KARPOV.COURSES >>>
КОНСПЕКТ

Конспект > 14 урок >Многоклассовая классификация: one vs rest, one vs one

>Оглавление

>Задача многоклассовой классификации

Задача классификации заключается в присвоении объекту некоторого класса. В зависимости от количества классов различают бинарную классификацию и многоклассовую классификацию.

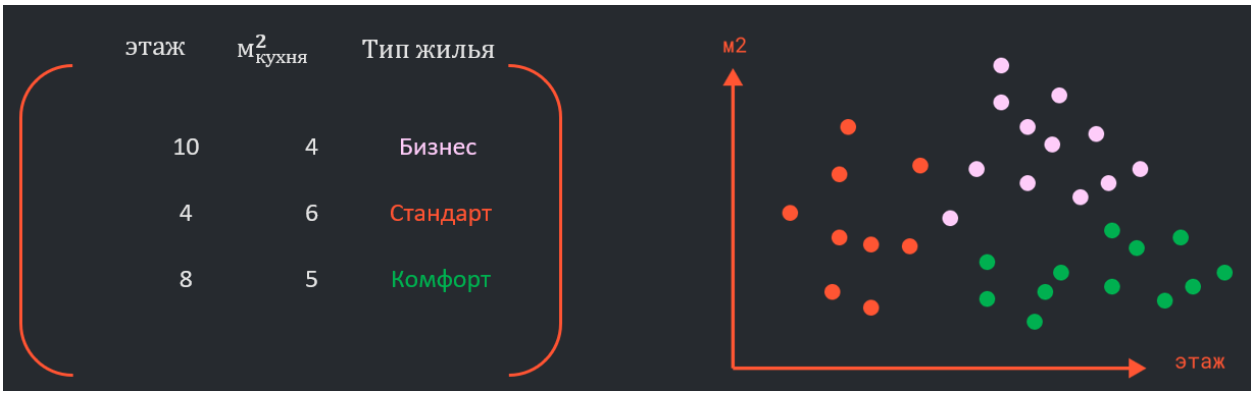
Задача многоклассовой классификации заключается в отнесении объекта к трем и более классам. Например, определение ***марки автомобиля*** по каким-то характеристикам, определение **стадии рака**, определение **возраста человека** по фотографии, определение уровня знания языка (A1, A2, B1) и т.д

Представим, что классов для предсказаний теперь не 2, а больше. Скажем, пусть их **K** штук (**K**>2). **K** различных уникальных значений, который может принимать таргет. Для решения задач многоклассовой классификации существует два основных подхода:

1. one vs rest или один против всех.
2. one vs one или один против одного.

Обратите внимание, далее в качестве примера (в том числе для визуализации), будем считать, что **K = 3.**

Пример с квартирами: представьте, что у нас есть выборка из 3 квартир. Мы знаем этаж и масштаб кухни($м^2$). И хотим по этим признакам узнать тип жилья (бизнес, стандарт и комфорт).



>Один против всех

One vs rest или one vs all

Обучим **K** различных моделей-классификаторов. Каждая модель будет ответственна за определенный класс. Выход модели под номером **i** — оценка

принадлежности объект к классу под номером **i** (т.е вероятность того, относится ли объект к классу). На нашем примере с квартирами, у нас будет 3 модели-классификатора: они будут оценивать вероятность того, что квартира принадлежит к указанному типу жилья (бизнес, стандарт или комфорт).

Метод называется один против всех потому что каждый раз, когда мы строим новый классификатор, мы сравниваем 2 класса: **i** и все остальные (бизнес или все остальные, стандарт или все остальные и т.д).

Пример

Построим модели для 3 классов: если объект относится к классу — ставим **1**, если не относится — ставим **0**. Также, выходы моделей классификации преобразуем в вероятности. Итого получится классификатор, который разбивает наши объекты на классы. Запишем все результаты в одну табличку:

этаж	$м^2_{кухня}$	Тип жилья	$a_{бизнес}(x)$	$a_{стандарт}(x)$	$a_{комфорт}(x)$	Итоговый прогноз
10	4	Бизнес	0.7	0.8	0.5	Стандарт
4	6	Стандарт	0.75	0.9	0.1	Стандарт
8	5	Комфорт	0.2	0.5	0.87	Комфорт

Теперь, мы имеем вероятность принадлежности к классу для каждого объекта. Но как из набора этих вероятностей получить итоговый ответ? Самый базовый и надёжный подход - это посмотреть на самый уверенный прогноз. То есть, посмотреть какая модель выдала наибольшую вероятность и указать соответствующий класс.

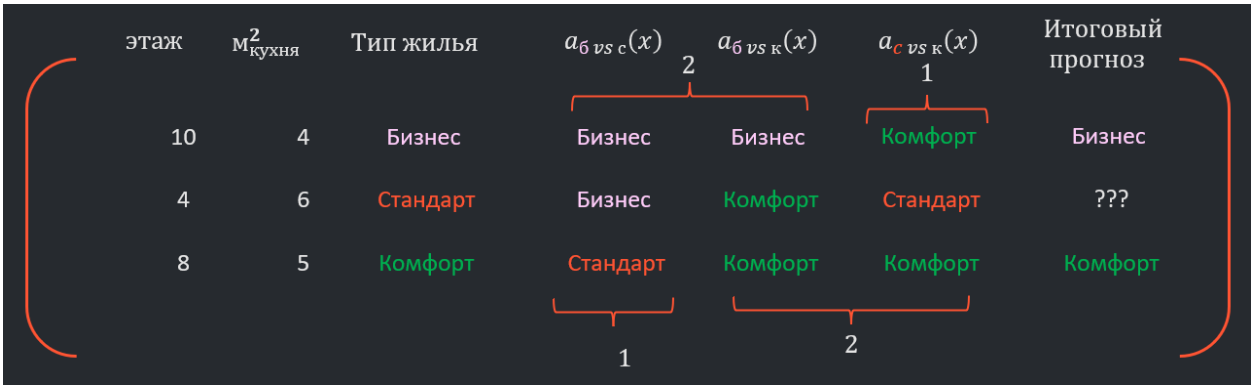
>Все против всех

Разделим выборку на множество подвыборок: в каждой будет всего лишь по 2 класса (**j** и i). Обучим $\frac{k!}{2! * (k-2)!}$ моделей-классификаторов для каждой подвыборки. Каждая модель для того или иного объекта будет отвечать на вопрос: к классу j или к классу i он принадлежит. То есть, после того как мы получили

выборки состоящие из двух классов (**j** и i), на каждой такой подвыборке мы обучаем модель, которая умеет определять к какому классу принадлежит объект (к классу j или к i). Итого, у нас получится много моделей-классификаторов, которые будут голосовать к какому классу относится объект

Пример

Разобьем выборку на 3 подвыборки: бизнес и стандарт, бизнес и комфорт, стандарт и комфорт. На каждой такой подвыборке обучим модель, которая будет сравнивать каждый класс и определит к какому классу принадлежит объект (бизнес или стандарт, бизнес или комфорт, стандарт или комфорт). И получается, что для каждой квартиры у нас будет какой-то прогноз (в каком классе больше вероятность - туда модель и отнесёт объект). Ну и чтобы определиться в прогнозе, прогоним каждый объект через модели и посмотрим за какой класс для каждого объекта модели проголосовали:



Во 2 объекте количество голосов за каждый класс совпало, что делать? Можно позаимствовать концепцию из one vs all. Нужно выбрать класс, за который модели проголосовали с большей уверенностью. То есть, для второго объекта есть какие-то прогнозы, можно в каждую из 3 моделей засунуть наш проблемный объект и получить какую-то вероятность:

этаж	$м^2_{кухня}$	Тип жилья	$a_{б\ vs\ с}(x)$	$a_{б\ vs\ к}(x)$	$a_{с\ vs\ к}(x)$	Итоговый прогноз
4	6	Стандарт	Бизнес	Комфорт	Стандарт	???
			$P(y_i = б\ \|x)$	$P(y_i = к\ \|x)$	$P(y_i = с\ \|x)$	
4	6	Стандарт	0.9	0.94	0.99	Стандарт

Вероятность у 3 модели оказался самым большим, поэтому для итогового ответа выберем самый уверенный классификатор (стандарт).

>Сравнение подходов

one vs all

Преимущества:

1. Необходимо построить всего лишь **K** моделей. Данные метод достаточно компактен, он тренирует меньше классификаторов, а значит, быстрее и, следовательно, обычно предпочтительнее.

Недостатки:

1. Зависимость от масштаба предсказаний модели.
2. В данном подходе часто встречается дисбаланс в количестве объектов в классе который сравнивается с другими. Например, если у вас есть 100 квартир с типом бизнес, 100 стандартных и 100 комфортных, то используя подход **ona vs all** вы будете сравнивать 100 и 200 квартир.
3. Если какой-то алгоритм оказался очень уверен на всех объектах - его нужно как-то калибровать, иначе одни модели будут перевешивать другие. Т.е необходимо калибровать вероятности друг на друга

all vs all

Преимущества:

1. Механизм голосования не требует калибровки выходов моделей

Недостатки:

1. Моделей оказывается очень много, особенно при больших **K**

>Метрики качества

Матрица ошибок (Confusion matrix) - это таблица, которая позволяет узнать эффективность алгоритма классификации путем сравнения прогнозируемого значения целевой переменной с ее фактическим значением. Есть 4 величины:

- **True Positive** — количество объектов, верно отнесенных к классу 1;
- **False Positive** — количество объектов, неверно отнесенных к классу 1;
- **False Negative** — количество объектов, неверно отнесенных к классу -1;
- **True Negative** — количество объектов, верно отнесенных к классу -1.

Эти величины и образуют матрицу ошибок:

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

Для расчета матрицы ошибок, точности и полноты в библиотеке sklearn.metrics существуют методы confusion_matrix, precision_score, recall_score.

Микро усреднение

Мы построили модель используя один из подходов, модель состоит из набора базовых алгоритмов. Тогда, для каждого из алгоритмов мы можем посчитать матрицу ошибок, найти средние значения и на их основе посчитать итоговые метрики. Например точность:

$$precision = \frac{TP_{\text{ср}}}{TP_{\text{ср}} + FP_{\text{ср}}}$$

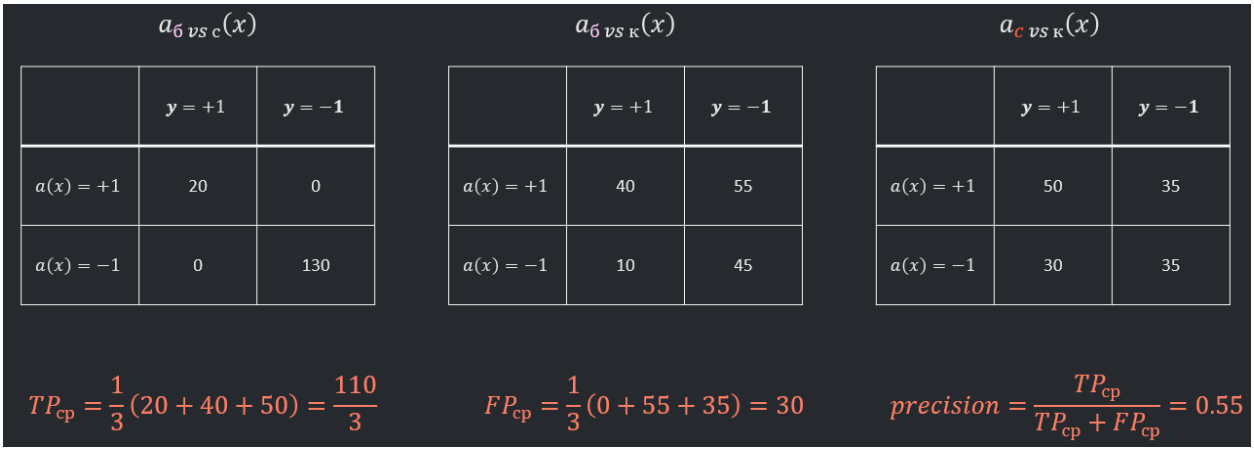
Или полноту:

$$recall = \frac{TP_{\text{ср}}}{TP_{\text{ср}} + FN_{\text{ср}}}$$

То есть, микро усреднение состоит в том, что в начале базовая ошибка каждого алгоритма усредняется и подставляется в итоговую, агрегированную метрику.

Пример

Допустим, мы решали задачу с помощью подхода all vs all, у нас получилось 3 модели и для каждой модели мы визуализировали матрицу ошибок. Затем усреднили все показатели и добавили их в агрегированную метрику (допустим, в `precision`):



И такую метрику можно использовать для оценки модели классификации.

Макро усреднение

Также, для каждого классификатора посчитаем идивидуальные TP, TN, FP, FN. Вычислим индивидуальные итоговые метрики, например `precision`. И уже итоговые индивидуальные метрики усредним

$$precision_{\text{ср}} = \frac{1}{n}\sum precision_i$$

Пример

Допустим, мы решали задачу с помощью подхода all vs all, у нас получилось 3 модели и для каждой модели мы визуализировали матрицу ошибок. Затем

считаем `precision` для каждой из моделей. И чтобы получить итоговую метрику для алгоритма можно усреднить `precision` всех моделей

$a_{б\ vs\ с}(x)$

	$y = +1$	$y = -1$
$a(x) = +1$	20	0
$a(x) = -1$	0	130

$$precision_1 = \frac{20}{20 + 0} = 1$$

$a_{б\ vs\ к}(x)$

	$y = +1$	$y = -1$
$a(x) = +1$	40	55
$a(x) = -1$	10	45

$$precision_2 = \frac{40}{40 + 55} = \frac{8}{19}$$

$a_{с\ vs\ к}(x)$

	$y = +1$	$y = -1$
$a(x) = +1$	50	35
$a(x) = -1$	30	35

$$precision_3 = \frac{50}{50 + 35} = \frac{10}{17}$$

Итоговый, усредненный `precision` будет равен 0.67.

Сравнение подходов

Особенность **микро усреднения** заключается в том, что при формировании итоговой метрики вклад каждого класса пропорционален его размеру. Это логично: чем больше тот или иной класс в выборке, тем больше **TP** и **FP** будет у данного класса. Поэтому метрику, которую получаем используя микро усреднение стоит оптимизировать только если можем позволить себе ошибиться на маленьких классах.

То есть, мы можем обучить модель на 1000 классов, из них 100 окажутся маленькими, мы на них будем сильно ошибаться и данная метрика не заметит этих ошибок, но будет давать хорошие прогнозы на больших классов.

А при **макро усреднении** мы для каждой модели считаем индивидуальные метрики, которые заранее не чувствительны к размеру классов. То есть, используемые индивидуально метрики не чувствительны к дисбалансу. Каждый класс, независимо от размера, одинаково важен для среднего замера.