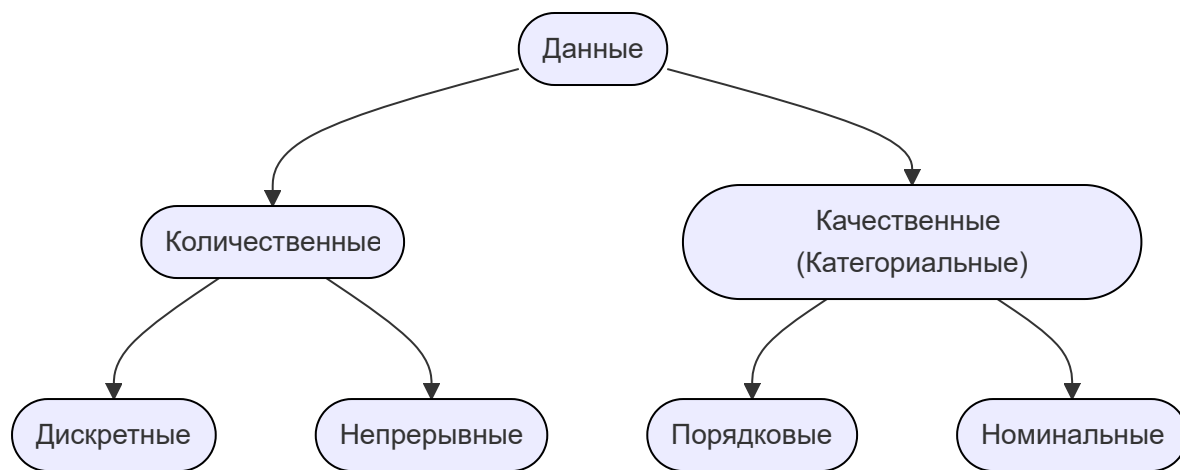


Тема 1. База

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 ✂ 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 Encoding**
 - М, Ж = 0, 1
 - Кодировем 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 - *isM*, второй - *isЖ*
 - Проклятие размерности
 - При большом количестве категорий матрица становится разреженной и большой

Масштабирование и нормализация

Уменьшение масштаба данных.

- **Стандартизирование**
 - —
 - Сведение к среднему μ и стандартному отклонению $\sigma = 1$
- **Min-max**
 - —
 - 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
 - $x_1^2 \cdot x_2$
- **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

Работа с выбросами

Убирание значений, сильно отклоняющихся от остальных наблюдений.

- Ящик с усами
 - Берём 25 и 75 перцентиль; отнимаем / прибавляем $1.5 \cdot$ (интерквартильный размах) (а. к. а. делаем усы); Смотрим, что будет вне границ
 - $< 1.5 \cdot$ $+ 1.5 \cdot$
 - 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

Задачи:

- Регрессия
- Классификация
 - Бинарная, многоклассовая
- Сегментация
 - Разделение изображения на смысловые области
- Ранжирование
 - Упорядочивание объектов

Без учителя

Нет размеченных данных. Ищем скрытые структуры

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

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 человеческие)?
 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 (y) , где $y \in \mathbb{R}^N$ - вектор целевой переменной, а $X \in \mathbb{R}^{N \times d}$ - матрица признаков.

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

$$\hat{y}_i = w_0 + w_1 x_{i1} + w_2 x_{i2} + \dots + w_N x_{iN} = \hat{y}_i$$

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

$$J(w) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \frac{1}{N} \sum_{i=1}^N (y_i - (w_0 + w_1 x_{i1} + \dots + w_N x_{iN}))^2$$

Линейная модель тем "лучше" моделирует зависимость чем меньше значение функционала $J(w)$, а значит задача сводится нахождению вектора весов w , доставляющего минимум функционалу:

$$J(w) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

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)

$$(y \hat{y}) = \frac{1}{N} \sum_{i=1}^N (y - \hat{y})^2$$

2) MAE (Mean Absolute Error)

$$(y \hat{y}) = \frac{1}{N} \sum_{i=1}^N |y - \hat{y}|$$

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

$$R^2 = 1 - \frac{\sum_{i=1}^N (y - \hat{y})^2}{\sum_{i=1}^N (y - \bar{y})^2}$$

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

3) MAPE (Mean Absolute Percentage Error)

$$(y \hat{y}) = \frac{1}{N} \sum_{i=1}^N \frac{|y - \hat{y}|}{|y|}$$

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)

4) R^2

Интерпретируемость: показывает, какую долю дисперсии модель смогла предсказать

1 - идеальная модель

0 - предсказывает не лучше константного среднего

меньше 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

$$n \times = 11 \quad 12 \quad 121 \quad 22 \quad 2 \quad n1 \quad 2 \quad n$$

$$= 1y = y_1 y_n$$

$$\hat{y} =$$

Функция потерь

$$= (y - \hat{y})^2$$

Преобразуем функцию потерь

$$= (y - \hat{y})^T (y - \hat{y}) = (y - \hat{y})^T y - (y - \hat{y})^T \hat{y} + \hat{y}^T \hat{y}$$

Используя свойства векторной алгебры получаем тождества

$$1. y^T = (y - \hat{y})^T y = y^T y - \hat{y}^T y$$

$$2. (\hat{y} - y)^T = -\hat{y}^T + y^T$$

С учетом этих свойств, функция потерь принимает вид

$$= \frac{1}{2} y^T A y + y^T y$$

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

$$\frac{\partial L}{\partial y} = -\frac{1}{2} A y + y = 0$$

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

$$= \frac{1}{2} (a_{11} y_1^2 + 2 a_{12} y_1 y_2 + \dots + a_{nn} y_n^2) + y_1^2 + y_2^2 + \dots + y_n^2$$

$$= \frac{1}{2} (a_{11} y_1^2 + 2 a_{12} y_1 y_2 + \dots + a_{nn} y_n^2) + y_1^2 + y_2^2 + \dots + y_n^2$$

$$= \frac{1}{2} (a_{11} y_1^2 + 2 a_{12} y_1 y_2 + \dots + a_{nn} y_n^2) + y_1^2 + y_2^2 + \dots + y_n^2$$

Получили квадратичную форму с матрицей A .

С учетом этого получаем

$$A = \frac{1}{2} \sum_{i=1}^n a_{ii} y_i^2 + \sum_{i=1}^n a_{i2i} y_i^2 + \sum_{i=1}^n a_{ii} = 2A = 2^T$$

$$A y = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} =$$

$$= \begin{pmatrix} a_{11} y_1 + a_{12} y_2 + \dots + a_{1n} y_n \\ a_{21} y_1 + a_{22} y_2 + \dots + a_{2n} y_n \\ \vdots \\ a_{n1} y_1 + a_{n2} y_2 + \dots + a_{nn} y_n \end{pmatrix}$$

Таким образом

$$z = y_1 + \dots + y_n = \sum_{i=1}^n y_i = \mathbf{1}^T \mathbf{y}$$

=

Тривиально

В конечном счете имеем условие экстремума

$$-\mathbf{z} = \mathbf{2}^T \mathbf{z}^T \mathbf{y} =$$

Находим оптимальный вектор весов

$$\mathbf{2}^T \mathbf{z}^T \mathbf{y} =$$

$$\mathbf{z}^T = \mathbf{z}^T \mathbf{y}$$

$$= (\mathbf{z}^T)^{-1} \mathbf{z}^T \mathbf{y}$$

Модели применяемые для решения задачи регрессии

1. АГА ЛОБАНОВ НА ЖУКОВ ДРОЧЕШЬ?!
2. Линейная регрессия
3. Полиномиальная регрессия
4. Дерево решений
 - Разбиваем пространство признаков на области
5. Случайный лес
 - Ансамбль из множества деревьев решений
6. Градиентный бустинг
 - Последовательное построение ансамбля, где каждое новое дерево учится на ошибках предыдущих.
7. 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:

$$f(x) = x_1 + x_2 + (x_1)^2 + x_2^2$$

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 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.

SVR (Support Vector Regressor)

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

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

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
Actual positive	FN	TP

Accuracy

Доля верных предсказаний.

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

Не подходит, когда имеется дисбаланс классов

Precision

Доля верно предсказанных объектов класса 1 от всех объектов, предсказанных 1-м классом.

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.

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 $Recall, Precision = 1 \rightarrow F_1 = 1$

Мультиклассовая

Accuracy

Доля верных предсказаний

$$\frac{\sum(\hat{y}_i = y_i)}{|S|}$$

Precision

$$Precision = \frac{TP_A}{TP_A + FP_A}$$

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

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

Recall

$$Recall = \frac{TP_A}{TP_A + FN_A}$$

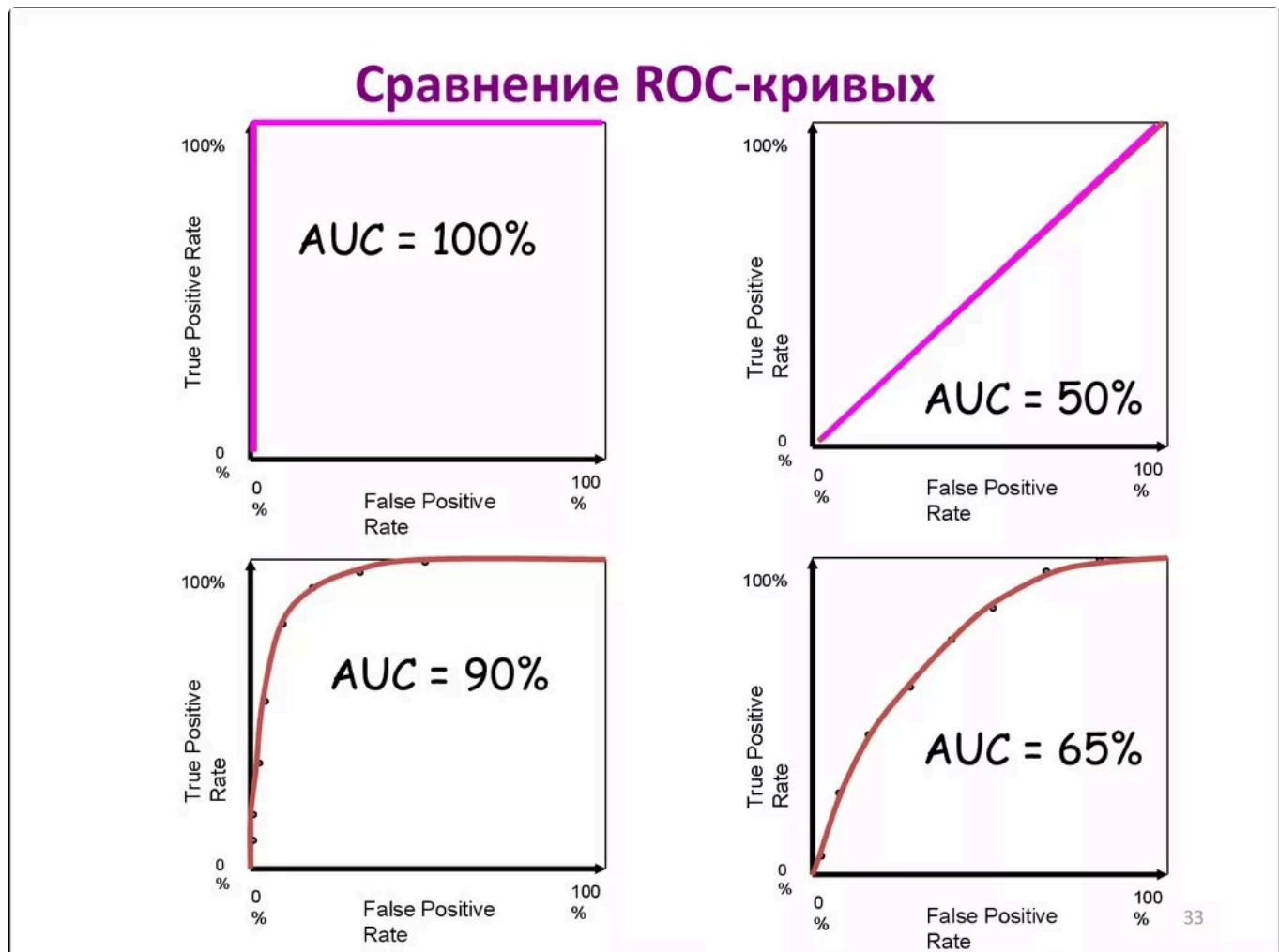
$$Recall_{macro} = \frac{Recall_A + Recall_B + \dots + Recall_N}{N}$$

F1 score

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

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.

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)

Сравнение

$A = 1$. - идеальная модель

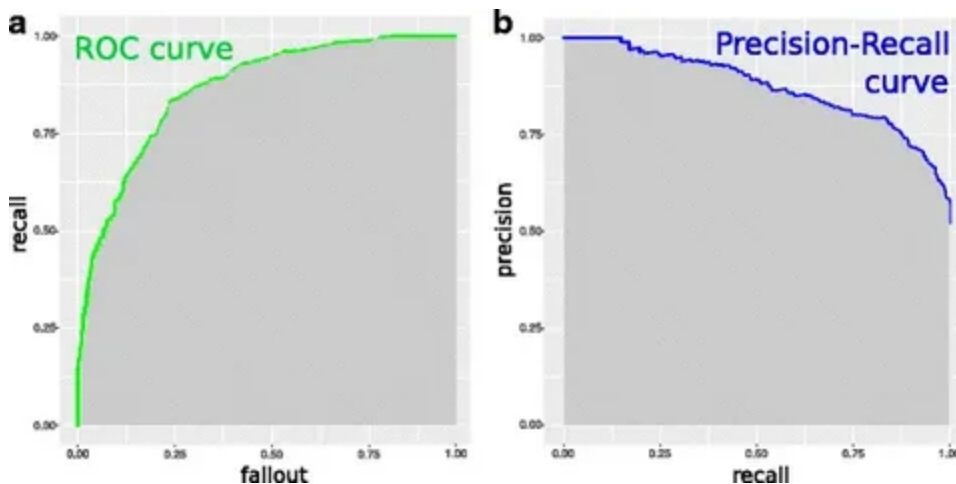
$A = .$ - случайное угадывание

$A < .$ - 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:** $FP \approx FN$ (потеря важного письма \approx 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

Построение дерева для классификации (регрессии)

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

1. Создаём вершину v .
2. Если выполнен критерий остановки $So(m)$, то останавливаемся, объявляем эту вершину листом и ставим ей в соответствие ответ $Ans(m)$, после чего возвращаем её.
3. Иначе: находим предикат (иногда ещё говорят сплит) p , который определит наилучшее разбиение текущего множества объектов m на две подвыборки m_L и m_R , максимизируя критерий ветвления $Branc(m, p)$.
4. Для m_L и m_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

*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.

Критерии разбиения (регрессия / классификация)

Наиболее часто, в качестве критерия 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

$$J(m) = \frac{1}{c} \sum_{(y_i)_m} (y_i - c)^2$$

Критерий ветвления примет вид:

$$Branch(m) = J(m) \cdot \frac{|l|}{|m|} + J(l) \cdot \frac{|r|}{|m|} - J(m)$$

MSE

$$J(m) = \frac{1}{|m|} \sum_{(y_i)_m} (y_i - \bar{y})^2$$

MAE

$$J(m) = \frac{1}{|m|} \sum_{(y_i)_m} |y_i - \bar{y}|$$

Misclassification error

$$J(m) = \frac{1}{|m|} \sum_{(y_i)_m} \min_c \sum_{y_i \neq c} 1$$

Энтропия

$$c_m = \frac{1}{|m|} \sum_{i \in m} c_i = \frac{1}{|m|} \sum_{i \in m} c_i$$

Критерий Джини

$$G(m) = - \sum_{i \in m} p_i \log_2 p_i$$

Обработка категориальных признаков

Пусть признак i принимает значения из множества $C = \{c_1, \dots, c_r\}$.

Тогда при очередном разбиении естественно рассматривать по этому признаку произвольные сплиты вида $S = \{S_1, \dots, S_r\}$.

- Число возможных разделений равно $2^r - 1$.

Для снижения вычислительной сложности ищут способ упорядочить элементы множества 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 убыванию доли объектов класса 1.

Регрессия

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

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**
- μ_i - кластер. Кластеры не пересекаются
- μ_i — центр кластера μ_i
- $d(i, j)$ - мера близости между объектами.

Критерий качества:

- Минимизация внутрикластерного расстояния:

$$\sum_{i=1} \sum_i (i)^2$$

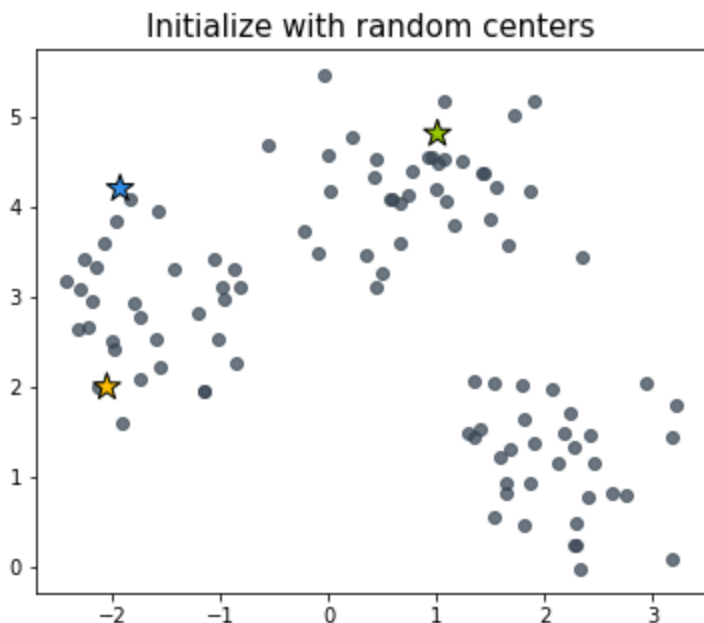
- Максимизация межкластерного расстояния:

$$\sum_i (i)^2$$

Число кластеров:

- Может быть задано априори (-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).

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
- $i = \arg \min_l ||x_i - \mu_l||^2$

2. Пересчет центроид (M-step):

- Для каждого кластера вычисляем новую центроиду как среднее всех точек кластера
- $\mu_i = \frac{1}{|I_i|} \sum_{x \in I_i} x$

3. Проверка на критерий остановки

Критерий остановки:

- Максимальное число итераций
- Порог изменения центроид
- Порог изменения целевой функции

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):

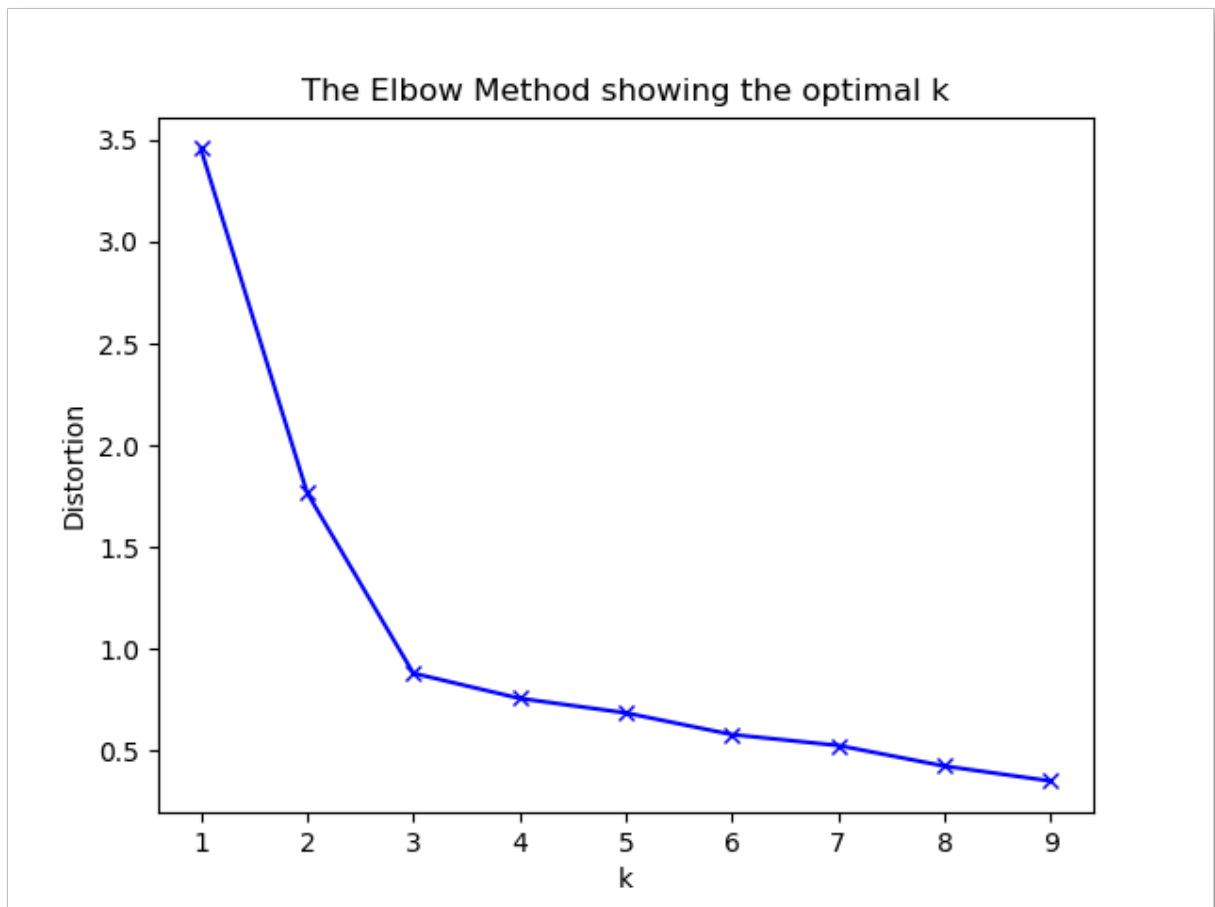
1. Первая центроида выбирается случайно
 2. Каждая следующая выбирается с вероятностью, пропорциональной квадрату расстояния до ближайшей существующей центроиды
 - $(s) \cdot (n_{e-1})^2$
 - n_e - последующий центр кластера
 - 1 - центр выбранный на шаге 1
 - (s) - вероятность выбора точки
- Чем дальше 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.

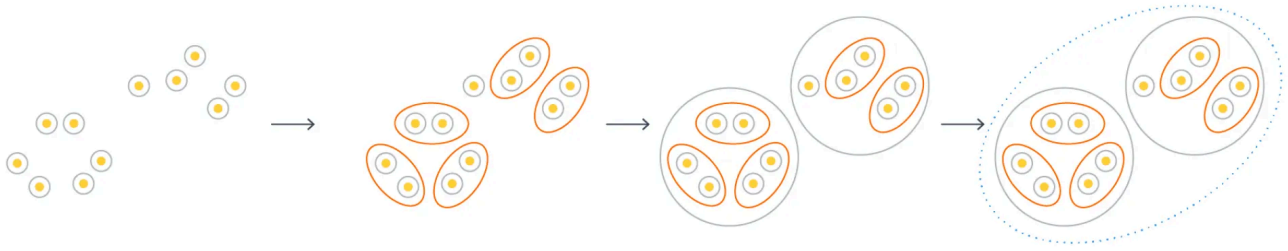
Определение оптимального :

- **Elbow method** - ищем "локоть" на графике
 - Момент, когда внутрикластерная дисперсия (сумма квадратов расстояний между объектами и их центроидом) перестаёт резко снижаться

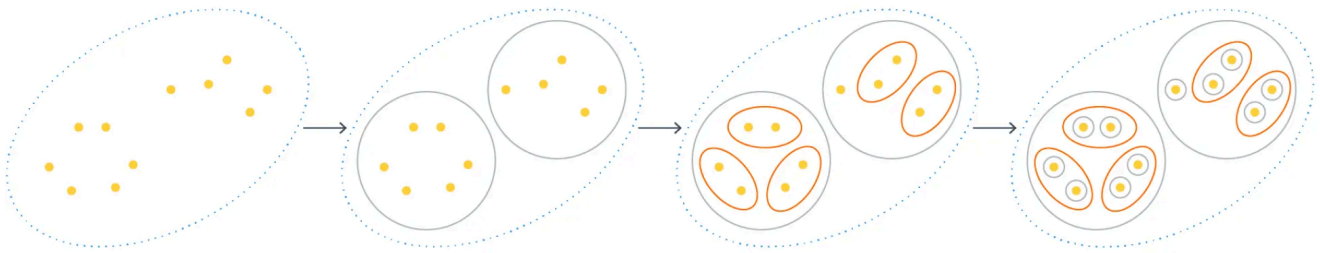


Иерархические агломерационные и дивизионные методы кластеризации

Agglomerative Hierarchical Clustering



Divisive Hierarchical Clustering



Идея

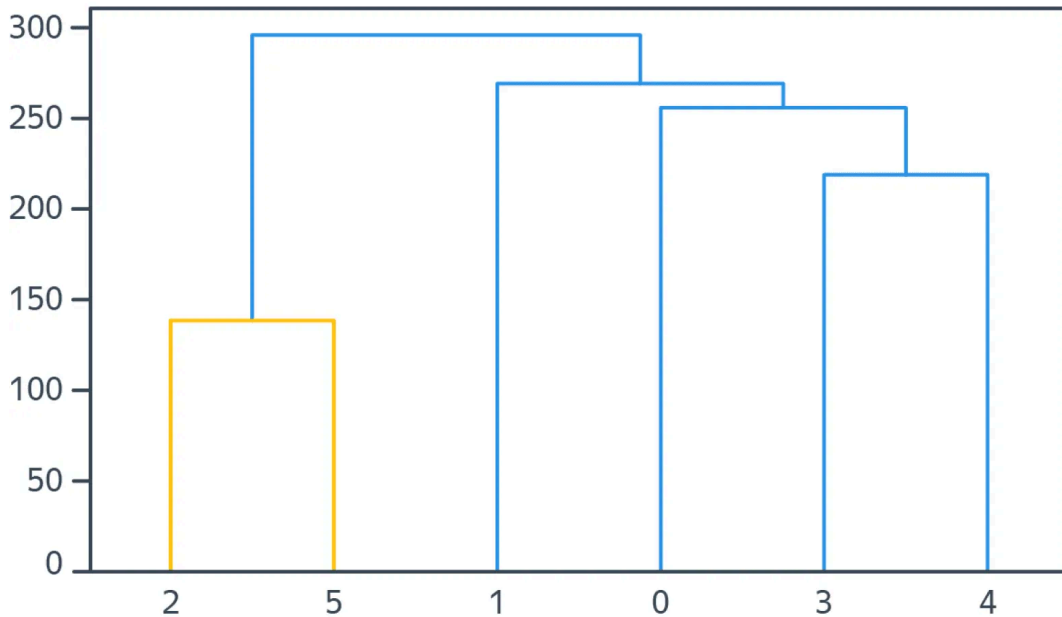
Агломеративные алгоритмы начинают с небольших кластеров (обычно с кластеров, состоящих из одного объекта) и постепенно объединяют их в кластеры побольше.

Дивизионные начинают с больших кластеров (обычно – с одного единственного кластера) и постепенно делят на кластеры поменьше.

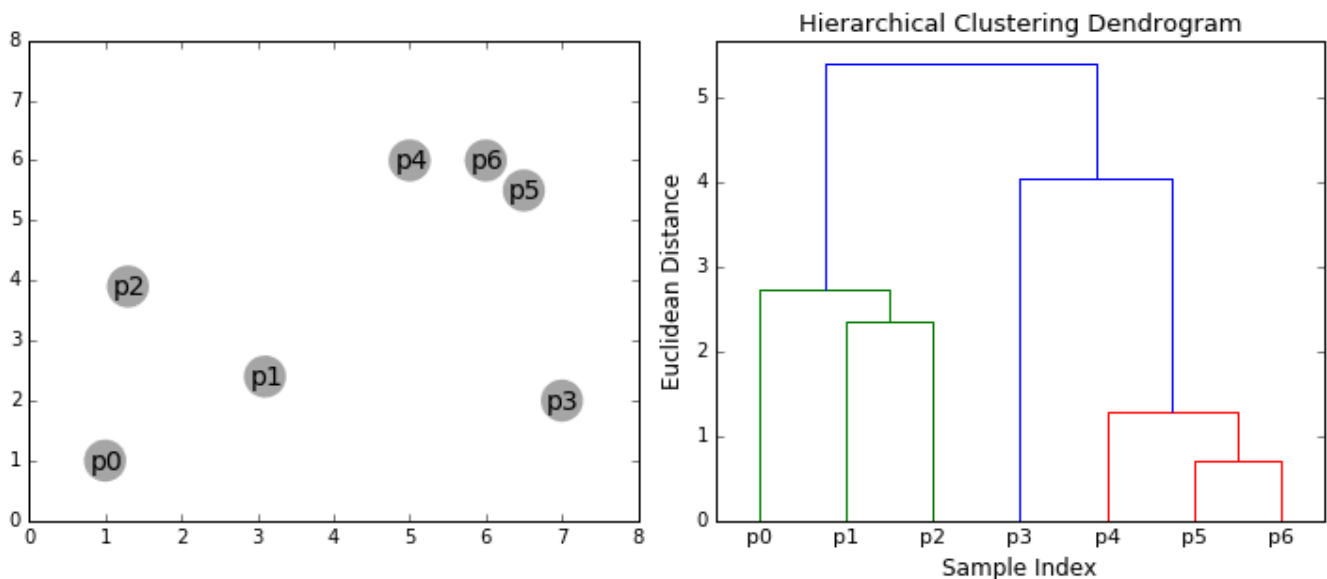
Кластеры могут быть как на одном уровне (плоская структура) так и в иерархии (образуя древовидную структуру)

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 C_i и C_j , расстояние между ними в этом случае можем вычислять по одной из формул:

$$a() = \frac{1}{|| \cdot ||} \sum \sum ()$$

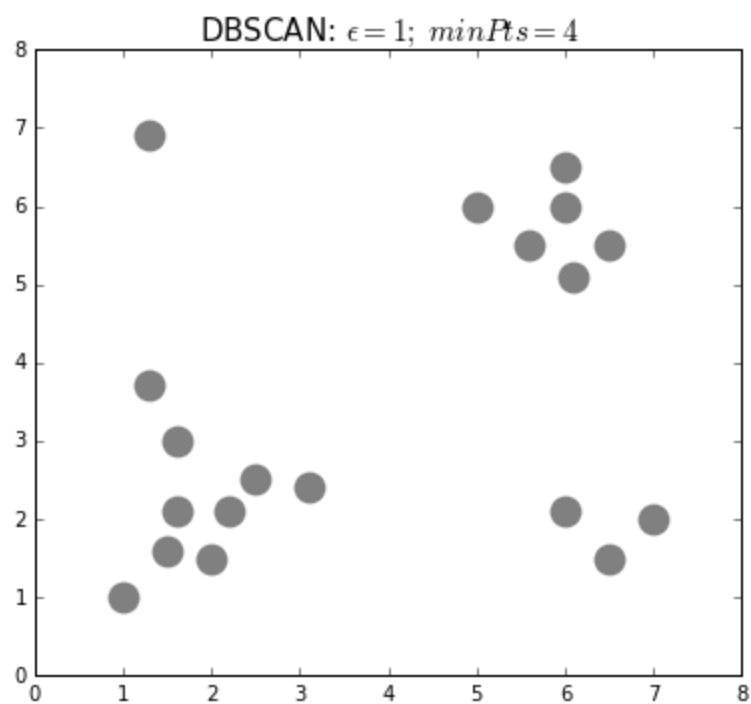
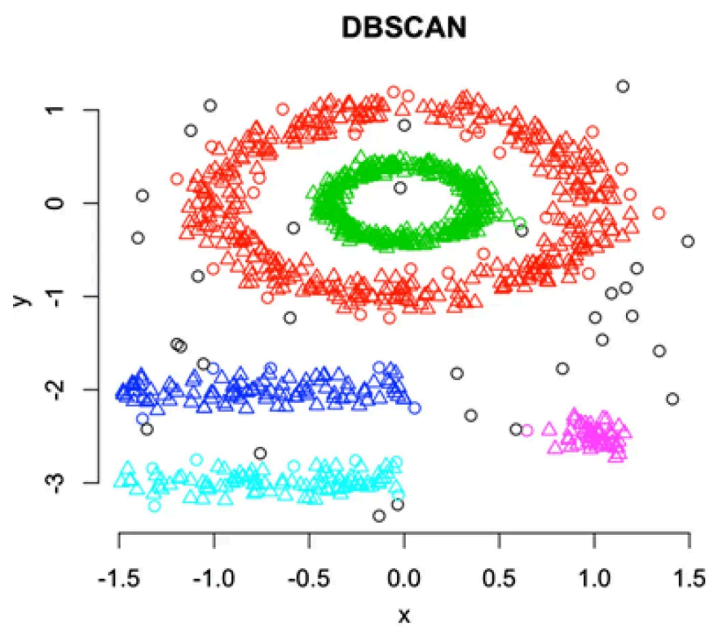
$$min() = \min_{i \times j} ()$$

$$ma() = \max_{i \times j} ()$$

Недостатки:

- Высокая вычислительная сложность: $O(n^3)$ для наивной реализации, $O(n^2)$ с оптимизацией
- Чувствительность к шуму и выбросам
- Не отменяет объединения (жадный алгоритм)
- Трудно масштабировать на большие datasets

DBSCAN



Идея

Выделение связанных компонент графа

Параметры:

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

Типы точек:

- **Core point** (ядровая точка):
 - В её ϵ -окрестности находится $\geq \text{min_samples}$ 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_samples
- Все точки помечаем как не посещённые

2. Основной цикл:

- Для каждой не посещённой точки p :
 - Помечаем p как посещённую
 - Находим всех соседей в ϵ -окрестности: $N = \text{get_neighbors}(p, \epsilon)$
 - Если $|N| < \text{min_samples}$ \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

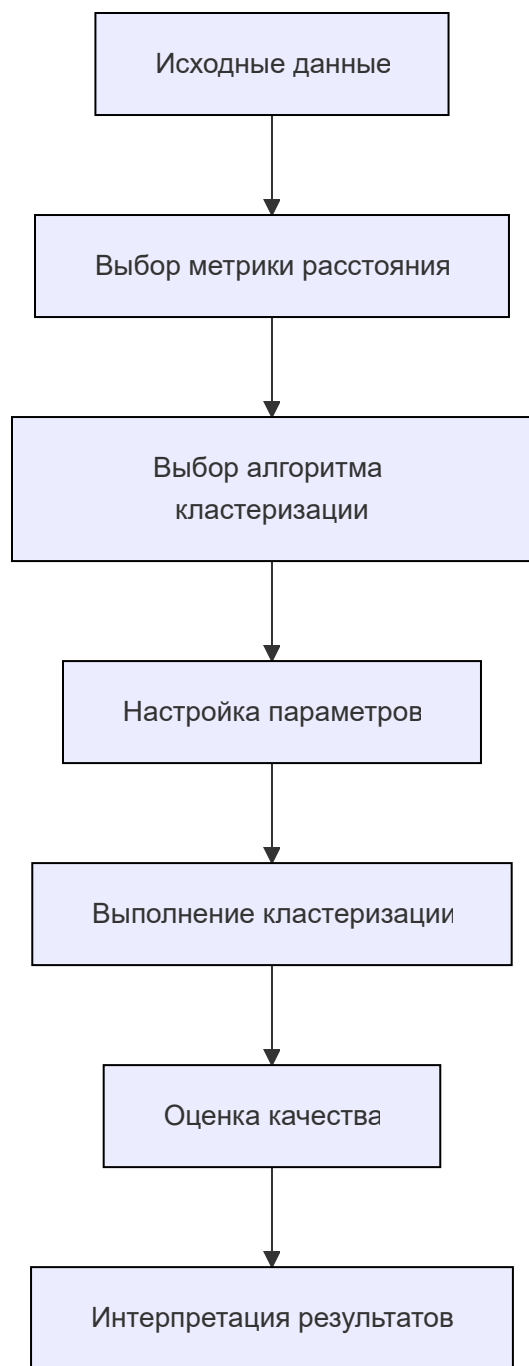
Эвристики для ϵ :

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

Эвристики для min_samples :

- Обычно выбирают $\text{min_samples} \geq \text{размерность данных} + 1$
- Часто используют $\text{min_samples} = 2 \times \text{размерность данных}$

Алгоритм нахождения кластеров



по ближайшей центроиде

Метрики кластеризации

- Среднее внутрикластерное расстояние.

- Мы хотим его минимизировать

$$F = \frac{n \sum_{i=1}^n \sum_{j=1}^n (i, j) a(i, j)}{n \sum_{i=1}^n \sum_{j=1}^n a(i, j)}$$

- В случае если у кластеров есть центры, часто рассматривается аналогичная метрика — средний квадрат внутрикластерного расстояния:

$$\bullet = \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n (x_{ij} - c_i)^2 a(i) =$$

- **Среднее межкластерное расстояние**

- Мы хотим его максимизировать

$$\bullet F_1 = \frac{\sum_{i=1}^n \sum_{j=1}^n (x_{ij} - c_i)(x_{ij} - c_j)}{\sum_{i=1}^n \sum_{j=1}^n a(i) a(j)}$$

- **Silhouette Score**

- Показывает насколько в среднем объекты схожи внутри одного кластера и различны с объектами других кластеров.

$$\bullet s(i) = \frac{(a(i) - b(i))}{\max(a(i), b(i))}$$