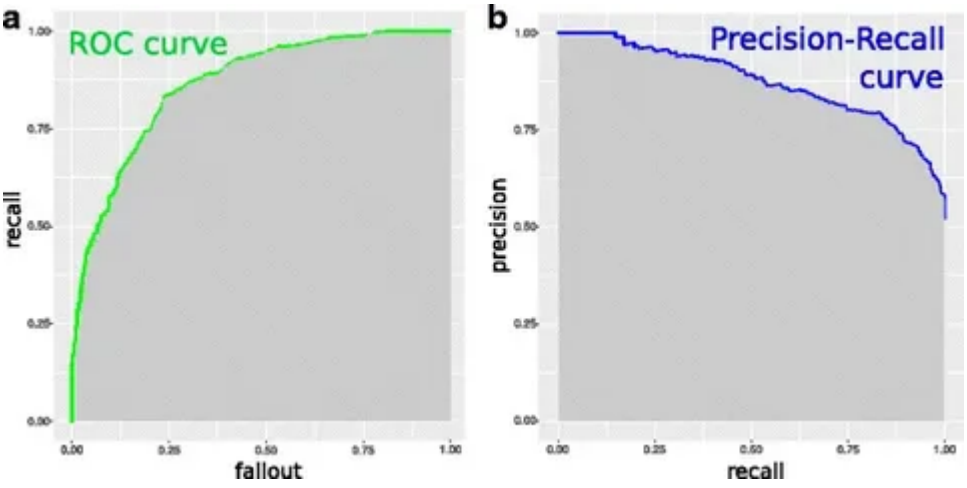
Сравнение

$AUC = 1.0$ - идеальная модель
$AUC = 0.5$ - случайное угадывание
$AUC < 0.5$ - хуже случайного угадывания

Ограничения

- **Сильный дисбаланс классов** → лучше Precision-Recall AUC
- **Разная стоимость ошибок** FP и FN → нужно выбирать конкретный порог
- **Нужен конкретный порог** для production системы

Гиперпараметр классификации (подозреваю, что речь про PR AUC)



Идея

PR AUC — это площадь под кривой Precision-Recall, которая показывает компромисс между точностью и полнотой модели.

Ключевое отличие от ROC AUC:

- **ROC AUC** фокусируется на обоих классах одинаково
- **PR AUC** фокусируется в основном на **положительном классе**

Построение

Оси кривой:

- **X-axis**: **Recall** (полнота) = $\frac{TP}{TP + FN}$
- **Y-axis**: **Precision** (точность) = $\frac{TP}{TP + FP}$

Процесс построения:

1. Сортируем объекты по убыванию вероятности положительного класса
2. Последовательно меняем порог классификации
3. Для каждого порога вычисляем Precision и Recall
4. Строим кривую по полученным точкам

Примеры выбора метрик для бинарной классификации

1. Медицинская диагностика (обнаружение болезни)

Контекст: Редкое заболевание (1% prevalence)

- **Цель:** Не пропустить ни одного больного
- **Стоимость ошибок:** FN >> FP (лучше ложная тревога, чем пропущенный больной)

Рекомендуемые метрики:

- **Recall** - главный приоритет
- **Specificity** - контроль ложных тревог
- **PR AUC** - оценка качества на положительном классе
- **F2-score** - с весом в пользу Recall

2. Спам-фильтрация email

Контекст: Баланс классов примерно 50/50

- **Цель:** Не пропустить спам, но не потерять важные письма
- **Стоимость ошибок:** FP ≈ FN (потеря важного письма ≈ получение спама)

Рекомендуемые метрики:

- **Precision** - минимизация ложных срабатываний
- **F1-score** - баланс между Precision и Recall
- **ROC AUC** - общее качество классификации
- **Accuracy** - так как классы сбалансированы

Модернизация метрик для задачи мультиклассификации.

Для каждого класса вычисляем метрики, рассматривая его как положительный, а все остальные - как отрицательный.

Примеры выбора метрик для задачи мультиклассификации.

1. Классификация изображений (CIFAR-10, ImageNet)

Контекст: 10-1000 сбалансированных классов
Цель: Высокая общая точность распознавания
Рекомендуемые метрики:

- **Top-1 Accuracy** - основная метрика
- **Top-5 Accuracy** - учитывает близкие классы
- **Macro F1-score** - сбалансированная оценка по всем классам
- **Confusion Matrix** - анализ типичных ошибок (например, "кошка vs собака")

2. Классификация текстов (новостей, отзывов)

Контекст: 5-20 классов, часто несбалансированных
Цель: Точно определять категории контента
Рекомендуемые метрики:

- **Weighted F1-score** - учет дисбаланса классов
- **Macro Precision** - важно для редких категорий
- **Per-class Recall** - убедиться, что все темы охвачены
- **Hamming Loss** - если возможна multi-label классификация

Тема 4. Деревья

https://education.yandex.ru/handbook/ml/article/reshayushchiye-derevya

Построение дерева для классификации (регрессии)

Пусть X — исходное множество объектов обучающей выборки, а X_m — множество объектов, попавших в текущий лист (в самом начале $X_m = X$). Тогда жадный алгоритм можно описать следующим образом:

1. Создаём вершину v.
2. Если выполнен критерий остановки $Stop(X_m)$, то останавливаемся, объявляем эту вершину листом и ставим ей в соответствие ответ $Ans(X_m)$, после чего возвращаем её.
3. Иначе: находим предикат (иногда ещё говорят сплит) $\beta_{j,t}$, который определит наилучшее разбиение текущего множества объектов X_m на две подвыборки X_ℓ и X_r, максимизируя критерий ветвления $Branch(X_m, j, t)$.

4. Для X_ℓ и X_r рекурсивно повторим процедуру.

Критерии останова

В качестве критерия может выступать простое правило:

- Достигнута нужная глубина
- Количество данных, попавших в лист < заданного
- Достигнут желаемый показатель однородности данных в листе
- Улучшение критерия ветвления меньше порога

Можно построить дерево жадно без ограничений, а затем провести *стрижку* (*pruning*), то есть удалить некоторые вершины из дерева так, чтобы итоговое качество упало не сильно, но дерево начало подходить под условия регуляризации.
Это помогает бороться с переобучением.

Критерии разбиения (регрессия / классификация)

Наиболее часто критерий ветвления - это функция, которая оценивает, насколько улучшится некоторая финальная метрика качества дерева в случае, если получившиеся два листа будут терминальными, по сравнению с ситуацией, когда сама исходная вершина — это лист.

Impurity (Хаотичность)
Показатель того, насколько хорошо объекты в листе можно приблизить константным значением

$$H(X_m) = \min_c \frac{1}{|X_m|} \sum_{(x_i,y_i)\in X_m} L(y_i, c)$$

Критерий ветвления примет вид:

$$Branch(X_m) = H(X_m) - \frac{|X_l|}{|X_m|} \cdot H(X_l) - \frac{|X_r|}{|X_m|} \cdot H(X_r)$$

Перед $H(x_l)$ и $H(x_r)$ именно такие множители, потому что в противном случае модель выбирала бы неинформативные разделения на малое подмножество с, например, элементами одного класса, и большое подмножество без особого изменения пропорций классов объектов.

Почему именно такие веса? Контрпример

Допустим, есть два возможных разбиения одного родительского узла (100 объектов, 50/50 классов, H=1, IG = H(до разбиения) - E[H(после разбиения)], E - матожидание):

Сценарий А (хорошее разбиение):

- Левый узел: 49 объектов (49 "Да", 0 "Нет"), H=0

- Правый узел: 51 объект (1 "Да", 50 "Нет"), H≈0.14
- Взвешенная энтропия: `(49/100)*0 + (51/100)*0.14 ≈ 0.07`
- IG = 1 - 0.07 = **0.93** (отлично!)

Сценарий B (плохое разбиение, но создающее один очень чистый маленький узел):

- Левый узел: 5 объектов (5 "Да", 0 "Нет"), H=0
- Правый узел: 95 объектов (45 "Да", 50 "Нет"), H≈1.0
- **Без весов (ошибка!):** Среднее было бы `(0 + 1)/2 = 0.5`, IG=0.5
- **С весами (правильно):** `(5/100)*0 + (95/100)*1 = 0.95`, IG=0.05
- Реальный IG = **0.05** (мизерный, и это правильно!)

Без коэффициентов дерево выбрало бы сценарий B, что абсурдно: мы создали крошечный чистый узел (5 объектов) ценой почти нулевого улучшения для 95% данных!

MSE

$$H(X_m) = \sum_{(x_i,y_i) \in X_m} \frac{(y_i - \overline{y})^2}{|X_m|}$$

MAE

$$H(X_m) = \sum_{(x_i,y_i) \in X_m} \frac{|y_i - \text{median}(Y)|}{|X_m|}$$

Misclassification error

$$H(X_m) = \min_{c \in Y} \frac{1}{|X_m|} \sum_{(x_i,y_i) \in X_m} \mathbb{I}[y_i \neq c]$$

Энтропия

$$H(X_m) = -\sum_{k=1}^{K} \frac{N_k}{|X_m|} \log c_k = -\sum_{k=1}^{K} p_k \log p_k$$

Измеряет непредсказуемость реализации случайной величины.
Если случайная величина принимает только одно значение, то она абсолютно предсказуема и её энтропия равна 0.
Наибольшего значения энтропия достигает для равномерно распределённой случайной величины — и это отражает тот факт, что среди всех величин с данной областью значений она наиболее «непредсказуема».

Критерий Джини

$$H(X_m) = \sum_{k=1}^{K} p_k(1 - p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

Обработка категориальных признаков

Пусть признак x^i принимает значения из множества $C = \{c_1, \ldots, c_M\}$.
Тогда при очередном разбиении естественно рассматривать по этому признаку произвольные сплиты вида $C = C_l \sqcup C_r$.

- *Число возможных разделений равно* $2^{M-1} - 1$.

Для снижения вычислительной сложности ищут способ упорядочить элементы множества C и работать с ними как с числами.
Для некоторых задач такое упорядочение можно построить вполне естественным образом.

Бинарная классификация

Для задачи бинарной классификации значения c_m можно упорядочить по неубыванию доли объектов класса 1.

Регрессия

Для задачи регрессии с функцией потерь MSE значения c_m можно упорядочить по среднему значению таргета в множестве.

Тема 5. Кластеризация

https://education.yandex.ru/handbook/ml/article/klasterizaciya

Постановка задачи

Задача **обучения без учителя**, целью которой является разбиение множества объектов на группы (кластеры) таким образом, чтобы:

- Объекты внутри одного кластера были **максимально похожи** друг на друга
- Объекты из разных кластеров были **максимально различны**
  C_i -кластер. Кластеры не пересекаются
  μ_i — центр кластера C_i
  $d\left(x_i, x_j\right)$ - мера близости между объектами.

Критерий качества:

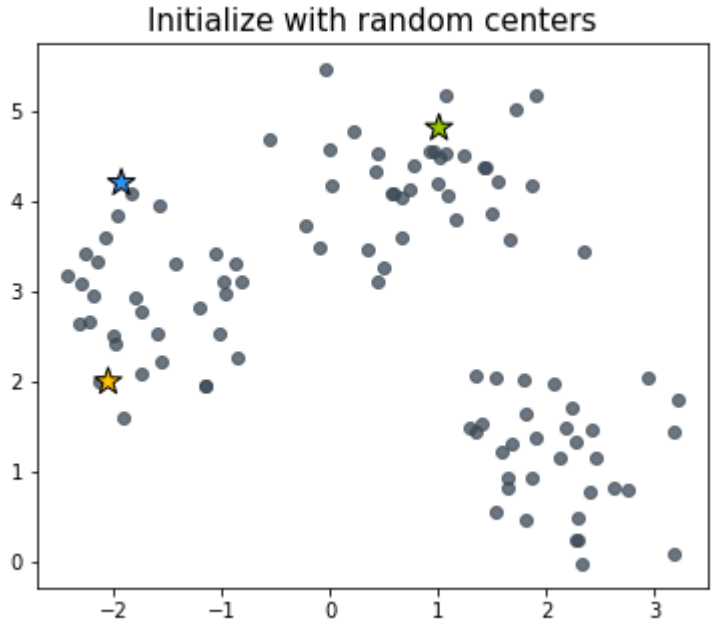
- Минимизация внутрикластерного расстояния:
  $\min \sum_{i=1}^{k} \sum_{x \in C_i} d(x, \mu_i)^2$

- Максимизация межкластерного расстояния:
  $\max \sum_{i \neq j} d(\mu_i, \mu_j)^2$

Число кластеров:

- Может быть задано априори (k-means)
- Может определяться автоматически (DBSCAN, иерархическая кластеризация)

K-means



Идея

Разбить данные на **k кластеров** так, чтобы минимизировать внутрикластерную вариацию (сумму квадратов расстояний от точек до центроиды их кластера).

$V = \sum_{i=1}^{N} \min_j ||x_i - \mu_j||^2 \to \min$, где x_i - точки, μ_j - центроиды.

Алгоритм

Инициализация:

- Выбираем число кластеров **k**
- Случайно инициализируем **k центроид** (центров кластеров)
  Основной цикл:

1. **Назначение кластеров (E-step):**
   - Для каждой точки находим ближайшую центроиду
   - Назначаем точку соответствующему кластеру

- $C_i = x_j : ||x_j - \mu_i||^2 \leq ||x_j - \mu_l||^2 \; \forall l$

2. **Пересчет центроид (M-step):**
   - Для каждого кластера вычисляем новую центроиду как среднее всех точек кластера
   - $\mu_i = \frac{1}{|C_i|} \sum_{x_j \in C_i} x_j$
3. **Проверка на критерий остановки**

Критерий остановки:

- Максимальное число итераций
- Порог изменения центроид (может требовать много итераций)
- Порог изменения целевой функции (внутрикластерной суммы квадратов)

Потенциальные проблемы

Кучное размещение центров. В этом случае их начальное положение с большой вероятностью окажется далёким от итогового положения центров кластеров.

Улучшения алгоритма

K-means++ (умная инициализация):

1. Первая центроида выбирается случайно
2. Каждая следующая выбирается с вероятностью, пропорциональной квадрату расстояния до ближайшей существующей центроиды
   - $p(s) \cdot \rho(x_{next}, x_1)^2$
     - x_{next} - последующий центр кластера
     - x_1 - центр выбранный на шаге 1
     - $p\,(s)$ - вероятность выбора точки

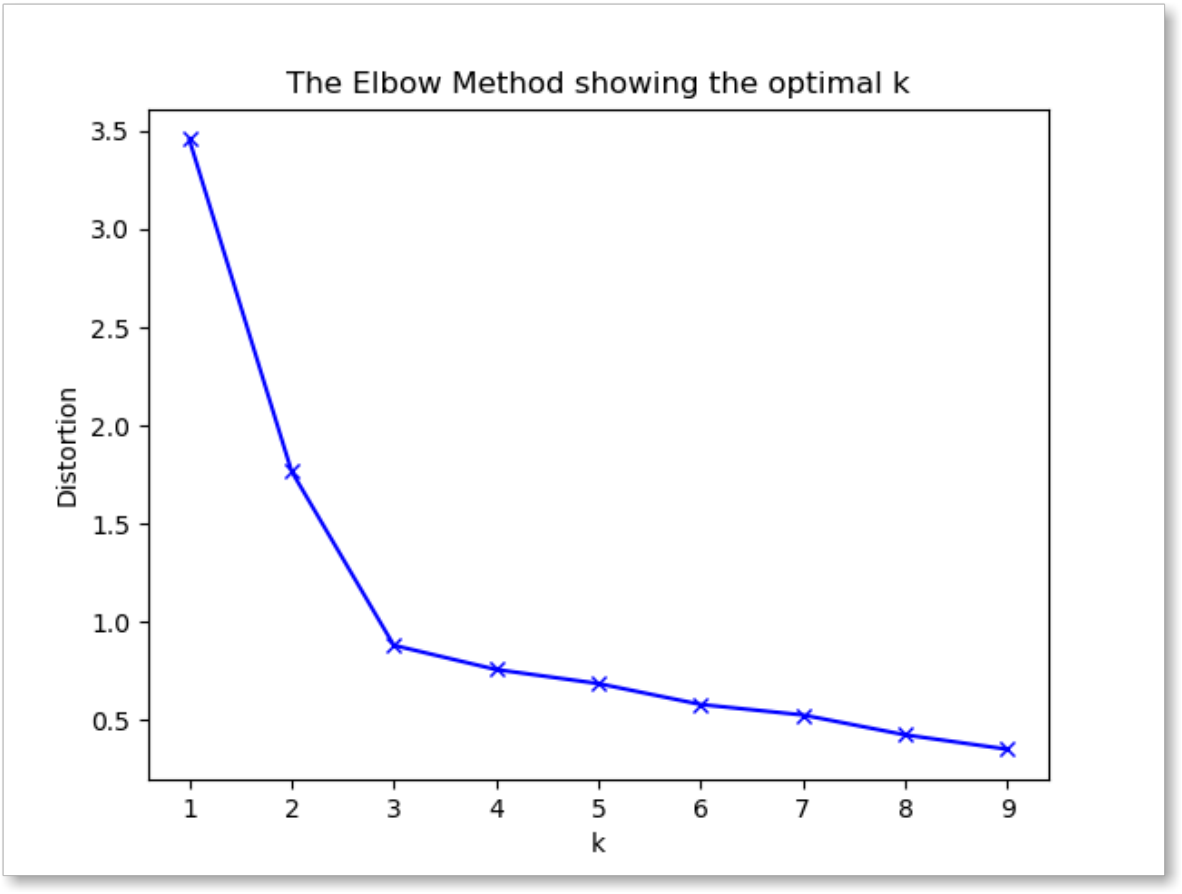
- Чем дальше точка тем выше шанс её взять.
- Значительно улучшает качество и скорость сходимости

mini-batch K-means

- На каждой итерации выбираем случайную подвыборку (мини-батч) и работаем на ней.
- В случае когда исходная выборка очень велика, переход к пакетной обработке не приводит к большой потере качества, зато значительно ускоряет работу алгоритма.

Определение оптимального k:

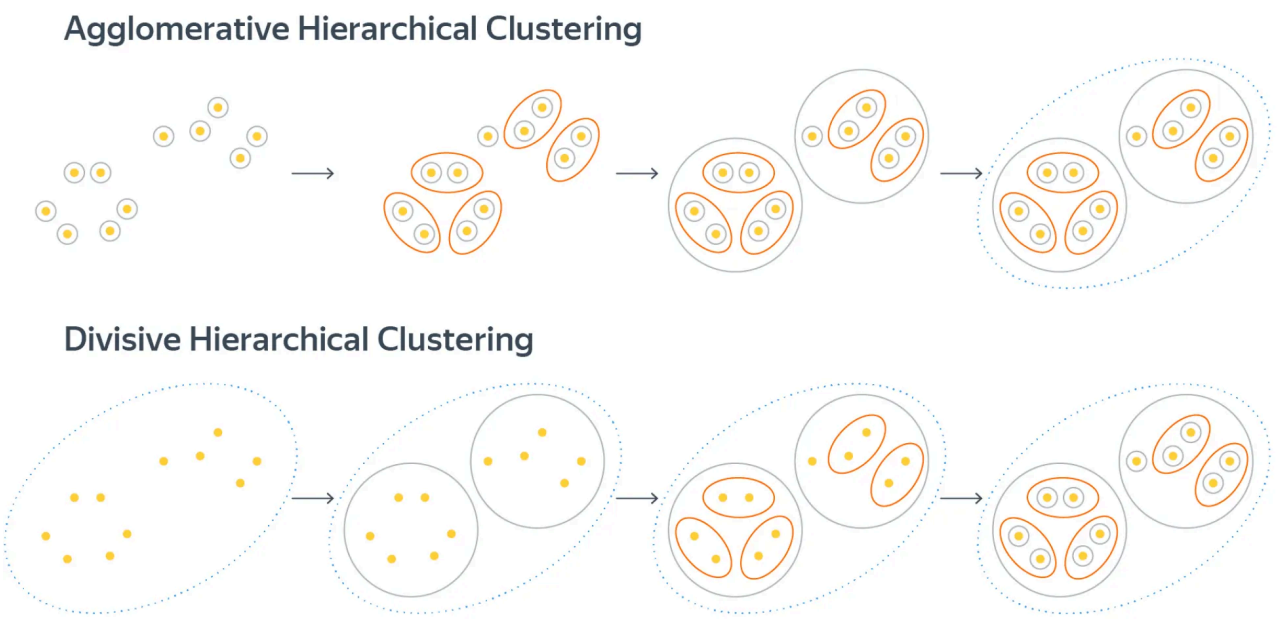
- **Elbow method** - ищем "локоть" на графике

- Момент, когда внутрикластерная (сумма квадратов расстояний между объектами и их центроидом) перестаёт резко снижаться
- 

Использование Манхеттенского расстояния вместо Евклидова

- $d_1(x, y) = \sum_{i=1}^{n} |x_i - y_i|$
- Менее чувствительно к выбросам, следовательно лучше справляется с проклятием размерности;
- Кластеры имеют ромбовидную форму, а не сферическую;
- Если разреженные данные (вычисления быстрее, т. к. не нужно возводить в квадрат и считать корень).

Иерархические агломерационные и дивизионные методы кластеризации



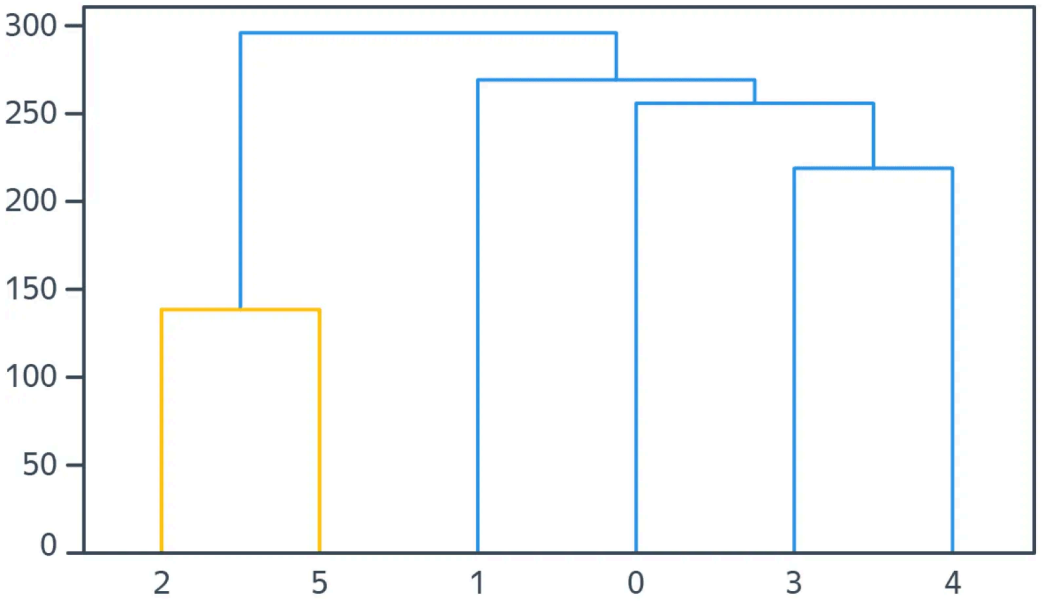
Идея

Агломеративные алгоритмы начинают с небольших кластеров (обычно с кластеров, состоящих из одного объекта) и постепенно объединяют их в кластеры побольше. **Дивизионные** начинают с больших кластеров (обычно – с одного единственного кластера) и постепенно делят на кластеры поменьше.

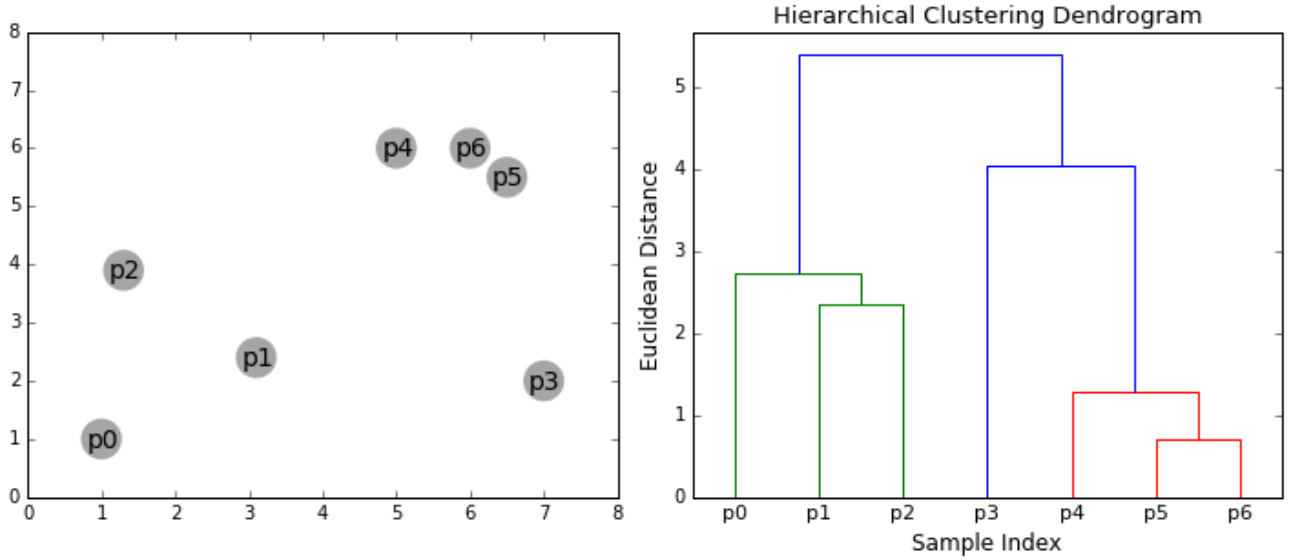
Кластеры могут быть как на одном уровне (плоская структура) так и в иерархии (образуя древовидную структуру)

По мере объединения кластеров, каждой итерации алгоритма соответствует пара объединяемых на этой итерации кластеров, а также расстояние между кластерами в момент слияния. Расстояния с ростом итерации будут только увеличиваться, поэтому

возникает возможность построить следующую схему, называемую **дендрограммой**:



Пример работы агломеративного алгоритма:



Здесь по горизонтали внизу отмечены объекты кластеризуемой выборки, под горизонтальной осью подписаны номера объектов, а их расположение вдоль оси продиктовано только эстетическими соображениями: нам удобно строить дендрограмму так, чтобы никакие дуги в ней не пересекались. По вертикали отложены расстояния между кластерами в момент слияния. Когда происходит объединение кластеров, состоящих из нескольких объектов, соответствующая этой итерации алгоритма дуга идёт не до конкретных объектов выборки, а до дуги другого кластера.

Метод "локтя" по дендрограмме:

1. Ищем самый длинный вертикальный отрезок, который не пересекается горизонтальными линиями
2. Проводим горизонтальную линию через этот отрезок
3. Число пересекаемых вертикальных линий = число кластеров

Алгоритм

1. Создаём столько кластеров, сколько у нас объектов в выборке, каждый объект — в своём отдельном кластере.
2. Повторяем итеративно слияние двух ближайших кластеров, пока не выполнится критерий останова.

Критерий останова

1. Нужное количество кластеров
2. Сильное изменение расстояния после шага итерации

Подсчёт расстояния

Если обозначить кластеры как U и V, расстояние между ними в этом случае можем вычислять по одной из формул:

$$d_{avg}(U,V) = \frac{1}{|U| \cdot |V|} \sum_{u \in U} \sum_{v \in V} \rho(u,v)$$

$$d_{min}(U,V) = \min_{(u,v) \in U \times V} \rho(u,v)$$

$$d_{max}(U,V) = \max_{(u,v) \in U \times V} \rho(u,v)$$

Метод Уорда (Ward), вычисление улучшения метрики внутрикластерной суммы квадратов после объединения кластеров:

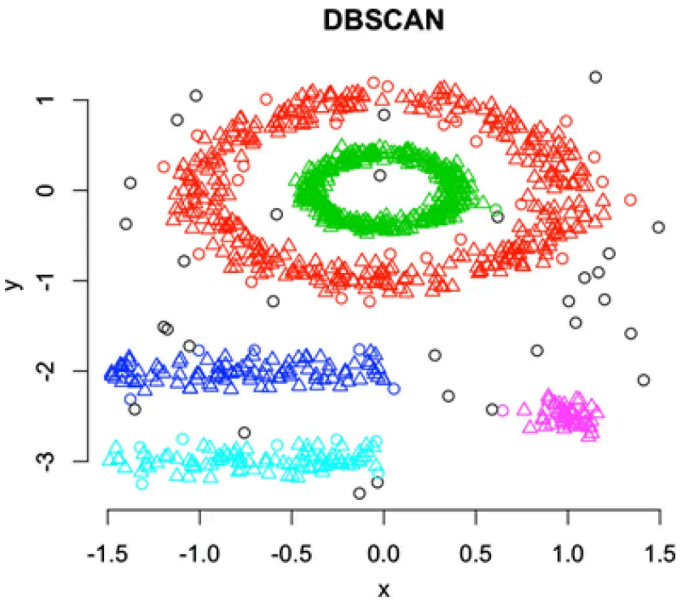
$$d_{ward}(U,V) = \sum_{x \in U \cup V} ||x - \overline{x}||^2 - \sum_{u \in U} \left|\left|u - \overline{U}\right|\right|^2 - \sum_{v \in V} \left|\left|v - \overline{V}\right|\right|^2,$$

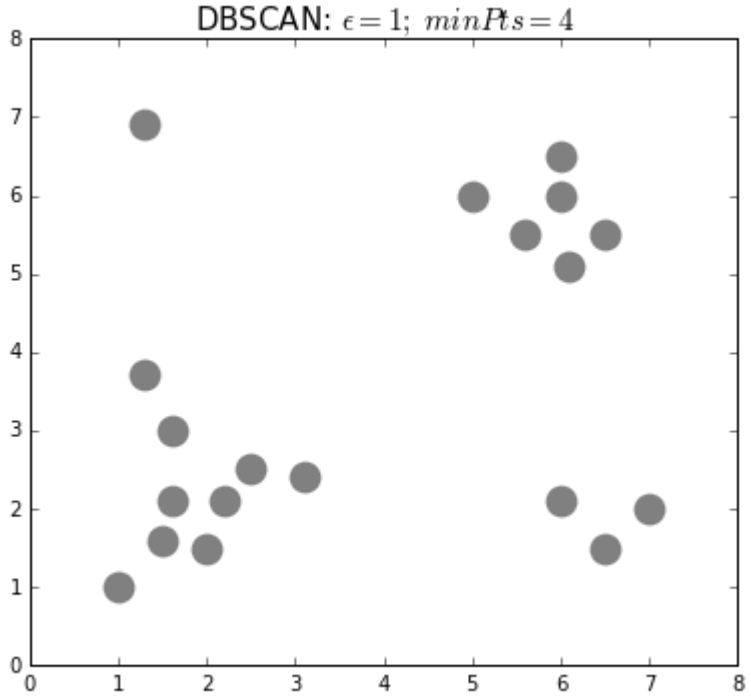
где $\overline{x} = \frac{1}{|U \cup V|} \sum_{x \in U \cup V} x, \quad \overline{U} = \frac{1}{|U|} \sum_{u \in U} u, \quad \overline{V} = \frac{1}{|V|} \sum_{v \in V} v$

Недостатки:

- **Высокая вычислительная сложность**: O(n³) для наивной реализации, O(n²) с оптимизацией
- **Чувствительность к шуму и выбросам**
- **Не отменяет объединения** (жадный алгоритм)
- **Трудно масштабировать на большие datasets**

DBSCAN





Идея

Выделение связных компонент графа

Параметры:

- ϵ - радиус окрестности
- **min_samples** - минимальное количество точек в ϵ-окрестности

Типы точек:

- **Core point** (ядровая точка):
  - В её ϵ-окрестности находится ≥ min_samples точек (включая саму точку)
- **Border point** (пограничная точка):

    - Не является core point, но попадает в ϵ-окрестность какой-либо core point
- **Noise point** (шумовая точка):
    - Не является ни core, ни border point

Алгоритм

1. **Инициализация:**
    - Выбираем параметры ϵ и $\min\text{samples}$
    - Все точки помечаем как не посещённые
2. **Основной цикл:**
    - Для каждой не посещённой точки p:
        - Помечаем p как посещенную
        - Находим всех соседей в ϵ-окрестности: $N = \{q \in X \mid \text{distance}(p, q) \leq \varepsilon\}$
        - Если $|N| < \min\text{samples} \rightarrow$ помечаем p как **NOISE**
        - Иначе:
            - Создаем новый кластер
            - Добавляем p в кластер
            - Расширяем кластер: добавляем все точки, достижимые из p

Недостатки:

- **Чувствителен к параметрам** ϵ и min_samples
- **Плохо работает с данными разной плотности** в одном наборе
- **Затруднительно выбирать параметры** в высокоразмерных пространствах
- **Не детерминирован** для border points (могут попасть в разные кластеры)

Выбор параметров

Эвристики для ϵ:

- **K-distance graph**: строим график расстояний до k-го ближайшего соседа (k = min_samples)
- Ищем "локоть" на графике - точка резкого излома

Эвристики для min_samples:

- Обычно выбирают min_samples ≥ размерность данных + 1
- Часто используют min_samples = 2 × размерность данных

Алгоритм нахождения кластеров

по ближайшей центроиде

Метрики кластеризации

- **Среднее внутрикластерное расстояние.**
  - Мы хотим его **минимизировать**
  - $F_0 = \dfrac{\sum\limits_{i=1}^{n}\sum\limits_{j=i}^{n}\rho(x_i,x_j)\mathbb{I}[a(x_i)=a(x_j)]}{\sum\limits_{i=1}^{n}\sum\limits_{j=i}^{n}\mathbb{I}[a(x_i)=a(x_j)]}$
  - В случае если у кластеров есть центры μ_k, часто рассматривается аналогичная метрика — средний квадрат внутрикластерного расстояния:
    - $\Phi_0 = \dfrac{1}{nK}\sum\limits_{k=1}^{K}\sum\limits_{i=1}^{n}\rho(\mu_k,x_i)^2\mathbb{I}[a(x_i)=k]$
- **Среднее межкластерное расстояние расстояние**
  - Мы хотим его **максимизировать**
  - $F_1 = \dfrac{\sum\limits_{i=1}^{n}\sum\limits_{j=i}^{n}\rho(x_i,x_j)\mathbb{I}[a(x_i)\neq a(x_j)]}{\sum\limits_{i=1}^{n}\sum\limits_{j=i}^{n}\mathbb{I}[a(x_i)\neq a(x_j)]}$
- **Silhouette Score**
  - Показывает насколько в среднем объекты схожи внутри одного кластера и различны с объектами других кластеров.
  - $s(i) = \dfrac{b(i)-a(i)}{\max(a(i),b(i))}$
  - $a(i)$ - среднее расстояние от точки i до точек в её кластере
  - $b(i)$ - среднее расстояние от точки i до точек другого ближайшего кластера
- **Индекс Дэвиса-Болдина**
  - Показывает, насколько центры кластеров удалены друг от друга и насколько объекты в кластерах близки к своим центроидам
  - $DB = \dfrac{1}{n}\sum\limits_{i=1}^{n}\max\limits_{i\neq j}\dfrac{\sigma_i+\sigma_j}{d(c_i,c_j)}$
  - n - число кластеров
  - c_i - центроид кластера i
  - σ_i - среднее расстояние точек кластера i до центроида c_i
  - Чем меньше метрика, тем лучше качество кластеризации
- **Индекс Калински-Харабаша**
  - Отношение межкластерой и внутрикластерной дисперсии
  - $CB = \dfrac{BCSS}{WCSS}\dfrac{n-k}{k-1} = \dfrac{\sum\limits_{i=1}^{k}n_i||c_i-c||^2}{\sum\limits_{i=1}^{k}\sum\limits_{x\in C_i}||x-c_i||^2}\dfrac{n-k}{k-1}$
  - n - всего объектов
  - k - число кластеров

- C_i - множество объектов кластера i
- n_i - число объектов в кластере C_i
- c - центроид для всего датасета (среднее по всем объектам)
- c_i - центроид кластера i
- $BCSS = \frac{1}{k-1}\sum_{i=1}^{k} n_i||c_i - c||^2$ - межкластерная сумма квадратов
- $WCSS = \frac{1}{n-k}\sum_{i=1}^{k}\sum_{x \in C_i} ||x - c_i||^2$ - внутрикластерная сумма квадратов
- Чем больше метрика, тем лучше кластеризация

Метрики кластеризации требующие разметку

Для измерения следующих метрик потребуется разметка выборки. Будем обозначать кластеры, к которым наш алгоритм кластеризации относит каждый объект, буквами $k \in \{1, \dots, K\}$, а классы, к которым объекты отнесены разметкой - буквами $c \in \{1, \dots, C\}$.
Пусть n — общее количество объектов в выборке, n_k — количество объектов в кластере номер k, m_c — количество объектов в классе номер c, а n_{ck} — количество объектов из класса c в кластере k. Определим следующие величины:

$$H_{class} = -\sum_{c=1}^{C} \frac{m_c}{n} \log \frac{m_c}{n}$$

$$H_{clust} = -\sum_{k=1}^{K} \frac{n_k}{n} \log \frac{n_k}{n}$$

$$H_{class|clust} = -\sum_{c=1}^{C}\sum_{k=1}^{K} \frac{n_{ck}}{n} \log \frac{n_{ck}}{n_k}$$

Гомогенность

Показывает "чистоту" кластеров (как много объектов кластера принадлежать истинному классу).

$$\boxed{Homogeneity = 1 - \frac{H_{class|clust}}{H_{class}}}$$

Отношение $\frac{H_{class|clust}}{H_{class}}$ показывает, во сколько раз энтропия распределения классов изменяется при получении информации о принадлежности объекта к кластеру

- *Тривиальный способ максимизировать гомогенность кластеризации - выделить каждый объект выборки в отдельный кластер*

Полнота

Показывает "полноту" кластеров (как много объектов истинного класса попали в один кластер).

Полнота задаётся аналогично гомогенности, с той лишь разницей, что вводится величина $H_{clust|class}$, симметричная $H_{class|clust}$:

$$\boxed{Completeness = 1 - \frac{H_{clust|class}}{H_{clust}}}$$

Отношение $\frac{H_{clust|class}}{H_{clust}}$ показывает, во сколько раз энтропия распределения меток принадлежности к кластеру изменяется при получении информации о истинных метках объектов

- *Тривиальный способ максимизировать полноту кластеризации - объединить всю выборку в один кластер*

V-мера

Является аналогом F-меры в задаче классификации

$$\boxed{V_\beta = \frac{(1+\beta) \cdot Homogeneity \cdot Completeness}{\beta \cdot Homogeneity + Completeness}}$$

- *V-мера комбинирует в себе гомогенность и полноту таким образом, чтобы максимизация итоговой метрики не приводила к тривиальным решениям*

Доп вопросы

OPTICS

Идея

Создание графа достижимости при фиксированном значении **min_samples** и для любого возможного радиуса ϵ.

Параметры:

- ϵ_{max} - максимальный рассматриваемый радиус окрестности
- **min_samples** - минимальное количество точек в ϵ-окрестности
- ξ - параметр автоматического определения кластеров

Определения:

core-dist - минимальный радиус ϵ, при котором данная точка считается ядровой.

$$\text{core-dist}_\epsilon(p) = \begin{cases} \text{неопределено} & , |N_\epsilon(p)| < \text{min-samples} \\ \epsilon : |N_\epsilon(p)| = \text{min-samples} & , |N_\epsilon(p)| \geq \text{min-samples} \end{cases}$$

, где $|N_\epsilon(p)|$ - число точек в ϵ-окрестности точки p.

reachability-dist - достижимость точки p относительно ядровой точки c - максимум между $\text{core-dist}(c)$ и расстоянием между этими точками.

$$\text{reachability-dist}_\epsilon(c,p) = \begin{cases} \text{неопределено} & ,|N_\epsilon(p)| < \text{min-samples} \\ \max\left(\text{core-dist}_\epsilon(c), \text{dist}(c,p)\right) & ,|N_\epsilon(p)| \geq \text{min-samples} \end{cases}$$

Алгоритм

1. **Инициализация:** Для всех точек вычисляется **core-distance** (используя **min_samples**). Все помечаются как необработанные.
2. **Обработка:**
   - Берется любая необработанная точка. Если она **ядро** (**core-distance** определено), ее соседи помещаются в **приоритетную очередь**, где приоритет — их **достижимое расстояние** от текущей точки.
   - Из очереди извлекается точка с наименьшим достижимым расстоянием, обрабатывается, и ее соседи добавляются/обновляются в очереди.
3. **Построение порядка:** Точки извлекаются из очереди и записываются в список **в порядке обработки**. Для каждой точки запоминается ее **достижимое расстояние**.
4. **Результат:** Алгоритм выдает не разметку кластеров, а **два упорядоченных массива**:
   - **Порядок обработки точек;**
   - **Достижимые расстояния для каждой точки в этом порядке.**

Далее можно кластеризовать ξ-методом, либо через DBSCAN, задав фиксированное ϵ.

ξ-метод

1. Проходимся по порядку обработки точек, начиная с первой.
2. Для каждой точки i проверяем изменение достижимого расстояния r_i, r_j:
   - если $r_i \leq (1-\xi)r_j$, то точке j присваивается та же метка кластера, что и у точки i,
   - иначе текущий кластер заканчивается, и точке j присваивается новая метка кластера.
3. **Проверка на пересечение**
   - Для последовательных пар кластеров (c_i, c_j) ищем точку минимума по достижимым расстояниям на границах $p = \arg\min\limits_{k \in c_i \cup c_j} r_k$
   - Объединяем кластеры, если $r_p \leq (1-\xi)r_{c_i^{left}}$ и $r_p \leq (1-\xi)r_{c_j^{right}}$,
     где c_i^{left} - точка левой границы кластера c_i,
     c_j^{right} - точка правой границы кластера c_j.

Отличия OPTICS от DBSCAN

1. DBSCAN - алгоритм для получения фиксированного разбиения на кластеры, OPTICS - алгоритм построения графа достижимости для дальнейшего анализа

структуры кластеров.

2. DBSCAN работает для одного фиксированного ϵ и предполагает единую плотность точек, OPTICS позволяет определять кластеры разной плотности.
3. Алгоритмическая сложность DBSCAN и OPTICS с оптимизацией поиска ближайших точек - $O(n \log n)$, но на практике OPTICS медленнее (*на константу, из-за использования очереди с приоритетом*).

HDBSCAN

Идея

Построение иерархии кластеров в зависимости от разных значений ϵ.

При увеличении значения ϵ от 0 до максимального расстояния между точками в датасете, выстраивается такая иерархия:

- при $\epsilon = 0$ каждая точка - отдельный кластер
- с увеличением ϵ начинают появляться пары точек, расстояние между которыми $\leq \epsilon$, и когда таких точек в увеличивающейся ϵ-окрестности становится $\geq$ **min_samples**, они объединяются в кластеры, как в обычном DBSCAN
- При максимальном имеющем смысл значении ϵ все точки попадают в один кластер

В зависимости от ϵ разное число кластеров, следовательно из такого разбиения можно построить дендограмму.