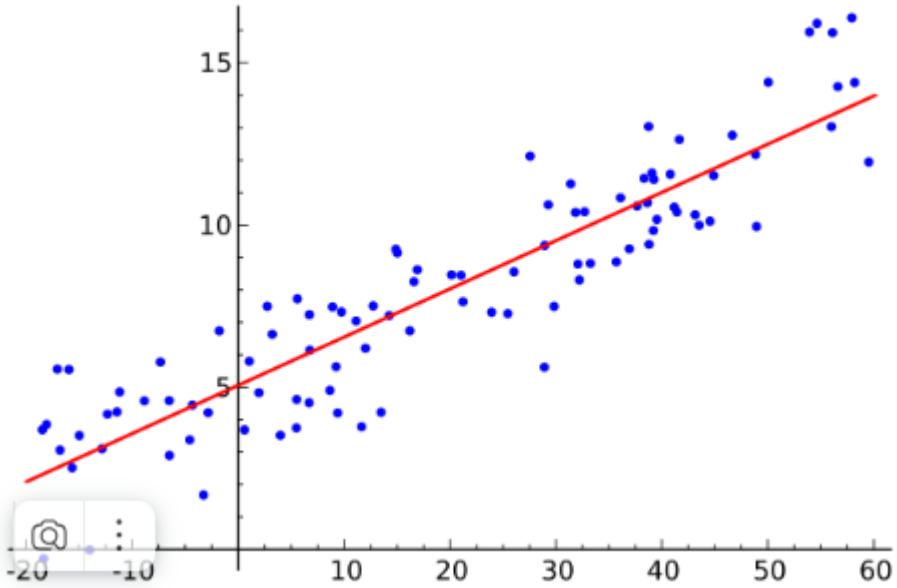
Методы для задач классификации

Линейная регрессия делает предсказание по формуле:

$$a(x) = w_0 + w_1 x_1 + \ldots + w_d x_d = (w, x)$$



Если немного модифицировать эту формулу, то с её помощью можно решать задачу *бинарной классификации*, чтобы формула для $a(x)$ выдавала одно из двух возможных значений, то есть классы (например, *+1* или *-1*).

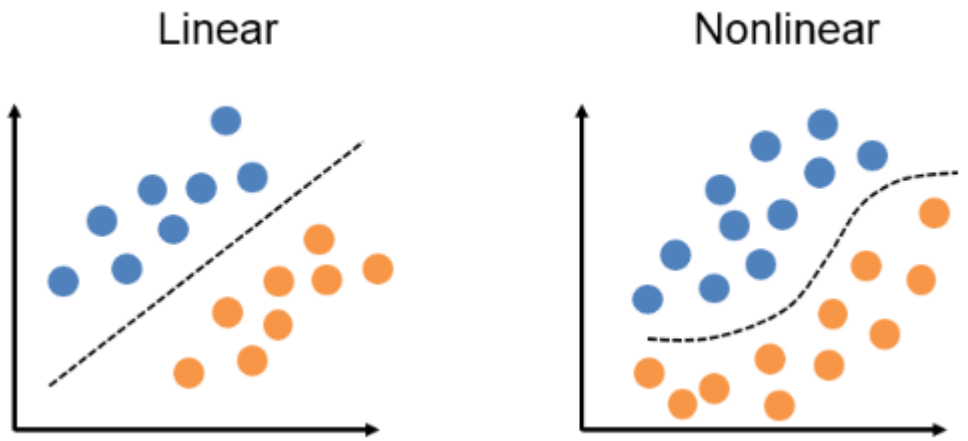
Формула для *бинарного линейного классификатора*:

$$a(x) = sign(w_0 + w_1 x_1 + \ldots + w_d x_d) = sign((w, x)),$$

- если $(w,x)>0$, то $a(x)=sign((w,x))=+1$, то есть объект принадлежит положительному классу
- если $(w,x)<0$, то $a(x)=sign((w,x))=-1$, то есть объект принадлежит отрицательному классу

Классификаторы бывают линейными и нелинейными.

Линейный классификатор - это классификатор, разделяющий классы линейной поверхностью (гиперплоскостью). Если же разделяющая поверхность нелинейная, то это уже нелинейный классификатор.



Классификатор, делающий предсказания по формуле $a(x)=sign(w,x)$, линейный.

Почему???

Знаем, что если $sign(w,x)>0$, то объект относится к классу +1,

а если $sign(w,x)<0$, то к классу −1.

То есть разделяющая граница между классами задается уравнением

$$(w,x) = w_0 + w_1 x_1 + \ldots + w_d x_d = 0,$$

а это гиперплоскость, то есть линейная поверхность

В частности, если у объекта два признака $x1$ и $x2$, то разделяющая граница – прямая линия

Если классы обозначены не как +1 и −1, а как 1 и 0, то иногда используют другую форму записи линейного классификатора:

$$a(x) = [(w, x) > 0].$$

Здесь $[(w,x)>0]$ - индикатор события $(w,x)>0$.
Он равен 1, если событие выполняется, т.е. если $(w,x)>0$,
и 0 иначе.

Почти каждой модели машинного обучения соответствуют **две** функции:

- формула, по которой модель делает предсказания
- функция потерь, которая минимизируется при обучении модели.

Линейный классификатор делает предсказания по формуле $a(x)=sign(w,x)$.

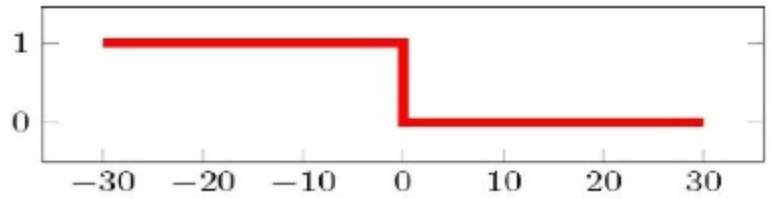
Остается понять, какую функцию потерь нужно минимизировать, чтобы найти оптимальные веса.

Самый разумный способ считать ошибку в задаче классификации - это посчитать долю ошибочных предсказаний модели, то есть

$$\frac{1}{l}\sum_{i=1}^{l}[a(x_i) \neq y_i]$$

где $[a(xi) \neq yi]$=1, если предсказанный класс не совпал с правильным, и 0 иначе.

Так выглядит пороговая функция потерь:



эту функцию можно было бы использовать как функцию потерь (она называется ***пороговой функцией потерь***), но с такой функцией связано одно большое неудобство.

Вместо неё обычно берут другую функцию потерь $L(w)$, которая

ограничивает сверху пороговую функцию потерь: $L(w) \leq L(w)$

является непрерывной (а лучше гладкой)

Тогда при решении задачи минимизации (то есть при обучении модели) функции потерь $L(w)$

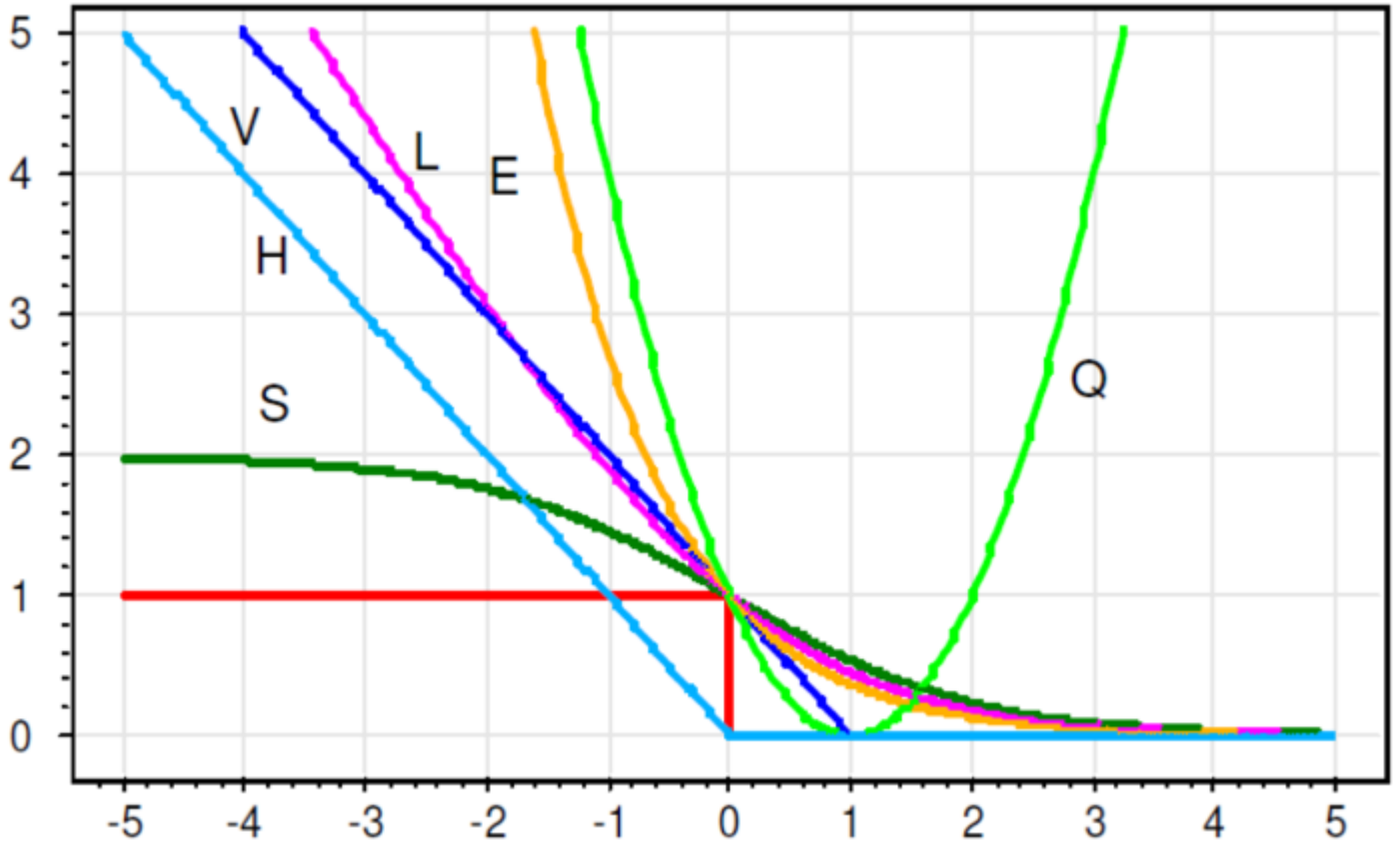
$L(w) \leq \rightarrow min$

автоматически минимизируется исходная, пороговая функция потерь $L(w)$

для решения задачи классификации нам нужно выбрать некоторую гладкую (или хотя бы непрерывную) функцию потерь $\tilde{L}(w)$, ограничивающую сверху пороговую функцию потерь. Таких функций $\tilde{L}(w)$ существует очень много, и ***от выбора конкретной функции будет зависеть то, какие в итоге получатся веса у модели после минимизации. Собственно, сама модель определяется тем, какая функция потерь будет выбрана!***

Существует несколько широко используемых функций потерь $\tilde{L}(w)$ в задаче классификации. Каждая из них задает свой классификатор, со своим названием и особенностями.

Различные функции потерь, применяемые в задачах классификации



Метод логистической регрессии

При решении задачи классификации можно предсказывать не только классы, но и их вероятности.

Задача классификации, в которой модель предсказывает только классы - это **жесткая классификация.**

Если же классификатор предсказывает вероятности классов - это **мягкая классификация.**

Логистическая регрессия - как раз такой классификатор, который умеет предсказывать не только классы объектов, но и их вероятности.

Формула для предсказания классов у линейного классификатора

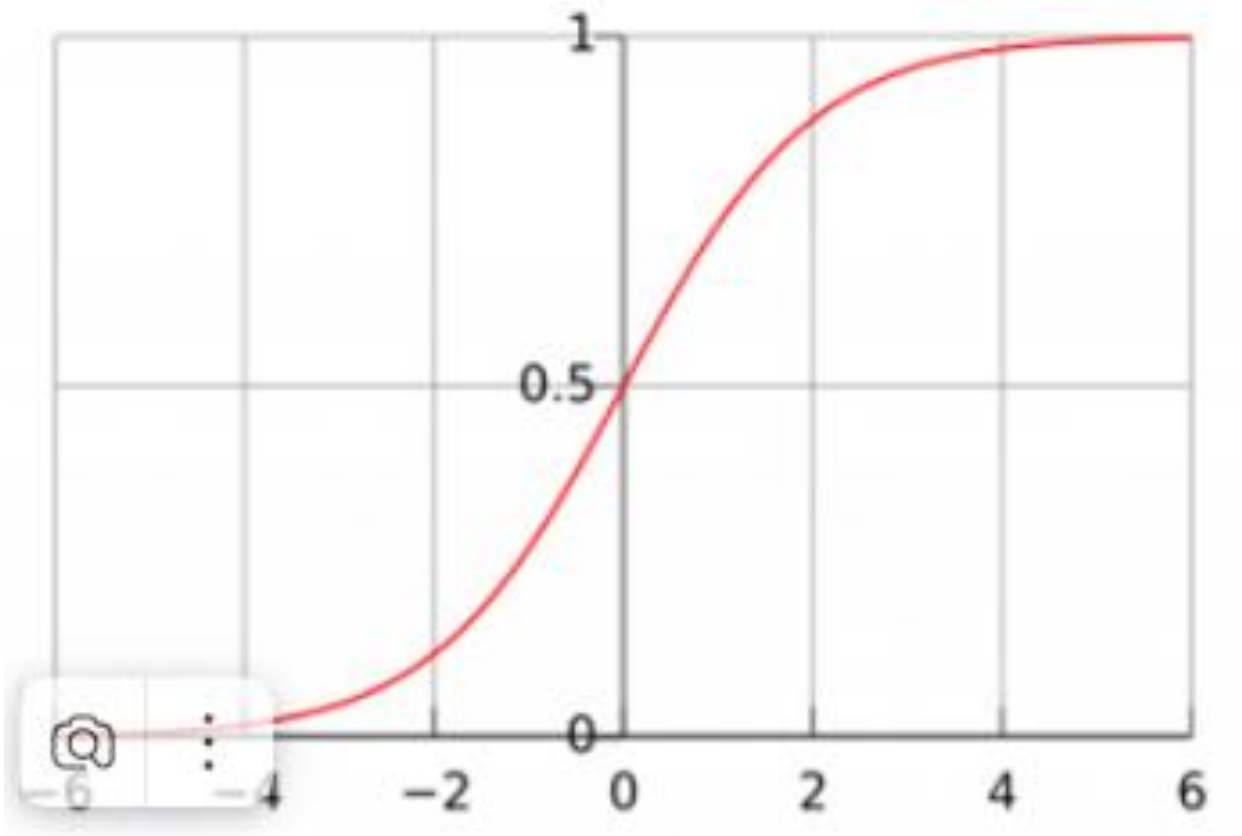
$$a(x) = sign(w, x)$$

Нужно видоизменить эту формулу, чтобы на выходе получать вероятность.

В логистической регрессии для предсказания вероятности используется **сигмоида:**

$$a(x)=\sigma(w,x)$$

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

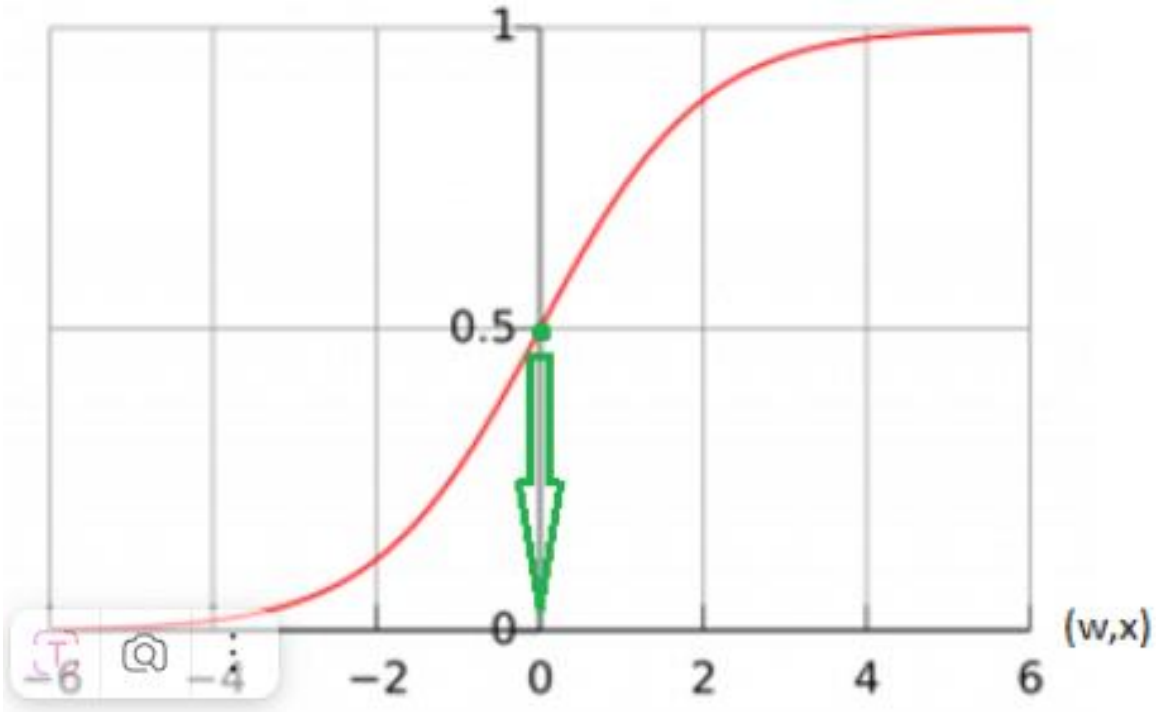


То есть формула для предсказания логистической регрессии имеет вид

$$a(x) = \sigma(w, x) = \frac{1}{1 + e^{-(w,x)}}.$$

По оси абсцисс (горизонтальная ось) на графике отложено значение скалярного произведения (w,x), а по оси ординат - предсказанная моделью вероятность того, что объект x принадлежит к классу +1.

Является ли логистическая регрессия *линейным* классификатором?

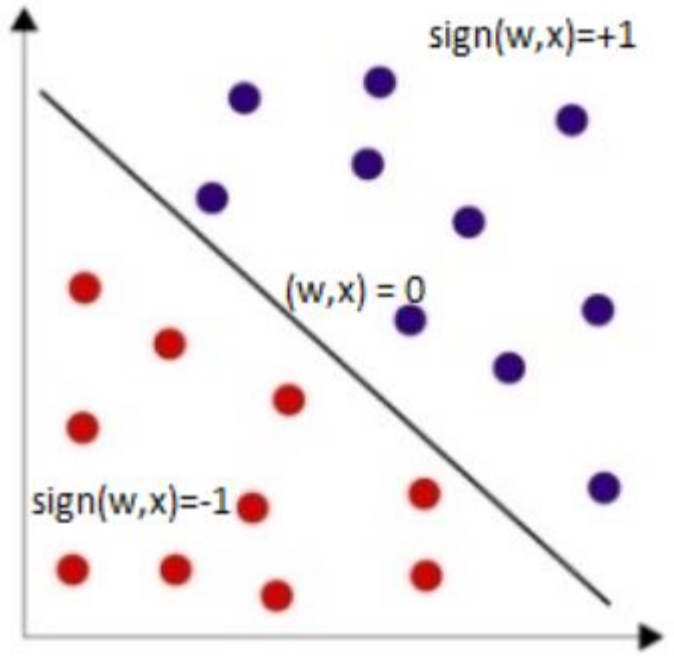


$a(x)=\sigma(w,x)>0.5 \Rightarrow$ класс+1

$a(x)=\sigma(w,x)< 0.5 \Rightarrow$ класс−1

Получаем, что разделяющая граница задается уравнением $\sigma(w,x)=0.5$. Посмотрим на график.
Значение $y=0.5$ соответствует точке $(w,x)=0$ на оси абсцисс.

Получается, что разделяющая поверхность задается уравнением (w,x)=0 или же $w0+w1x1+...+wdxd$=0, а это линейная поверхность (более точно - гиперплоскость).



Логистическая регрессия является **линейным классификатором.**

Функция потерь в логистической регрессии

К***акая функция потерь используется при обучении модели***?

Так как логистическая регрессия предсказывает вероятность, то есть непрерывную величину, то можно попробовать взять функцию потерь регрессии, например, среднеквадратичную ошибку (MSE). Тогда задача оптимизации для логистической регрессии будет выглядеть так:

$$MSE = \frac{1}{l}\sum_{i=1}^{l}(a(x_i) - y_i)^2 = \frac{1}{l}\sum_{i=1}^{l}(\frac{1}{1+e^{-(w,x_i)}} - y_i)^2 \to \min_w$$

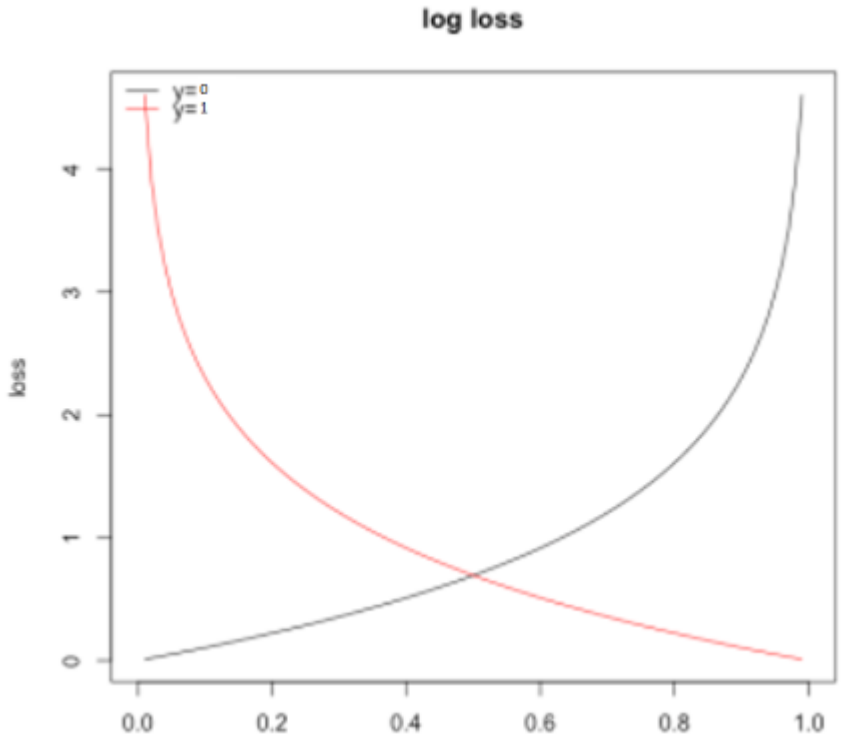
- полученная функция *невыпуклая* (то есть может иметь несколько локальных минимумов),
- MSE не очень разумно выдает штрафы

MSE не очень хорошо подходит для обучения логистической регрессии

Для обучения логистической регрессии хорошо подходит ***логистическая функция потерь (или же log-loss)***.

Она вычисляет ошибку на объекте следующим образом:

$$\text{log-loss} = -(y \cdot log(a(x)) + (1 - y) \cdot log(1 - a(x)))$$



для положительного класса
это $-log(a(x))$- **красная** кривая

для отрицательного класса
это $-log(1-a(x))$ - **черная** кривая

Логистическая функция потерь очень хорошо подходит для обучения логистической регрессии:

- она выпуклая, то есть у неё один минимум, и градиентный спуск его найдет
- она корретно штрафует ошибки
- log-loss обладает замечательным свойством: **модель, обученная при помощи минимизации log-loss, предсказывает корректные математические вероятности!**

В результате:

- Логистическая регрессия делает предсказания по формуле

$$a(x) = \sigma(w, x) = \frac{1}{1+e^{-(w,x)}},$$

и это предсказание означает вероятность того, что объект x относится к классу +1

- При обучении логистической регрессии минимизируется функция потерь log-loss:

$$\text{log-loss} = -\frac{1}{l}\sum_{i=1}^{l} y_i \cdot log(a(x_i)) + (1 - y_i) \cdot log(1 - a(x_i))$$

- Минимизация этой функции потерь гарантирует то, что обученный алгоритм будет выдавать не просто числа из отрезка [0;1], а ожидаемые вероятности

Метрики классификации

- ***Accuracy*** - это доля правильных ответов модели

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

- где $a(xi)$ - предсказанный класс на объекте xi
- yi - правильный ответ (класс) на объекте xi
- $[a(xi)=yi]$ - индикатор, то есть величина, равная 1, если $a(xi)=yi$, и 0 иначе

Задача:

Модель на 100 тестовых клиентах правильно угадала ответ 72 раза (то есть для 72 клиентов из 100 модель верно предсказала, вернет человек кредит или не вернет).

Чему равна метрика на этих данных?

Пусть в тренировочных данных 1000 транзакций, и 50 из них мошеннические.

Возьмем модель, которая для всех транзакций предсказывает, что они легальные: $a(x)$=0

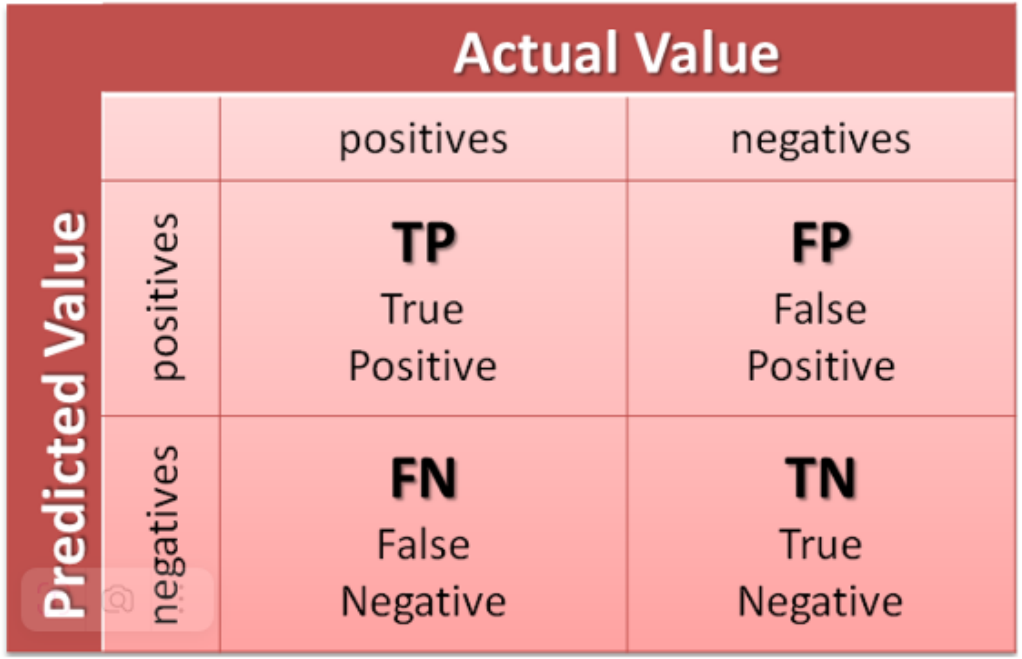
Какая у неё *accuracy*?

Пусть решается задача скоринга на сбалансированных классах (то есть в тренировочных данных поровну клиентов, которые вернули кредит и которые не вернули кредит).

Пусть в тестовых данных в задаче скоринга 1000 клиентов, и accuracy = 0.8.

Можно ли сказать, что модель имеет хорошее качество?

Матрица ошибок



- *True Positive* (***TP***) –
- количество объектов положительного класса, предсказанных моделью как положительные (верные предсказания)
- *False Positive* (***FP***) - количество объектов отрицательного класса, предсказанных моделью как положительные (ошибки модели)
- *False Negative* (***FN***) - количество объектов положительного класса, предсказанных моделью как отрицательные (ошибки модели)
- *True Negative* (***TN***) - количество объектов отрицательного класса, предсказанных моделью как отрицательные (верные предсказания)

Задача:

Пусть в тестовых данных 1000 объектов, из них 500 положительных и 500 отрицательных.

Модель имеет accuracy = 0.8, при этом ошибается на 150 объектах положительного класса.

Сопоставьте элементы матрицы ошибок с числами

Существуют две удобные метрики, с помощью которых можно измерить качество модели, используя матрицу ошибок:

- **точность (*precision*)**
- **полнота (*recall*)**

Пусть решается задача скоринга - модель предсказывает, выдавать клиенту кредит или нет.

Сравним между собой две модели. Даны их матрицы ошибок на тестовых данных (данные сбалансированы: всего 200 клиентов, по 100 клиентов каждого класса):

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	80	20
$a(x) = -1$ Не получили кредит	20	80

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	48	2
$a(x) = -1$ Не получили кредит	52	98

Какая модель лучше?

Precision (точность)

Оценим, насколько точна модель среди тех клиентов, которым она выдала кредит?
То есть посчитаем долю правильных ответов модели среди всех её положительных предсказаний - это и есть метрика *precision*:

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

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	80	20
$a(x) = -1$ Не получили кредит	20	80

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	48	2
$a(x) = -1$ Не получили кредит	52	98

Recall (полнота)

Как много кредитоспособных клиентов (тех, кто вернет кредит) находят модели?
Среди всех клиентов, кто вернет кредит, посчитаем долю клиентов, которым модель выдала кредит - это метрика *recall*:

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

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	80	20
$a(x) = -1$ Не получили кредит	20	80

	$y = 1$ Могут вернуть	$y = -1$ Не могут вернуть
$a(x) = 1$ Получили кредит	48	2
$a(x) = -1$ Не получили кредит	52	98

Нельзя однозначно ответить, какая из этих двух моделей лучше. Возможно, банку, выдающему кредиты, важно выдавать кредиты только тем, кто их вернёт, и не выдавать кредиты тем, кто не вернет - тогда нужна высокая точность (*precision*). А если же банк хочет выдать кредиты наибольшему количеству клиентов, кто теоретически может вернуть кредит - максимизируется полнота (*recall*).

Так что метрика выбирается, исходя из требований заказчика!

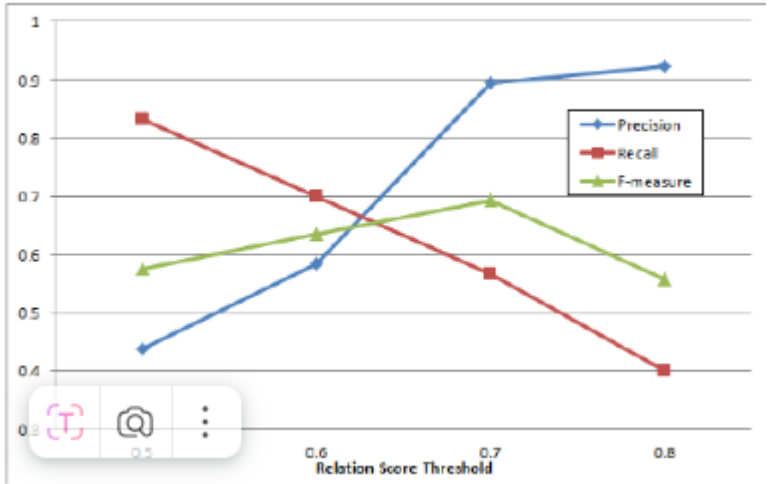
Иногда важно, чтобы и точность, и полнота были побольше, при этом нет отдельных требований к точности и полноте. Тогда можно использовать *accuracy*.

f1-score

В случае дисбаланса классов измеряют некоторую усредненную по точности и полноте величину.

$$f1 = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

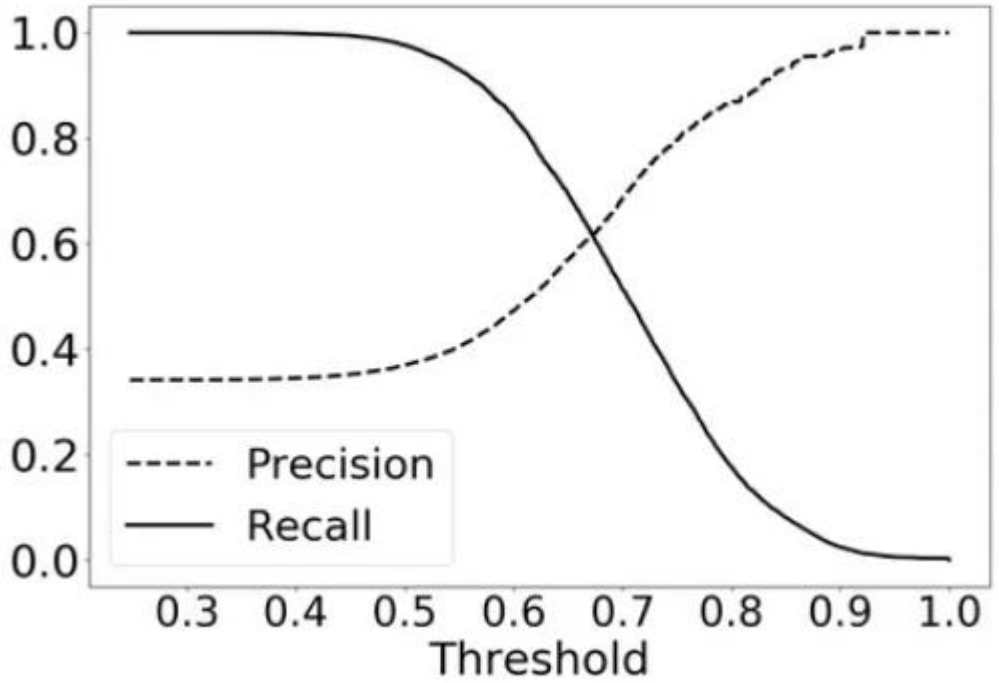
F1-score - это среднее гармоническое точности и полноты



Улучшение *precision* и *recall* (подбор порога)

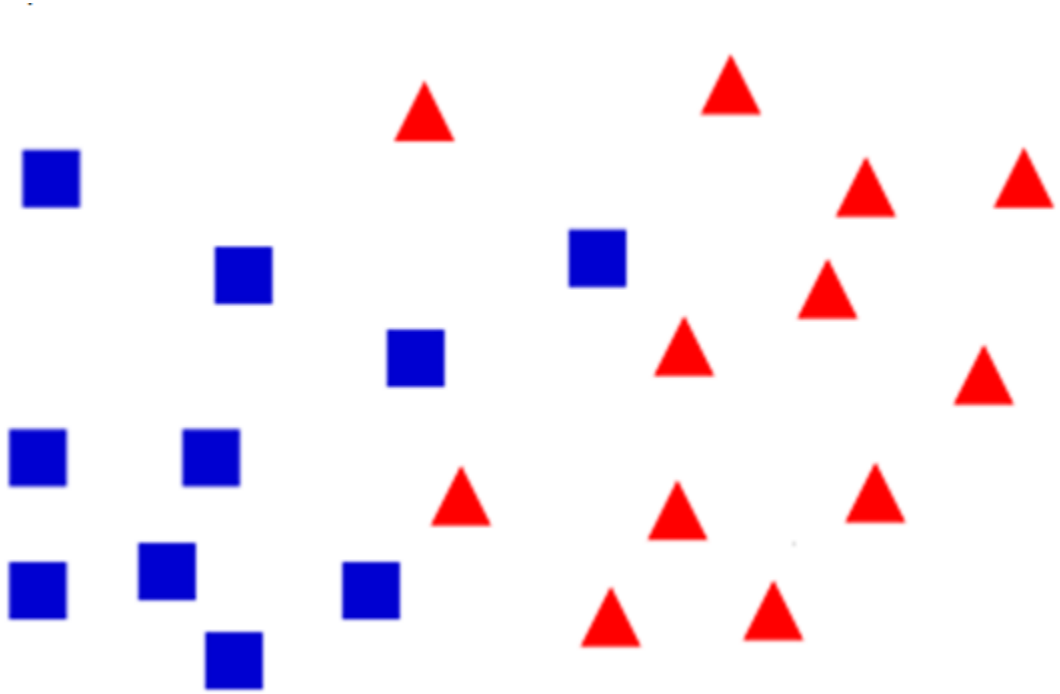
На *precision* и *recall* (а значит, и на *f1*) можно влиять путём подбора порога перевода вероятности в классы.

Подбором порога нельзя увеличить и точность, и полноту, они ведут себя обратным образом. Но можно добиться увеличения более важной для задачи метрики.



Метод К ближайших соседей (KNN)

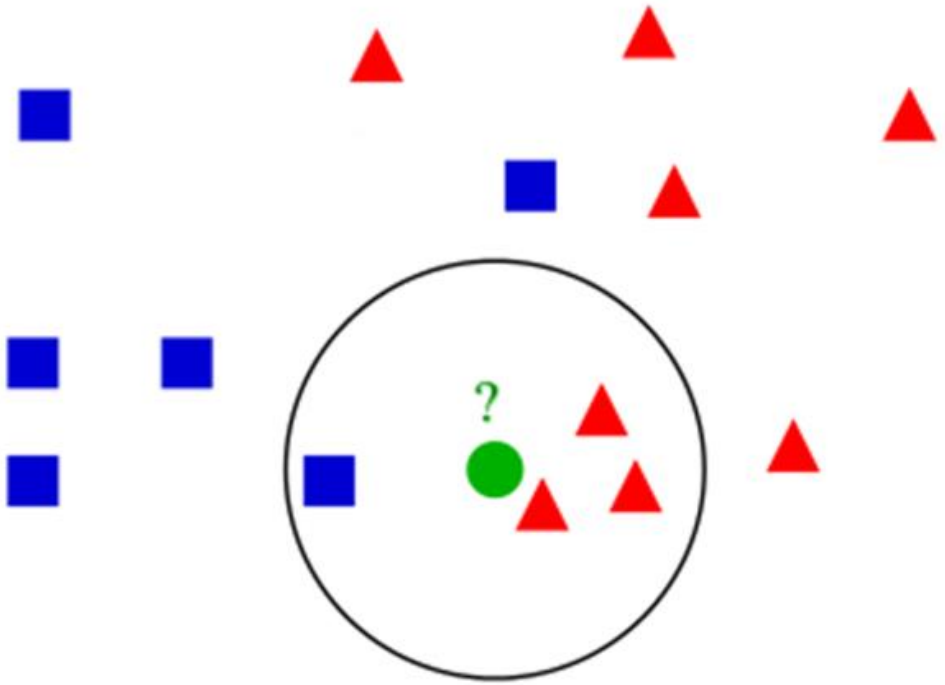
Идея метода простая, она называется ***гипотезой компактности***: схожие объекты находятся близко друг к другу в пространстве признаков.



У метода есть гиперпараметр k - *число соседей*.

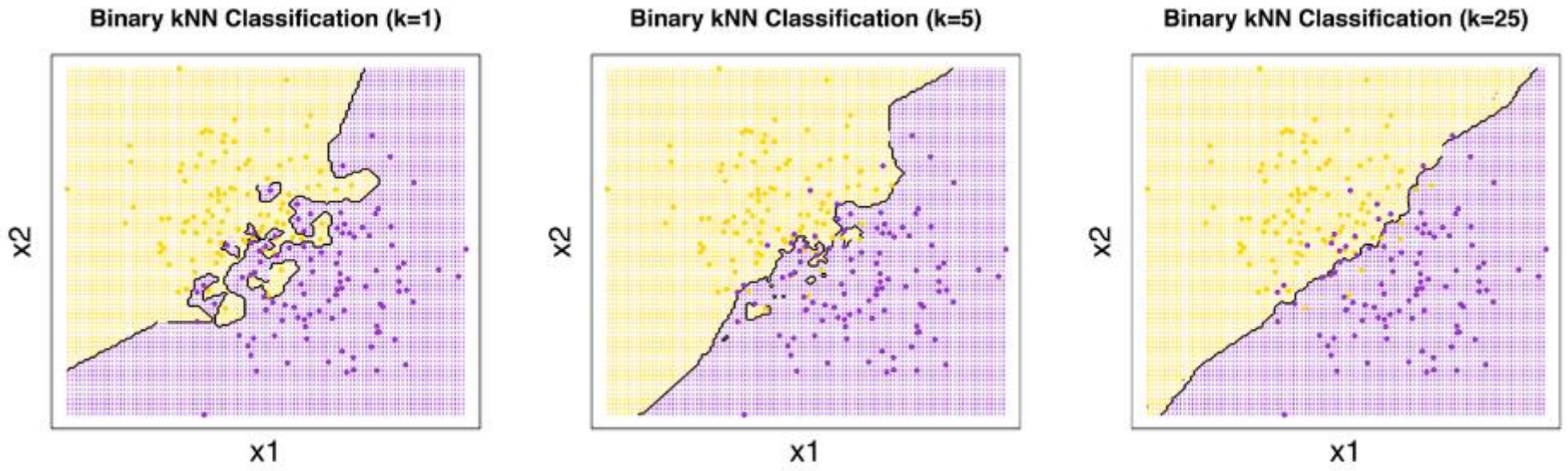
Чтобы определить, к какому классу относится объект, нужно:

- вычислить расстояние от объекта до каждого объекта выборки;
- выбрать k объектов выборки с наименьшим расстоянием (k ближайших соседей);
- класс искомого объекта - это наиболее часто встречающийся класс среди k ближайших соседей.



Алгоритм определения классов очень простой. На самом деле ***у метода нет фазы обучения***, потому что нет параметров, которые подбираются в процессе обучения по выборке. В этом смысле метод очень простой.

Несмотря на свою простоту, метод легко переобучается. Например, если взять число соседей очень маленьким (k=2 или k=3), метод будет делать предсказания только по двум или трем самым близким точкам, и поэтому сильно подгонится под данные. В этом можно убедиться, посмотрев на разделяющую поверхность метода при разных значениях k:



В методе ближайших соседей вычисляем расстояния между объектами. Способов вычислить расстояние очень много, каждый из них задается своей формулой (метрикой). Каждая метрика больше подходит для своего типа задач.

Наиболее используемые метрики:

1. **Евклидова метрика** - классический способ измерить расстояние между объектами. Пусть объекты a и b имеют координаты $a = (x_1, y_1), b = (x_2, y_2)$.

Тогда евклидово расстояние между ними

$$\rho(a, b) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

2. Манхеттенское расстояние - другой способ посчитать расстояние между двумя точками:

$$\rho(a,b) = |x_1 - x_2| + |y_1 - y_2|$$

3. **Расстояние Хемминга** - это число различных позиций в координатах двух векторов

и т.д.

Перед применением KNN необходимо привести данные к одному масштабу!

Многоклассовая классификация

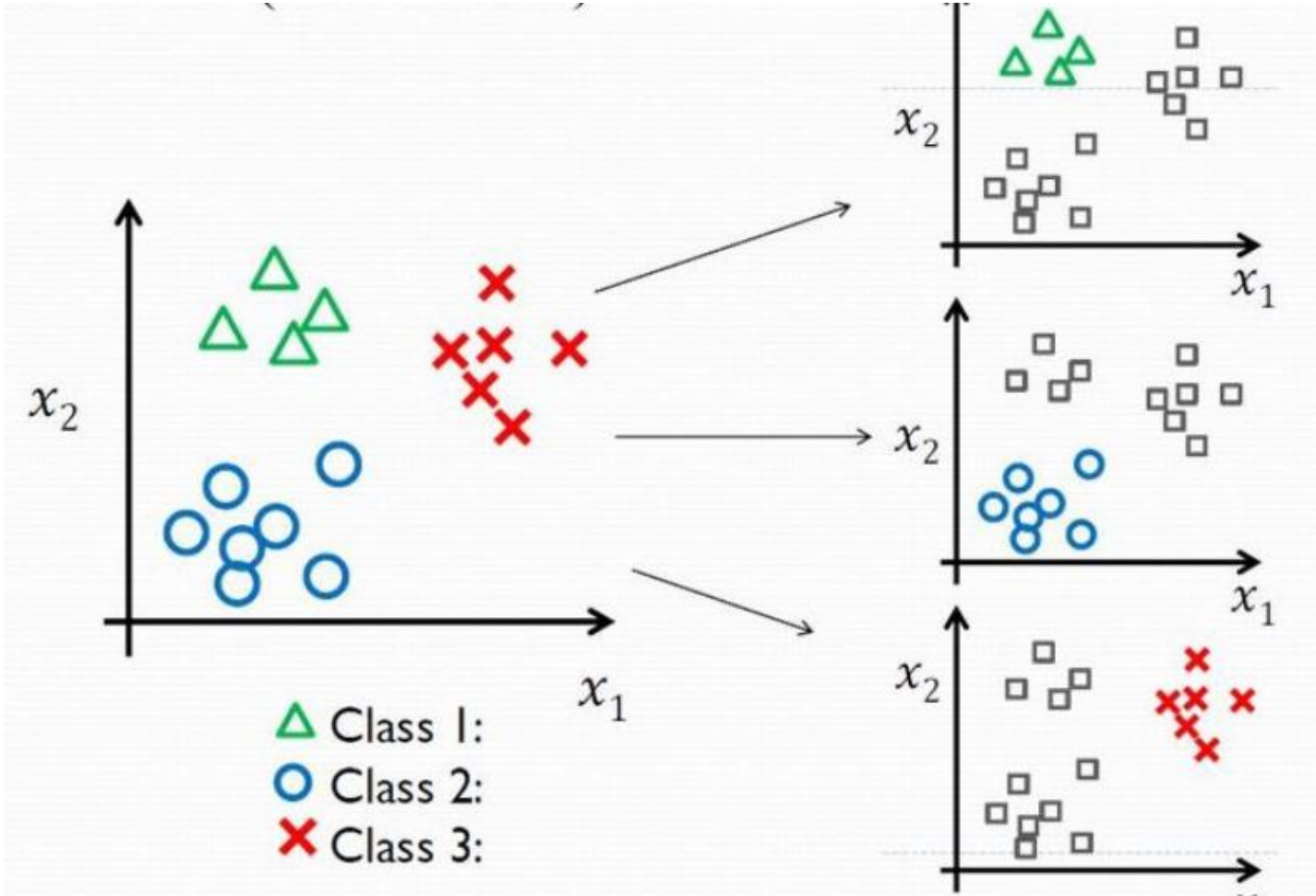
Multiclass-классификация - задача, в которой целевая переменная представляет собой один из нескольких классов (где классов может быть больше, чем два)

Задачи многоклассовой (multiclass) классификации можно свести к серии бинарных задач. Два основных подхода:

- *one-versus-one (OVO)*
- *one-versus-rest (OVR)*

One-versus-rest (OVR) подход

Подход сводит задачу к серии бинарных классификаторов, где i-й классификатор определяет, относится объект к классу i или не относится (два варианта, то есть бинарный классификатор).



One-versus-one (OVO) подход

В этом подходе каждый бинарный классификатор пытается отличить, принадлежит объект классу i или классу j. Каждый такой классификатор обучается только на объектах i-го и j-го классов.

У каждого из подходов есть особенности, связанные со скоростью обучения:

Если выбираем подход one-versus-all, то при большом количестве объектов в данных обучение будет долгим, ведь нужно обучить несколько бинарных классификаторов на большом датасете.

В случае one-versus-one каждый бинарный классификатор (с номером ij) обучается только на части датасета, состоящей из классов i и j.

Если же классов в задаче много, то подход one-versus-one будет работать долго, так как он сводится к обучению бинарных классификаторов на каждой паре классов, то есть, если в задаче N классов, то необходимо обучить $N(N-1)/2$ бинарных классификаторов

Метрики качества многоклассовой классификации

В случае мультикласса также можно построить матрицу ошибок.

Например, в задаче с тремя классами может получиться следующая матрица

		True/Actual		
		Cat ()	Fish ()	Hen ()
Predicted	Cat ()	4	6	3
	Fish ()	1	2	0
	Hen ()	1	2	6

Для вычисления точности и полноты в этом случае существует несколько подходов:

- Микроусреднение (micro-average)
- Макроусреднение (macro-average)
- Взвешенное усреднение (weighted-average)

Макроусреднение (macro-average)

В этом подходе вычисляем значение выбранной метрики для каждой бинарной ситуации (кошка/не кошка, рыба/не рыба, курица/не курица), а затем усредняем полученные числа.

Посчитайте точность и полноту для ситуации кошка/не кошка:

		True/Actual		
		Cat	Fish	Hen
Predicted	Cat	4	6	3
	Fish	1	2	0
	Hen	1	2	6

$$precision(cat) = \frac{TP}{TP+FP}$$

$$recall(cat) = \frac{TP}{TP+FN}$$

Тогда macro-average:

$$precision = \frac{precision(cat) + precision(fish) + precision(hen)}{3}$$

Взвешенное усреднение (weighted-average)

В этом подходе мы усредняем посчитанные для каждого класса метрики с весами, пропорциональными количеству объектов класса.

То есть weighted average

$$precision = \frac{6}{25} \cdot precision(cat) + \frac{10}{25} \cdot precision(fish) + \frac{9}{25} \cdot precision(hen)$$

так как всего 25 объектов, и из них 6 кошек, 10 рыб и 9 куриц

Микроусреднение (micro-average)

В этом подходе мы вычисляем значения TP, TN, FP, FN по всей матрице ошибок сразу, исходя из их определения. Затем по полученным числам вычисляем выбранные метрики.

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

TP - это количество верно угаданных объектов положительного класса. В нашем случае TP=4+2+6=12

FP - это суммарное количество false positive-предсказаний. Например, если cat предсказана как fish, то это false positive для fish. Таким образом, FP - это сумма всех неверных предсказаний, то есть FP=6+3+1+0+1+2=13

Получаем micro-average $$precision = \frac{12}{12+13} = \frac{12}{25}$$

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

FN - это сумма false negative-предсказаний. Например, если cat предсказана как fish, то это false negative для cat. Таким образом, FN - это опять же сумма всех неверных предсказаний, то есть FN=6+3+1+0+1+2=13

Получается, что *в случае микроусреднения*

precision=recall

И так как f1-score - это среднее гармоническое точности и полноты, то *при микроусреднении*

*precision=recall=f*1