

# Машинное обучение

Лекция 13

Поиск аномалий

НИУ ВШЭ, 2021

# План

- Несбалансированные данные
- Оценка качества и построение качественных моделей на несбалансированных выборках
- Методы балансировки данных

Несбалансированные данные

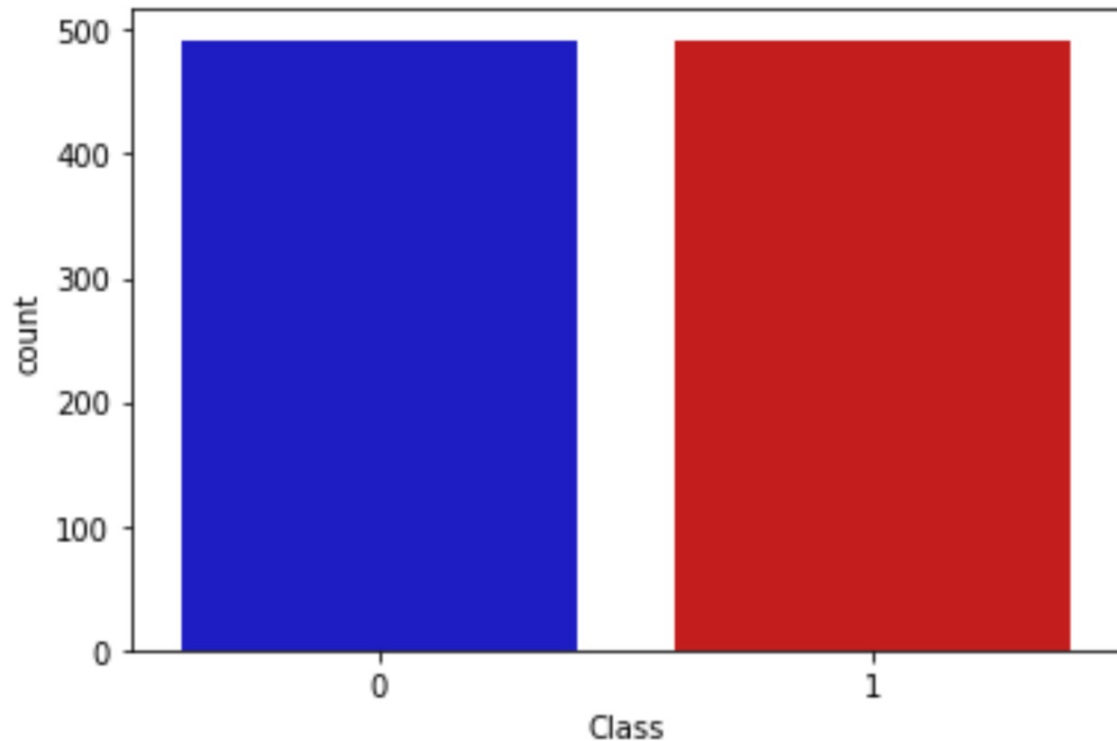
# Задача класифікації

- Посмотрим на баланс классов:

# Задача классификации

- Посмотрим на баланс классов:

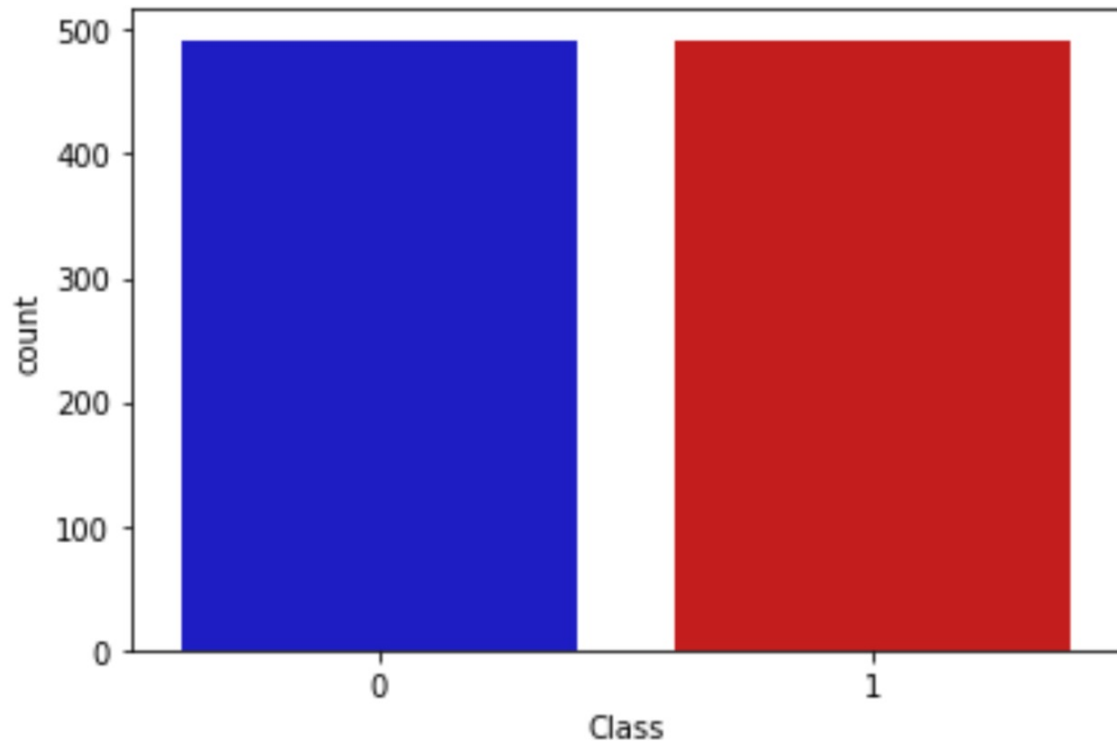
**Ожидание**



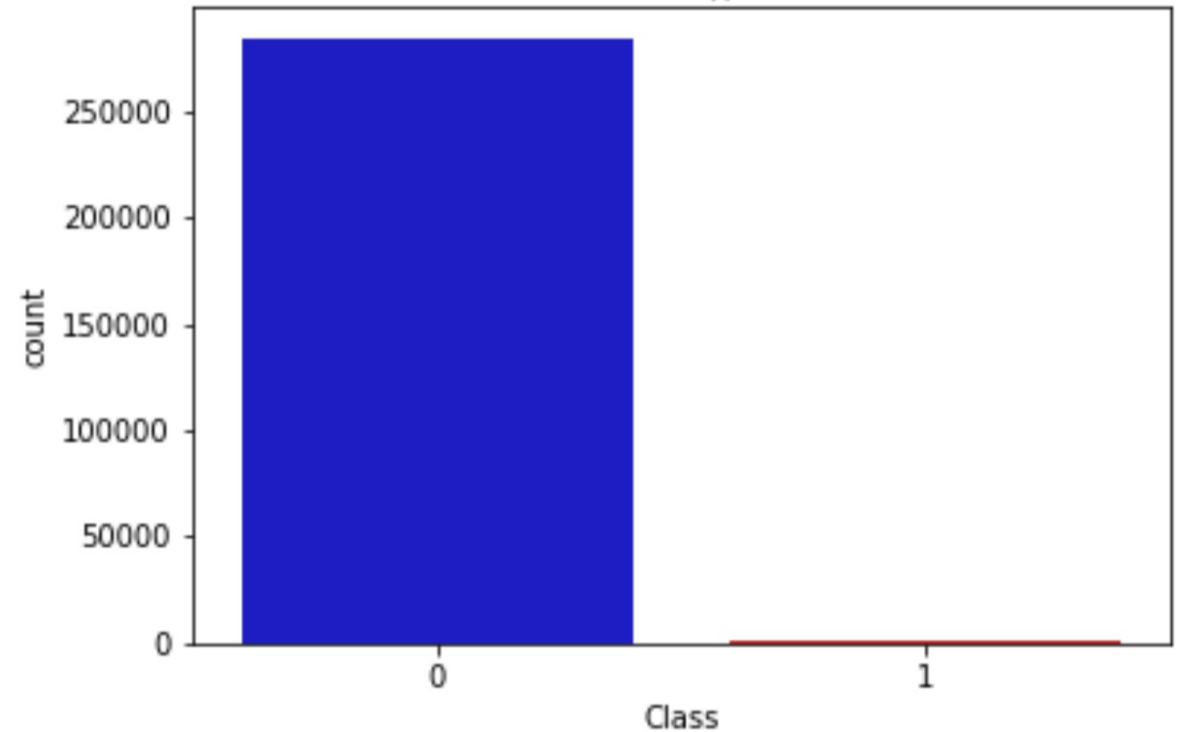
# Задача классификации

- Посмотрим на баланс классов:

**Ожидание**



**Реальность**



# Примеры

- [Детектирование «неискренних» вопросов](#)
- [Обнаружение мошеннических транзакций](#)
- [Классификация рентгеновских снимков](#)
- [Предсказание сейсмической активности](#)
- Выявление аномалий на производстве (predictive maintenance)
- Фильтрация спама
- Предсказание оттока клиентов
- Предсказание CTR (Click-Through Rate)

# Проблемы

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



# Проблемы

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

# 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

# Определение несбалансированности

- Данные **несбалансированы**, если число наблюдений одного класса сильно больше, чем число наблюдений других классов
- Что значит *сильно больше*?

# 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

# 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?

# Ассигасу: 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 ассигасу – да, в терминах бизнеса – нет

# Ассурасу: проблемы

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

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

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

# 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?

# 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

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

# F-мера: проблемы

- Какой случай лучше?

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

precision = 0.8

recall = 0.4

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

precision = 0.8

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

$$\begin{aligned} \text{TPR} &= 0.4 \\ \text{TNR} &\approx 0.9999 \end{aligned}$$

$$\text{Balanced accuracy} \approx 0.7$$

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

$$\begin{aligned} \text{TPR} &= 0.4 \\ \text{TNR} &\approx 0.91 \end{aligned}$$

$$\text{Balanced accuracy} \approx 0.65$$

# MCC

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

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

# MCC

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

$$\text{MCC} \approx 0.566$$

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

$$\text{MCC} \approx 0.362$$

# Метрики качества ранжирования

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

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

# Метрики качества ранжирования

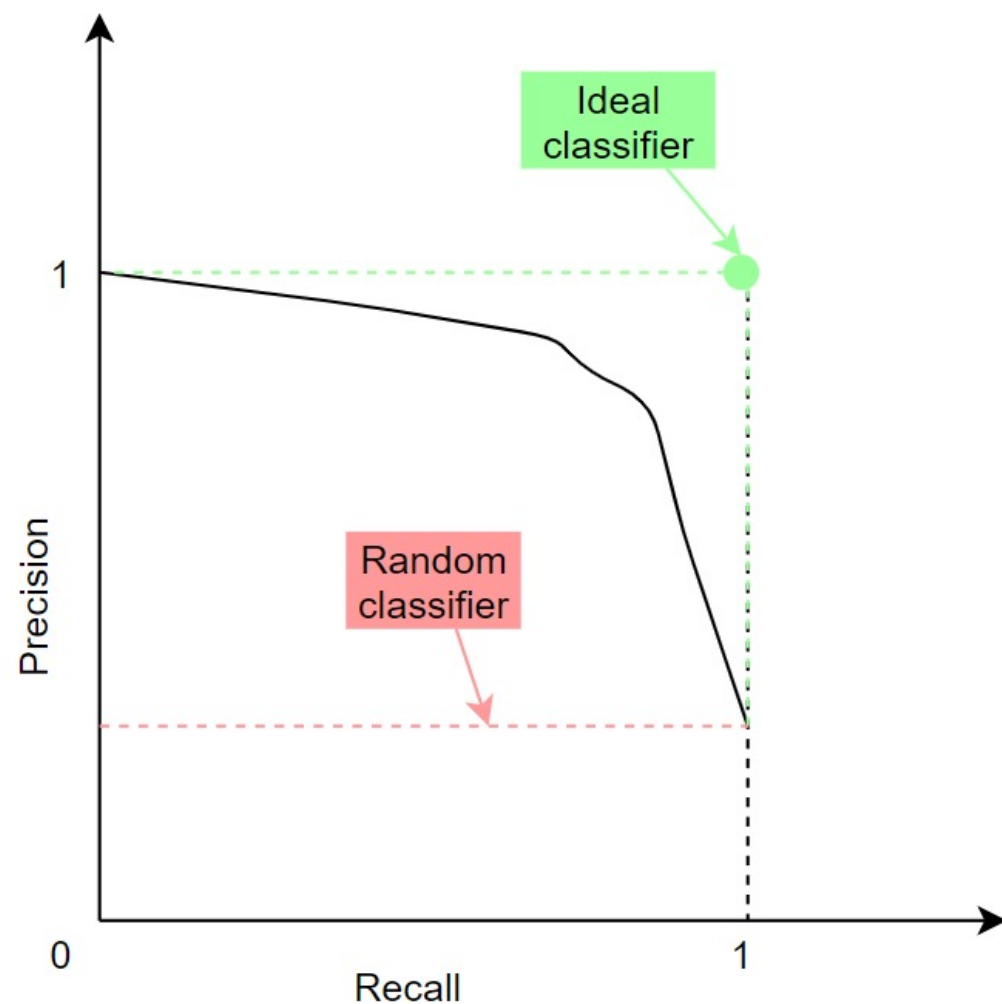
- Значения точности и полноты зависят от порога  $t$

$y$	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





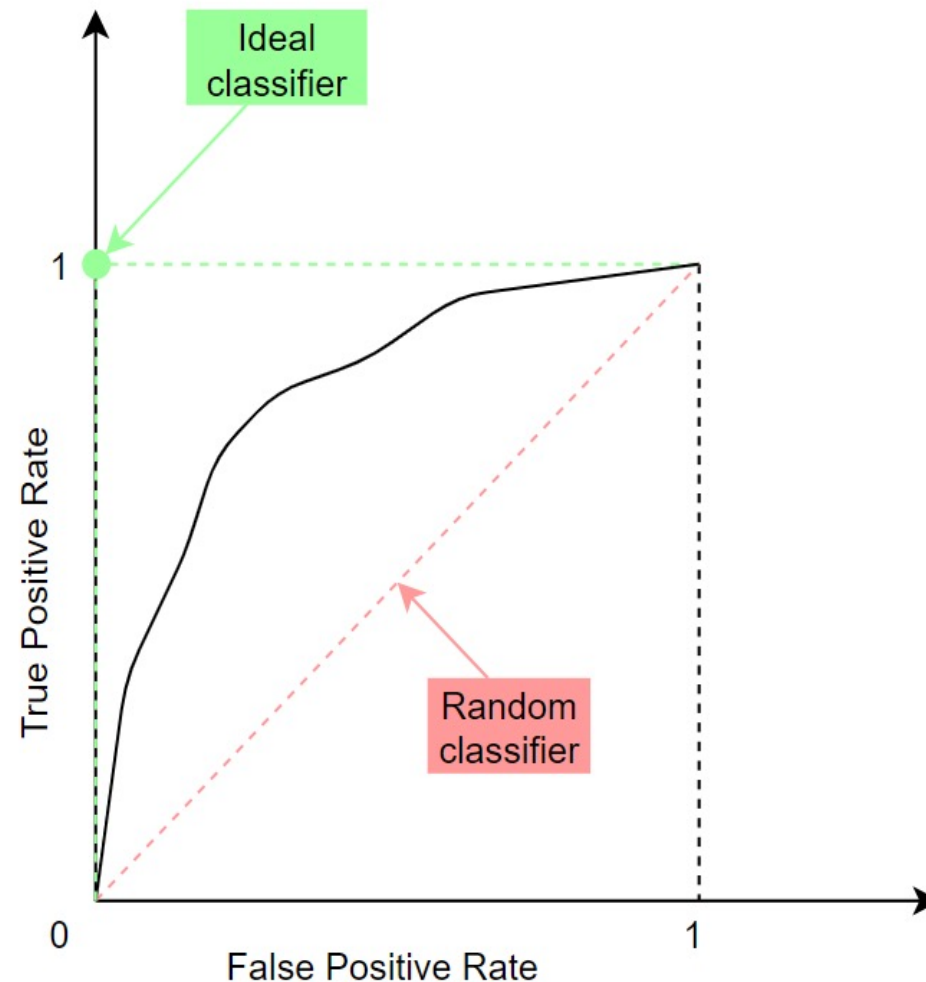
# ROC-кривая и AUC-ROC

- 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

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

# Веса классов

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

- Функция потерь для логистической регрессии

# 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

# 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`)



# 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:  $-\log(1 - z)$

# 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)

# Random undersampling

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

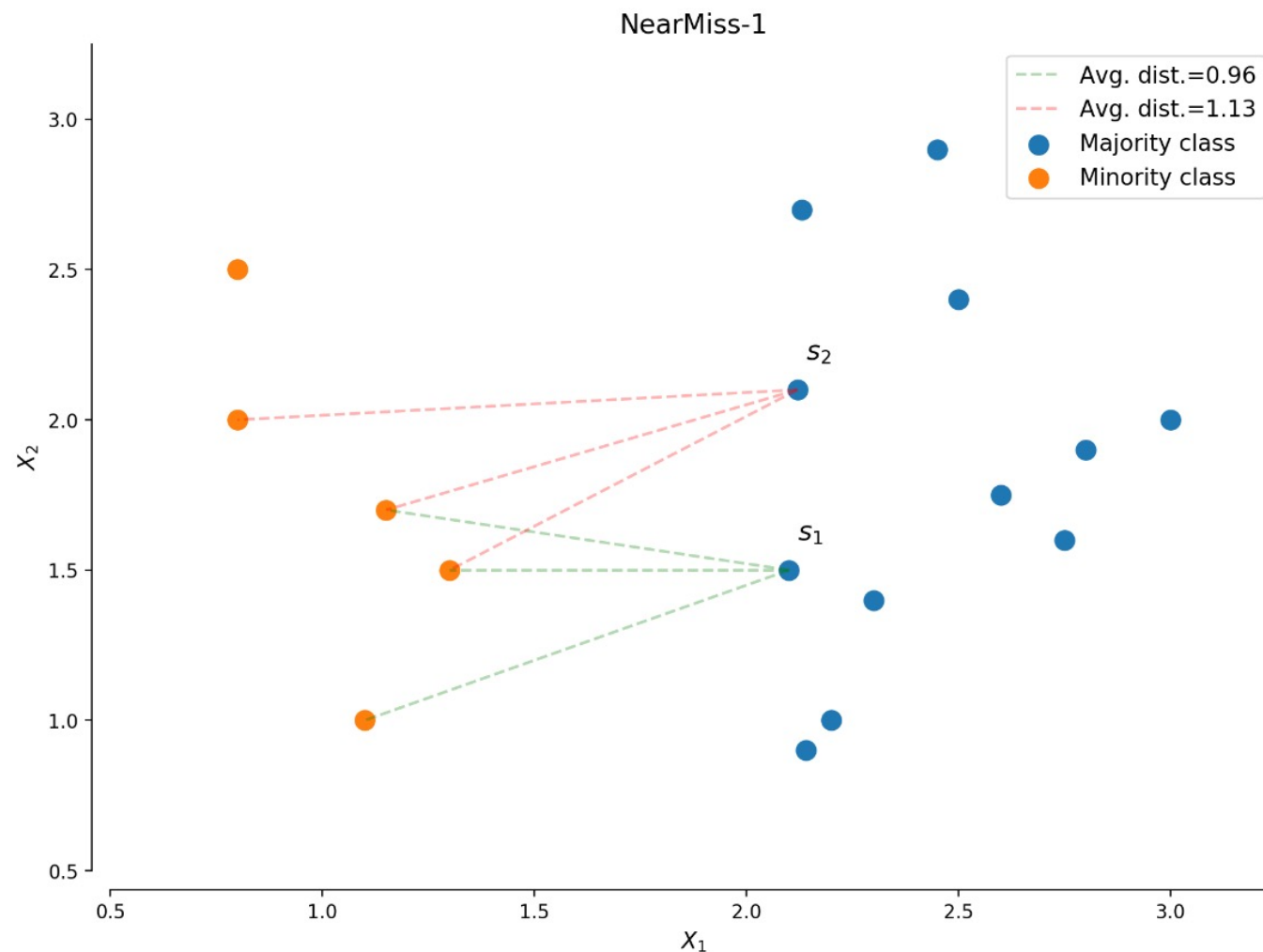
# NearMiss

- 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



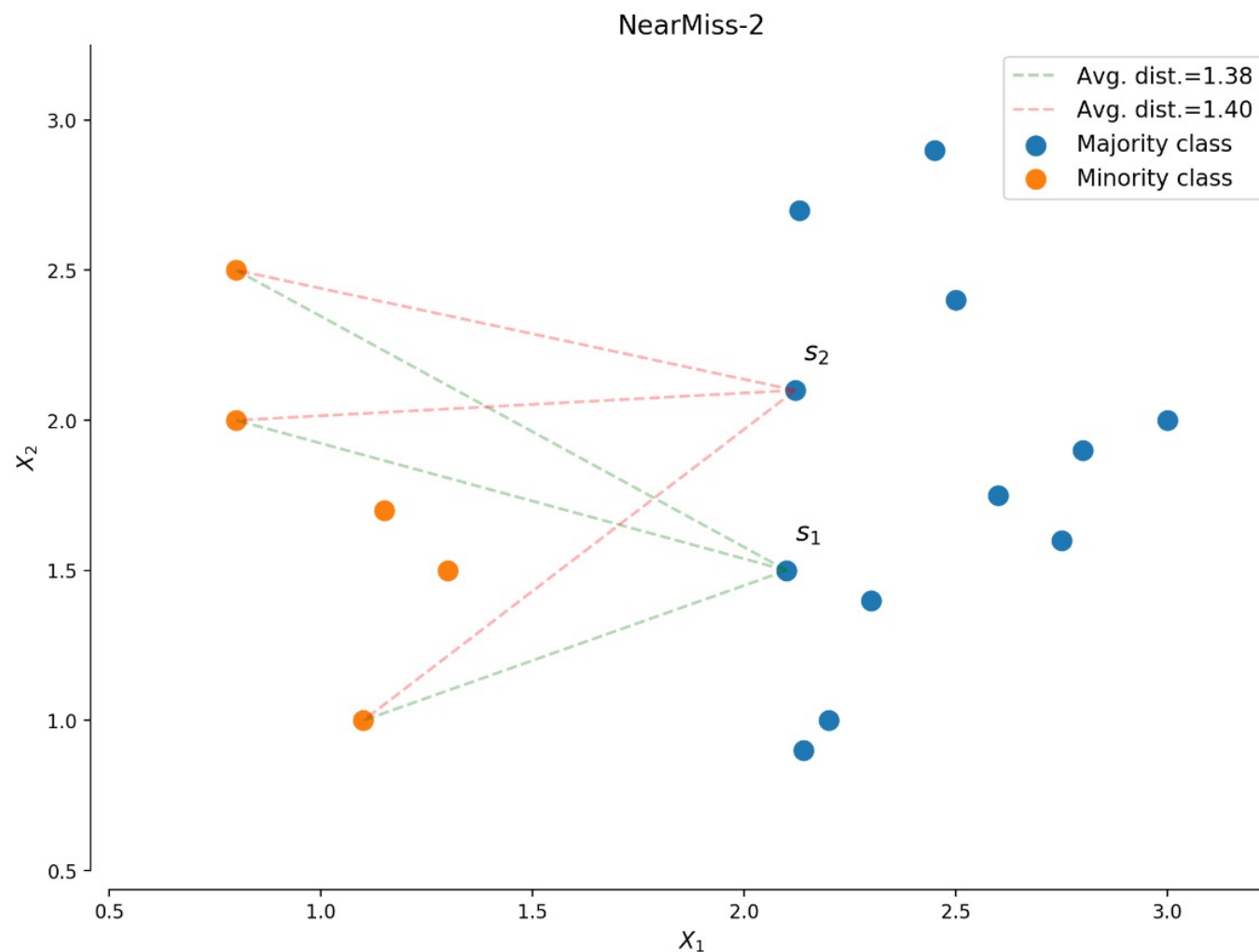
# NearMiss-1

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



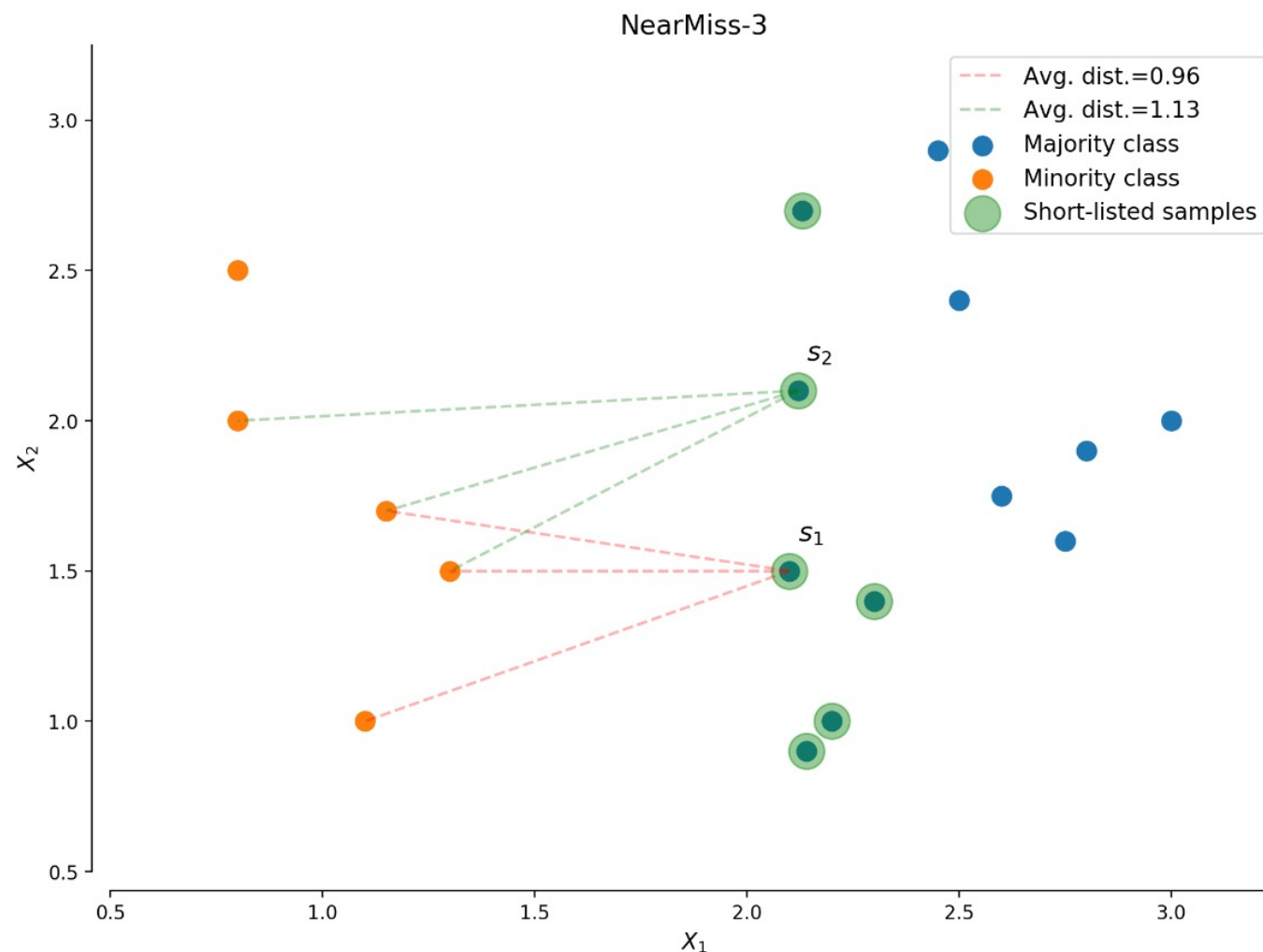
# NearMiss-2

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



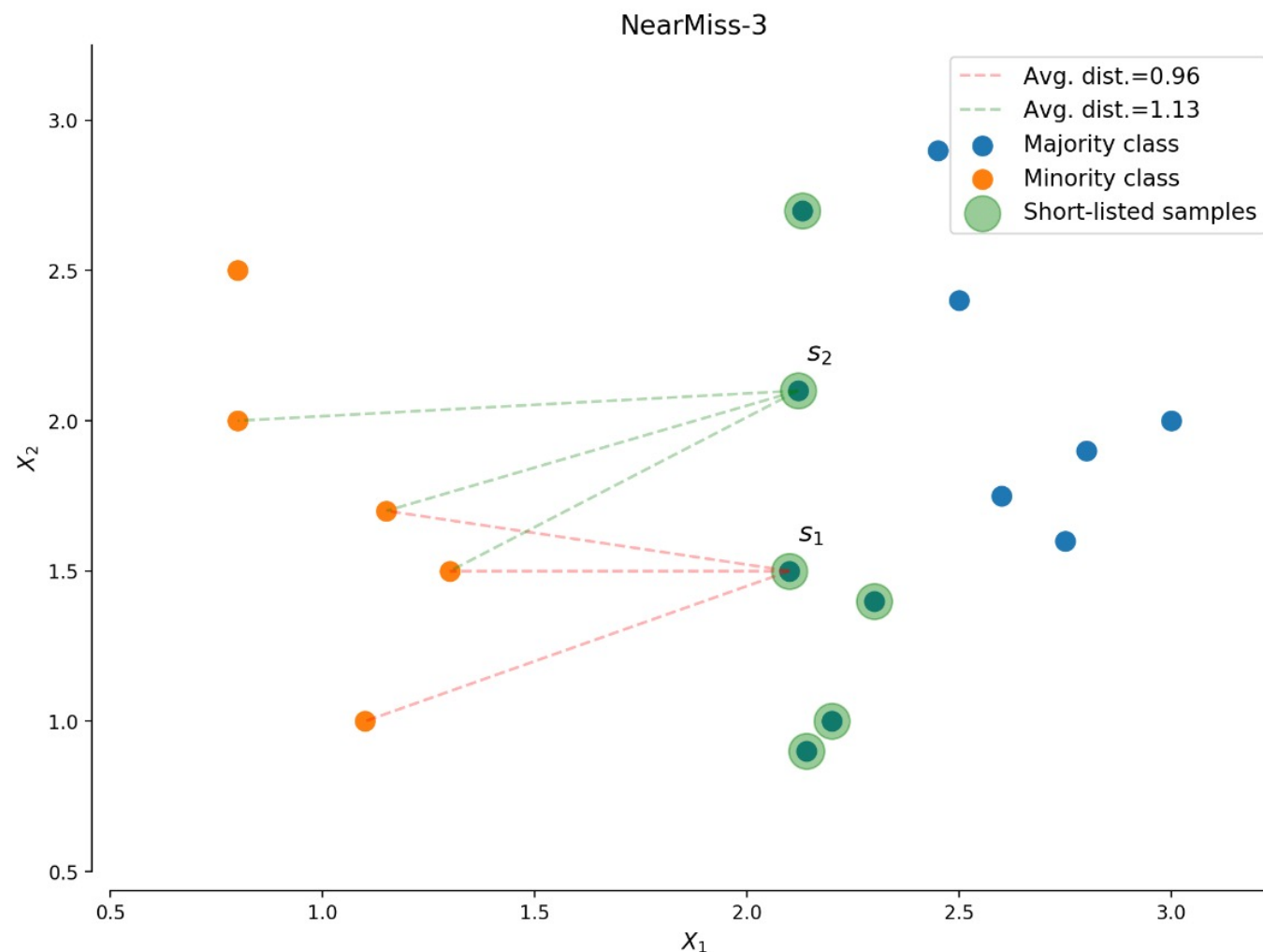
# NearMiss-3

- 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



# 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

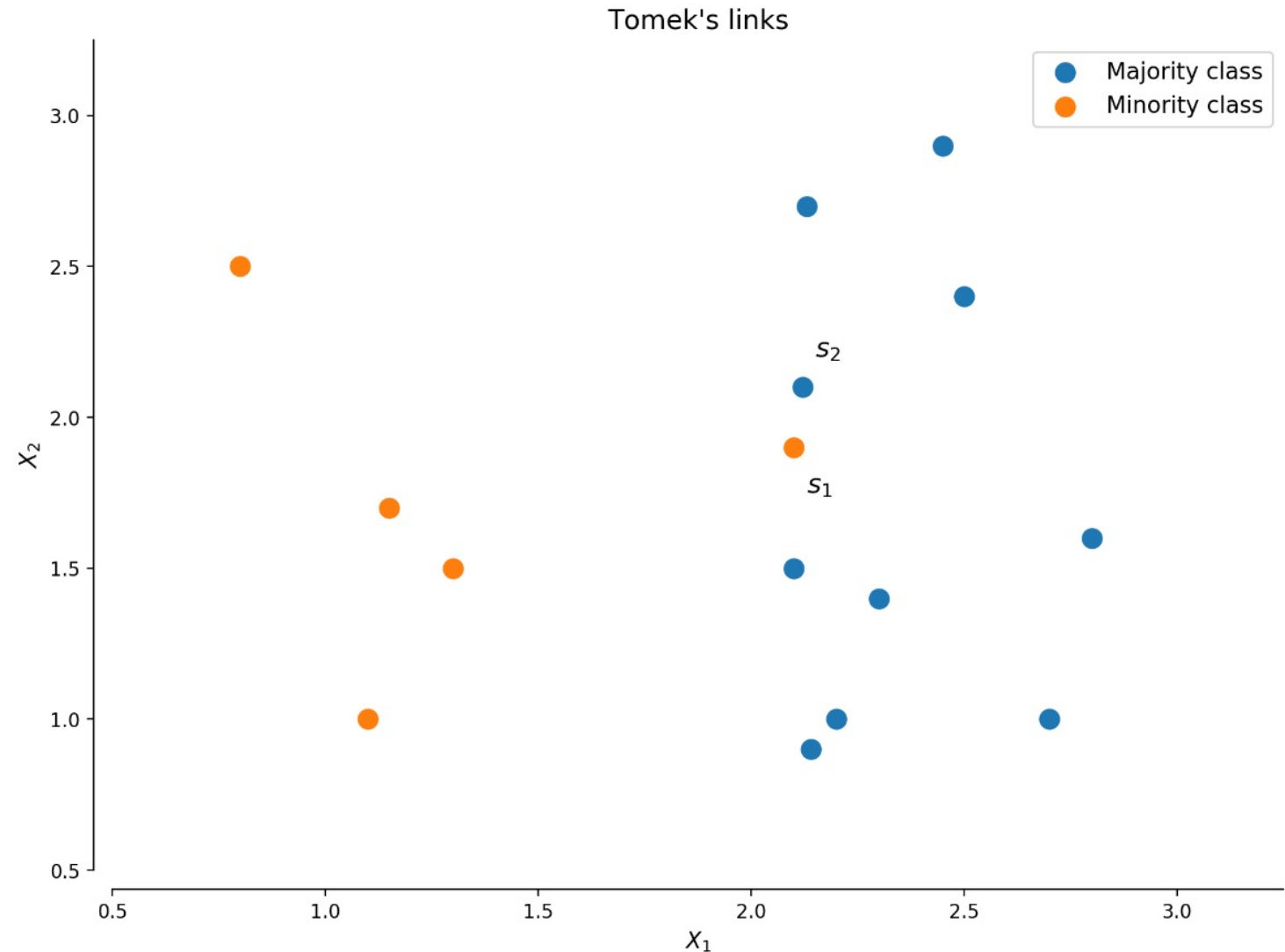
- Вместо сэмплирования напрямую, используем эвристики, которые позволят нам очистить данные
- Между объектами  $x$  и  $y$  разных классов существует **связь Томека**, если они являются ближайшими соседями друг друга:

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

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

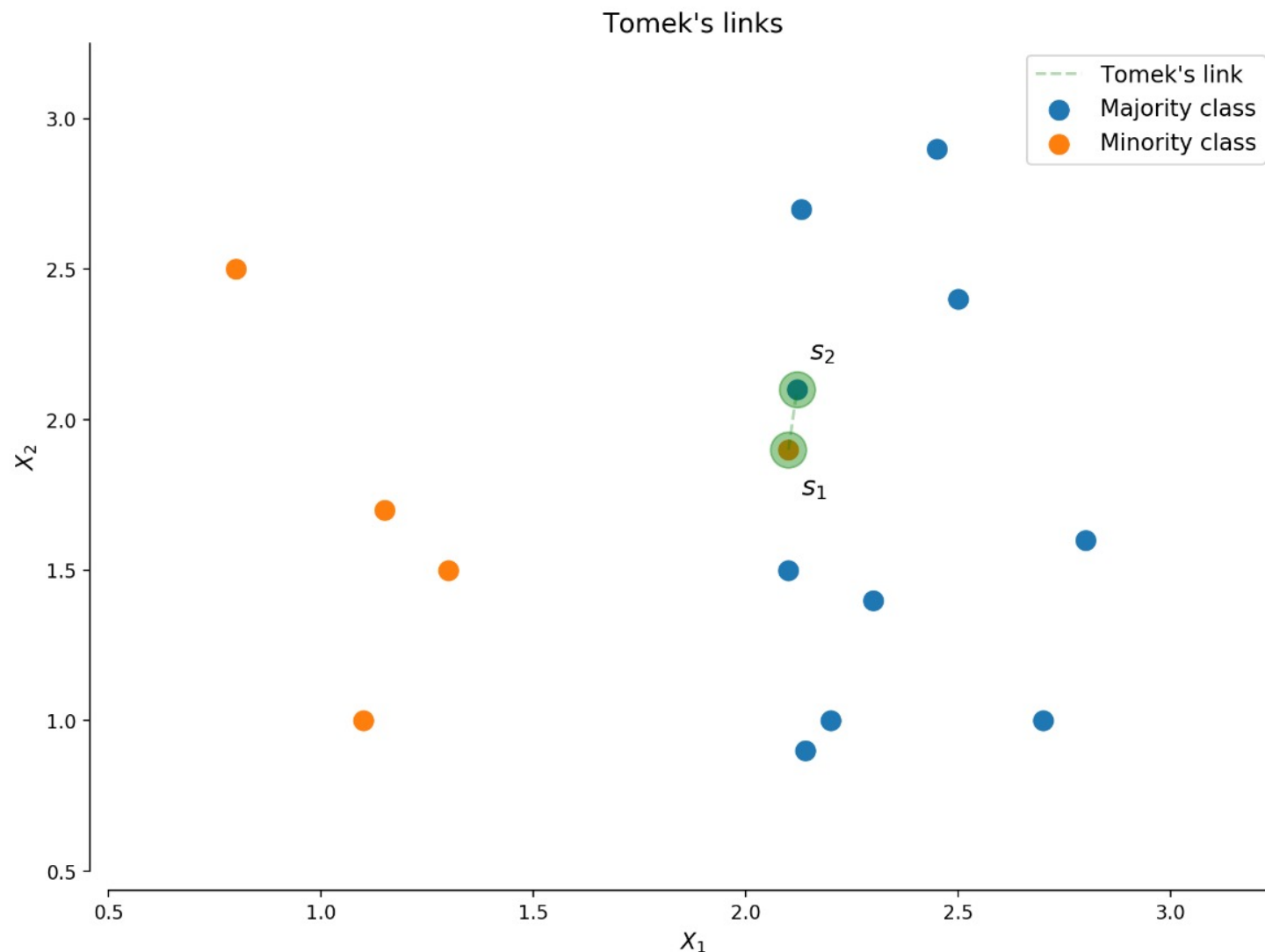
# Связи Томека

- 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

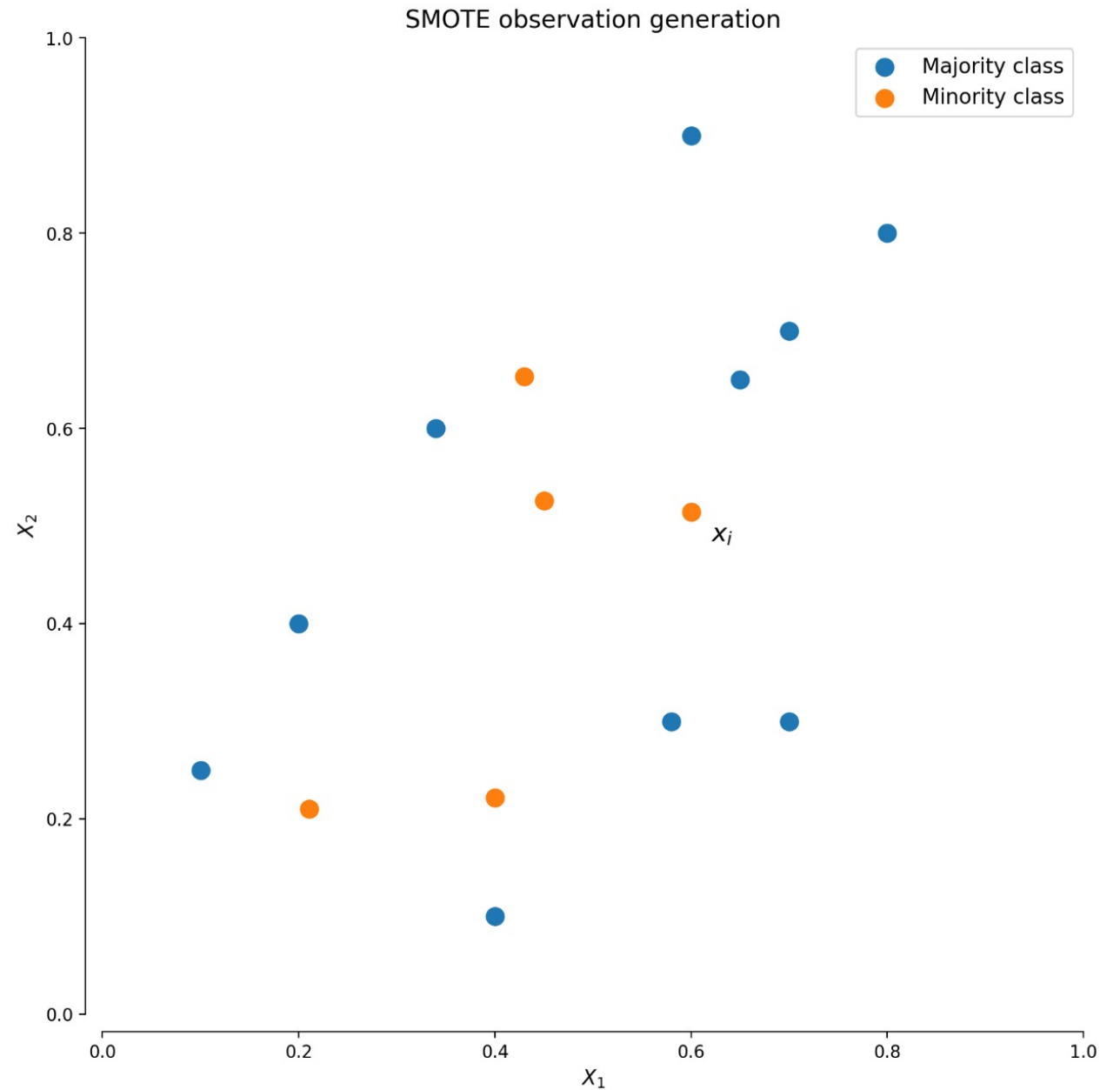
- Простейший метод: **random oversampling** (случайно клонируем объекты миноритарного класса)

# 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.

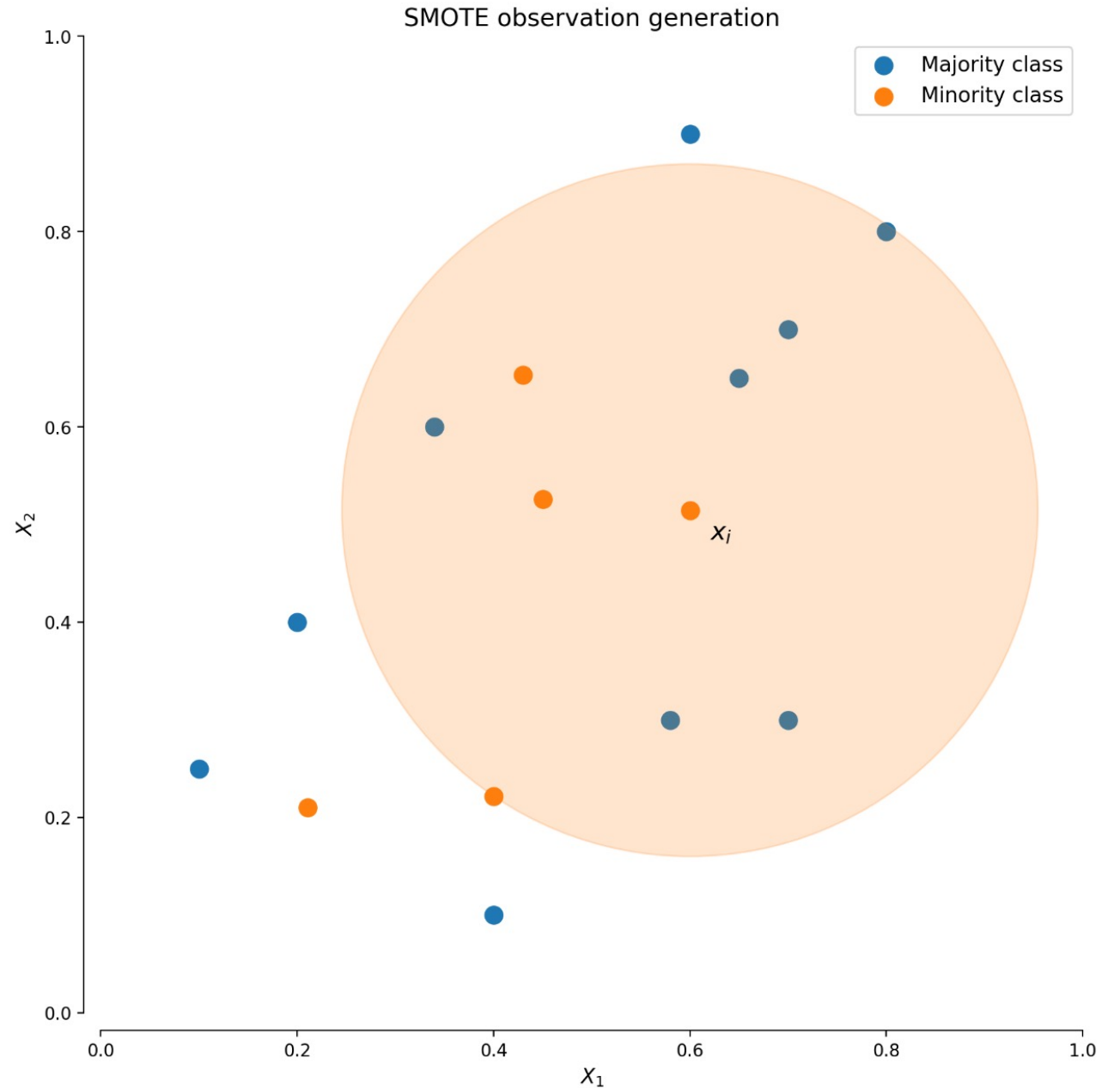
# SMOTE

- SMOTE: Synthetic Minority Over-sampling TEchnique



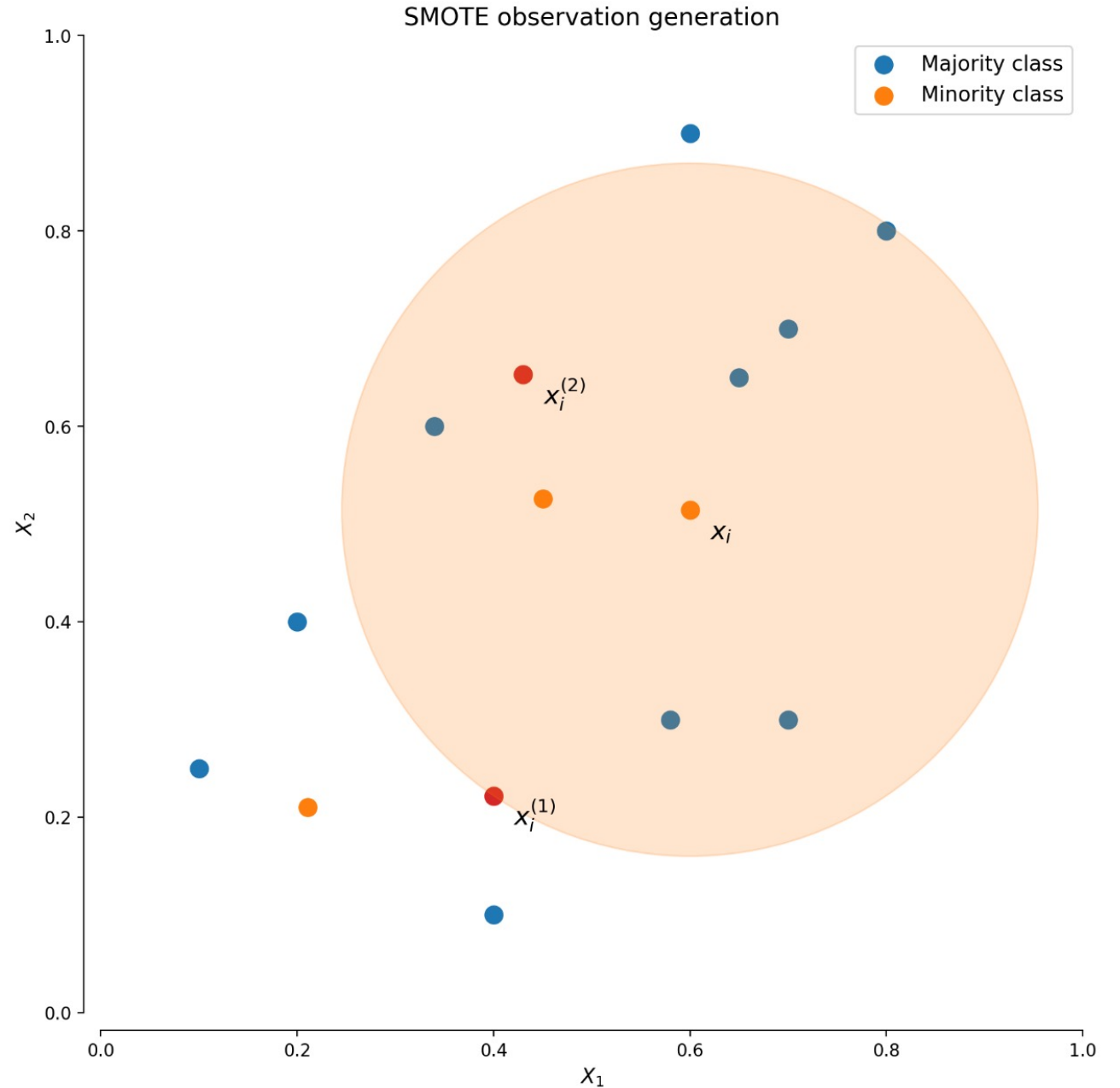
# SMOTE

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



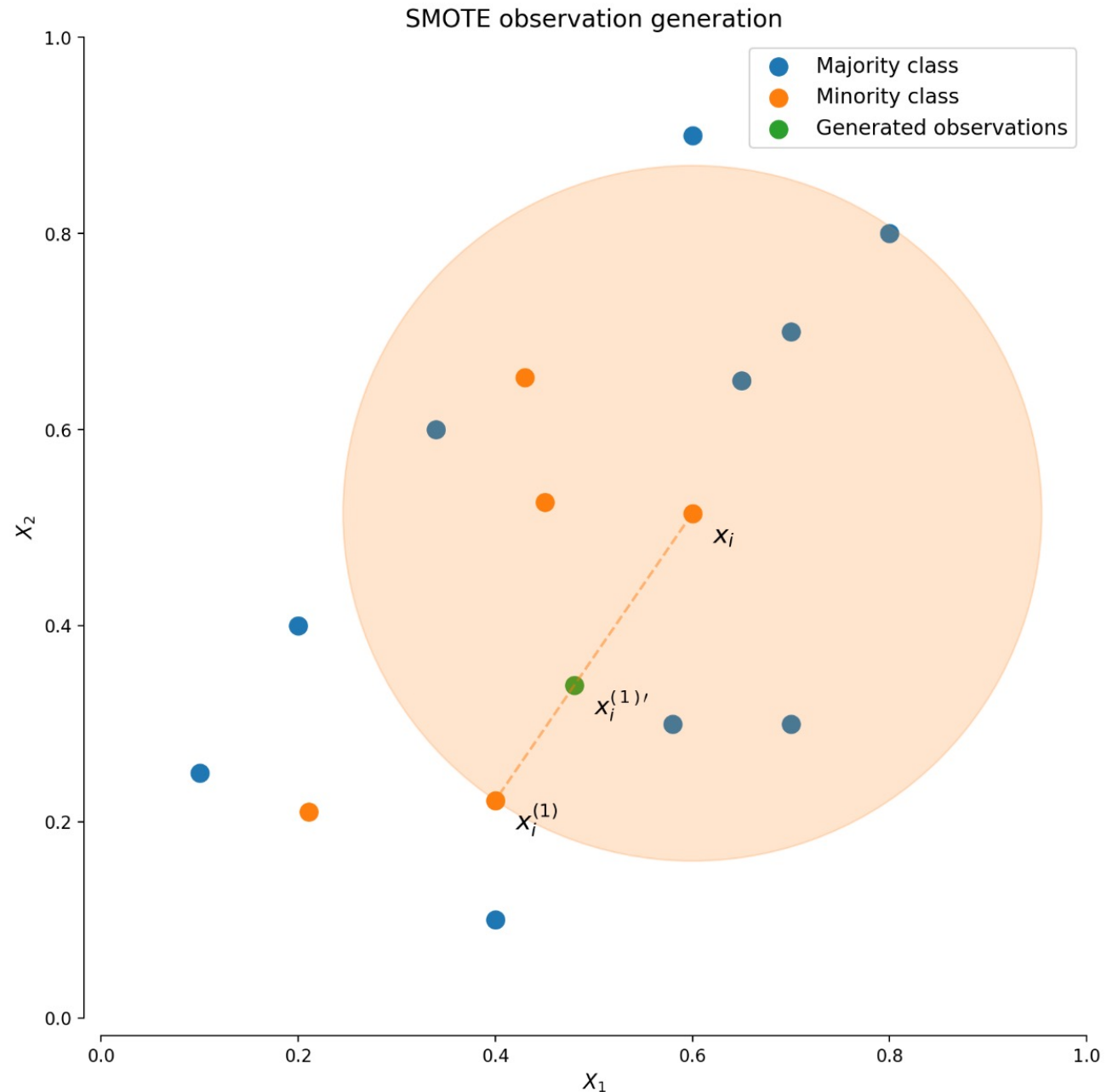
# SMOTE

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



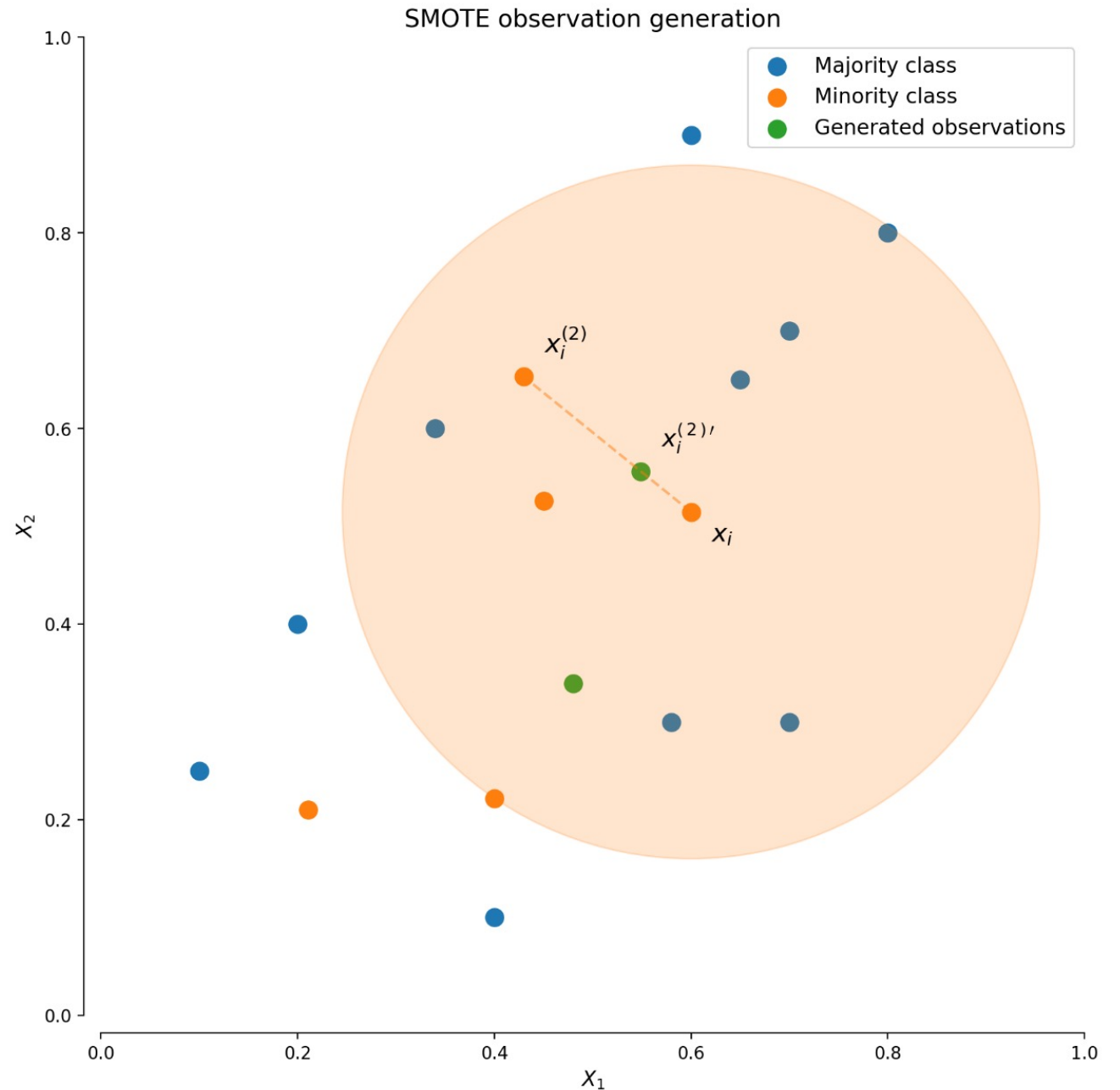
# SMOTE

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



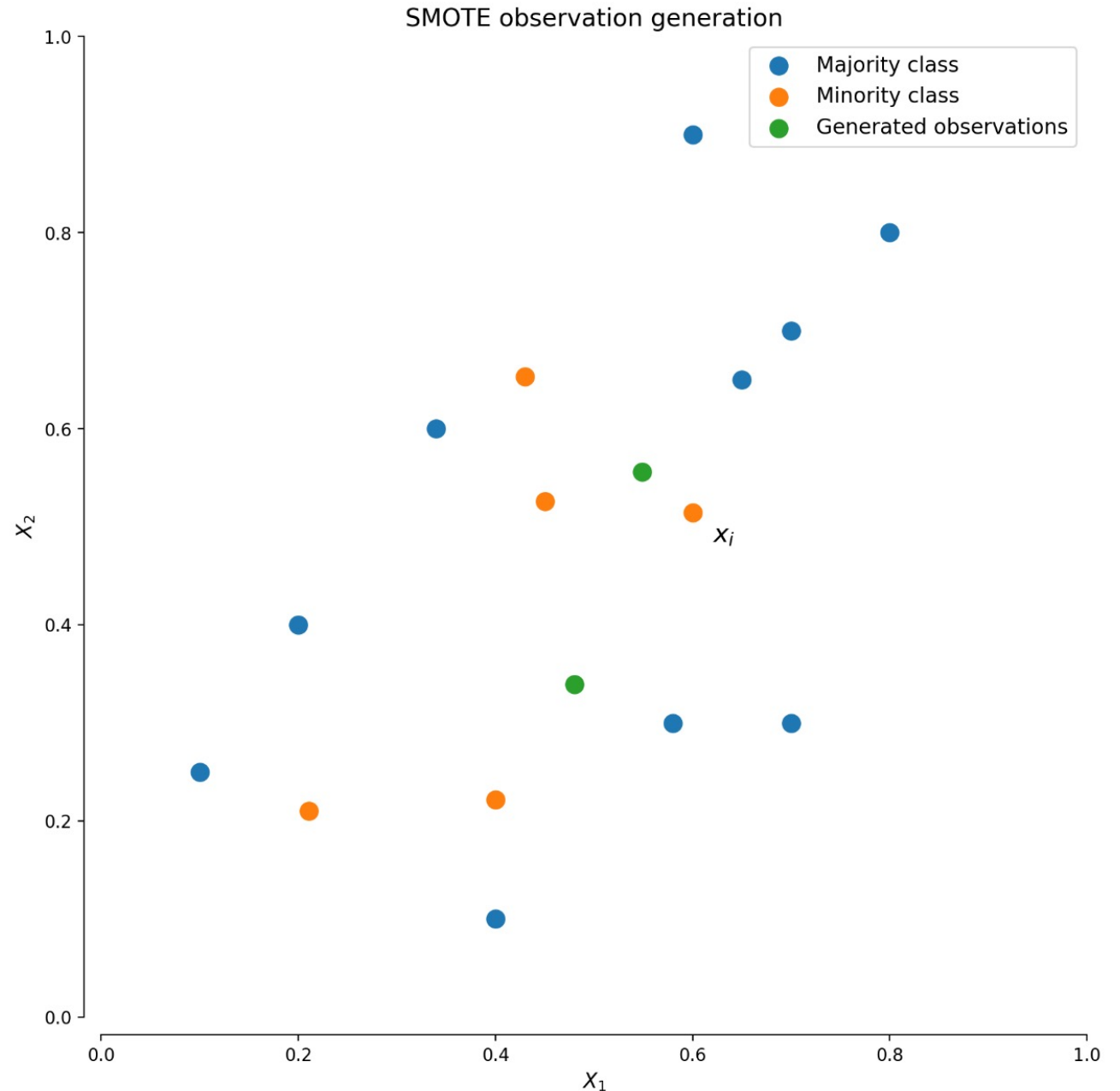
# SMOTE

- 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:

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

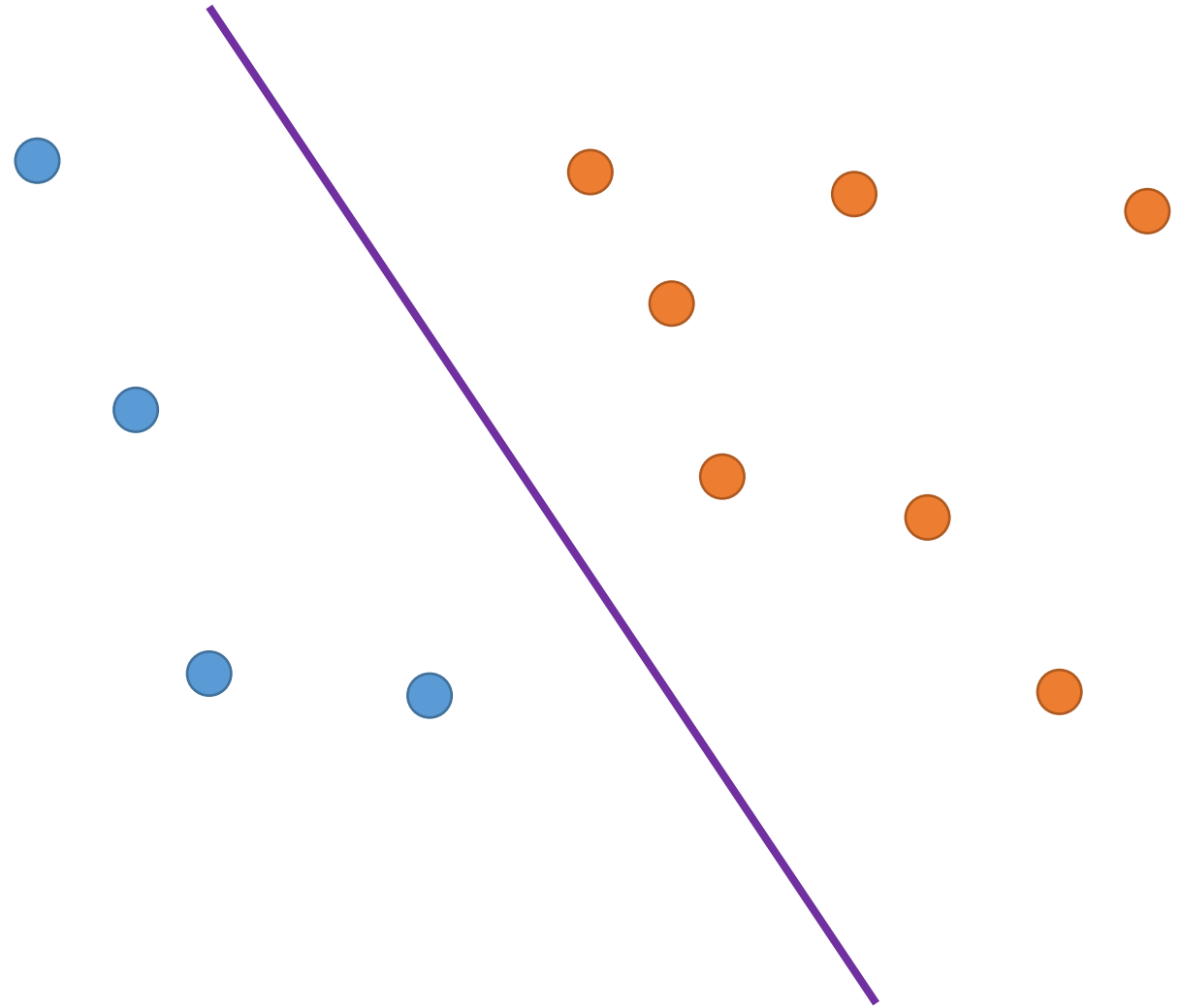
# ADASYN

- $C_{\text{maj}}, C_{\text{min}}$  – 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  $x_i$ :

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

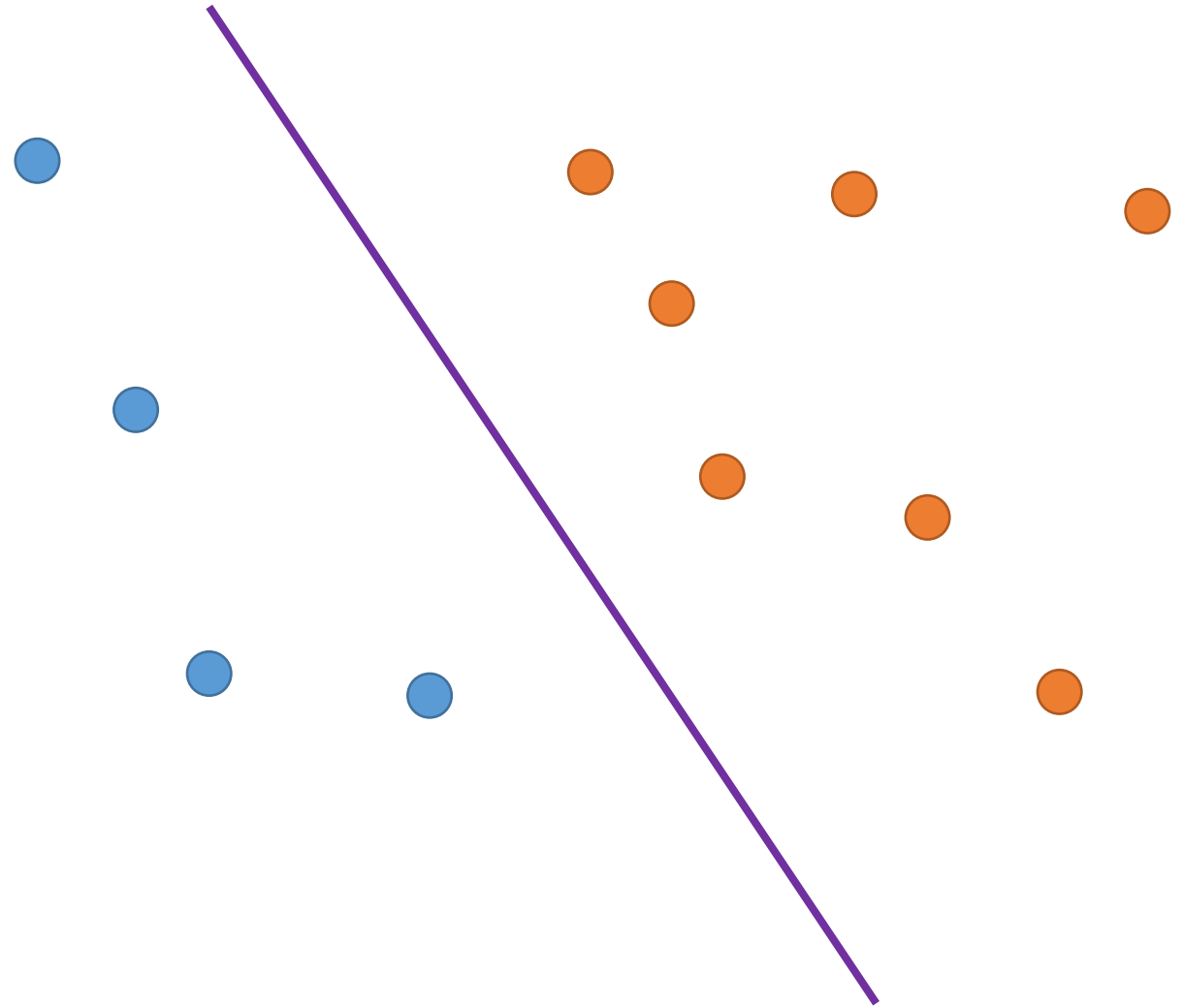
# Borderline-SMOTE

- Для моделей классификации очень важно выучить границу между классами



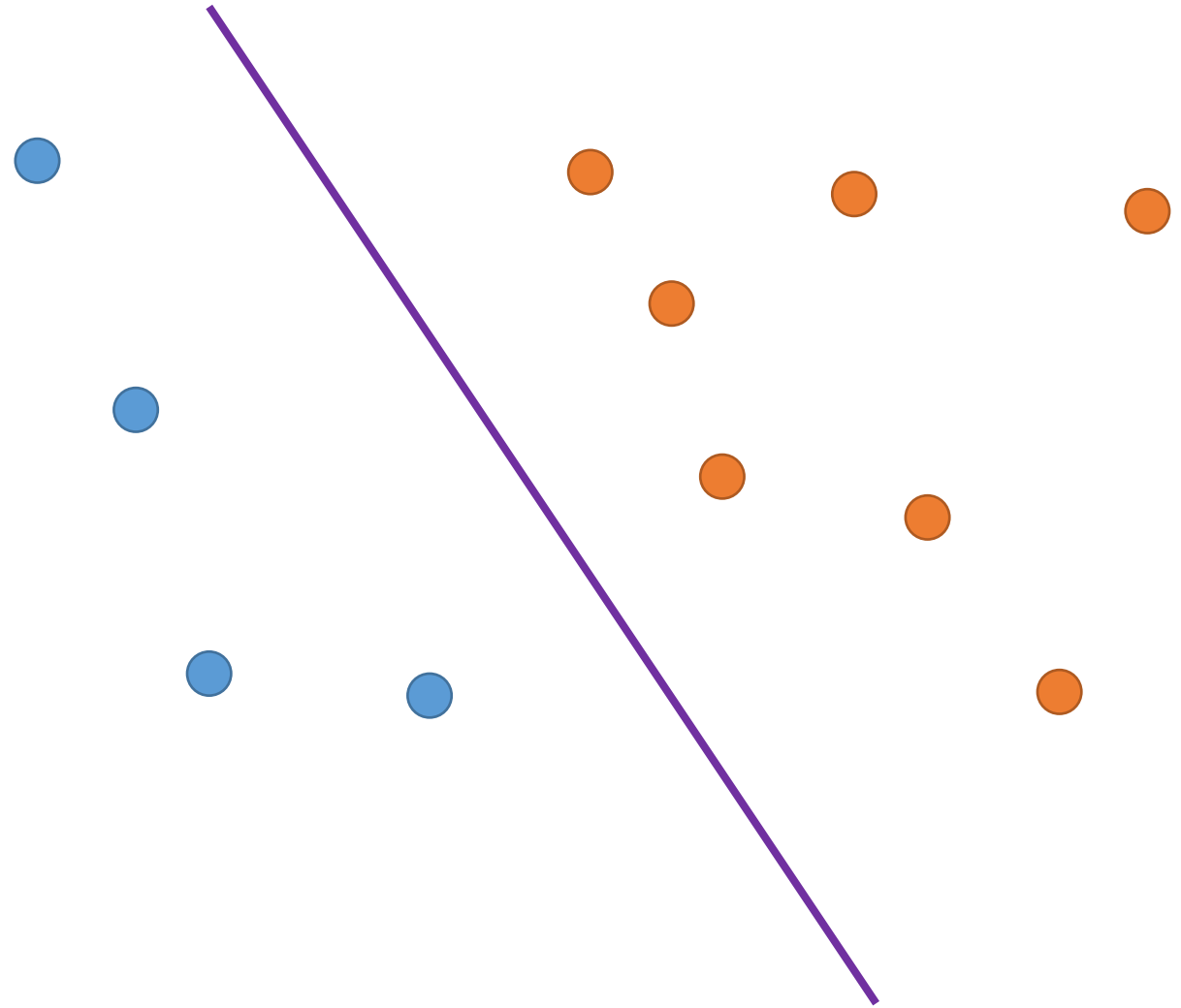
# Borderline-SMOTE

- Для моделей классификации очень важно выучить границу между классами
- Объекты около границ крайне важны



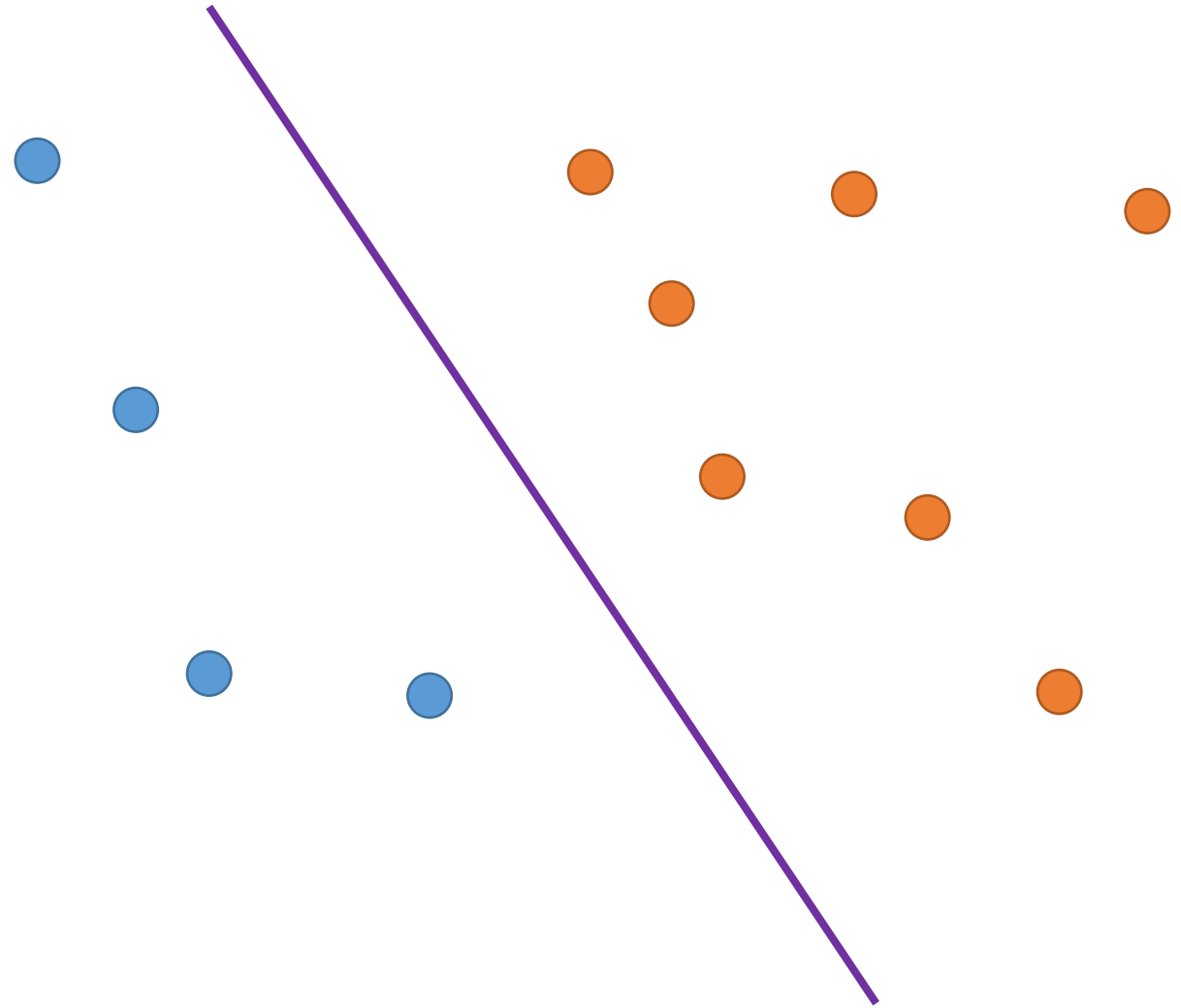
# Borderline-SMOTE

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



# 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



# 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

# 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)

# Borderline-SMOTE

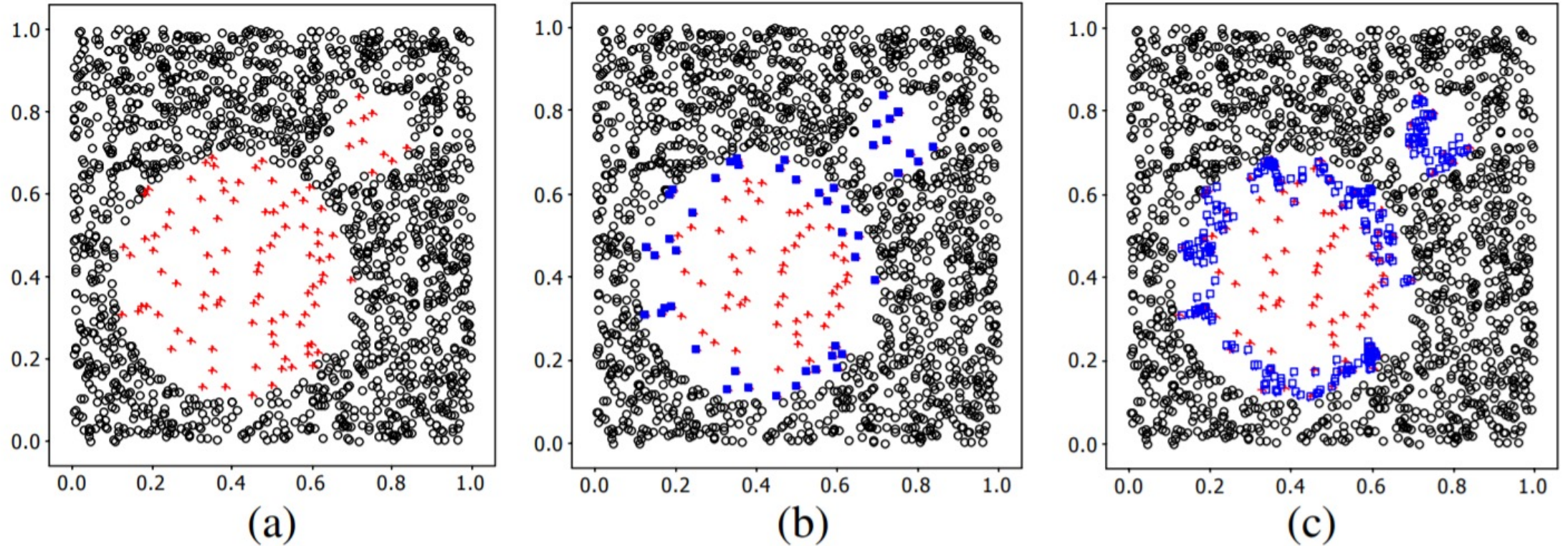
- Найти  $k$  ближайших соседей для каждого объекта  $x_i$  миноритарного класса
  - Затем для каждого  $x_i$  вычислить  $k' \in [0, k]$  – число соседей, принадлежащих к мажоритарному классу
1. Если  $k' = k$ , то  $x_i$  считаем шумом
  2. Если  $k' \in \left[0, \frac{k}{2}\right)$ , то  $x_i$  – «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

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

- **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**

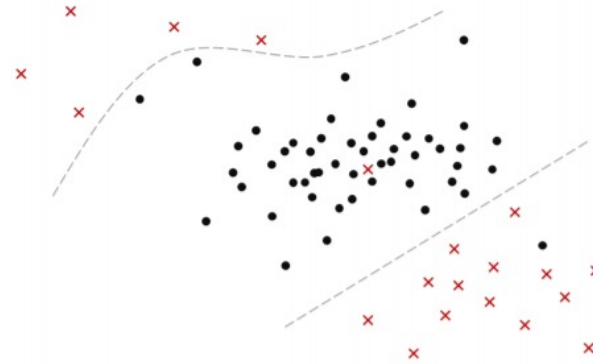
- определить границы классов с помощью SVM
- сгенерировать объекты на основе опорных векторов

- **K-Means SMOTE**

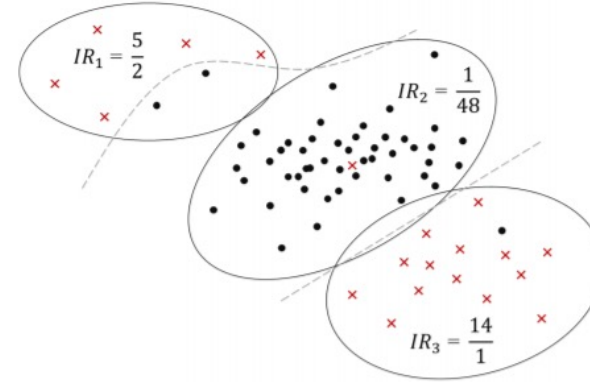
- кластеризовать объекты
- посчитать дисбаланс классов в кластерах
- сгенерировать объекты внутри кластеров с большим числом объектов миноритарного класса
- чтобы определить, сколько объектов сгенерировать, определить разреженность кластера

# K-Means SMOTE

Input data



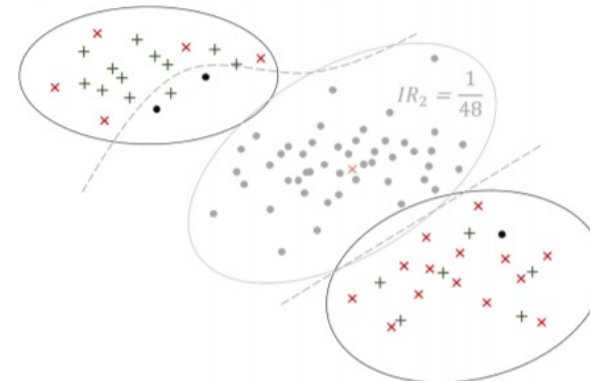
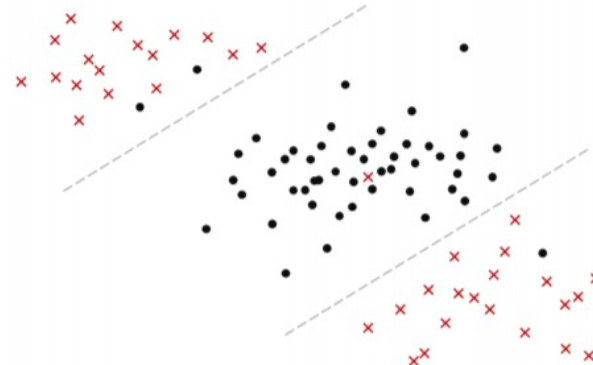
Find  $k = 3$  clusters and compute imbalance ratio (IR)



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

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

Oversampled data rectifies decision boundary



# Undersampling/oversampling

- В обоих методах модифицируется обучающая выборка – не валидация/тест!



# Undersampling/oversampling

- В обоих методах модифицируется обучающая выборка – не валидация/тест!
- Как вы думаете, разбиение на фолды для кросс-валидации нужно делать **до** oversampling, **после** или можно **и так, и так**?

# Undersampling/oversampling

- В обоих методах модифицируется обучающая выборка – не валидация/тест!
- Разбиение на фолды для кросс-валидации нужно делать **до** oversampling

# 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

# Определение аномалий

## Популярные техники [ [править](#) | [править код](#) ]

---

В литературе было предложено несколько техник выявления аномалий<sup>[7]</sup>. Вот некоторые популярные техники:

- Техники, основанные на плотности (**к-ближайшие соседи**<sup>[8][9][10]</sup>, **локальный уровень выброса**<sup>[11]</sup>, **изолирующие леса**<sup>[12]</sup> и многие другие варианты этой концепции<sup>[13]</sup>).
- Обнаружение выбросов на основе подпространств<sup>[14]</sup> и на основе корреляции<sup>[15]</sup> для данных высокой размерности<sup>[16]</sup>.
- **Метод опорных векторов** для одного класса<sup>[17]</sup>.
- Репликатор **нейронных сетей**<sup>[18]</sup>.
- **Байесовские сети**<sup>[18]</sup>.
- **Скрытые марковские модели** (СММ)<sup>[18]</sup>.
- Выявление выбросов на основе **кластерного анализа**<sup>[19][20]</sup>.
- Отклонения от **ассоциативных правил** и часто встречающихся наборов.
- Выявление выбросов на основе нечёткой логики.
- **Техника создания ансамблей**, использующая **бэггинг признаков**<sup>[en][21][22]</sup>, усреднение оценки<sup>[23][24]</sup> и различение источников несхожести<sup>[25][26]</sup>.

Эффективность различных методов зависит от данных и параметров и имеют слабые систематические преимущества один перед другим, если сравнивать по многим наборам данных и параметров<sup>[27][28]</sup>.

[https://ru.wikipedia.org/wiki/Выявление\\_аномалий](https://ru.wikipedia.org/wiki/Выявление_аномалий)

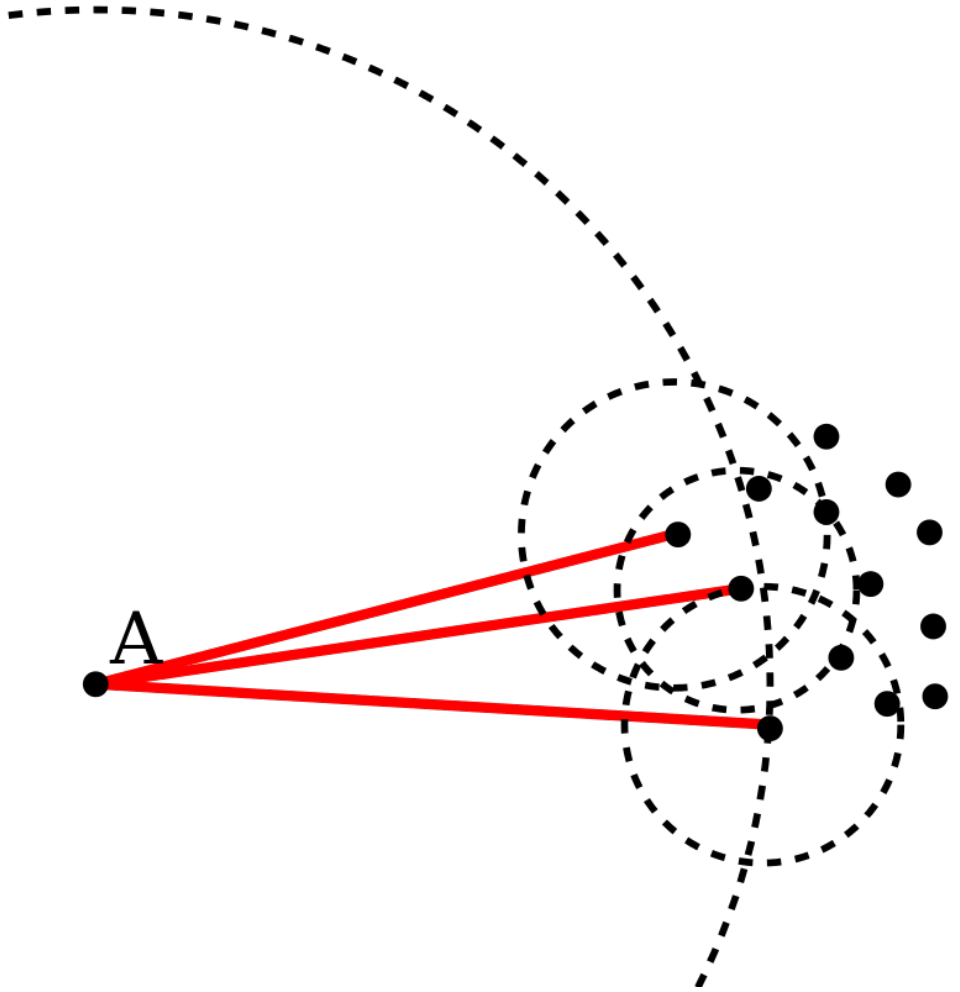
# Методы на основе kNN

- 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

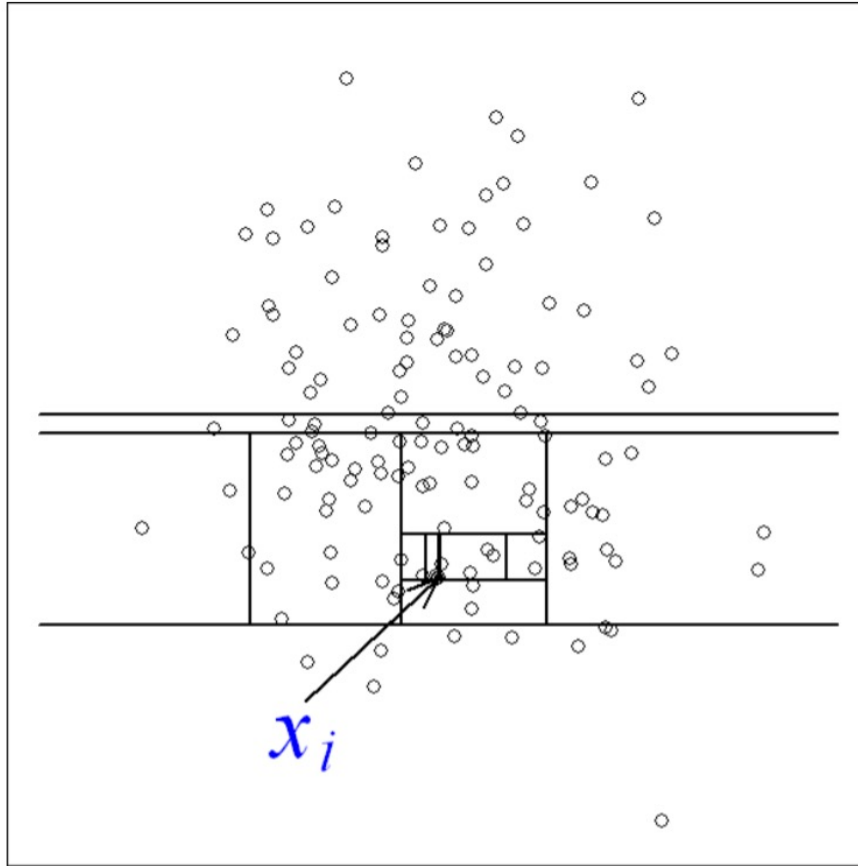
- **LOF: Local Outlier Factor**
- Наблюдение аномально, если его локальная плотность намного меньше локальной плотности его ближайших соседей



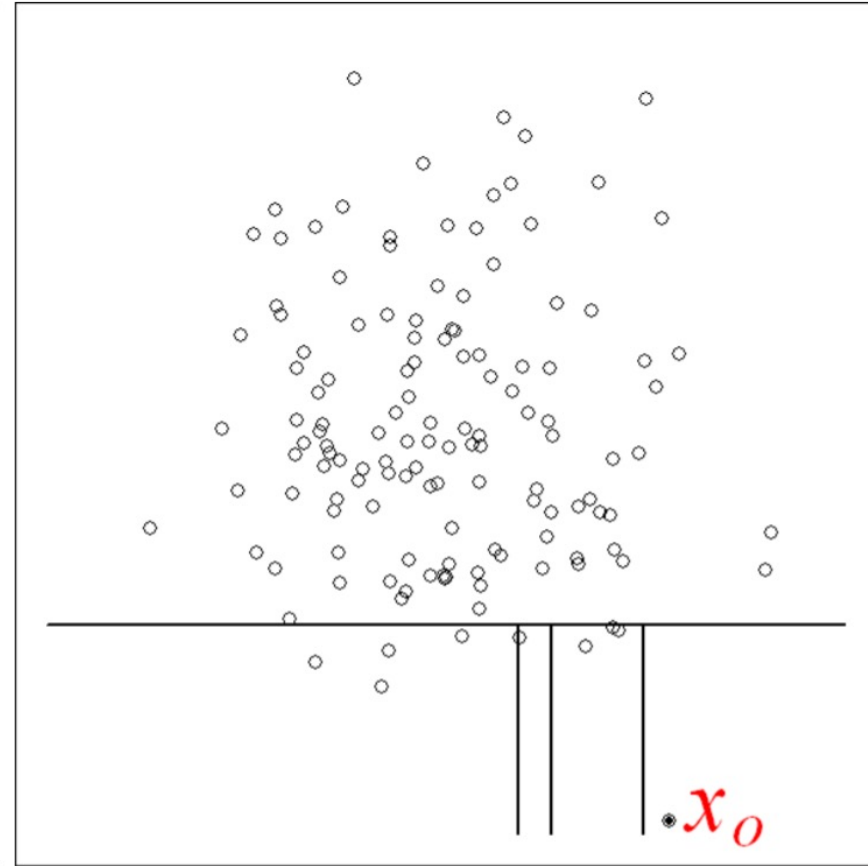
# Isolation Forest

- **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