



НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ

Машинное обучение в маркетинге

ЗАДАЧА КЛАССИФИКАЦИИ

Элен Теванян

Москва, 2019

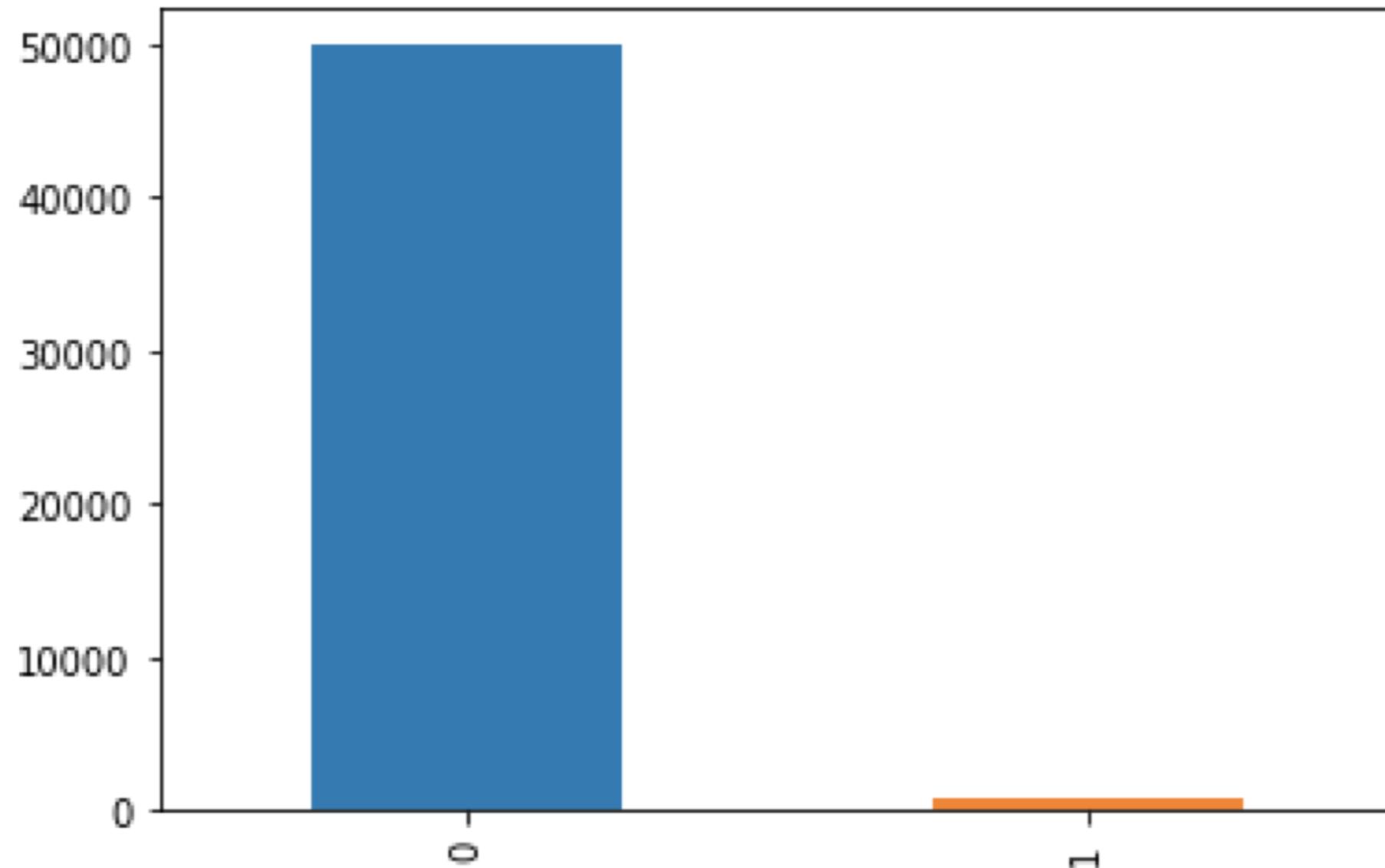
Постановка задачи

ПРИМЕР ЗАДАЧИ КЛАССИФИКАЦИИ

- Страховая компания планирует ценовую стратегию на следующий год. За последние 3 года они тратили на страховые компенсации денег больше, чем рассчитывали. Они решили обучить модель, которая спрогнозирует обращение за компенсацией клиентом
- Задача: клиент → [затребовал компенсацию (1)/ не затребовал (0)]
- x_i – объект, для которого строим предсказания (клиент)
- y_i – целевая переменная: 1 или 0
- (x_i, y_i) – прецедент
- Обучающая выборка – все клиенты страховой компании

Баланс классов

СБАЛАНСИРОВАНА ЛИ ВЫБОРКА?



ЕСЛИ ПРОИГНОРИРОВАТЬ ДИСБАЛАНС?

- практически все классификаторы пытаются минимизировать число ошибок на обучающей выборке
- классификаторы не учитывают цены ошибок

Классификаторы могут получаться очень плохие с точки зрения точности или полноты =>

Мы не научились прогнозировать целевой класс

ВАРИАНТ 1: ПЕРЕВЗЕШИВАНИЕ

В стандартных алгоритмах почти всегда есть параметр `class_weight` и для несбалансированных данных рекомендуется прописывать параметр `class_weight = 'balanced'`

Тогда внутри алгоритма каждому классу присваивается вес по формуле:

$$\text{class weight} = \frac{n_{samples}}{n_{classes} * np. bincount(y))}$$

ВАРИАНТ 2: UPSAMPLING

- Искусственно увеличим размер датасета и за счет новых данных сбалансируем выборку.
- Баланс достигается за счет добавления дубликатов редкого класса в выборку.

```
1 k = 20
2 X_train_upsampled = X_train[y_train == 0]
3 minor_class = X_train[y_train == 1]
4
5 # В цикле k раз допишите в X_train_upsampled датафрейм minor_class
6 for i in range(k):
7     X_train_upsampled = pd.concat([X_train_upsampled, minor_class], axis=0)
8     print(X_train_upsampled.shape[0])
9
10 # Сбросьте индекс, чтобы в таблице не было дублирующихся индексов
11 X_train_upsampled = X_train_upsampled.reset_index(drop=True)
12
13 # Создайте вектор ones
14 ones = np.ones(len(minor_class)*k)
15 # Удлините вектор целевой переменной тренировочной выборки
16 y_train_upsampled = pd.concat([y_train[y_train==0], pd.Series(ones)])
17 # Сбросьте индекс, чтобы в таблице не было дублирующихся индексов
18 y_train_upsampled = y_train_upsampled.reset_index(drop=True)
```

ВАРИАНТ 3: SUBSAMPLING

Если при Upsampling мы раширяли датасет дубликатами редкого класса, то при Subsampling мы будем датасет уменьшать.

- редкий класс останется неизменным - он полностью включается в датасет
- сократится преобладающий класс: мы сделаем из него подвыборку размером с датасет редкого класса.

ВАРИАНТ 3: SUBSAMPLING

```
1 # - Выделите данные для объектов с меткой 1 в отдельную переменную minor_class.  
2 minor_class = x_train[y_train == 1]  
3 # В переменной minor_class_size сохраните количество строк датафрейма minor_class  
4 minor_class_size = minor_class.shape[0]  
5 # Выделите данные для объектов с меткой 0 в отдельную переменную major_class  
6 major_class = x_train[y_train == 0]  
7 # В переменной major_class_size сохраните количество строк датафрейма major_class  
8 major_class_size = major_class.shape[0]  
9  
10 # Сгенерируйте подвыборку размера minor_class_size  
11 major_class_sample = major_class.sample(minor_class_size)  
12 # Объедините major_class_sample и minor_class в одну таблицу  
13 x_train_subsampled = pd.concat([major_class_sample, minor_class], axis=0)  
14 # Сбросьте индекс  
15 x_train_subsampled.reset_index(drop=True, inplace=True)  
16  
17 # Создайте y_major_class, состоящий из нулей  
18 y_major_class = pd.Series(np.zeros(minor_class_size))  
19 # Создайте y_minor_class, состоящий из единиц  
20 y_minor_class = pd.Series(np.ones(minor_class_size))  
21 # Объедините y_major_class и y_minor_class в y_train_subsampled  
22 y_train_subsampled = pd.concat([y_major_class, y_minor_class])  
23 # Сбросьте индекс методом reset_index()  
24 y_train_subsampled.reset_index(drop=True, inplace=True)
```

СТРАТИФИКАЦИЯ

Вне зависимости от способа балансировки данных советуем делить данные на train и test/validation стратифицированно:

- `train_test_split(....., stratified=y)`: для единоразового разбиения
- `StratifiedKFold()`: для крос—валидации
- `StratifiedShuffleSplit`: для отложенных выборок

Метрики качества

ПРИМЕР ЗАДАЧИ КЛАССИФИКАЦИИ

- Пусть в обучающей выборке 100 клиентов
- В таблице приведены результаты обучения модели машинного обучения

	Клиент обратился $y = 1$	Клиент не обратился $y = 0$
Модель спрогнозировала, что клиент обратился $a(x) = 1$	60 True Positive (TP)	5 False Positive(FP)
Модель спрогнозировала, что клиент не обратился $a(x) = 0$	25 False Negative (FN)	10 True Negative (TN)

ДОЛЯ ПРАВИЛЬНЫХ ОТВЕТОВ

- Доля правильных ответов (accuracy) показывает долю объектов в выборке, которым классификатор присвоил их истинный класс.

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

	$y = 1$	$y = 0$
$a(x) = 1$	60	5
$a(x) = 0$	25	10

ДОЛЯ ПРАВИЛЬНЫХ ОТВЕТОВ

- Доля правильных ответов (accuracy) показывает долю объектов в выборке, которым классификатор присвоил их истинный класс.

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

	$y = 1$	$y = 0$
$a(x) = 1$	60	5
$a(x) = 0$	25	10

$$\text{Accuracy} = \frac{60+10}{100} = 0.7$$

ТОЧНОСТЬ

- Точность (precision) показывает уровень доверия к классификатору при $a(x) = 1$

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

	$y = 1$	$y = 0$
$a(x) = 1$	60	5
$a(x) = 0$	25	10

ТОЧНОСТЬ

- Точность (precision) показывает уровень доверия к классификатору при $a(x) = 1$

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

	$y = 1$	$y = 0$
$a(x) = 1$	60	5
$a(x) = 0$	25	10

$$P = \frac{60}{60+5} = 0.92$$

- Отвечаем на вопрос: можно ли доверять классификатору при $a(x) = 1$

ПОЛНОТА

- Полнота (recall) показывает, какую долю правильно-положительных ответов находит классификатор

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

	$y = 1$	$y = 0$
$a(x) = 1$	60	5
$a(x) = 0$	25	10

ПОЛНОТА

- Полнота (recall) показывает, какую долю правильно-положительных ответов находит классификатор

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

	$y = 1$	$y = 0$
$a(x) = 1$	60	5
$a(x) = 0$	25	10

$$P = \frac{60}{60+25} = 0.71$$

- Отвечаю на вопрос: много ли положительных ответов находит классификатор?

ОЦЕНКА ПРИНАДЛЕЖНОСТИ К КЛАССУ

- Будем жить в мире, где два класса: +1 и -1
 - t – порог
 - $b(x)$ – оценка принадлежности к классу +1
-
- Классифицируем так:

$$a(x) = [b(x) > t]$$

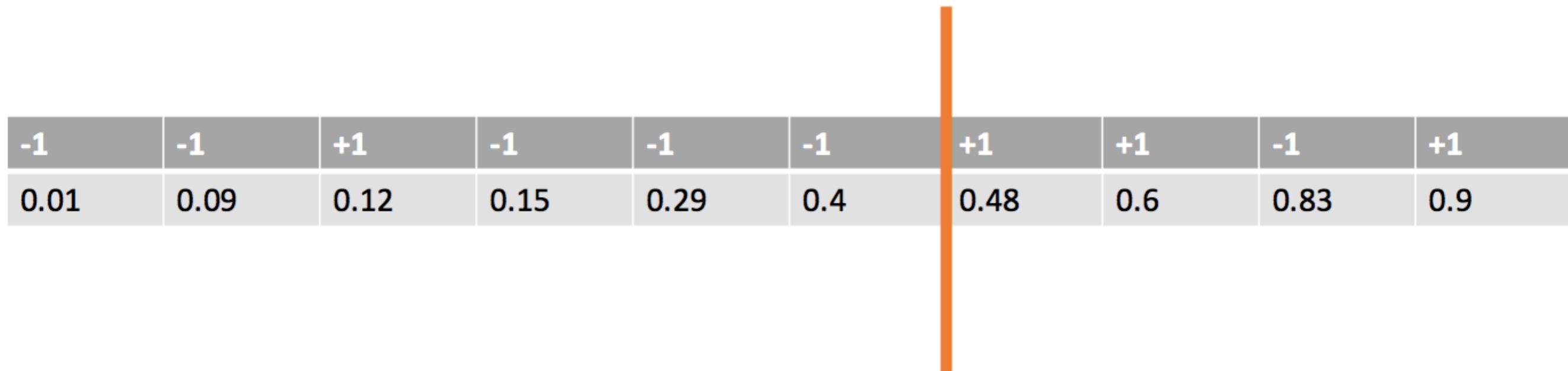
ОЦЕНКА ПРИНАДЛЕЖНОСТИ К КЛАССУ

- Как оценить качество $b(x)$?
- Как выбрать порог?

ОЦЕНКА ПРИНАДЛЕЖНОСТИ К КЛАССУ

- Как оценить качество $b(x)$?
- Как выбрать порог?
- Порог выбирается позже
 - Порог высокий:
 - Мало объектов к классу +1
 - Точность выше
 - Полнота ниже
 - Порог низкий:
 - Много объектов относим к +1
 - Точность ниже
 - Полнота выше

ОЦЕНКА ПРИНАДЛЕЖНОСТИ К КЛАССУ



ОЦЕНКА ПРИНАДЛЕЖНОСТИ К КЛАССУ

-1	-1	+1	-1	-1	-1	+1	+1	-1	+1
0.01	0.09	0.12	0.15	0.29	0.4	0.48	0.6	0.83	0.9

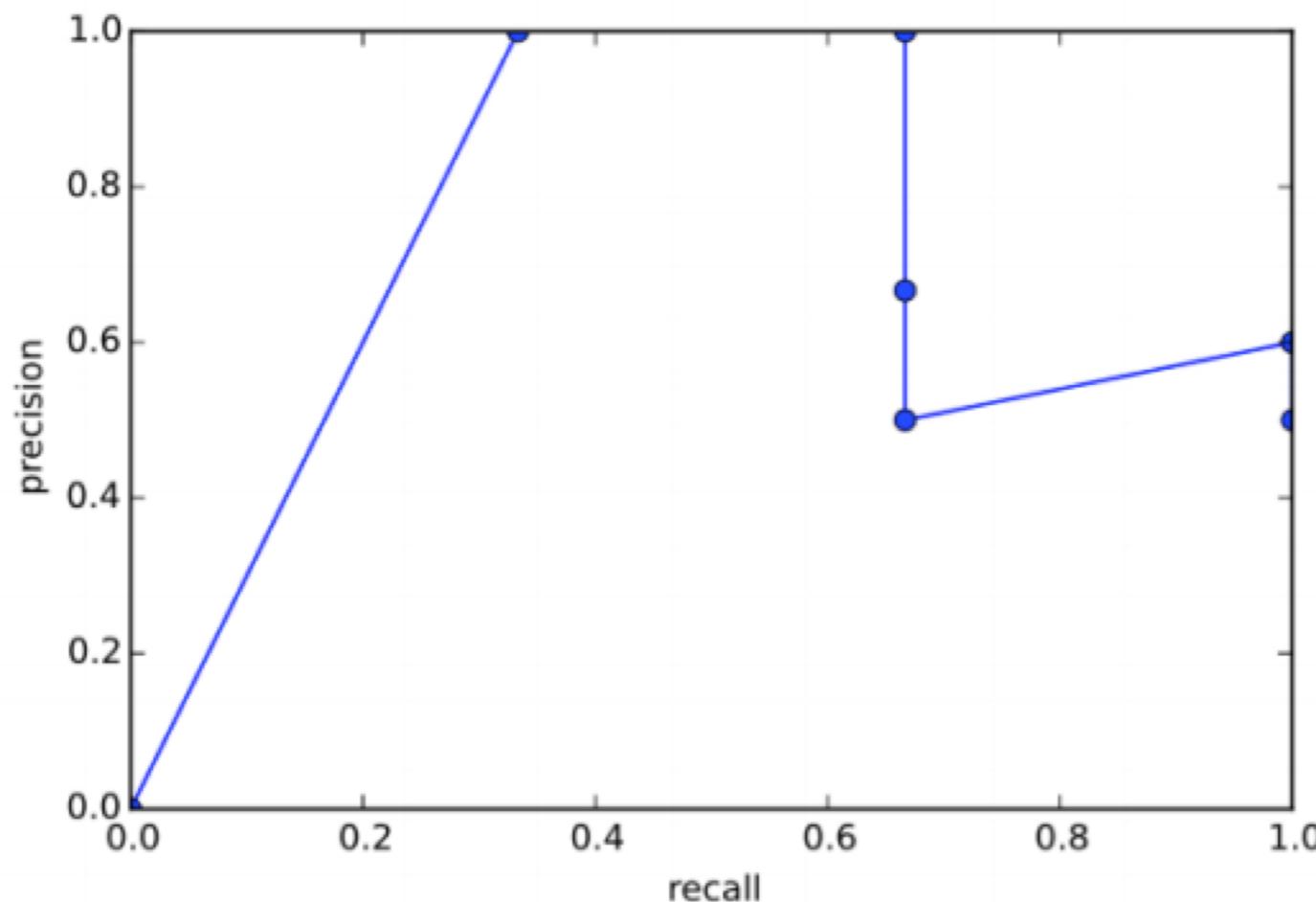


ОЦЕНКА ПРИНАДЛЕЖНОСТИ К КЛАССУ

- Пример: кредитный scoring
- $b(x)$ – оценка вероятности возврата кредита
- $a(x) = [b(x) > 0.5]$
- precision = 0.1
- recall = 0.7
- ??????

PR-КРИВАЯ

- Кривая точности-полноты
- Ось X – полнота
- Ось Y – точность
- Точки значения полноты и точности при последовательных порогах



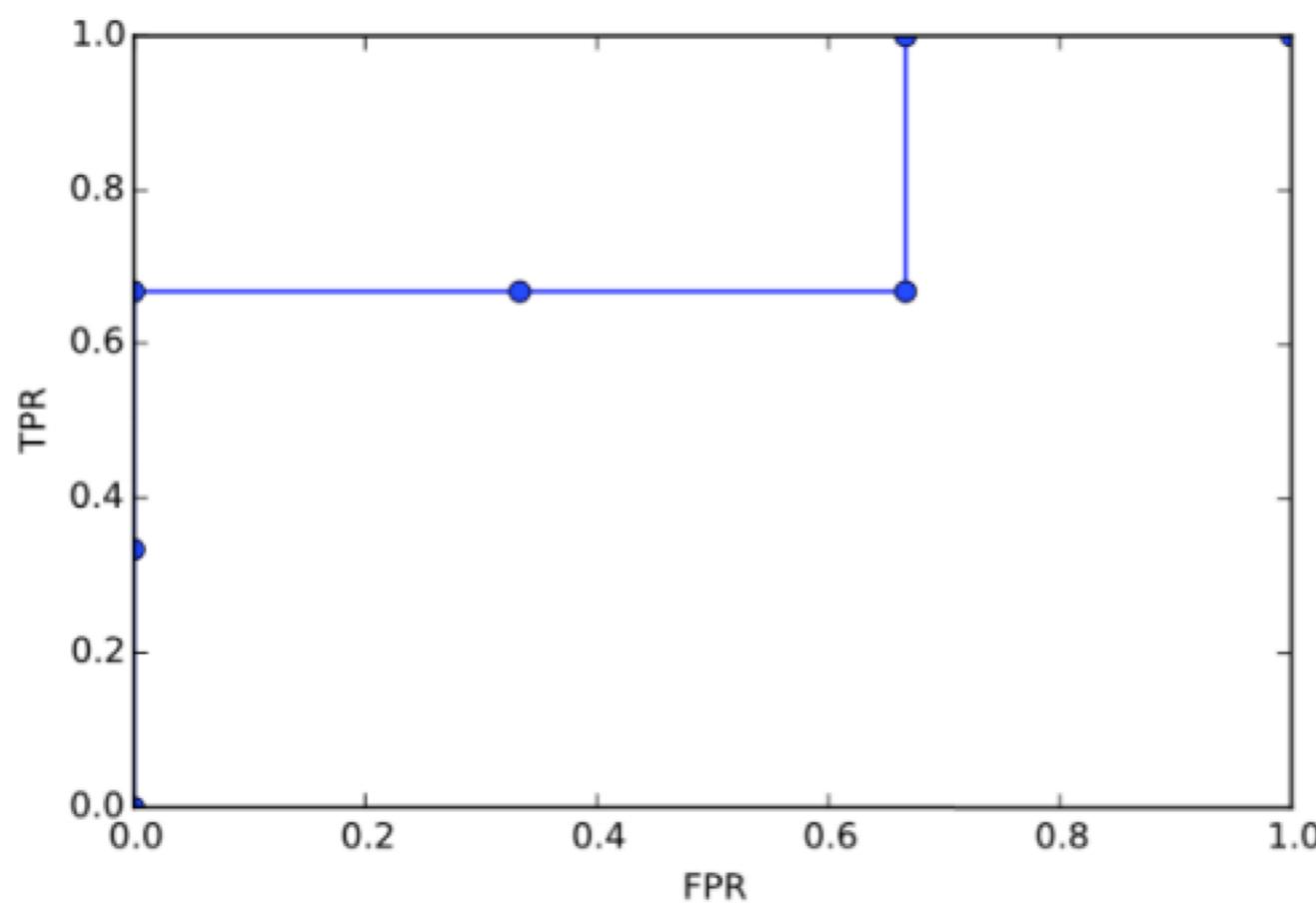
ROC- КРИВАЯ

- Receiver Operating Characteristic
- Ось X – False Positive Rate

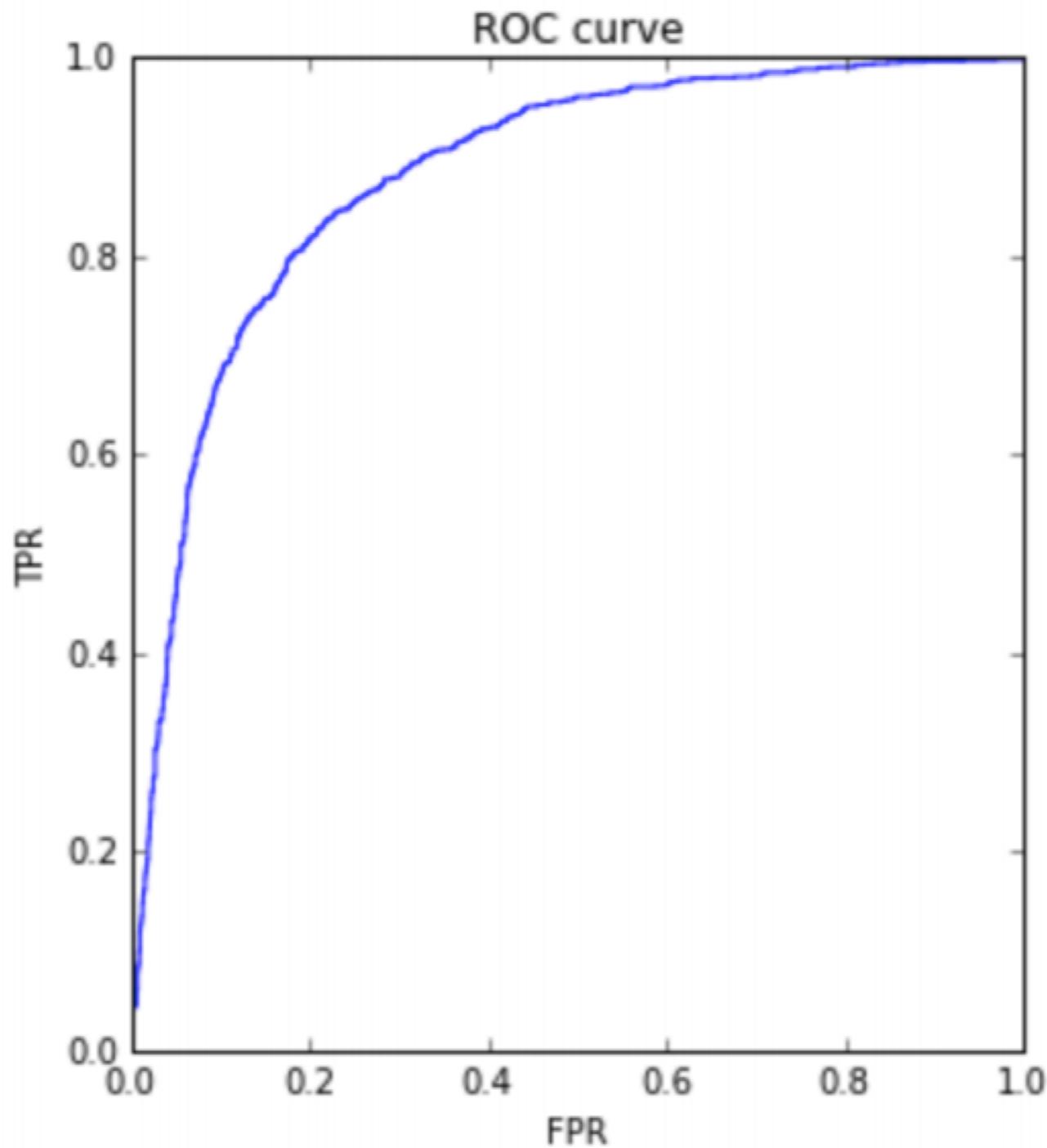
$$FPR = \frac{FP}{FP+TN}$$

- Ось Y – True Positive Rate

$$TPR = \frac{TP}{TP+TN}$$

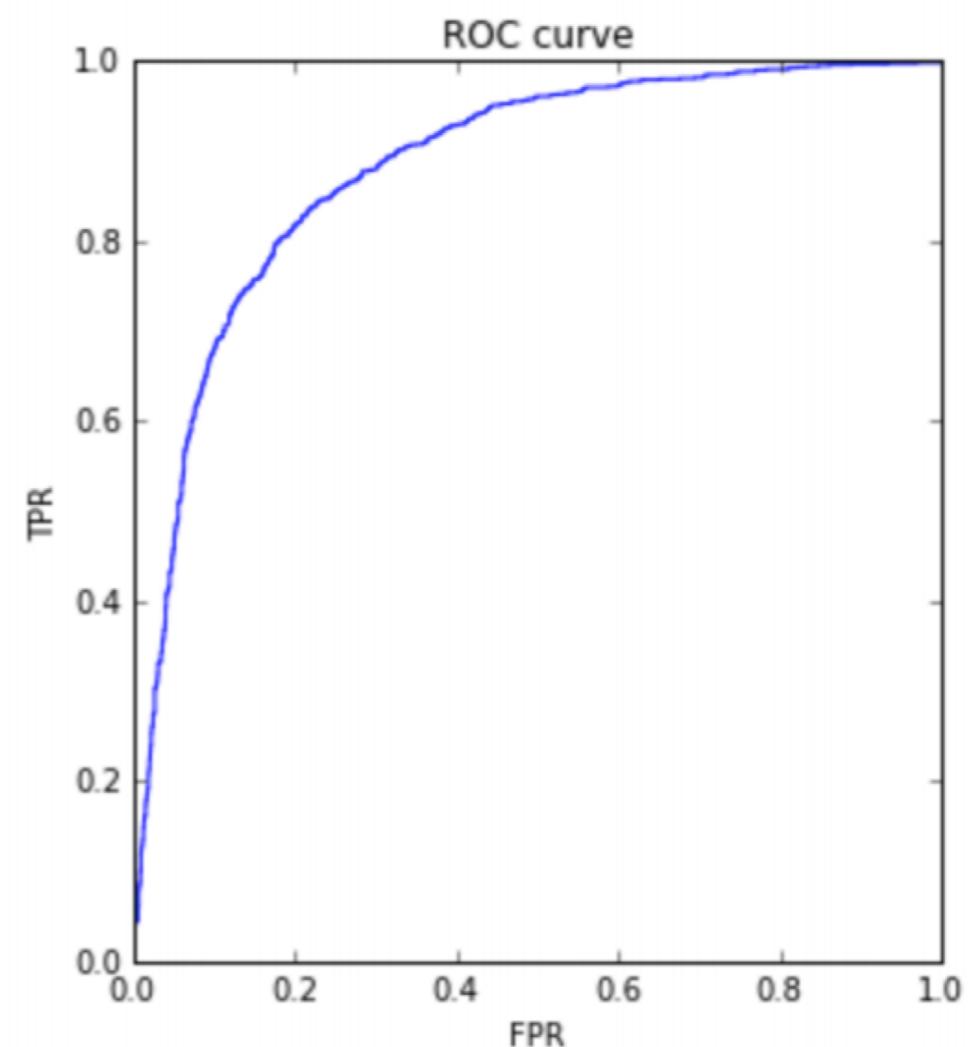


ROC- КРИВАЯ В РЕАЛЬНОСТИ



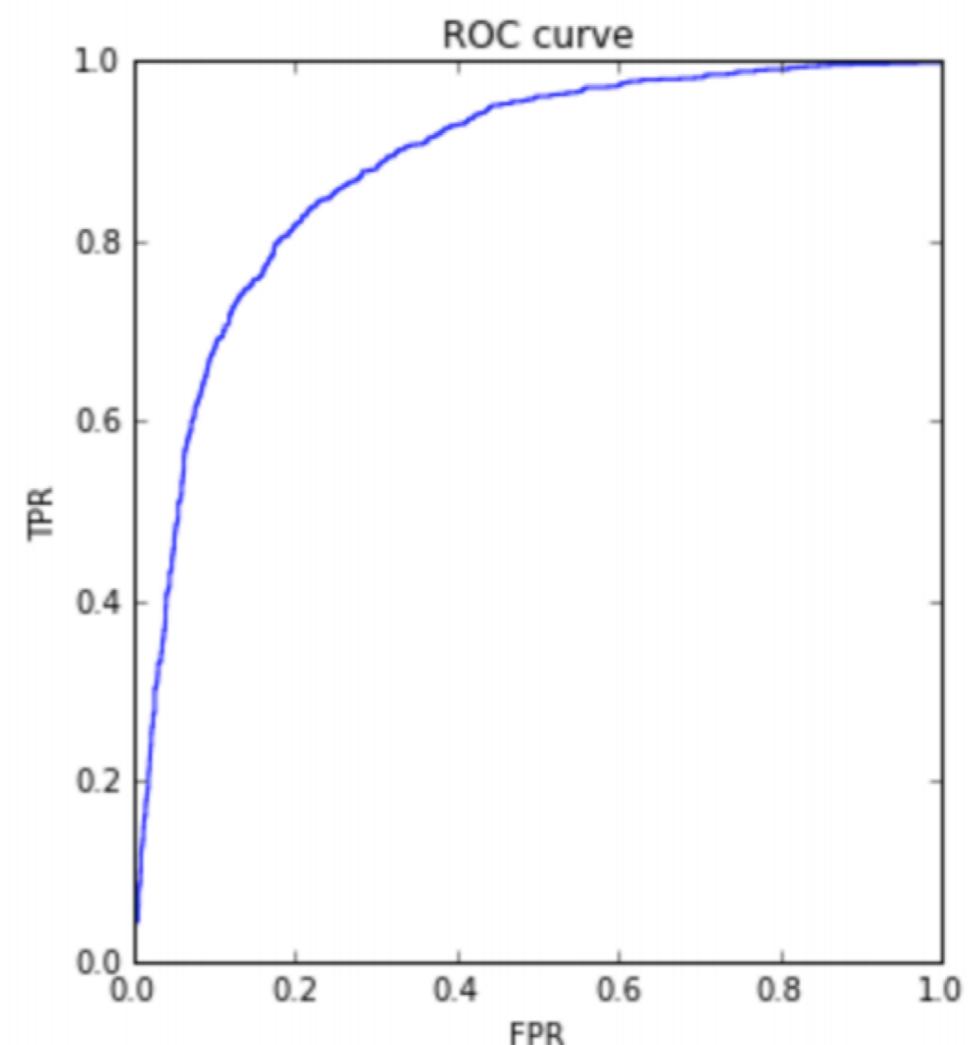
ROC-КРИВАЯ

- Левая точка: (0, 0)
 - Правая: (1,1)
 - Идеально:
захватили точку (0, 1)
-
- Считаем еще AUC-ROC –
площадь под кривой



ROC-КРИВАЯ

- Левая точка: (0, 0)
 - Правая: (1,1)
 - Идеально:
захватили точку (0, 1)
-
- Считаем еще AUC-ROC –
площадь под кривой
 - Идеально: AUC-ROC = 1
 - Плохо: AUC-ROC = 0.5



Алгоритмы

KNN

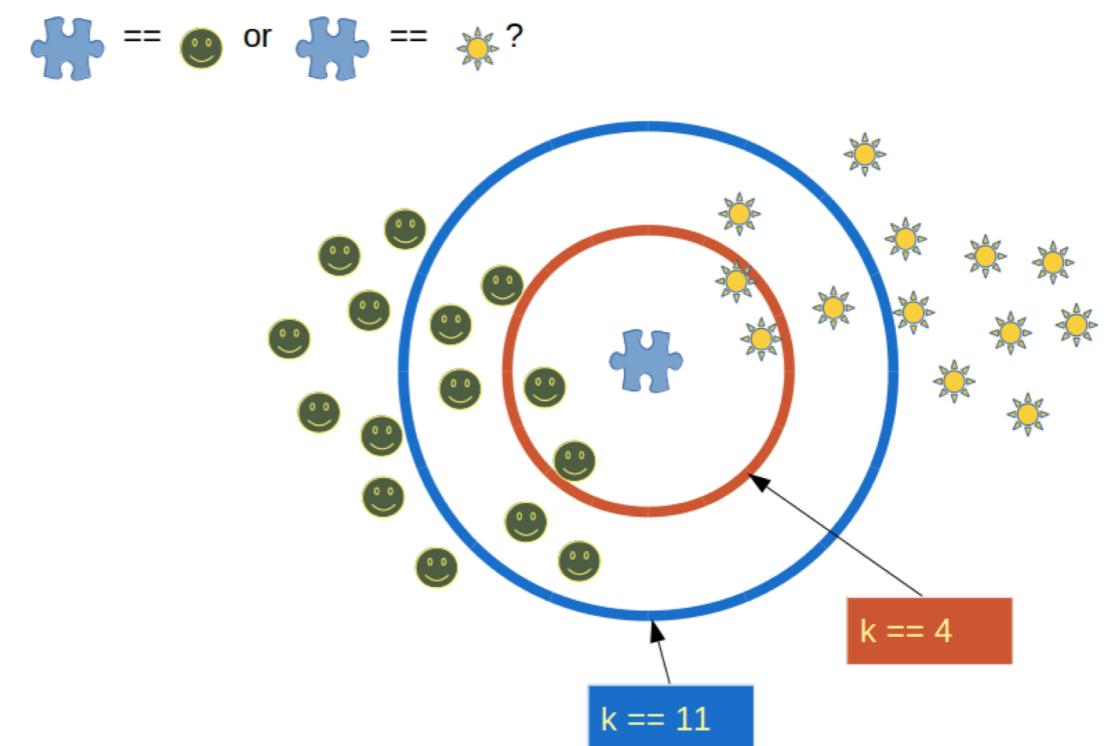
- Метод k-ближайших соседей – метрический, т.е. нужно считать расстояния
- Дано: (x_i, y_i) – прецеденты
- Обучение: запоминаем выборку

KNN

- Получаем новый объект x
- Сортируем объекты выборки по расстоянию до x

- Присваиваем x класс, наиболее популярный в его окружении

$$a(x) = \arg \max_{y \in \mathbb{Y}} \sum_{i=1}^k [y_{(i)} = y]$$

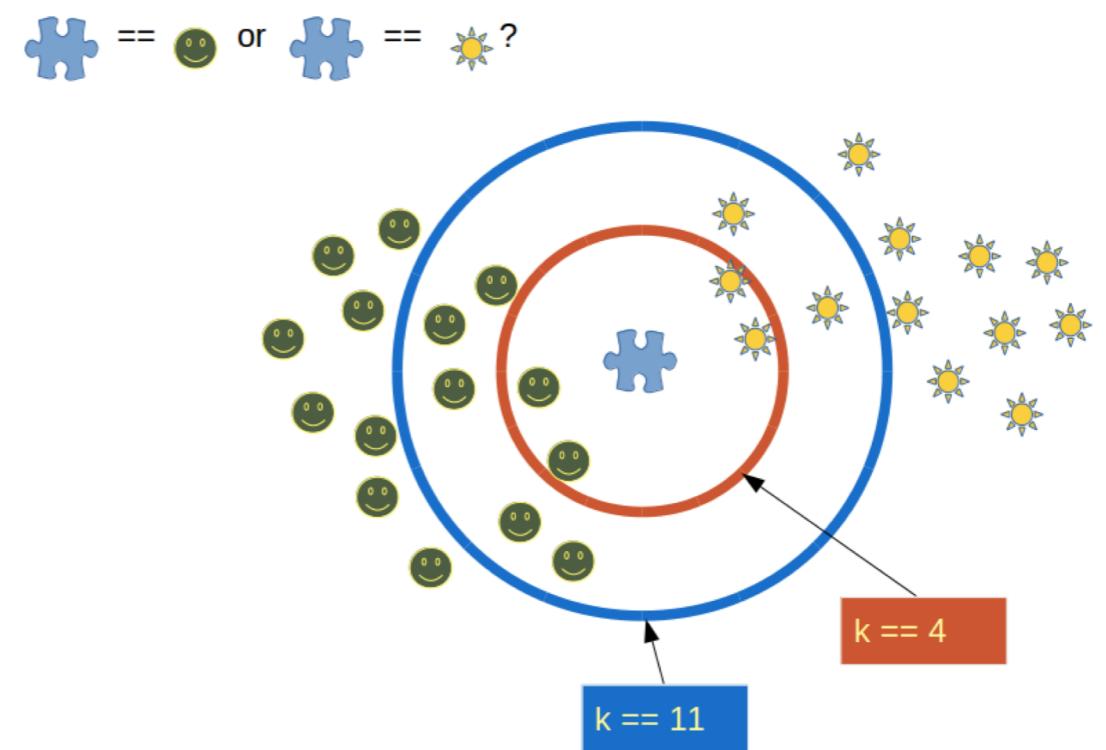


KNN

- Получаем новый объект x
- Сортируем объекты выборки по расстоянию до x

- Присваиваем x класс, наиболее популярный в его окружении

$$a(x) = \arg \max_{y \in \mathbb{Y}} \sum_{i=1}^k [y_{(i)} = y]$$



- k – гиперпараметр
- Никак не учитывается расстояние до ближайших соседей
- Как такового обучения модели нет

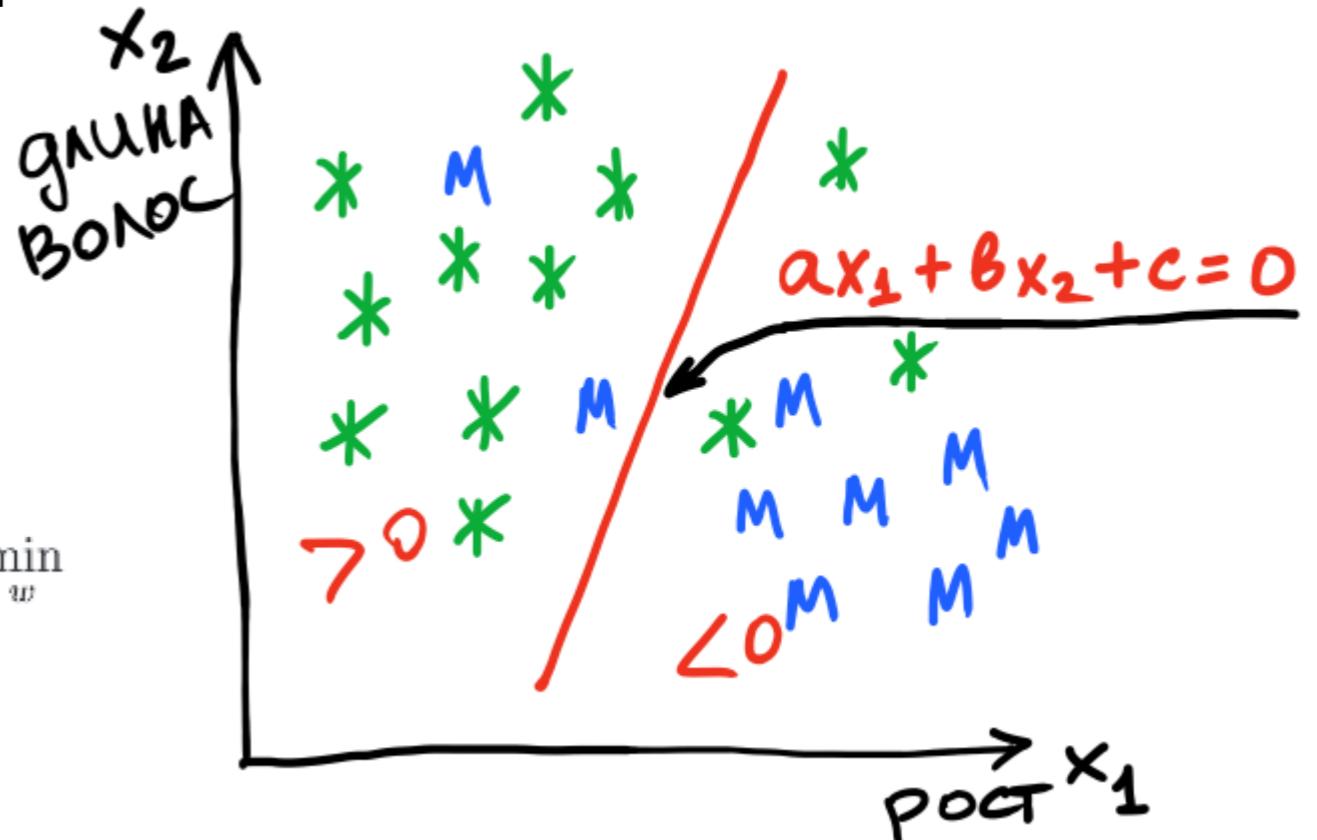
ЛИНЕЙНЫЙ КЛАССИФИКАТОР

- Модель:

$$a(x) = \text{sign} (\langle w, x \rangle + w_0) = \text{sign} \left(\sum_{j=1}^d w_j x_j + w_0 \right)$$

- Простейшая функция потерь для обучения:

$$Q(a, X) = \frac{1}{\ell} \sum_{i=1}^{\ell} [a(x_i) \neq y_i] = \frac{1}{\ell} \sum_{i=1}^{\ell} [\text{sign} \langle w, x_i \rangle \neq y_i] \rightarrow \min_w$$



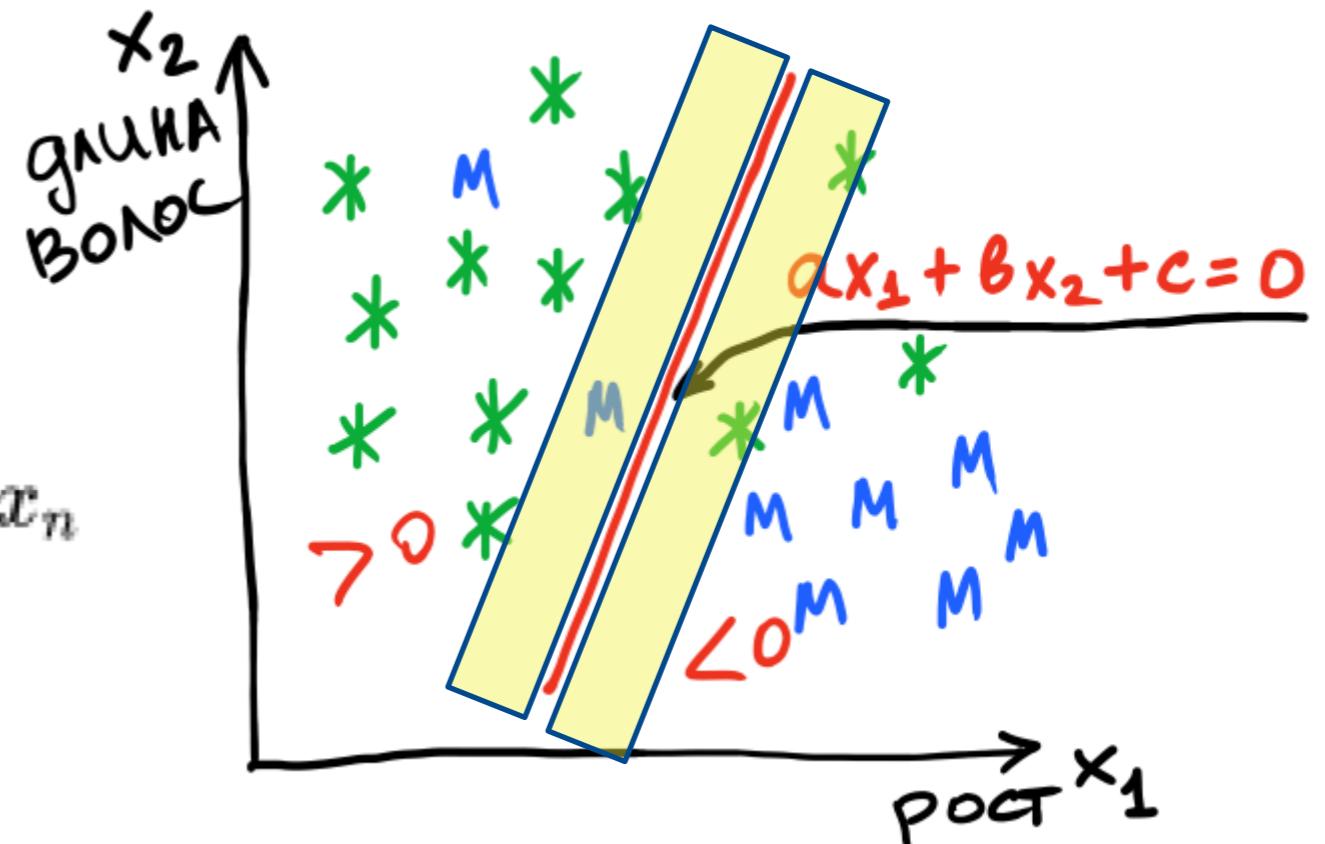
ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

- Модель:

$$\mathbb{P}\{y = 1 \mid x\} = f(z),$$

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \dots + \theta_n x_n$$

$$f(z) = \frac{1}{1 + e^{-z}}.$$



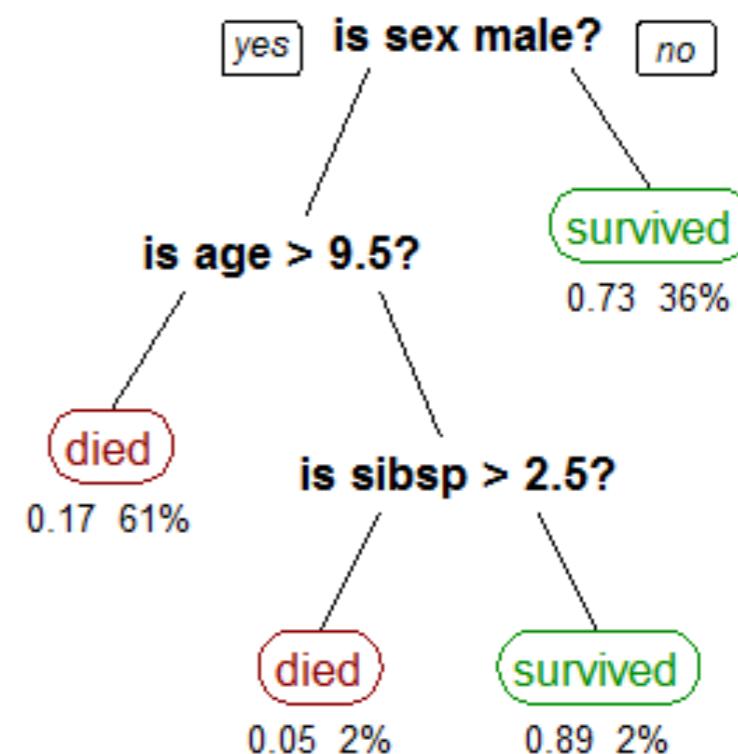
$$\mathbb{P}\{y \mid x\} = f(\theta^T x)^y (1 - f(\theta^T x))^{1-y}, \quad y \in \{0, 1\}.$$

$$\hat{\theta} = \operatorname{argmax}_{\theta} L(\theta) = \operatorname{argmax}_{\theta} \prod_{i=1}^m \mathbb{P}\{y = y^{(i)} \mid x = x^{(i)}\}.$$

<https://habr.com/ru/company/io/blog/265007/>

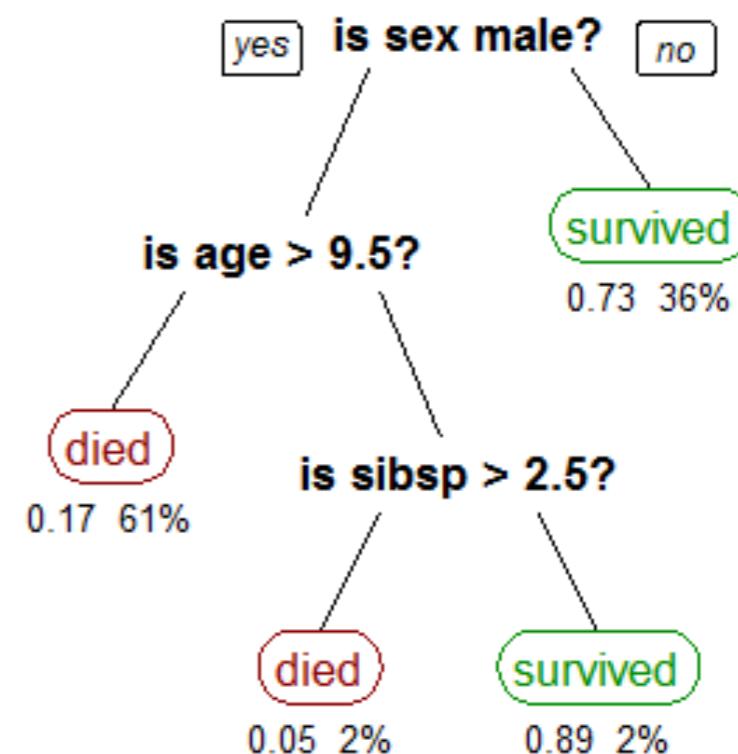
РЕШАЮЩЕЕ ДЕРЕВО

- В каждом узле записано условие
- В зависимости от ответа переходим по одной из веток
далее
- Если из узла выходят только по две ветки, то дерево –
бинарное
- В листьях записаны метки класса



РЕШАЮЩЕЕ ДЕРЕВО

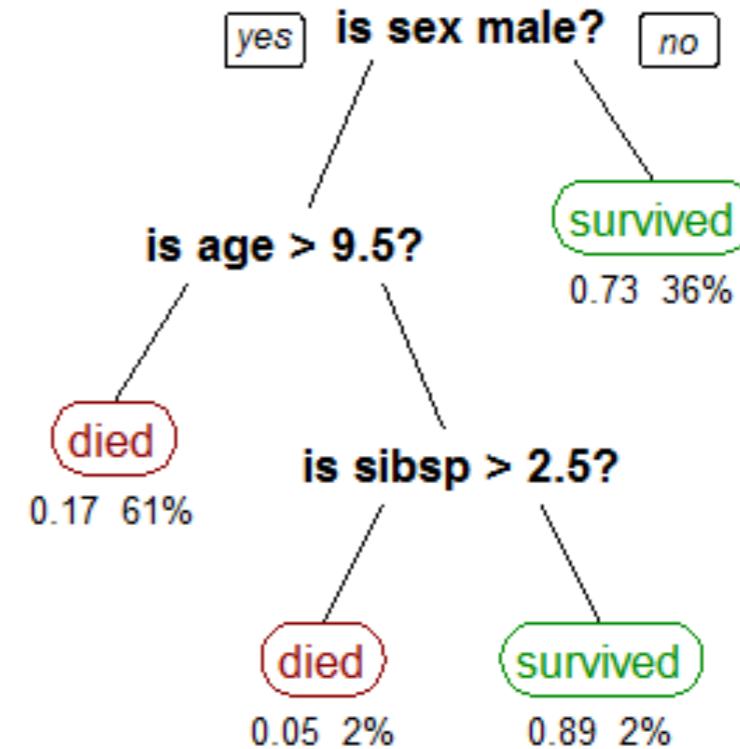
- В каждом узле записано условие
- В зависимости от ответа переходим по одной из веток
далее
- Если из узла выходят только по две ветки, то дерево –
бинарное
- В листьях записаны метки класса



РЕШАЮЩЕЕ ДЕРЕВО

5 условий построения дерева:

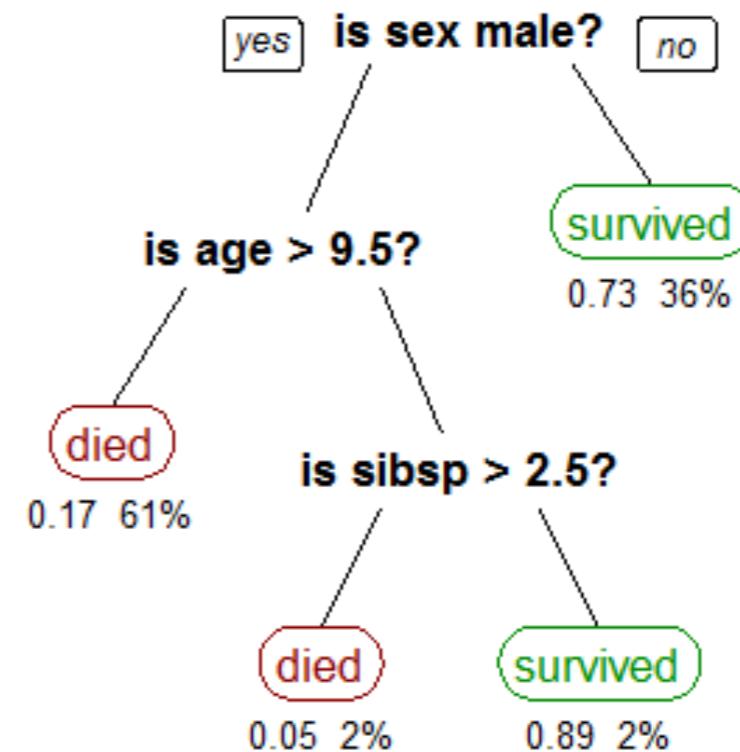
- 1) Вид условий в вершинах
- 2) Функционал качества
- 3) Критерий останова
- 4) Метод обработки пропусков
- 5) Метод стрижки



РАЗНЫЕ ПОДХОДЫ К ПОСТРОЕНИЮ ДЕРЕВА

- В зависимости от функционала качества, которые определяются через критерии информативности

- 1) Ошибка классификации
- 2) Энтропийный критерий
- 3) Критерий Джинни



СЛУЧАЙНЫЙ ЛЕС

- Генерируем подвыборку с повторениями
- Строим решающее дерево на t случайных признаках из M
- Признаки выбираются по критериям информативности
- Дерево строится до полного исчерпания подвыборки
- Дерево не подвергается стрижке

Многоклассовая классификация

ЧАСТЬ ПРОБЛЕМ РЕШЕНА

Часть классификаторов sklearn умеет работать с многоклассовым случаем, :

- KNeighborsClassifier
- RandomForestClassifier
- SVC

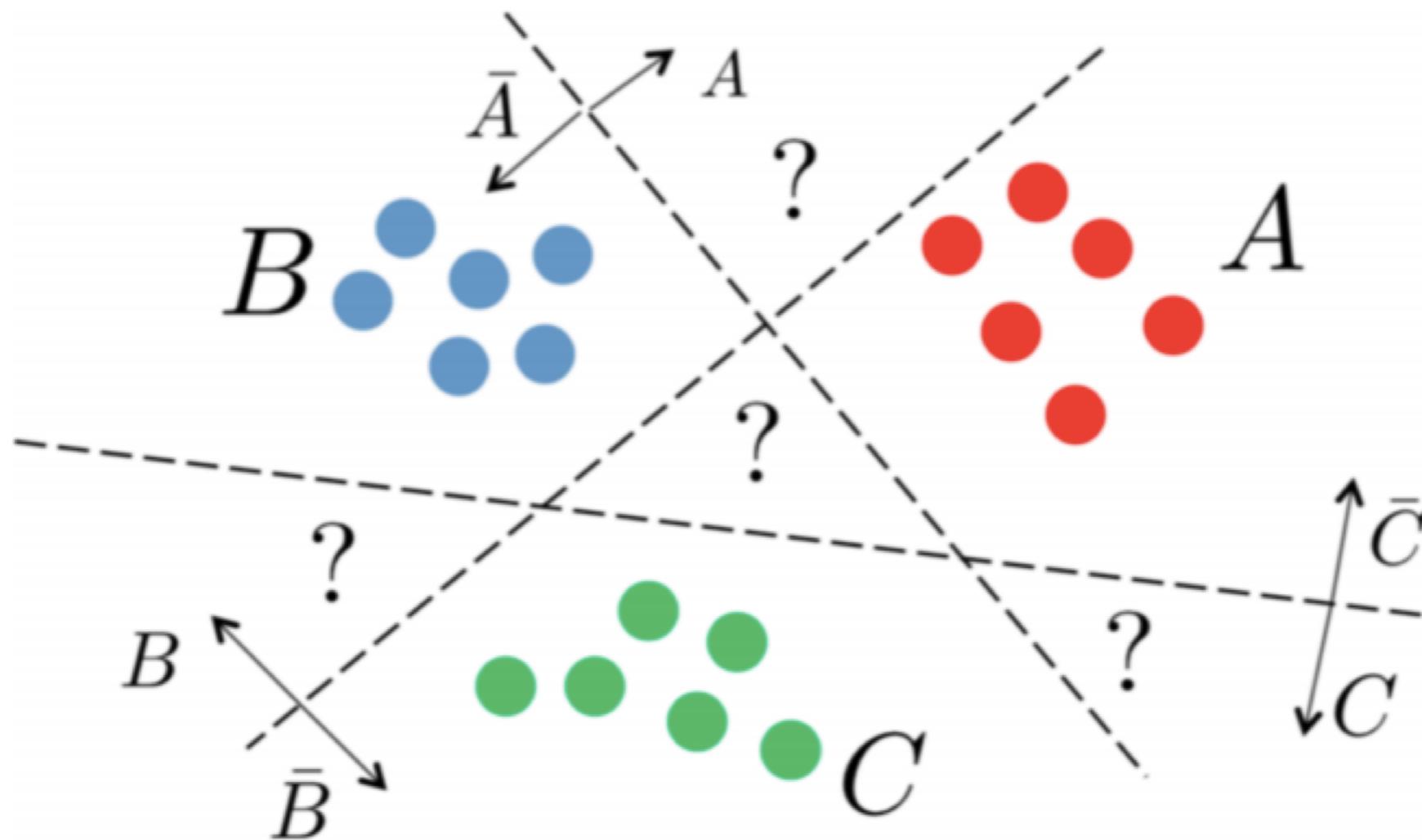
Либо:

- OneVsRestClassifier
- OneVsOneClassifier

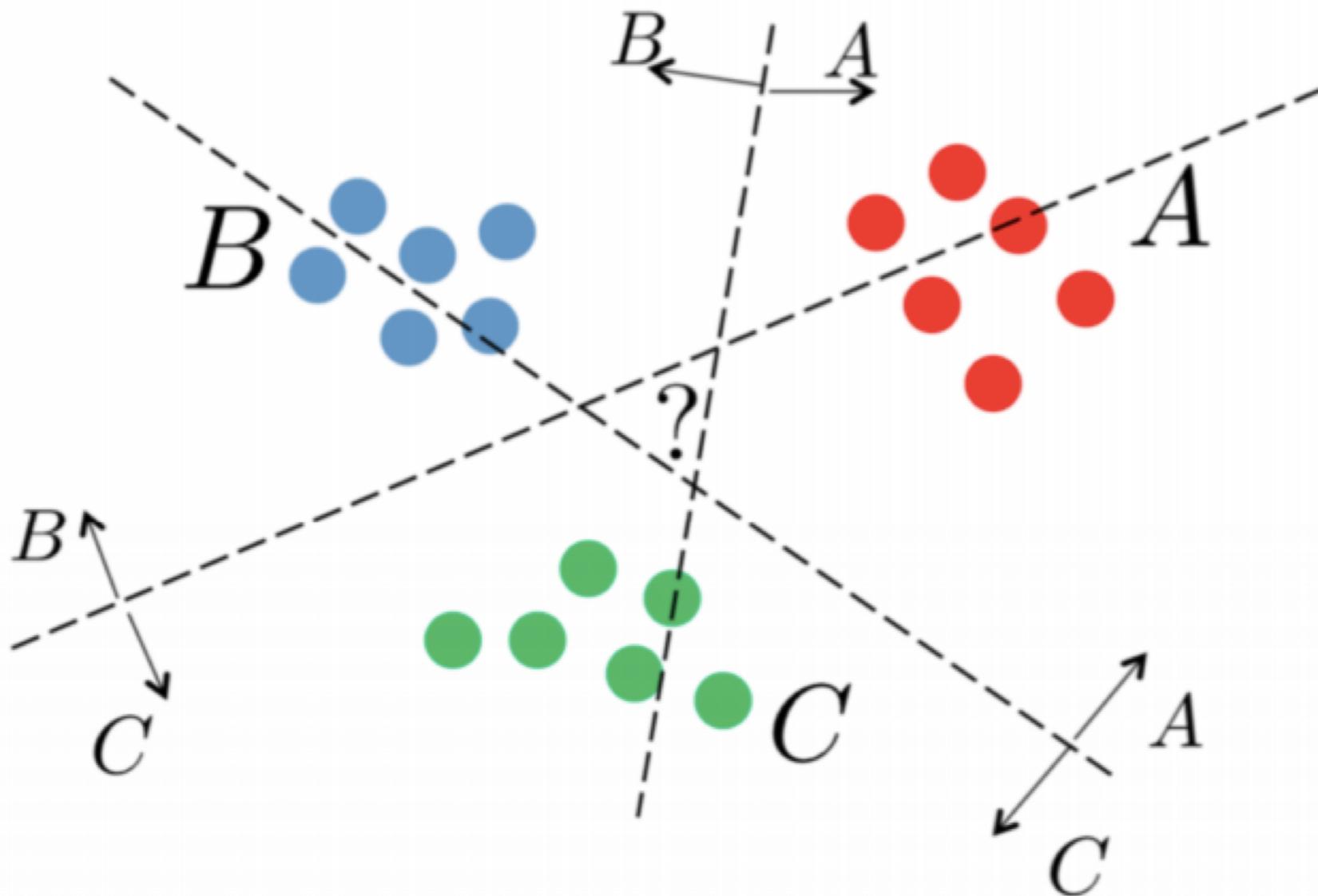
<https://scikit-learn.org/0.21/modules/multiclass.html>

ONE-VS-ONE

ONE-VS-ALL



ONE-VS-ONE





НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ