Метрики качества классификации

Метрики качества могут использоваться:

- Для задания функционала ошибки (используется при обучении).
- Для подбора гиперпараметров (используется при измерении качества на кросс-валидации). В том числе можно использовать другую метрику, которая отличается от метрики, с помощью которой построен функционал ошибки.
- Для оценивания итоговой модели: пригодна ли модель для решения задачи.

1. Доля правильных ответов

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

Python: sklearn.metrics.accuracy_score

Недостатки:

Первая проблема связана с несбалансированными выборками. Показателен следующий пример.

Пусть в выборке 1000 объектов, из которых 950 относятся к классу −1 и 50 — к классу +1. Рассматривается бесполезный (поскольку не восстанавливает никаких закономерностей в данных) константный классификатор, который на всех объектах возвращает ответ −1. Но доля правильных ответов на этих данных будет равна 0.95, что несколько много для бесполезного классификатора. Чтобы «бороться» с этой проблемой, используется следующий факт. Пусть q_0 — доля объектов самого крупного класса, тогда доля правильных ответов для разумных алгоритмов $\text{accuracy} \in [q_0, 1]$, а не [1/2, 1], как это можно было бы ожидать. Поэтому, *если получается высокий процент правильных ответов, это может быть связано не с тем, что построен хороший классификатор, а с тем, что какого-то класса сильно больше, чем остальных*.

Вторая проблема с долей верных ответов состоит в том, что она никак не учитывает разные цены разных типов ошибок. Тогда как цены действительно могут быть разными. Например, в задаче кредитного скоринга, то есть в задаче принятия решения относительно выдачи кредита, сравниваются две модели. При использовании первой модели кредит будет выдан 100 клиентам, 80 из которых его вернут. Во второй модели, более консервативной, кредит был выдан только 50 клиентам, причем вернули его в 48 случаях. То, какая из двух моделей лучше, зависит от того, цена какой из ошибок выше: не дать кредит клиенту, который мог бы его вернуть, или выдать кредит клиенту, который его не вернет. Таким образом, нужны дополнительные метрики качества, которые учитывают цены той или иной ошибки.

Как соотносятся между собой результат работы алгоритма и истинный ответ удобно оценивать с помощью *матрицы ошибок*

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

Когда алгоритм относит объект к классу 1, говорят, что алгоритм срабатывает. Если алгоритм сработал и объект действительно относится к классу 1, имеет место верное срабатывание (true positive), а если объект на самом деле относится к классу −1, имеет место ложное срабатывание (false positive). Если алгоритм дает ответ −1, говорят, что он пропускает объект. Если имеет место пропуск объекта класса 1, то это ложный пропуск (false negative). Если же алгоритм пропускает объект класса −1, имеет место истинный пропуск (true negative). Таким образом, существуют два вида ошибок: ложные срабатывания и ложные пропуски. *Для каждого из них нужна своя метрика качества, чтобы измерить, какое количество ошибок какого типа совершается*.

Python: sklearn.metrics.confusion_matrix,
sklearn.metrics. multilabel_confusion_matrix

2. Точность и полнота

Пусть для примера рассматриваются две модели $\alpha_1(x)$ и $\alpha_2(x)$. Выборка состоит из 200 объектов, из которых 100 относятся к классу 1 и 100 — к классу −1. Матрицы ошибок имеют вид:

	$y=1$	$y=-1$
$a_1(x)=1$	80	20
$a_1(x)=-1$	20	80

	$y=1$	$y=-1$
$a_2(x)=1$	48	2
$a_2(x)=-1$	52	98

Введем две метрики. Первая метрика, *точность* (*precision*), показывает, насколько можно доверять классификатору в случае срабатывания:

$$precision(a, X) = \frac{TP}{TP + FP}.$$

Python: sklearn.metrics.precision_score

Вторая метрика, *полнота* (*recall*), показывает, на какой доле истинных объектов первого класса алгоритм срабатывает:

$$recall(a, X) = \frac{TP}{TP + FN}.$$

Простыми словами: Как много объектов класса 1 находит классификатор?

Python: [sklearn.metrics.recall_score]

$$\begin{aligned} precision(a_1, X) &= 0.8, & precision(a_2, X) &= 0.96 \\ recall(a_1, X) &= 0.8, & recall(a_2, X) &= 0.48 \end{aligned}$$

В примере точность и полнота первого алгоритма оказываются равными. Вторая модель является очень точной, но в ущерб полноте.

Пример 1. Пусть в задаче кредитного скоринга ставится условие, что неудачных кредитов должно быть не больше 5%. В таком случае задача является задачей максимизации полноты при условии $precision(\alpha, X) \geq 0.95$.

Пример 2. Использование в медицинской диагностике. Необходимо построить модель, которая определяет, есть или нет определенное заболевание у пациента. При этом требуется, чтобы были выявлены как минимум 80% пациентов, которые действительно имеют данное заболевание. Тогда ставят задачу максимизации точности при условии $recall(a, X) \geq 0.8$. Следует особо обратить внимание на то, как точность и полнота работают в случае несбалансированных выборок.
Пусть рассматривается выборка со следующей матрицей ошибок:

	$y = 1$	$y = -1$
$a(x) = 1$	10	20
$a(x) = -1$	90	10000

Доля верных ответов (*accuracy*), точность(*precision*) и полнота(*recall*) для данного случая: $accuracy(a, X) = 0.99, \quad precision(a, X) = 0.33, \quad recall(a, X) = 0.1.$

То, что доля верных ответов равняется 0.99, ни о чем не говорит: алгоритм все равно делает 66% ложных срабатываний и выявляет только 10% положительных случаев. Благодаря введению точности и полноты становится понятно, что алгоритм нужно улучшать.

3. Объединение точности и полноты – F-мера

F-мера – гармоническое среднее точности и полноты:

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

Если необходимо отдать предпочтение точности или полноте, следует использовать расширенную F-меру, в которой есть параметр β:

$$F = (1 + \beta^2) \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}.$$

Python: sklearn.metrics.f1_score

Удобно использовать: sklearn.metrics.precision_recall_support
sklearn.metrics.classification_report

Качество оценок принадлежности к классу

Многие алгоритмы бинарной классификации устроены следующим образом: сначала вычисляется некоторое вещественное число $b(x)$, которое сравнивается с порогом t.

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

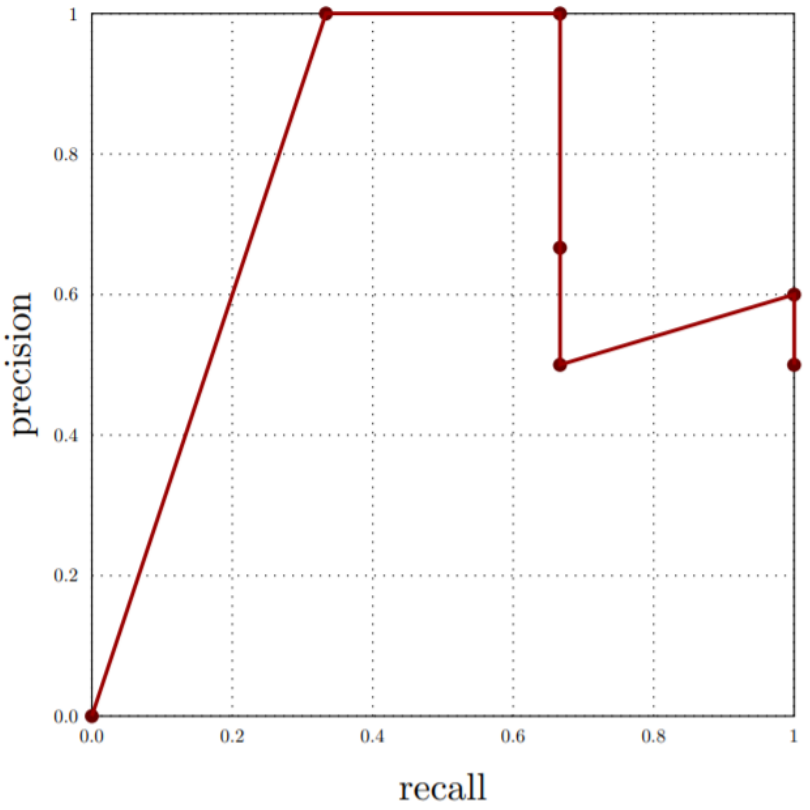
где $b(x)$ — оценка принадлежности классу +1. Другими словами, $b(x)$ выступает в роли некоторой оценки уверенности, что x принадлежит классу +1. В случае линейного классификатора $\alpha(x) = [\langle w, x\rangle > t]$ оценка принадлежности классу +1 имеет вид $b(x) = \langle w, x\rangle$. Часто бывает необходимо оценить качество именно оценки принадлежности, а порог выбирается позже из соображений на точность или полноту.

1. PR-кривая

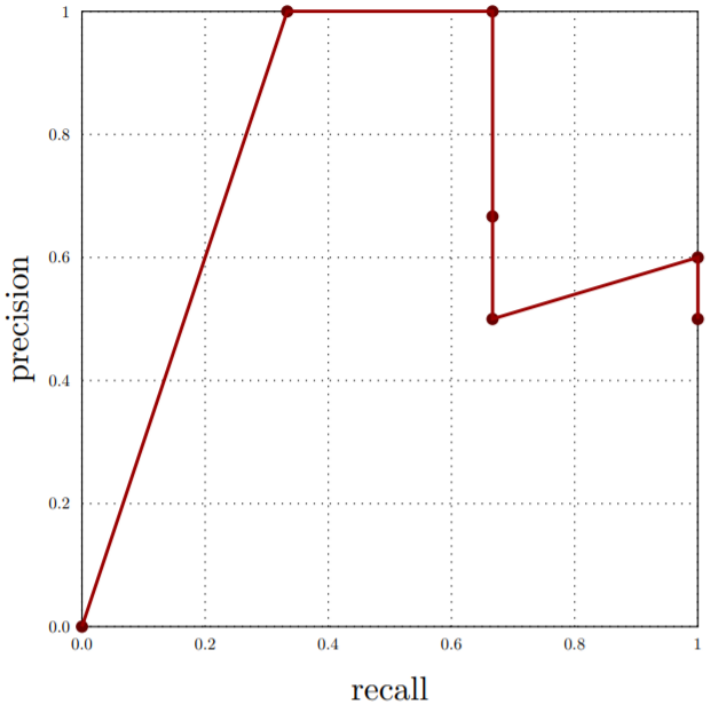
По оси X откладывается полнота, а по оси Y — точность. Каждой точке на этой кривой будет соответствовать классификатор с некоторым значением порога.

Для примера будет приведено построение PR-кривой для выборки из 6 объектов, три из которых относятся к классу 1 и 3 — к классу 0.

b(x)	0.14	0.23	0.39	0.54	0.73	0.90
y	0	1	0	0	1	1



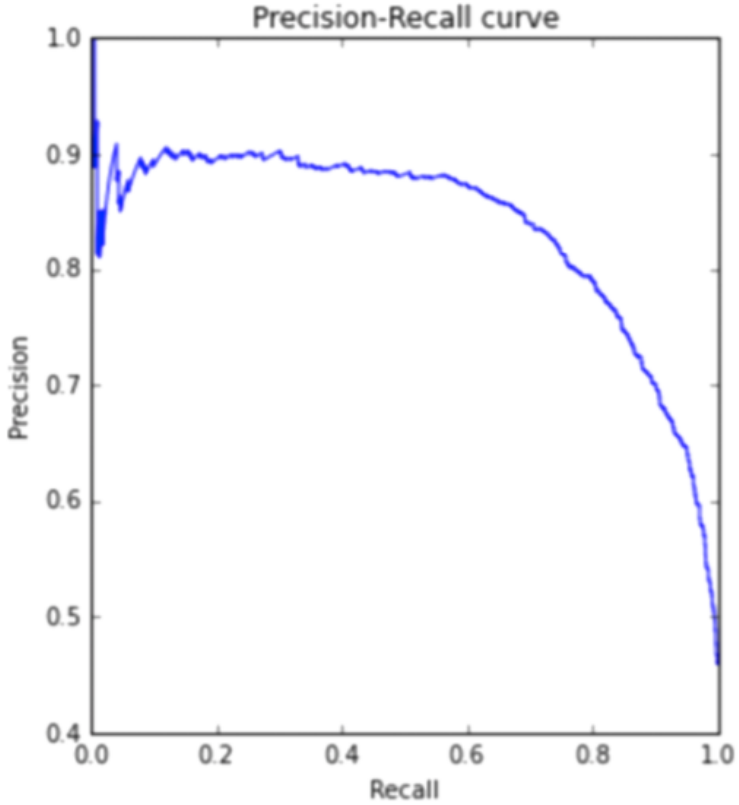
1. При достаточно большом пороге ни один объект не будет отнесен к классу 1. В этом случае и точность и полнота равны 0.
2. При таком пороге, что ровно один объект отнесен к классу 1, точность будет 100% (поскольку этот объект действительно из 1 класса), а полнота — 1/3 (поскольку всего 3 объекта 1 класса).
3. При дальнейшем уменьшении порога уже два объекта отнесены к классу 1, точность также остается 100%, а полнота становится равной 2/3.
4. При таком пороге, что уже три объекта будут отнесены к классу 1, точность становится равной 2/3, а полнота остается такой же.
5. При таком пороге, что четыре объекта отнесены к классу 1, точность уменьшится до 0.5, а полнота опять не изменится.
6. При дальнейшем уменьшении порога уже 5 объектов будут отнесены к 1 классу, полнота станет равной 100%, а точность — 3/5.



Следует отметить, что начинается PR-кривая всегда из точки (0, 0), а заканчивается точной (1, r), где r — доля объектов класса 1. В случае идеального классификатора, то есть если существует такой порог, что и точность, и полнота равны 100%, кривая будет проходить через точку (1, 1). Таким образом, чем ближе кривая пройдет к этой точке, тем лучше оценки. Площадь под этой кривой может быть хорошей мерой качества оценок принадлежности к классу 1. Такая метрика называется AUC–PRC, или площадь под PR-кривой.

PR-кривая строится в осях precision и recall, а следовательно *изменяется при изменении баланса классов*.

Python: sklearn.metrics.precision_recall_curve



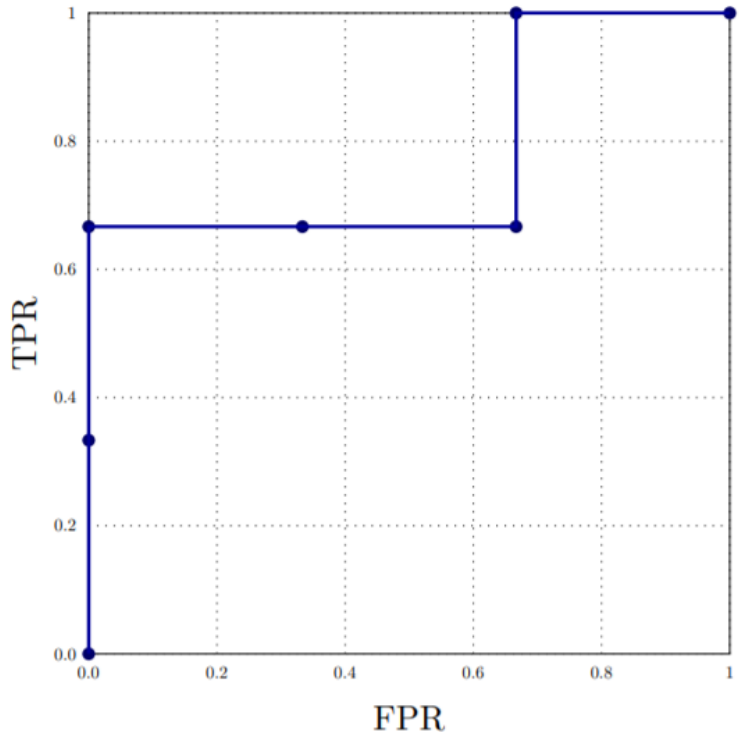
Кривая точности-полноты в реальных задачах с десятками тысяч объектов

1. ROC-кривая

Второй способ измерить качество оценок принадлежности к классу 1 — ROC-кривая, которая строится в осях False Positive Rate (ось X) и True Positive Rate (ось Y):

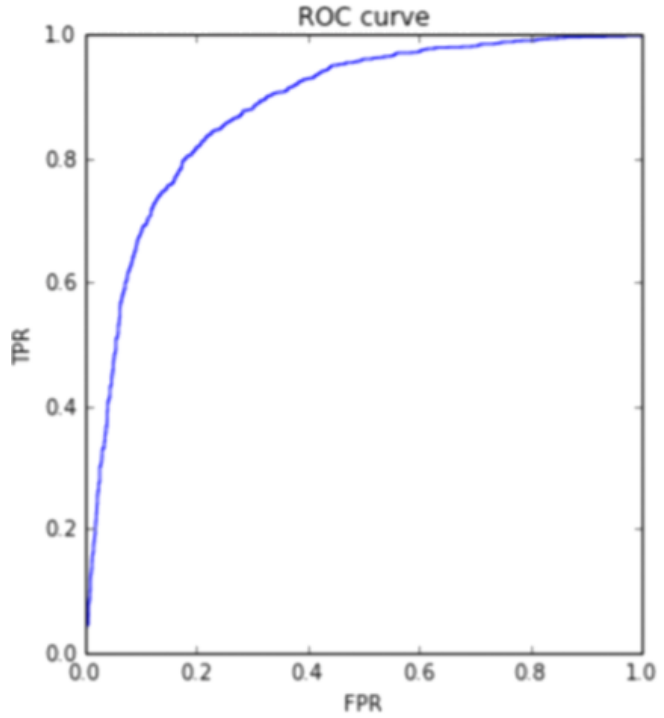
$$FPR = \frac{FP}{FP + TN}, \qquad TPR = \frac{TP}{TP + FN}.$$

ROC-кривая строится аналогично PR-кривой: постепенно рассматриваются случаи различных значений порогов и отмечаются точки на графике. Для упомянутой ранее выборки ROC-кривая имеет следующий вид:



Кривая стартует с точки (0, 0) и приходит в точку (1, 1). При этом, если существует идеальный классификатор, кривая должна пройти через точку (0, 1). Чем ближе кривая к этой точке, тем лучше будут оценки, а площадь под кривой будет характеризовать качество оценок принадлежности к первому классу. Такая метрика называется AUC–ROC, или площадь под ROC-кривой.

Как было написано выше, ROC-кривая строится в осях FPR и TPR, которые нормируются на размеры классов. Следовательно, при изменении баланса классов величина AUC-ROC и неизменных свойствах объектов выборки площадь под ROC-кривой не изменится.



ROC-кривая в реальных задачах с десятками тысяч объектов

Python: sklearn.metrics.roc_curve
sklearn.metrics.roc_auc_score