

# Поиск аномалий и несбалансированные данные

# Повторение

# Кластеризация

- Дано: матрица «объекты-признаки»  $X$
- Найти:
  1. Множество кластеров  $Y$
  2. Алгоритм кластеризации  $a(x)$ , который приписывает каждый объект к одному из кластеров
- Каждый кластер состоит из похожих объектов
- Объекты из разных кластеров существенно отличаются

# Отличия

## Обучение с учителем

- Цель: минимизация функционала ошибки
- Множество ответов известно заранее
- Конкретные способы измерения качества

## Кластеризация

- Нет строгой постановки
- Множество кластеров неизвестно
- Правильные ответы отсутствуют (в большинстве случаев) — нельзя измерить качество

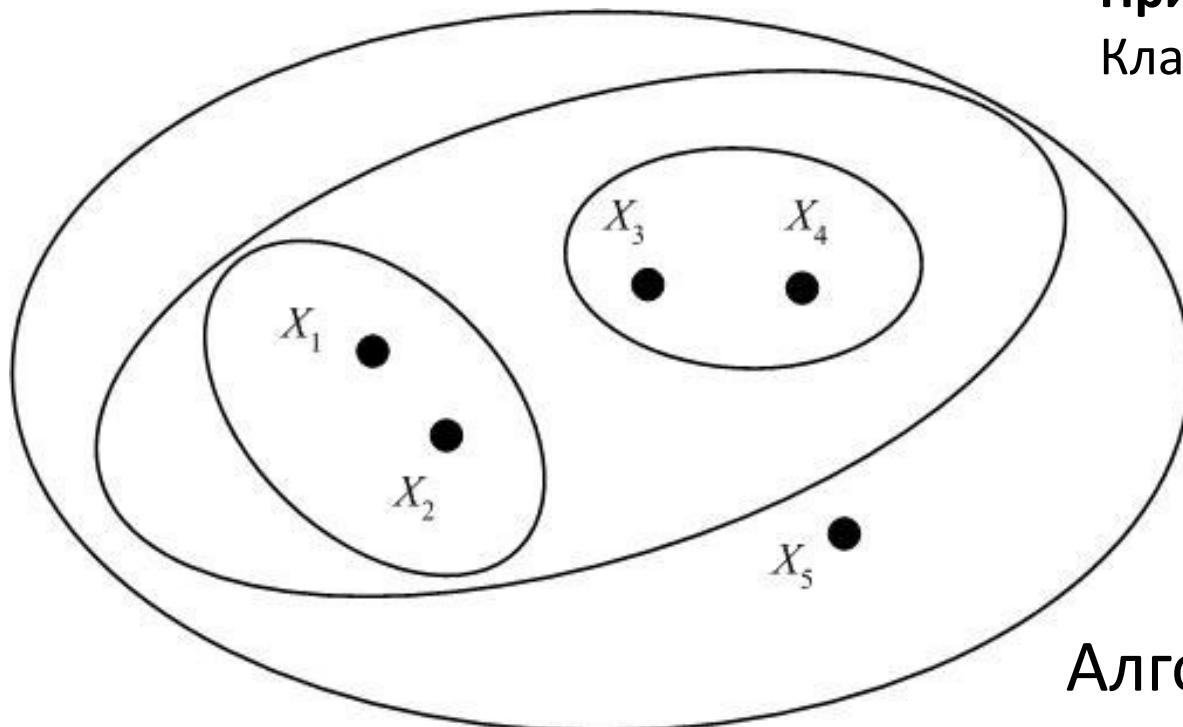
# Зачем кластеризовать?

- Маркетинг: искать похожих клиентов
  - Модерация: проверять только одно сообщение из кластера
  - Соц. опросы: выделять группы схожих анкет
  - Соц. сети: искать сообщества
- 
- Выявлять типы людей и формировать поведенческие паттерны для каждого типа

# Форма кластеров



# Иерархическая кластеризация



Пример:

Кластеризация статей на Хабре

IT

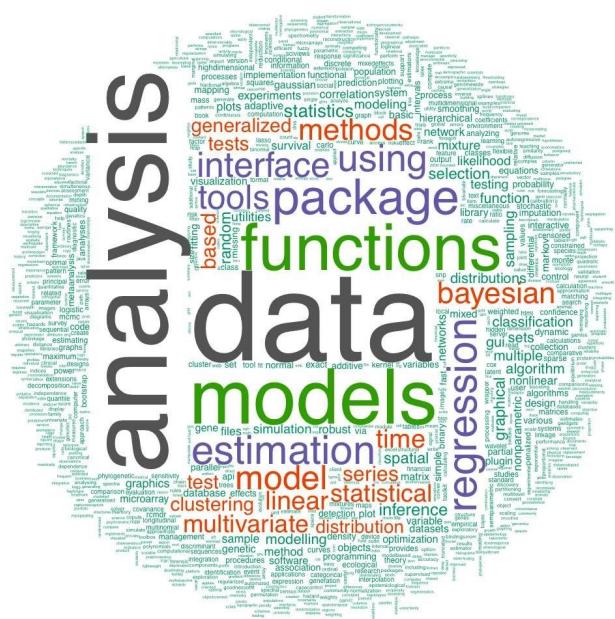
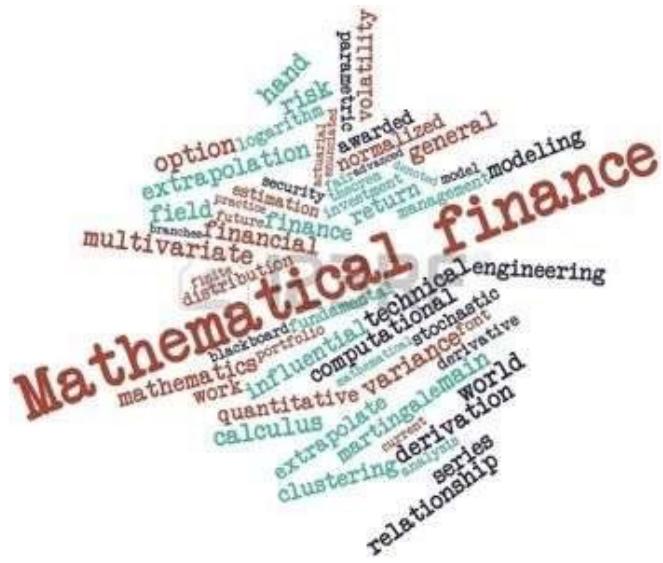
Алгоритмы

Алгоритмы  
и структуры  
данных

Методы  
машинного  
обучения

# «Жесткая» и «мягкая» кластеризации

# Кластеризация для выделения «тем»



# Типы задач кластеризации

- Форма кластеров, которые нужно выделять
- Плоская или древовидная структура
- Размер кластеров
- Конечная задача или вспомогательная
- Жесткая или мягкая кластеризация

# K-Means

- Дано: выборка  $x_1, \dots, x_\ell$
- Параметр: число кластеров  $K$
- Начало: случайно выбрать  $K$  центров кластеров  $c_1, \dots, c_K$
- Повторять по очереди до сходимости:

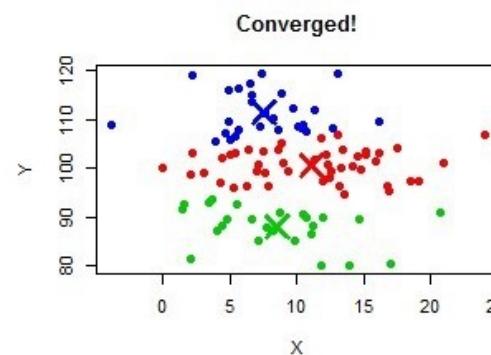
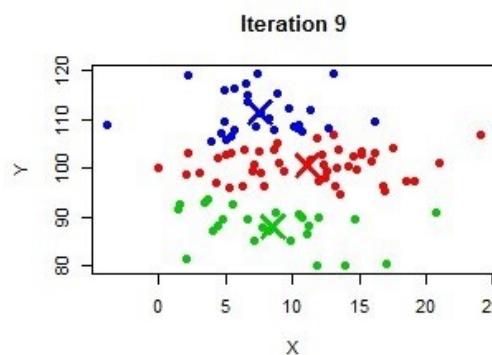
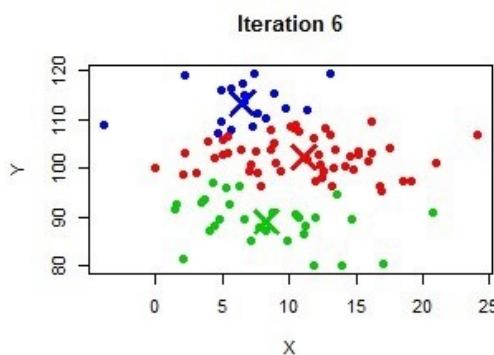
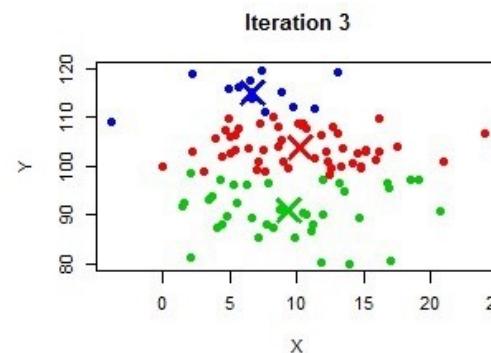
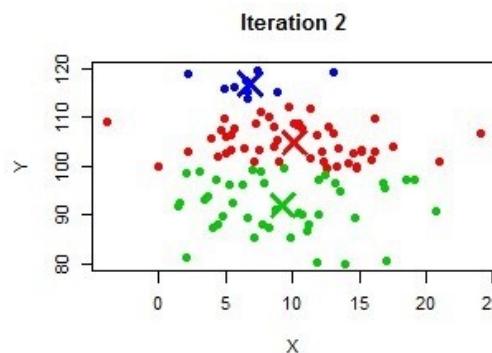
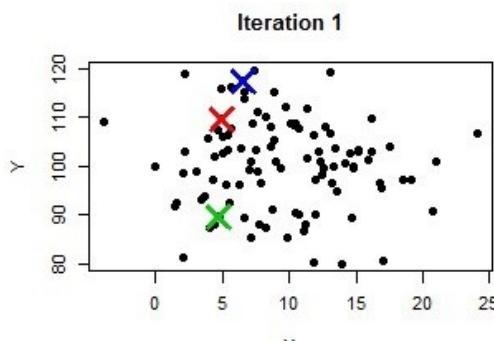
- Шаг А: отнести каждый объект к ближайшему центру

$$y_i = \arg \min_{j=1, \dots, K} \rho(x_i, c_j)$$

- Шаг Б: переместить центр каждого кластера в центр тяжести

$$c_j = \frac{\sum_{i=1}^{\ell} x_i [y_i = j]}{\sum_{i=1}^{\ell} [y_i = j]}$$

# K-Means



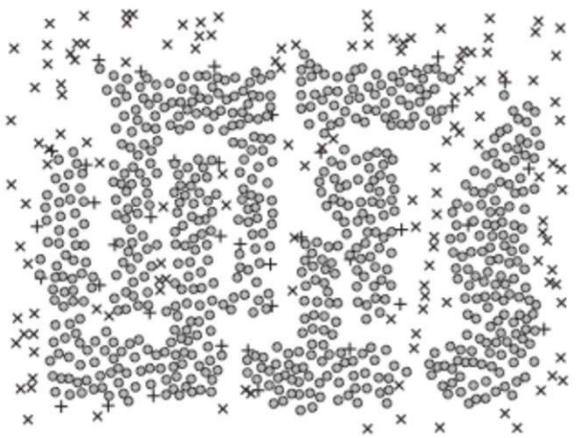
# Параметры DBSCAN

- Размер окрестности (eps)
- Минимальное число объектов в окрестности — для определения основных точек

# DBSCAN



(a) Clusters found by DBSCAN.



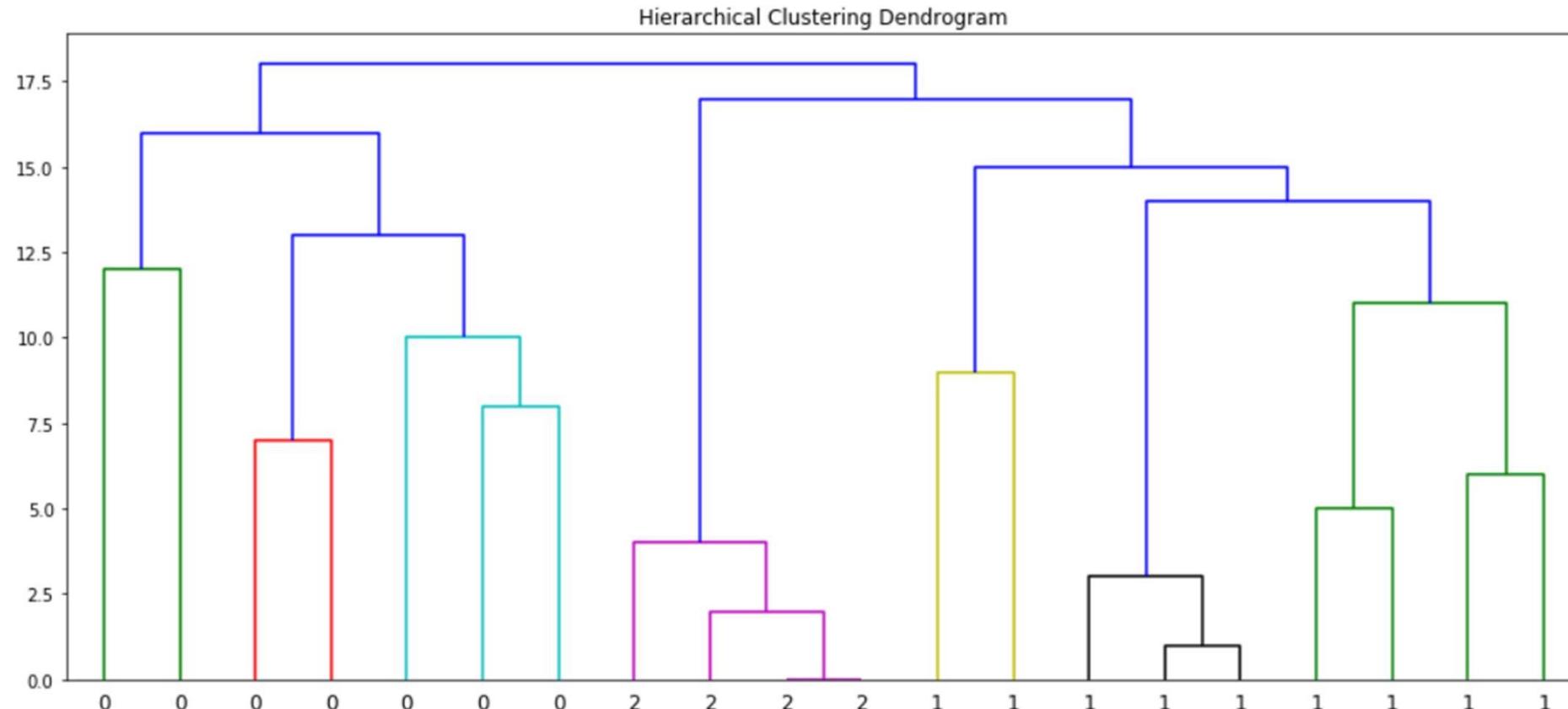
(b) Core, border, and noise points.

1. Выбрать точку без метки
2. Если в окрестности меньше  $N$  точек, то пометить как шумовую
3. Создать новый кластер, поместить в него текущую точку
4. Для всех точек из окрестности  $S$ : (а) если точка шумовая, то отнести к данному кластеру, но не использовать для расширения; (б) если точка основная, то отнести к данному кластеру, а её окрестность добавить к  $S$
5. Перейти к шагу 1

# Виды иерархической кластеризации

- Аггломеративная – на каждой итерации объединяем два меньших кластера в один побольше
- Дивизивная – на каждой итерации делим один большой кластер на два поменьше

# Виды иерархической кластеризации



# Агglomerативная кластеризация

1. Инициализация – каждая точка = кластер
2. Самые близкие (относительно какой-то метрики) кластеры объединяются
3. Повторяем до того момента, когда все точки будут в одном кластере
4. Останавливаемся, когда достигаем фиксированного числа кластеров, либо когда расстояние между кластерами больше заданного порога

# Векторные представления слов

Хотим представить каждое слово в виде вещественного вектора:

$$w \rightarrow \vec{w} \in \mathbb{R}^d$$

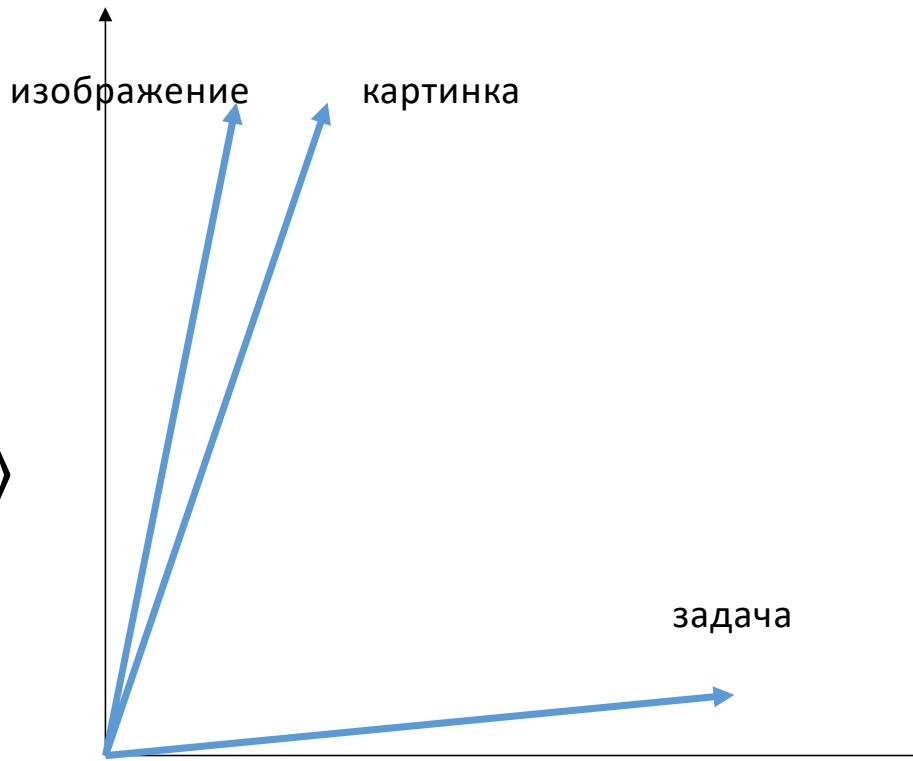
Требования к представлениям (embeddings):

- Размерность  $d$  должна быть не очень большой
- Похожие слова должны иметь близкие векторы
- Арифметические операции над векторами должны иметь смысл

# word2vec

Задача:

- Для каждого слова  $w$  построить вектор  $\vec{w}$
- Если два слова  $w_1$  и  $w_2$  идут рядом, то скалярное произведение  $\langle \vec{w}_1, \vec{w}_2 \rangle$  должно быть большим



# Тематическое моделирование

- Рассматриваем каждый документ как мешок слов
- Всего  $K$  тем
- Тема — распределение на словах
- Документ — распределение на темах

# Модель PLSA

- Probabilistic Latent Semantic Analysis

$$p(w|d) = \sum_{t \in T} p(w|t)p(t|d) = \sum_{t \in T} \varphi_{wt}\theta_{td}$$

- $T$  — множество тем
- $p(w|t) = \varphi_{wt}$  — распределение слов в теме  $t$
- $p(t|d) = \theta_{td}$  — распределение тем в документе  $d$

# Модель PLSA

- Probabilistic Latent Semantic Analysis

$$\sum_{d \in D} \sum_{w \in d} n_{dw} \log p(w|d) \rightarrow \max_{\varphi_{wt}, \theta_{td}}$$

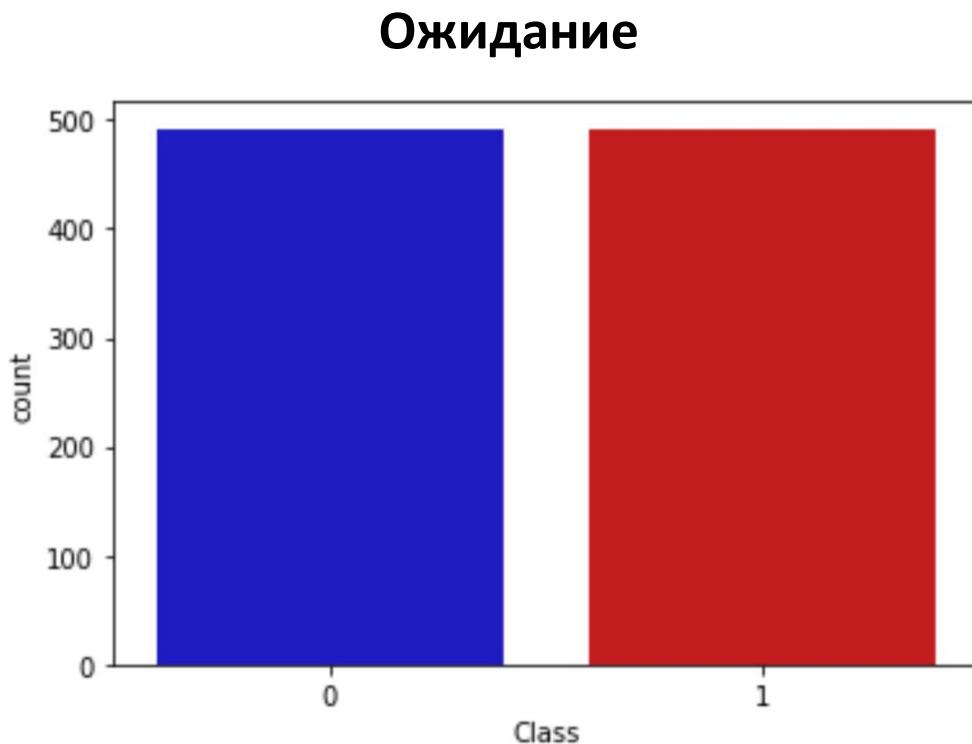
Ограничения:  $\varphi_{wt} \geq 0, \theta_{td} \geq 0, \sum_{w \in W} \varphi_{wt} = 1, \sum_{t \in T} \theta_{td} = 1$

- $D$  — множество документов
- $W$  — множество слов

# Несбалансированные данные

# Задача классификации

- Посмотрим на баланс классов:



# Проблемы

- Определение дефектных деталь на производстве
- Вы построили модель, которая корректно определяет, дефектная ли деталь, в 99.9% случаев
- Хорошая ли это модель?
- Ответ: необязательно.

# Проблемы

- Хороший случай:
  - 30% деталей дефектны
  - 70% деталей не дефектны
- Модель, которая дает 99.9% правильных ответов – вполне осмысленная

# Проблемы

- Плохой случай:
  - 0.1% деталей дефектны
  - 99.9% деталей не дефектны
- Модель, которая дает 99.9% правильных ответов – не особо осмысленная
- Она просто выдает константные предсказания!

# Определение несбалансированности

- Данные **несбалансированы**, если число наблюдений одного класса сильно больше, чем число наблюдений других классов
- Что значит *сильно больше*?
- Явного порога нет, это зависит от задачи
- Соотношение классов 10:1 можно считать несбалансированностью

# Резюме

- В задаче классификации данные могут быть несбалансированы, то есть наблюдений одного класса существенно больше, чем других
- Если модель дает много правильных ответов, это не значит, что она хорошая
- Проверьте баланс классов

# Метрики качества

# Accuracy

$$\text{accuracy} = \frac{\#(\text{correct predictions})}{\#(\text{observations})}$$

- $(x_i, y_i)$  – наблюдения и метки классов
- $\ell$  – общее число наблюдений
- $a$  – классификатор

$$\text{accuracy} = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(x_i) = y_i]$$

# Accuracy: проблемы

- Число дефектных деталей (класс 1): 10
- Число нормальных деталей (класс -1): 10001

$$\forall x: a(x) = -1$$

- Хорошая ли это модель?
- Ответ: в терминах accuracy – да, в терминах бизнеса – нет

# Accuracy: проблемы

- Что хуже?
  - Ошибиться, назвав нормальную деталь дефектной
  - Ошибиться, назвав дефектную деталь нормальной
- Одна ошибка в нормальных деталях:  $\approx -0.01\%$  accuracy
- Одна ошибка в дефектных деталях:  $\approx -0.01\%$  accuracy
- Возможно, ошибка в дефектной детали хуже

# Accuracy: проблемы

- Алгоритму удобно предсказывать мажоритарный класс для всех наблюдений
- Мы должны изменить процедуру обучения и/или метрику качества

# Точность и полнота

|             | $y = 1$             | $y = -1$            |
|-------------|---------------------|---------------------|
| $a(x) = 1$  | True Positive (TP)  | False Positive (FP) |
| $a(x) = -1$ | False Negative (FN) | True Negative (TN)  |

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

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

# Точность и полнота

- **Точность (precision)** показывает, насколько сильно мы можем доверять нашему алгоритму, если он предсказывает положительный класс
- **Полнота (recall)** показывает долю наблюдений положительного класса, верно предсказываемых алгоритмом

# Точность и полнота

- Число дефектных деталей (класс 1): 10
- Число нормальных деталей (класс -1): 10001
- Модель корректно распознает 4 дефектных детали из 10
- Модель корректно распознает 10000 нормальных деталей из 10001
- Хорошая ли это модель?
- Ответ: зависит от того, что вы хотите.

# Точность и полнота

|             | $y = 1$ | $y = -1$ |
|-------------|---------|----------|
| $a(x) = 1$  | 4       | 1        |
| $a(x) = -1$ | 6       | 10000    |

$$\text{precision} = \frac{4}{4 + 1} = 0.8$$

$$\text{recall} = \frac{4}{4 + 6} = 0.4$$

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} = \frac{4 + 10000}{10011} \approx 0.9993$$

# Точность и полнота

- Случай 1. Низкая точность, высокая полнота
  - Часто отмечаем нормальные детали как дефектные
  - Зато редко пропускаем дефектные детали
- Случай 2. Высокая точность, низкая полнота
  - Редко отмечаем нормальные детали как дефектные
  - Зато часто пропускаем дефектные детали

# F-мера

- F-мера является гармоническим средним точности и полноты

$$F\text{-score} = 2 \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

- $F_\beta$ -мера является взвешенной версией F-меры, где можно сделать больший акцент на точность либо полноту

$$F_\beta\text{-score} = (1 + \beta^2) \frac{\text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$$

## F-мера: проблемы

- Точность, полнота и F-мера не учитывают True Negatives (TN) – количество верных предсказаний для наблюдений отрицательного класса
- Однако, если вас не интересуют True Negatives, это вполне нормально

# F-мера: проблемы

- Какой случай лучше?

|             | $y = 1$ | $y = -1$ |
|-------------|---------|----------|
| $a(x) = 1$  | 4       | 1        |
| $a(x) = -1$ | 6       | 10000    |

$$\text{precision} = 0.8$$

$$\text{recall} = 0.4$$

|             | $y = 1$ | $y = -1$ |
|-------------|---------|----------|
| $a(x) = 1$  | 4       | 1        |
| $a(x) = -1$ | 6       | 10       |

$$\text{precision} = 0.8$$

$$\text{recall} = 0.4$$

# Balanced accuracy

- True Positive Rate (полнота):

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- True Negative Rate (специфичность):

$$\text{TNR} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

# Balanced accuracy

- **Balanced accuracy** – это среднее TPR and TNR

$$\text{Balanced accuracy} = \frac{\text{TPR} + \text{TNR}}{2}$$

# Balanced accuracy

|             | $y = 1$ | $y = -1$ |
|-------------|---------|----------|
| $a(x) = 1$  | 4       | 1        |
| $a(x) = -1$ | 6       | 10000    |

$$\text{TPR} = 0.4$$

$$\text{TNR} \approx 0.9999$$

Balanced accuracy  $\approx 0.7$

|             | $y = 1$ | $y = -1$ |
|-------------|---------|----------|
| $a(x) = 1$  | 4       | 1        |
| $a(x) = -1$ | 6       | 10       |

$$\text{TPR} = 0.4$$

$$\text{TNR} \approx 0.91$$

Balanced accuracy  $\approx 0.65$

# MCC

- **Matthews correlation coefficient (MCC)** – это сбалансированная метрика, которая отражает корреляцию между правильными ответами и предсказаниями

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \in [-1, 1]$$

# MCC

|             | $y = 1$ | $y = -1$ |
|-------------|---------|----------|
| $a(x) = 1$  | 4       | 1        |
| $a(x) = -1$ | 6       | 10000    |

$MCC \approx 0.566$

|             | $y = 1$ | $y = -1$ |
|-------------|---------|----------|
| $a(x) = 1$  | 4       | 1        |
| $a(x) = -1$ | 6       | 10       |

$MCC \approx 0.362$

# Метрики качества ранжирования

- Пусть классификатор  $b(x)$  выдает вероятности принадлежности классам
- Подобрав порог  $t$ , можно построить следующий классификатор:

$$a(x) = \text{sign}(b(x) - t)$$

# Метрики качества ранжирования

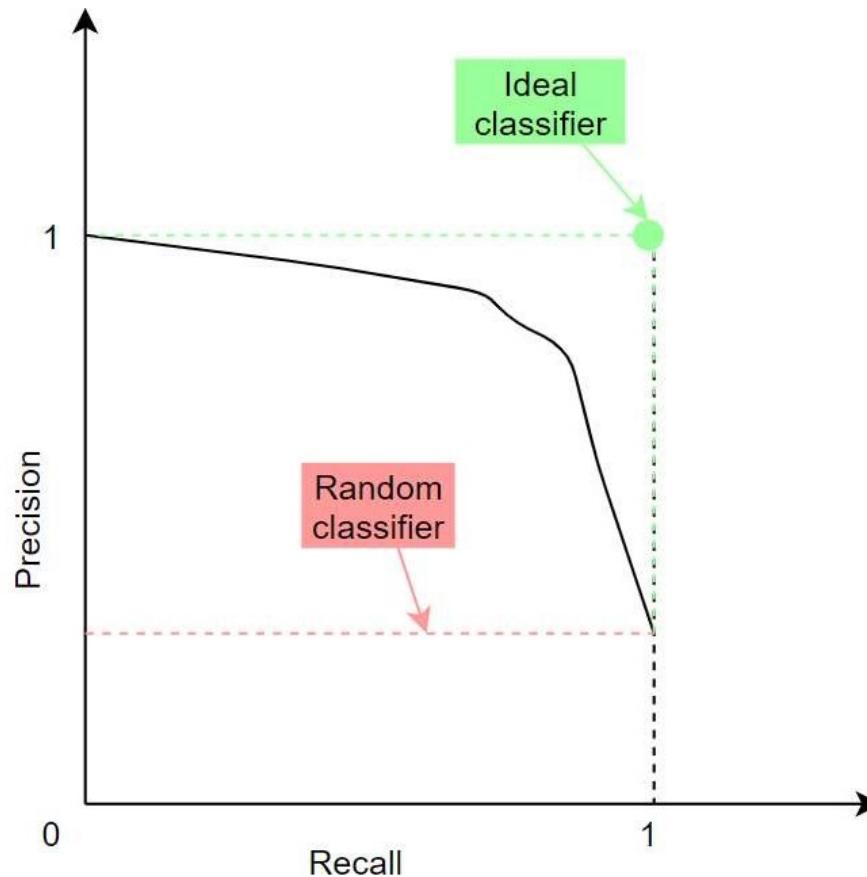
- Значения точности и полноты зависят от порога  $t$

| $y$    | 1   | -1  | 1    | -1  | -1   | 1   | 1   |
|--------|-----|-----|------|-----|------|-----|-----|
| $b(x)$ | 0.1 | 0.2 | 0.25 | 0.4 | 0.45 | 0.7 | 0.9 |

- $t = 0.3$ :
  - precision = 0.5
  - recall = 0.5
- $t = 0.8$ :
  - precision = 1
  - recall = 0.25

# PR-кривая и AUC-PR

- При изменении  $t$  меняются значения точности и полноты
- AUC-PR – площадь под PR-кривой



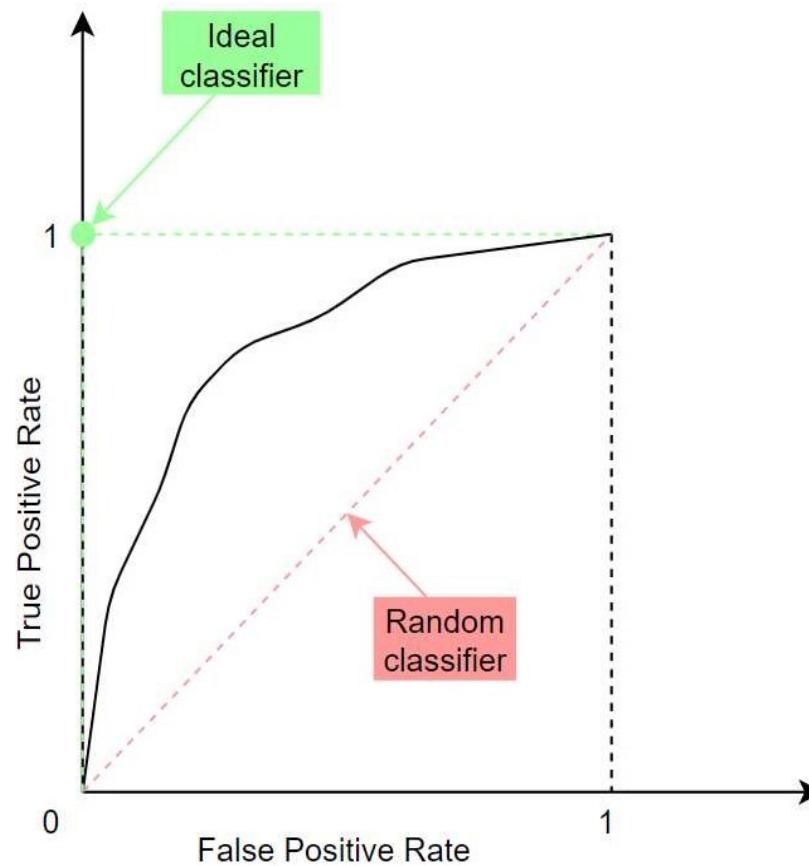
# ROC-кривая и AUC-ROC

- При изменении  $t$  меняются значения TPR и FPR

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

- AUC-ROC – площадь под ROC-кривой



# AUC-PR vs AUC-ROC

- AUC-PR и AUC-ROC зачастую ведут себя похоже
- В случае сильного дисбаланса, если хочется учитывать TN, возможно, стоит использовать AUC-ROC
- Если не хочется учитывать TN, то AUC-ROC может вводить в заблуждение
- AUC-PR может быть более интерпретируемой и подходящей метрикой, если задачу нужно решить в терминах точности/полноты

# Резюме

- Есть много метрик для проверки качества модели на несбалансированных данных – какую именно использовать, зависит от постановки задачи
- Accuracy может вводить в заблуждение
- Точность, полнота и F-мера учитывают стоимость ошибки
- Balanced accuracy и MCC также учитывают True Negatives
- Метрики качества ранжирования: AUC-PR и AUC-ROC

# Балансирование данных

# Веса классов

$$L(y, z) = -[y = 1] \times \log(z) - [y = -1] \times \log(1 - z)$$

- Функция потерь для логистической регрессии
- Класс 1: 10 наблюдений, класс -1: 10000 наблюдений
- `class_weight = {1: 1000, -1: 1}` (`#negatives/#positives`)

$$L(y, z) = -\mathbf{1000}[y = 1] \times \log(z) - [y = -1] \times \log(1 - z)$$

# Веса классов

$$L(y, z) = -[y = 1] \times \log(z) - [y = -1] \times \log(1 - z)$$

- Штраф за ошибку в положительном наблюдении:  $-\log(z)$
- Штраф за ошибку в отрицательном наблюдении:  $-\log(1 - z)$

$$L(y, z) = -\mathbf{1000}[y = 1] \times \log(z) - [y = -1] \times \log(1 - z)$$

- Штраф за ошибку в положительном наблюдении:  $-\mathbf{1000} \times \log(z)$
- Штраф за ошибку в отрицательном наблюдении:  $-\log(1 - z)$

# Веса классов

- Логистическая регрессия, SVM, случайный лес: class\_weight
- XGBoost, LightGBM, CatBoost: scale\_pos\_weight

# Undersampling

- **Undersampling** – это техника балансирования данных, при которой уменьшается число наблюдений мажоритарного класса

# Random undersampling

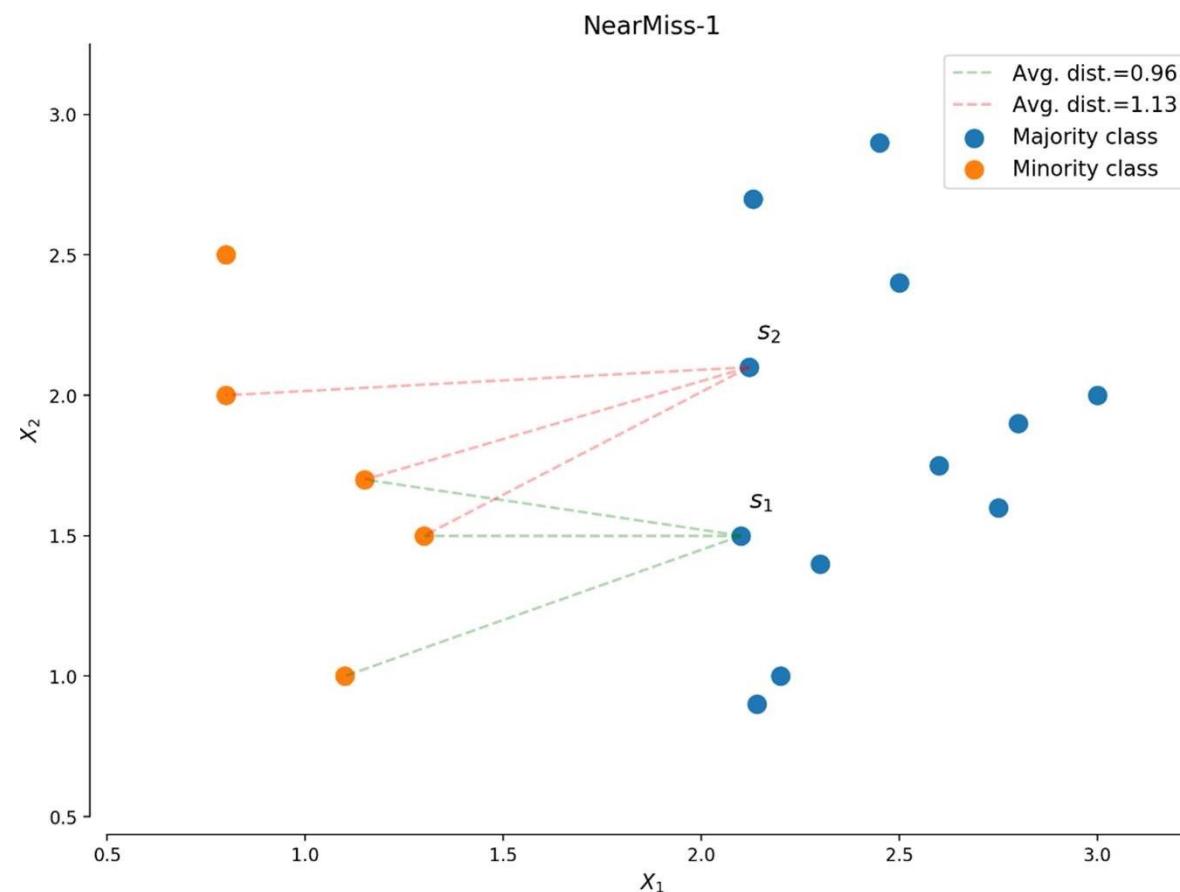
- Простейший метод: **random undersampling** (удаляем случайные объекты мажоритарного класса)
- Скорее всего, повлечет за собой потерю качества (можем удалить важные объекты)

# NearMiss

- Хотим контролировать процесс удаления объектов мажоритарного класса и сделать его менее случайным
- Будем использовать расстояния между объектами положительного и отрицательного классов
- Используем алгоритм kNN (k Nearest Neighbors) для определения близких и далеких объектов

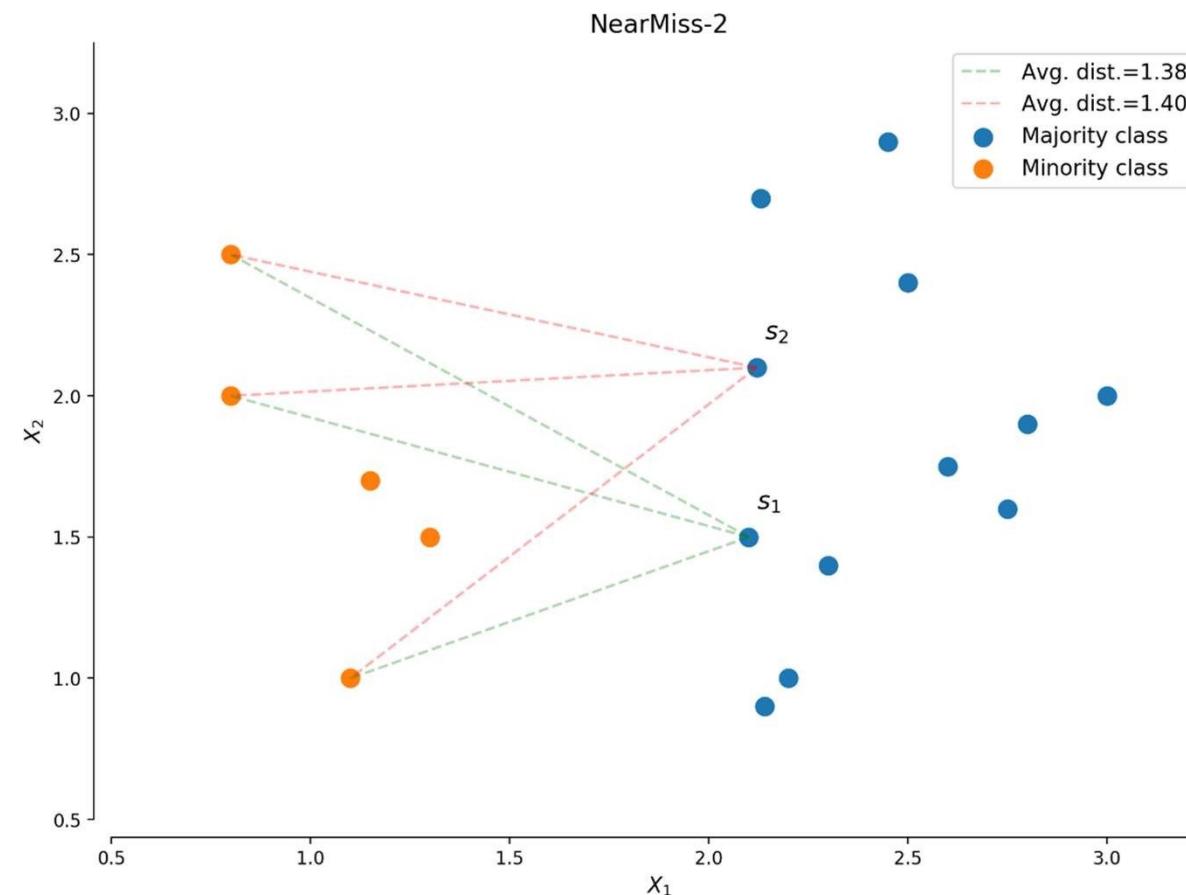
# NearMiss-1

- Сохраняем  $M$  объектов мажоритарного класса, имеющих наименьшее среднее расстояние до  $k$  самых близких объектов миноритарного класса
- Пример:  $M = \#\text{(minority class observations)}$ ,  $k = 3$



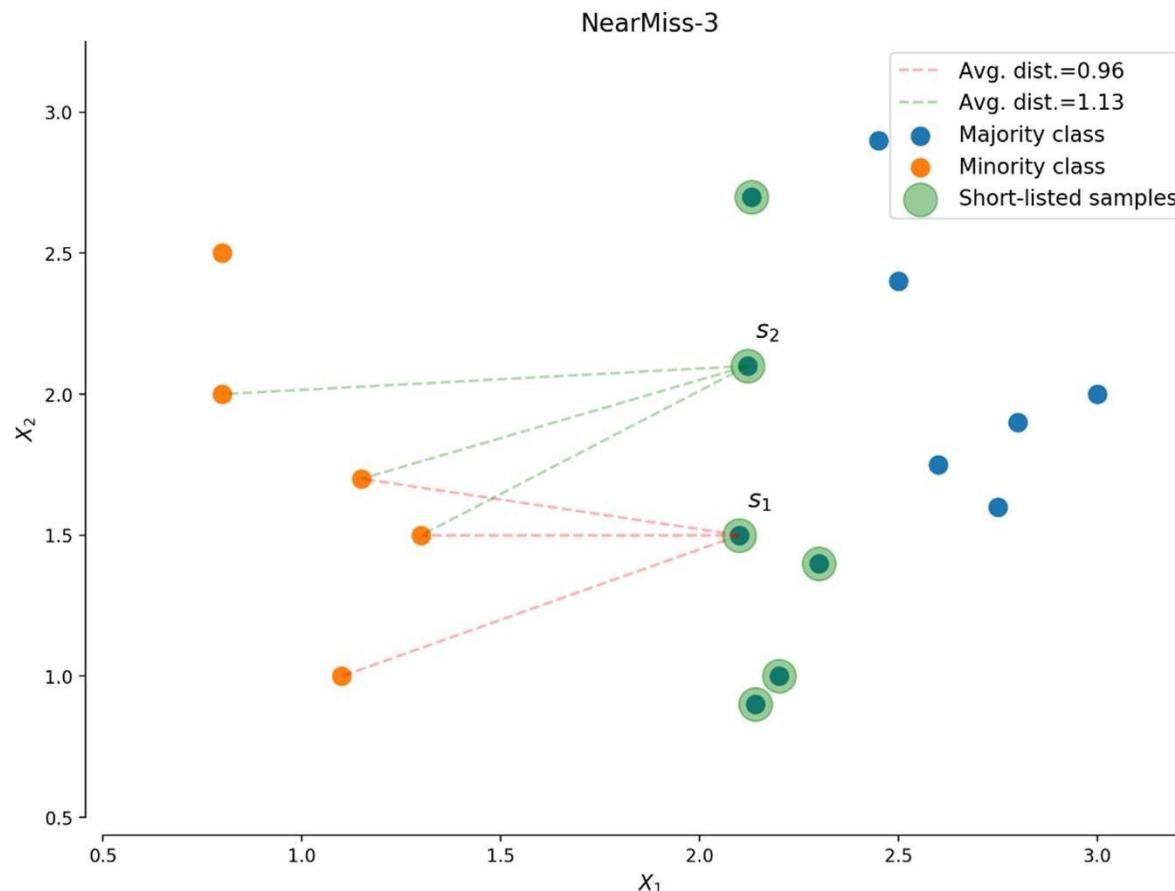
# NearMiss-2

- Сохраняем  $M$  объектов мажоритарного класса, имеющих наименьшее среднее расстояние до  $k$  самых дальних объектов миноритарного класса



# NearMiss-3

- Сделаем «шорт-лист» объектов мажоритарного класса, наиболее близких к объектам миноритарного класса
- Сохраним  $M$  объектов мажоритарного класса из «шорт-листа» с наибольшим средним расстоянием до  $k$  ближайших объектов миноритарного класса



# Связи Томека

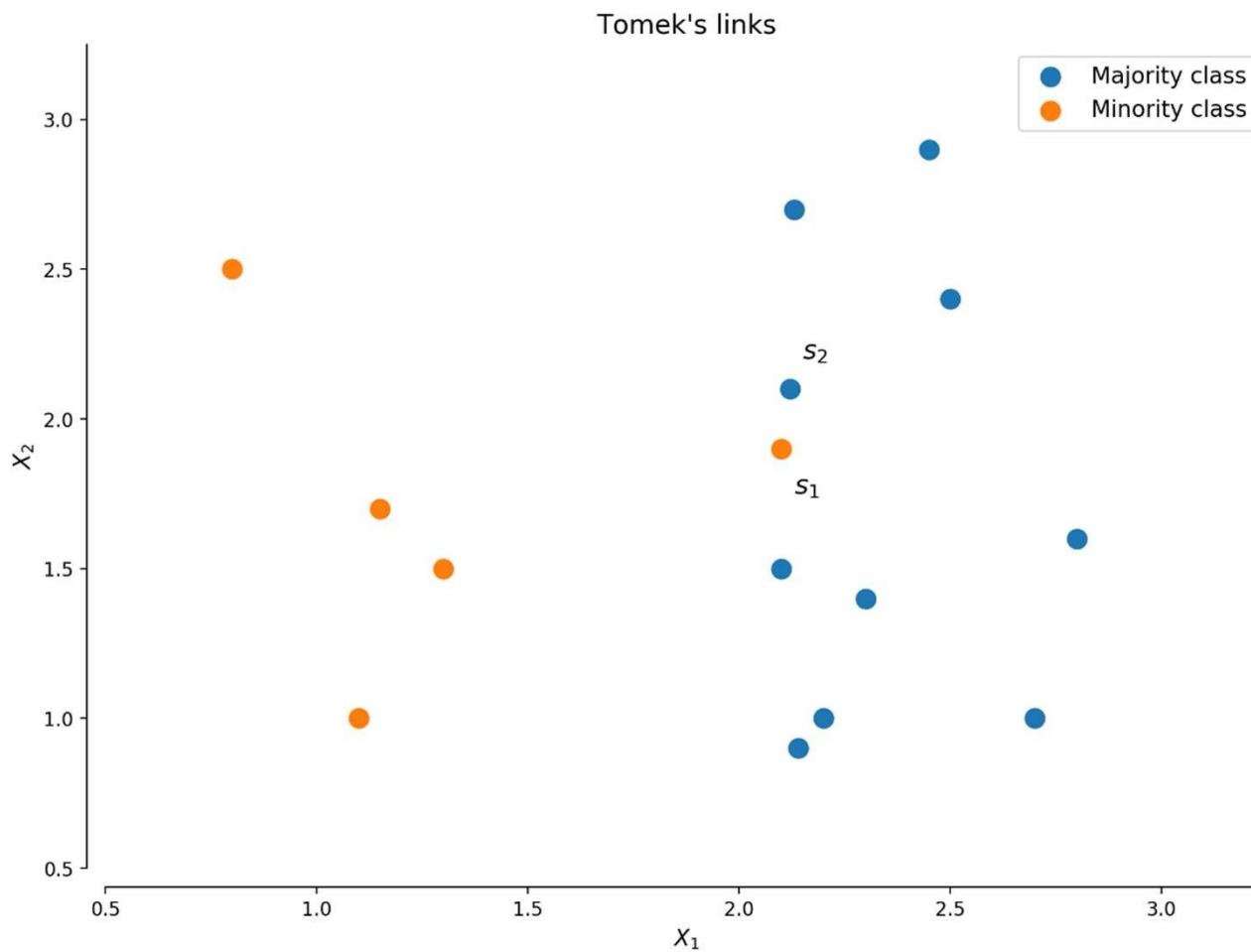
- Вместо сэмплирования напрямую, используем эвристики, которые позволяют нам очистить данные
- Между объектами  $x$  и  $y$  разных классов существует **связь Томека**, если они являются ближайшими соседями друг друга:

$$\forall z: \quad d(x, y) < d(x, z) \text{ and } d(x, y) < d(y, z)$$

- $z$  – другой объект
- $d(x, y)$  – расстояние между  $x$  и  $y$

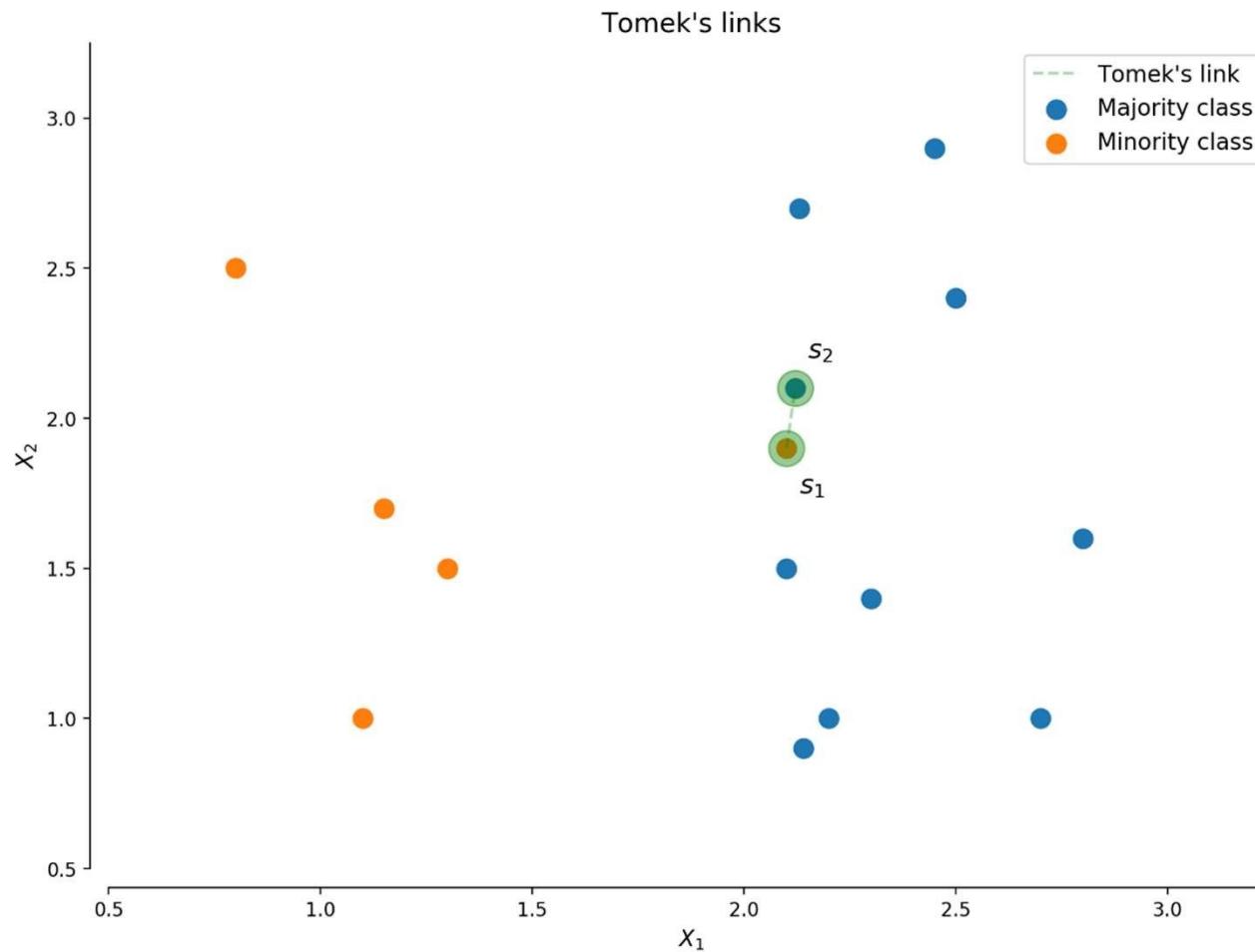
# Связи Томека

- Не хотим хранить избыточные объекты мажоритарного класса, которые находятся слишком близко к миноритарному классу



# Связи Томека

- Не хотим хранить избыточные объекты мажоритарного класса, которые находятся слишком близко к миноритарному классу
- Найдя связь Томека, мы можем либо удалить объект мажоритарного класса, либо оба объекта



# Oversampling

- **Oversampling** – это техника балансирования данных, при которой увеличивается число наблюдений миноритарного класса

# Random oversampling

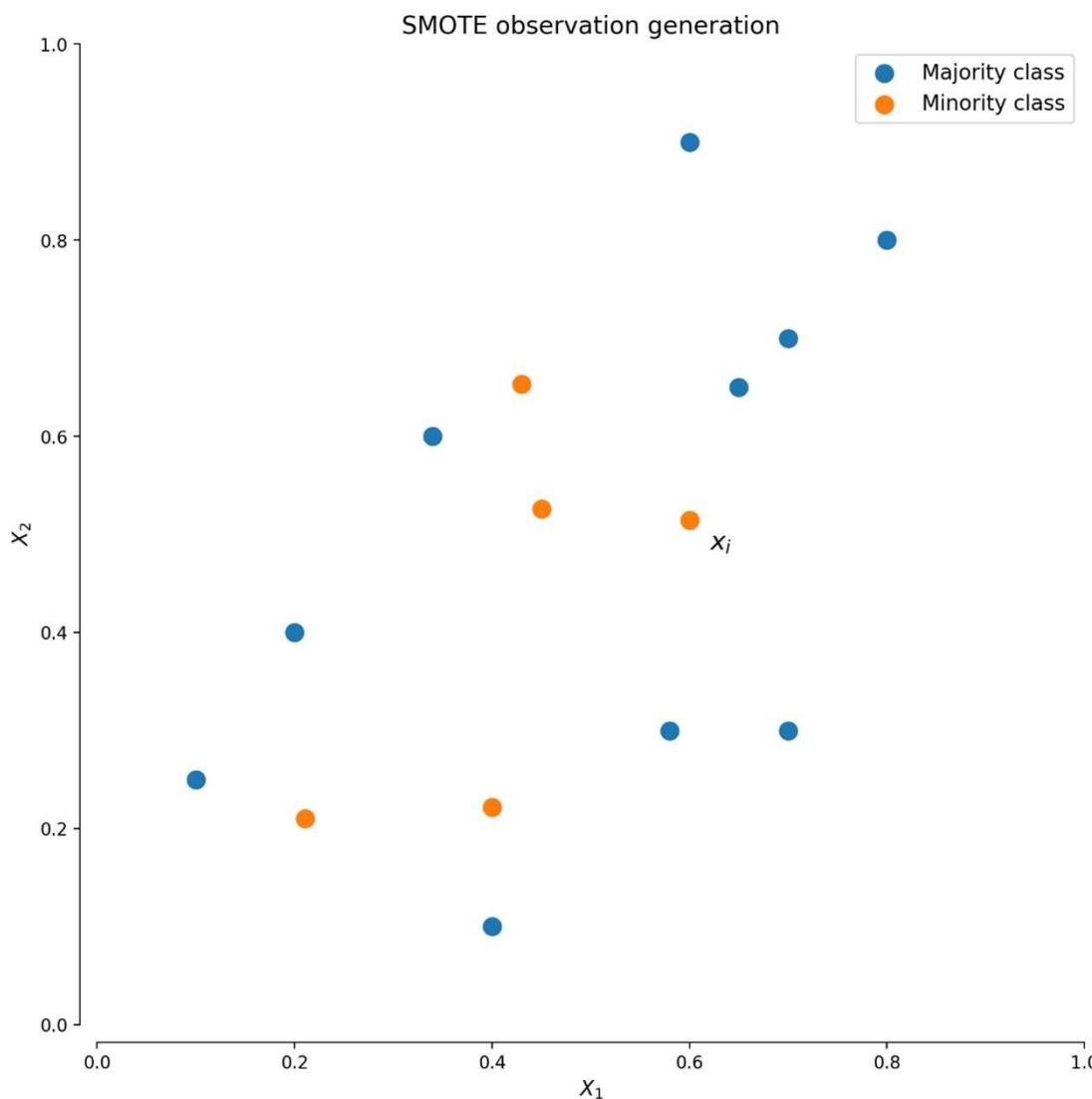
- Простейший метод: **random oversampling** (случайно клонируем объекты миноритарного класса)

# SMOTE

- **SMOTE: Synthetic Minority Over-sampling Technique**
- **Шаг 1.** Для каждого объекта миноритарного класса  $x_i$  найти  $k$  его ближайших соседей
- **Шаг 2.** Для каждого  $x_i$  выбрать среди его соседей  $M$  случайных:  
 $x_i^{(1)}, \dots, x_i^{(M)}$
- **Шаг 3.** Для каждой пары  $(x_i, x_i^{(j)})$  сгенерировать новый объект:  
$$x_i^{(j)'} = x_i + \lambda (x_i^{(j)} - x_i),$$
где  $\lambda \in [0, 1]$  – случайное число.

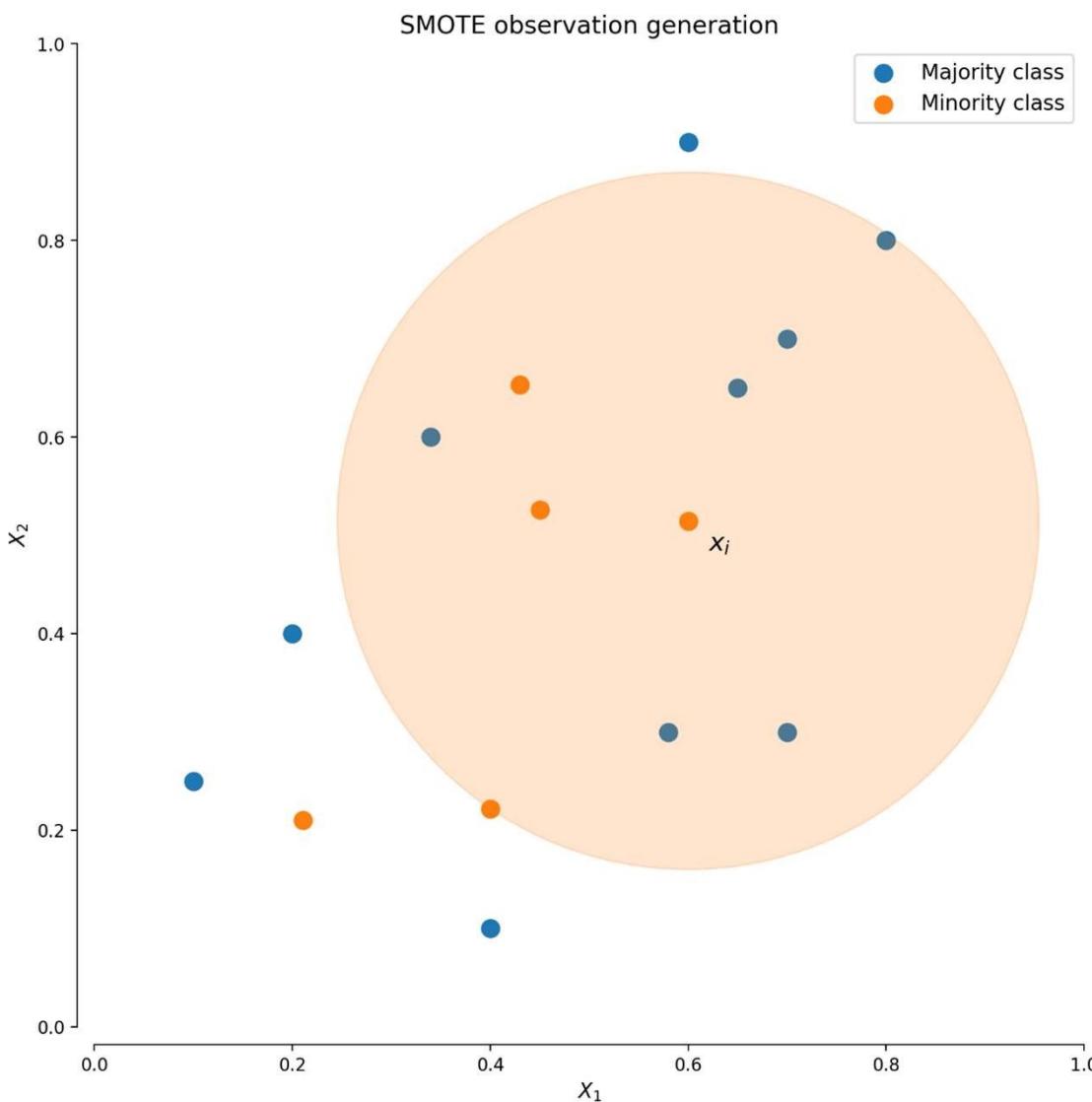
# SMOTE

- SMOTE: Synthetic Minority Over-sampling TErnique



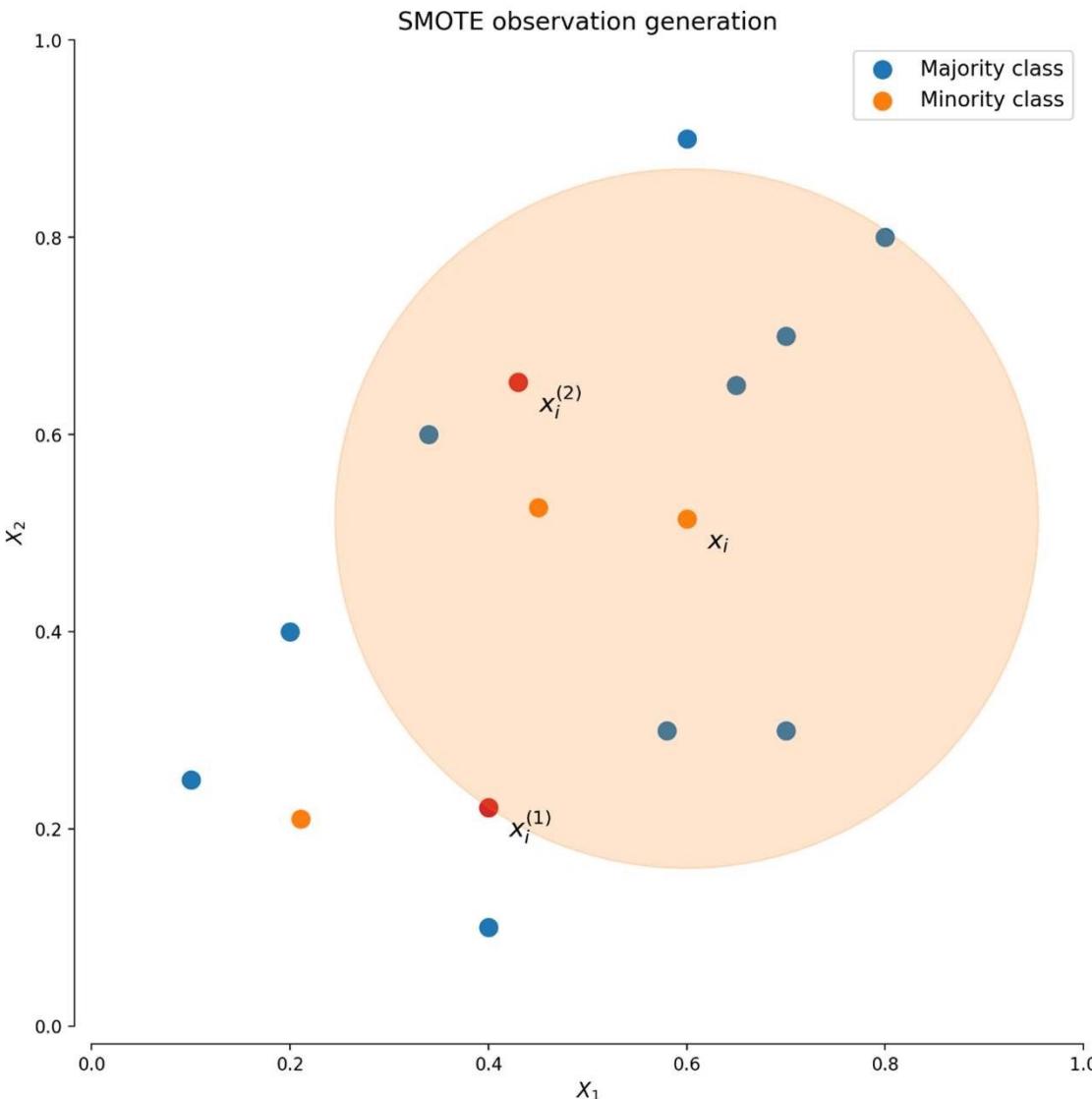
# SMOTE

- SMOTE: Synthetic Minority Over-sampling Technique
- Шаг 1. Ищем соседей



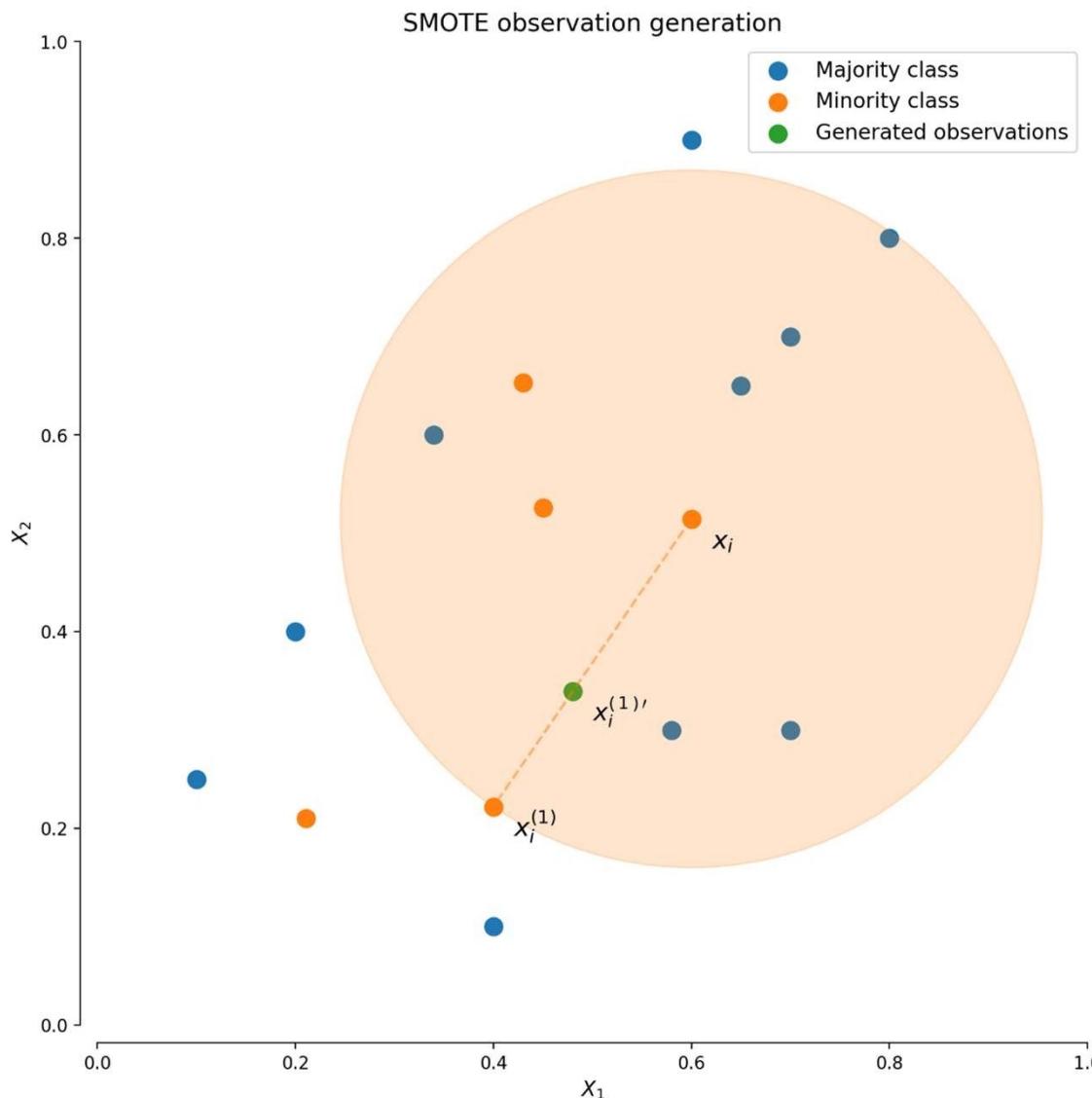
# SMOTE

- SMOTE: Synthetic Minority Over-sampling Technique
- Шаг 1. Ищем соседей
- Шаг 2. Выбираем случайных соседей



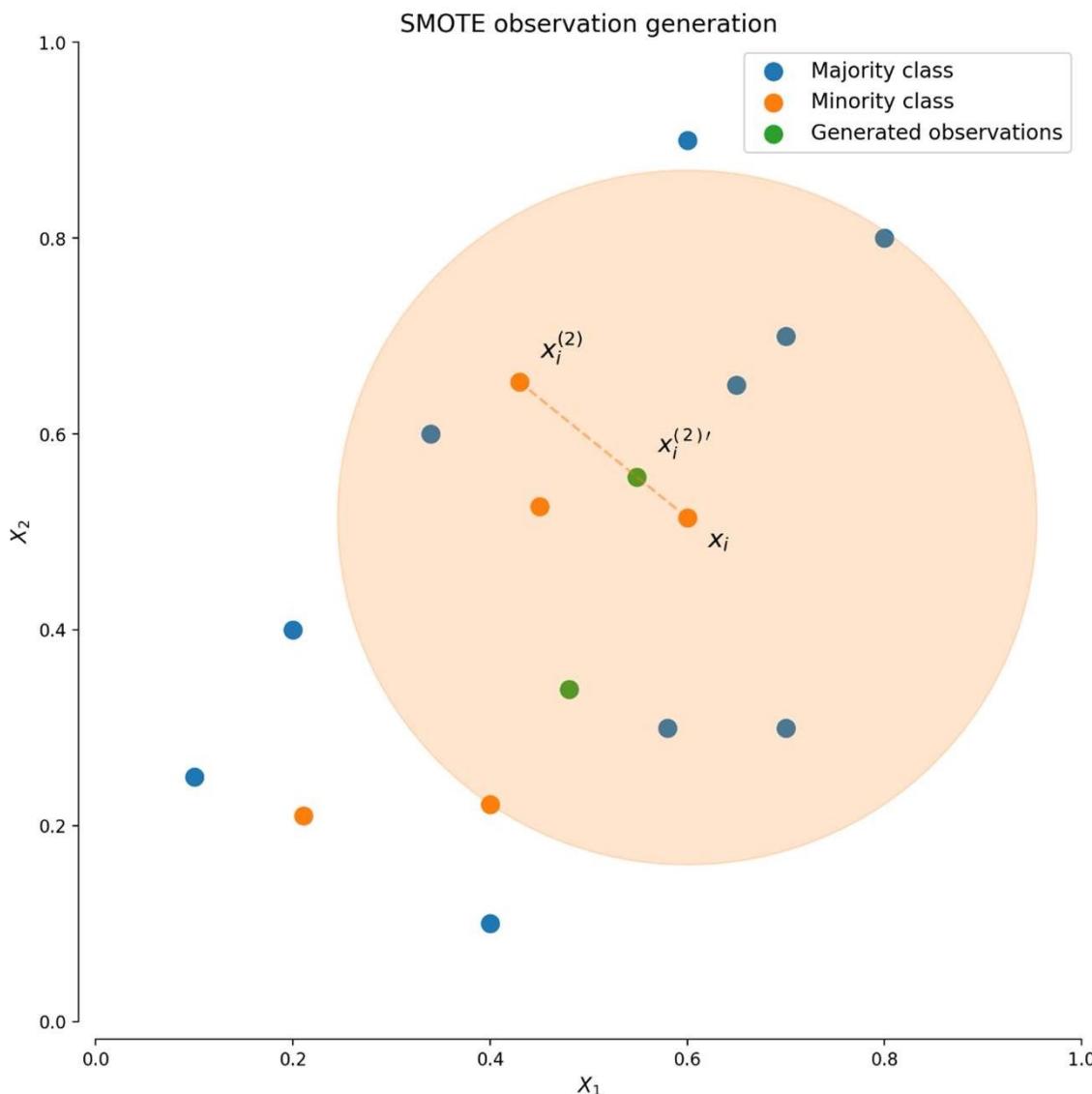
# SMOTE

- SMOTE: Synthetic Minority Over-sampling Technique
- Шаг 1. Ищем соседей
- Шаг 2. Выбираем случайных соседей
- Шаг 3. Генерируем объекты



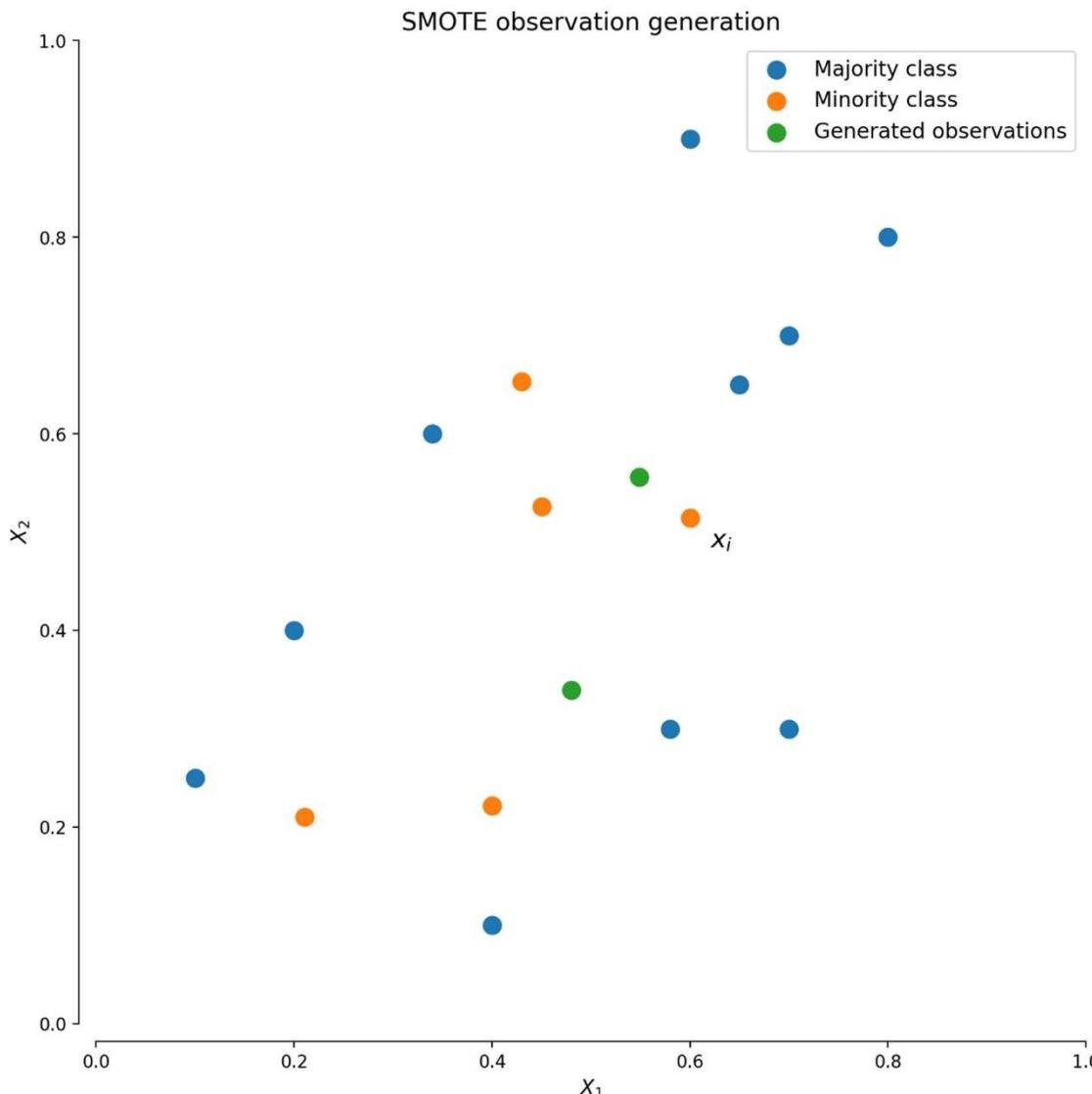
# SMOTE

- SMOTE: Synthetic Minority Over-sampling Technique
- Шаг 1. Ищем соседей
- Шаг 2. Выбираем случайных соседей
- Шаг 3. Генерируем объекты



# SMOTE

- SMOTE: Synthetic Minority Over-sampling Technique
- Шаг 1. Ищем соседей
- Шаг 2. Выбираем случайных соседей
- Шаг 3. Генерируем объекты



# ADASYN

- ADASYN: ADAptive SYNthetic Sampling Approach
- SMOTE:
  - Шаг 2. Для каждого объекта  $x_i$  сгенерировать  $M$  новых наблюдений
- ADASYN:
  - Шаг 2. Для каждого объекта  $x_i$  сгенерировать  $g_i$  новых наблюдений

# ADASYN

- $c_{\text{maj}}, c_{\text{min}}$  – число наблюдений мажоритарного/миноритарного классов
- $\beta \in [0, 1]$  – балансирующий параметр
- Общее число объектов, которые нужно сгенерировать:

$$G = (c_{\text{maj}} - c_{\text{min}}) \times \beta$$

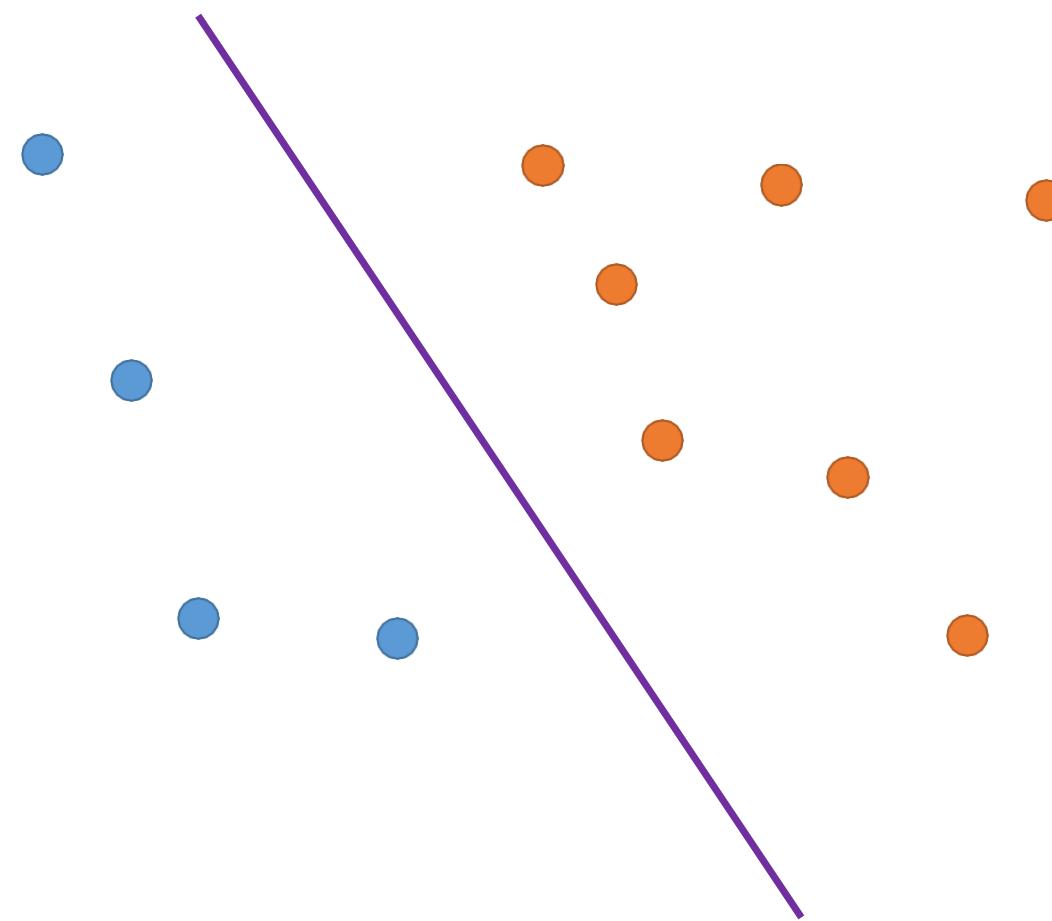
# ADASYN

- $c_{\text{maj}}, c_{\text{min}}$  – число наблюдений мажоритарного/миноритарного классов
- $\Delta_i$  – число соседей  $x_i$  мажоритарного класса
- $G$  – общее число объектов, которые нужно сгенерировать
- Число объектов, которые нужно сгенерировать для  $x_i$ :

$$g_i = \frac{\Delta_i}{\sum_{j=1}^{c_{\text{min}}} \Delta_j} \times G$$

# Borderline-SMOTE

- Для моделей классификации очень важно выучить границу между классами
- Объекты около границ крайне важны
- Следовательно, давайте генерировать объекты возле границ
- Как определить границы?



# Borderline-SMOTE

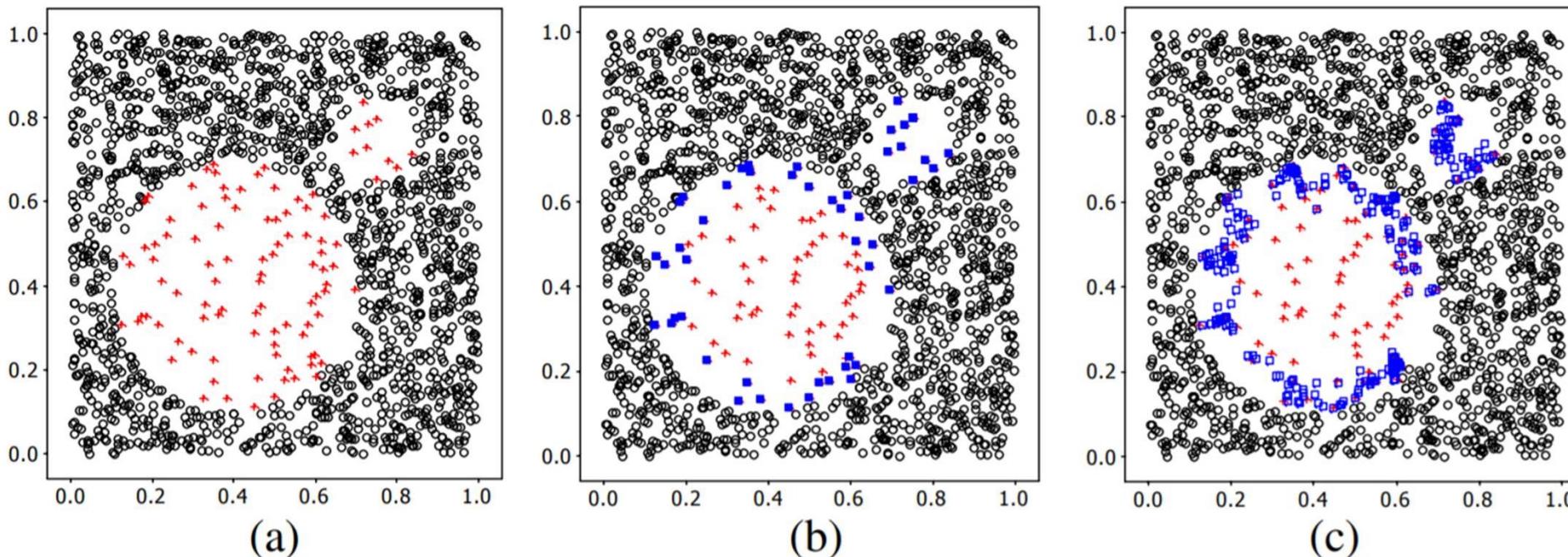
- Найти  $k$  ближайших соседей для каждого объекта  $x_i$  миноритарного класса
  - Затем для каждого  $x_i$  вычислить  $k' \in [0, k]$  – число соседей, принадлежащих к мажоритарному классу
1. Если  $k' = k$ , то  $x_i$  считаем шумом
  2. Если  $k' \in \left[0, \frac{k}{2}\right)$ , то  $x_i$  – «надежный» объект (далеко от границы)
  3. Если  $k' \in \left[\frac{k}{2}, k\right)$ , то  $x_i$  – объект «в опасности» (близко к границе)

# Borderline-SMOTE

$$x_i^{(j)'} = x_i + \lambda (x_i^{(j)} - x_i)$$

- **Borderline-SMOTE1:** используем для генерации объекты «в опасности» и их соседей миноритарного класса
- **Borderline-SMOTE2:** аналогично, но также использовать соседи мажоритарного класса, с  $\lambda \in [0, 0.5]$

# Borderline-SMOTE



**Fig. 1.** (a) The original distribution of Circle data set. (b) The borderline minority examples (*solid squares*). (c) The borderline synthetic minority examples (*hollow squares*).

# SMOTE: другие вариации

- **SVM SMOTE**

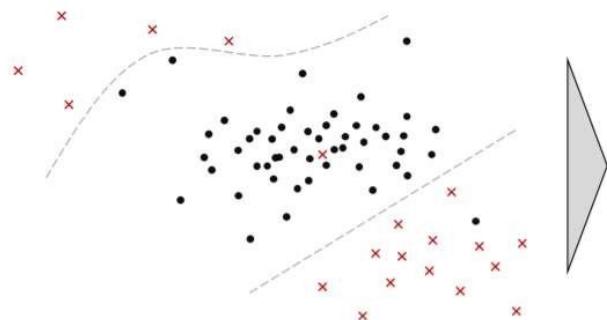
- определить границы классов с помощью SVM
- сгенерировать объекты на основе опорных векторов

- **K-Means SMOTE**

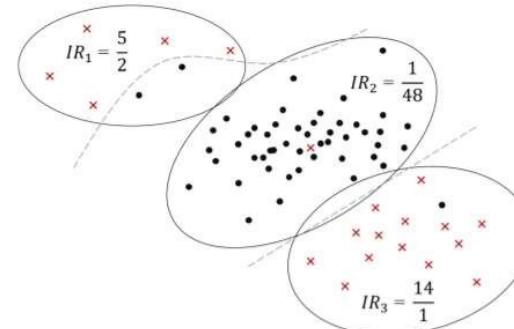
- кластеризовать объекты
- посчитать дисбаланс классов в кластерах
- сгенерировать объекты внутри кластеров с большим числом объектов миноритарного класса
- чтобы определить, сколько объектов сгенерировать, определить разреженность кластера

# K-Means SMOTE

Input data

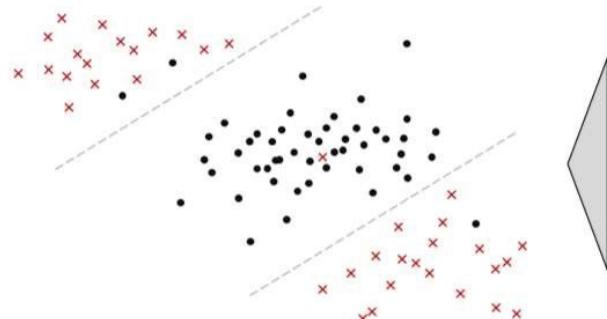


Find  $k = 3$  clusters and compute imbalance ratio (IR)

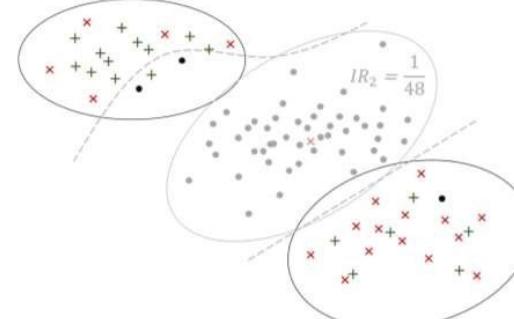


- ✖ Minority Sample
- Majority Sample
- Decision Boundary
- + Generated Sample

Oversampled data rectifies decision boundary



Use SMOTE to oversample clusters with  $IR > 1$ , generating more samples in sparse clusters



# Undersampling/oversampling

- В обоих методах модифицируется обучающая выборка – не валидация/тест!
- Разбиение на фолды для кросс-валидации нужно делать **до** oversampling
- Комбинация из undersampling и oversampling может неплохо сработать

# Резюме

- Можно балансировать данные множеством разных методов:
  - установка весов классов внутри алгоритмов
  - undersampling (random, NearMiss, связи Томека)
  - oversampling (random, методы на основе SMOTE)

# Определение аномалий

# Определение аномалий

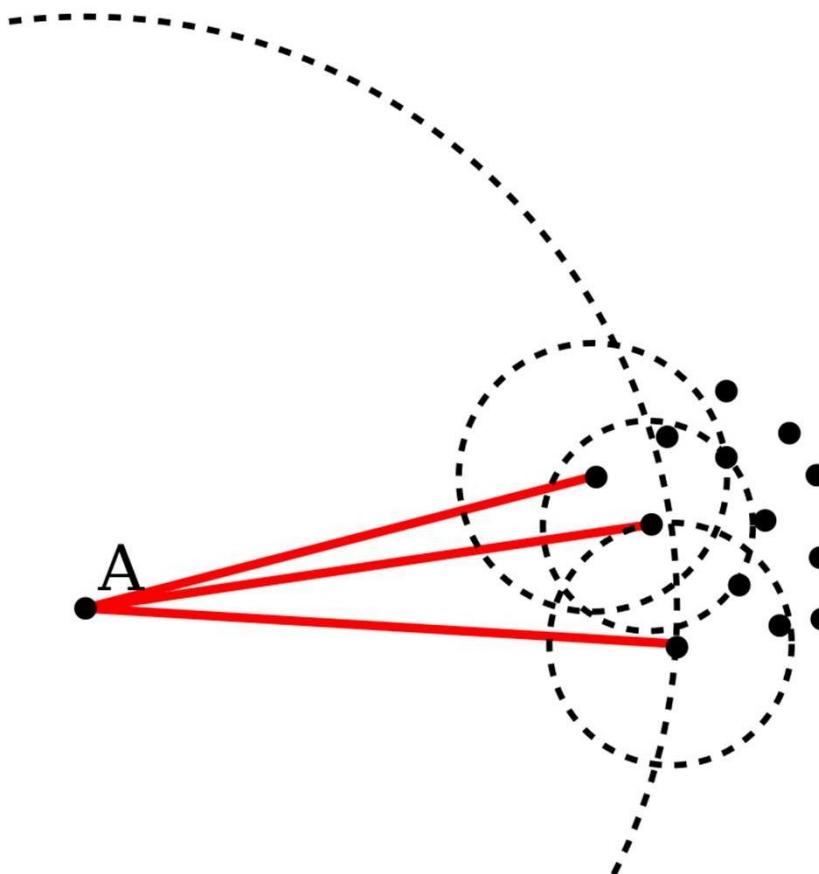
- Целенаправленное определение аномалий (выбросов, новизны) – объектов, которые не подходят под оригинальное распределение
- Очень большой дисбаланс в данных
- Может формулироваться как задача обучения без учителя
- Примеры: обнаружение вторжений в систему, предсказание сбоев и поломок

# Методы на основе kNN

- Используем алгоритм kNN для детекции объектов, которые лежат далеко от остальных
- **Метод 1:** как далеко находится объект от своего k-ого ближайшего соседа
- **Метод 2:** какое среднее расстояние от объекта до k ближайших соседей?

# Local Outlier Factor

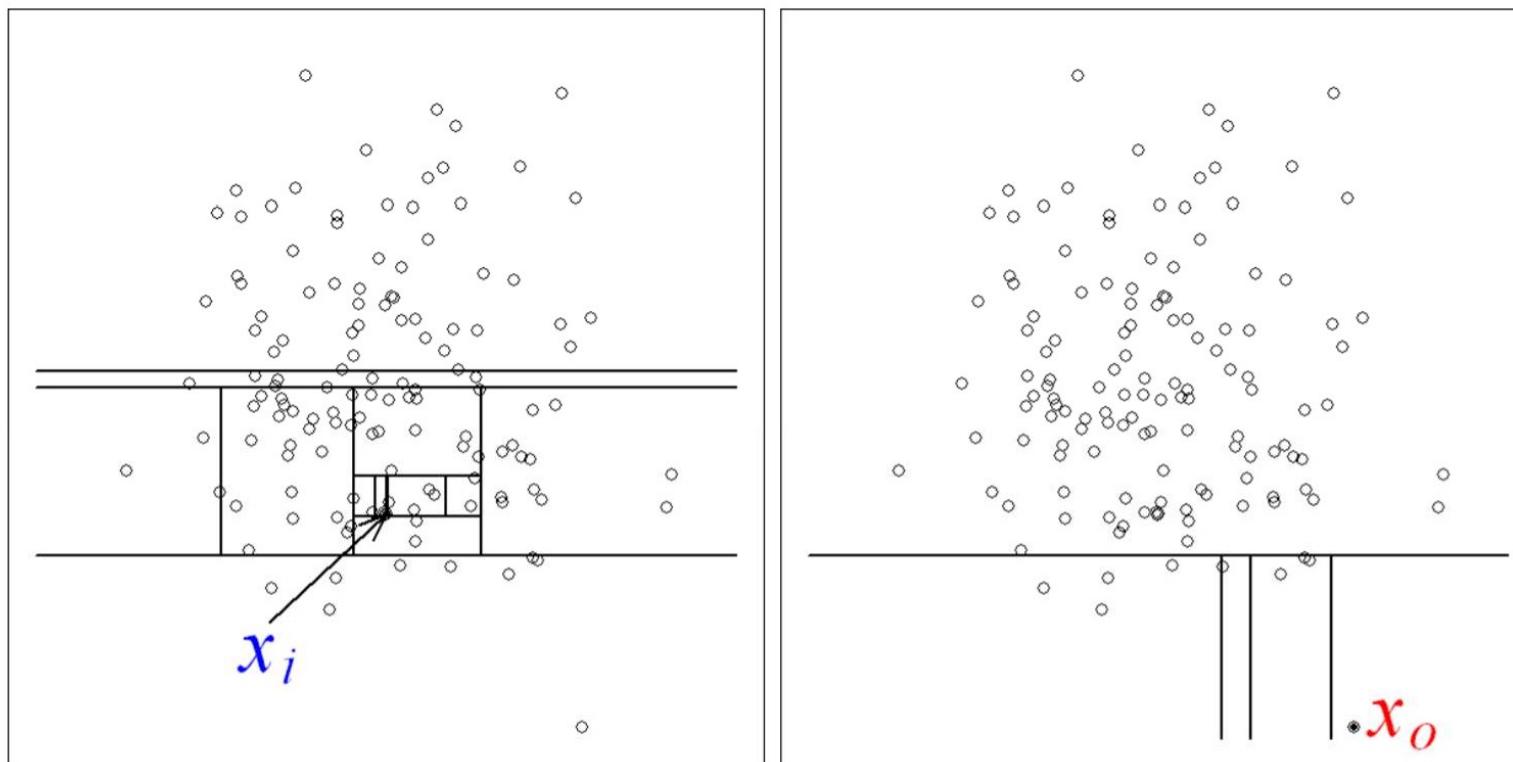
- LOF: Local Outlier Factor
- Наблюдение аномально, если его локальная плотность намного меньше локальной плотности его ближайших соседей



# Isolation Forest

- **Isolation Forest** «изолирует» наблюдения, делая случайные разбиения в решающих деревьях
- Идея: если наблюдение аномально, то чтобы его изолировать, нужно очень мало разбиений
- Построим лес и посчитаем оценку аномальности для каждого наблюдения

# Isolation Forest



(a) Isolating  $x_i$

(b) Isolating  $x_o$

# Резюме

- Определение аномалий – специфичная задача с дисбалансом данных
- Есть много методов для решения такой задачи
- kNN, Local Outlier Factor, Isolation Forest

# Спасибо за внимание!



Ildar Safilo

@Ildar\_Saf

[irsafilo@gmail.com](mailto:irsafilo@gmail.com)

<https://www.linkedin.com/in/isafilo/>