Метрики в задачах машинного обучения

Блог компании Open Data Science

Оглавление

Figure 1: Картинка для привлечения внимания.

В задачах машинного обучения для оценки качества моделей и сравнения различных алгоритмов используются метрики, а их выбор и анализ – непременная часть работы датасатаниста.

В этой статье мы рассмотрим некоторые критерии качества в задачах классификации, обсудим, что является важным при выборе метрики и что может пойти не так.

Метрики в задачах классификации

Для демонстрации полезных функций *sklearn* и наглядного представления метрик мы будем использовать датасет по оттоку клиентов телеком-оператора.

Загрузим необходимые библиотеки и посмотрим на данные:

	State	Account length	Area code	International plan	Voice mail plan	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total eve calls	Total eve charge	Total night minutes	Total night calls	Total night charge	Total intl minutes	Total intl calls	Total intl charge	Customer service calls	Churn
0	KS	128	415	No	Yes	25	265.1	110	45.07	197.4	99	16.78	244.7	91	11.01	10.0	3	2.70	1	False
1	OH	107	415	No	Yes	26	161.6	123	27.47	195.5	103	16.62	254.4	103	11.45	13.7	3	3.70	1	False
2	NJ	137	415	No	No	0	243.4	114	41.38	121.2	110	10.30	162.6	104	7.32	12.2	5	3.29	0	False
3	OH	84	408	Yes	No	0	299.4	71	50.90	61.9	88	5.26	196.9	89	8.86	6.6	7	1.78	2	False
4	OK	75	415	Yes	No	0	166.7	113	28.34	148.3	122	12.61	186.9	121	8.41	10.1	3	2.73	3	False

Figure 2: Первые 5 строк данных

Accuracy, precision и recall

Перед переходом к самим метрикам необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации – *confusion matrix* (матрица ошибок). Допустим, что у нас есть два класса и алгоритм, предсказывающий принадлежность каждого

объекта одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

Table 1: Матрица ошибок классификации

	$y=1$	$y=0$
$\hat{y}=1$	True Positive (TP)	False Positive (FP)
$\hat{y}=0$	False Negative (FN)	True Negative (TN)

Здесь $\hat{y}$ – это ответ алгоритма на объекте, а y – истинная метка класса на этом объекте. Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).

Обучение алгоритма и построение матрицы ошибок.

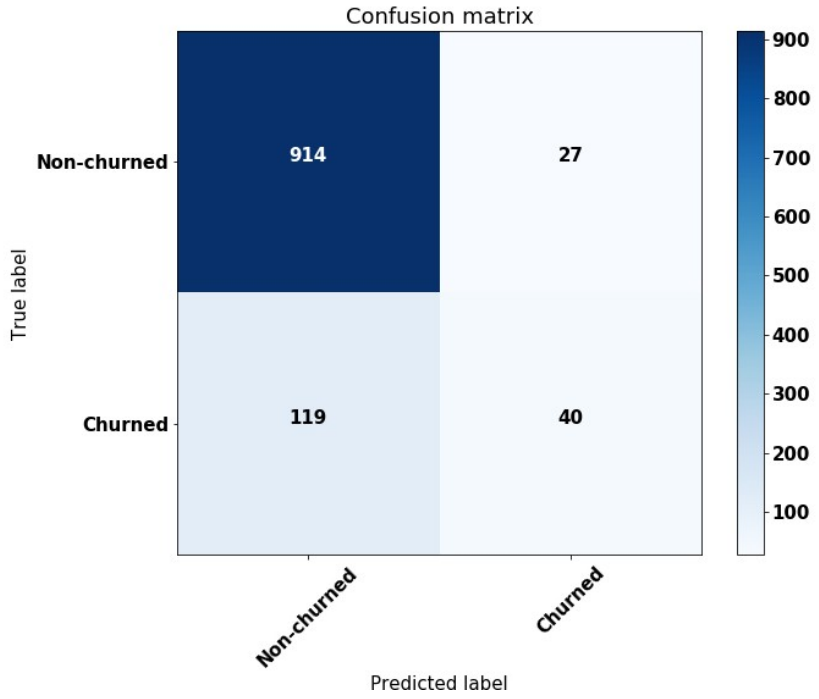

Figure 3: Матрица ошибок обученного алгоритма

Accuracy

Интуитивно понятной, очевидной и почти неиспользуемой метрикой является accuracy – доля правильных ответов алгоритма:

$$\text{accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}.$$

Эта метрика бесполезна в задачах с неравными классами, и это легко показать на примере.

Допустим, мы хотим оценить работу спам-фильтра почты. У нас есть 100 не-спам писем, 90 из которых наш классификатор определил верно (True Negative = 90, False Positive = 10), и 10

спам-писем, 5 из которых классификатор также определил верно (True Positive = 5, False Negative = 5). Тогда accuracy:

$$\text{accuracy} = \frac{5+90}{5+90+10+5} = 86{,}4.$$

Однако если мы просто будем предсказывать все письма как не-спам, то получим более высокую accuracy:

$$\text{accuracy} = \frac{0+100}{0+100+0+10} = 90{,}9.$$

При этом, наша модель совершенно не обладает никакой предсказательной силой, так как изначально мы хотели определять письма со спамом. Преодолеть это нам поможет переход с общей для всех классов метрики к отдельным показателям качества классов.

Precision, recall и F-мера

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).

$$\text{precision} = \frac{TP}{TP+FP}, \qquad \text{recall} = \frac{TP}{TP+FN}.$$

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

Именно введение precision не позволяет нам записывать все объекты в один класс, так как в этом случае мы получаем рост уровня False Positive. Recall демонстрирует способность алгоритма обнаруживать данный класс вообще, а precision – способность отличать этот класс от других классов.

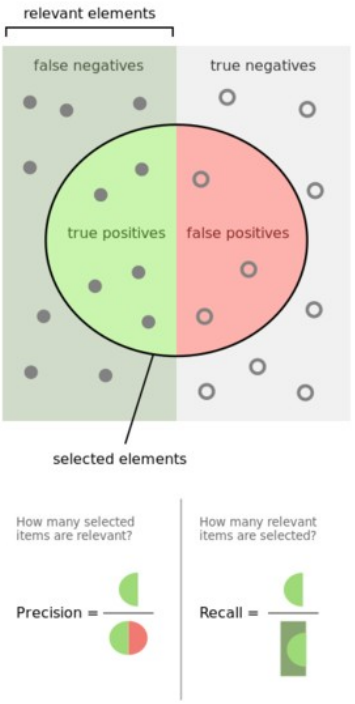

Figure 4: Иллюстрация к метрикам качества классификации.

Как мы отмечали ранее, ошибки классификации бывают двух видов: False Positive и False Negative. В статистике первый вид ошибок называют ошибкой I-го рода, а второй – ошибкой II-го рода. В нашей задаче по определению оттока абонентов, ошибкой первого рода будет принятие лояльного абонента за уходящего, так как наша нулевая гипотеза состоит в том, что никто из абонентов не уходит, а мы эту гипотезу отвергаем. Соответственно, ошибкой второго рода будет являться "пропуск" уходящего абонента и ошибочное принятие нулевой гипотезы.

Precision и recall не зависят, в отличие от accuracy, от соотношения классов и потому применимы в условиях несбалансированных выборок. Часто в реальной практике стоит задача найти оптимальный (для заказчика) баланс между этими двумя метриками. Классическим примером является задача определения оттока клиентов. Очевидно, что мы не можем находить **всех** уходящих в отток клиентов и **только** их. Но, определив стратегию и ресурс для удержания клиентов, мы можем подобрать нужные пороги по precision и recall. Например, можно сосредоточиться на удержании только высокодоходных клиентов или тех, кто уйдет с большей вероятностью, так как мы ограничены в ресурсах колл-центра.

Обычно при оптимизации гиперпараметров алгоритма (например, в случае перебора по сетке *GridSearchCV*) используется одна метрика, улучшение которой мы и ожидаем увидеть на тестовой выборке. Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. F-мера (в общем случае F_β) – среднее гармоническое precision и recall:

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

β в данном случае определяет вес точности в метрике, и при $\beta=1$ это среднее гармоническое (с множителем 2, чтобы в случае precision = 1 и recall = 1 иметь $F_1=1$)
F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю. В sklearn есть удобная функция `metrics.classificationreport`, возвращающая recall, precision и F-меру для каждого из классов, а также количество экземпляров каждого класса.

Table 2: Метрики качества алгоримов классификации

Class	Precision	Recall	f1-score	support
Non-churned	0,88	0,97	0,93	941
Churned	0,60	0,25	0,35	159
avg / total	0,84	0,87	0,84	1100

Здесь необходимо отметить, что в случае задач с несбалансированными классами, которые превалируют в реальной практике, часто приходится прибегать к техникам искусственной модификации датасета для выравнивания соотношения классов. Их существует много, и мы не будем их касаться, здесь можно посмотреть некоторые методы и выбрать подходящий для вашей задачи.

AUC-ROC и AUC-PR

При конвертации вещественного ответа алгоритма (как правило, вероятности принадлежности к классу, отдельно см. SVM) в бинарную метку, мы должны выбрать какой-либо порог, при котором 0 становится 1. Естественным и близким кажется порог, равный 0.5,

но он не всегда оказывается оптимальным, например, при вышеупомянутом отсутствии баланса классов.

Одним из способов оценить модель в целом, не привязываясь к конкретному порогу, является AUC-ROC (или ROC AUC) – площадь (*A*rea *U*nder *C*urve) под кривой ошибок (*R*eceiver *O*perating *C*haracteristic curve). Данная кривая представляет из себя линию от (0, 0) до (1, 1) в координатах True Positive Rate (TPR) и False Positive Rate (FPR):

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

TPR нам уже известна, это полнота, а FPR показывает, какую долю из объектов negative класса алгоритм предсказал неверно. В идеальном случае, когда классификатор не делает ошибок (FPR = 0, TPR = 1) мы получим площадь под кривой, равную единице; в противном случае, когда классификатор случайно выдает вероятности классов, AUC-ROC будет стремиться к 0.5, так как классификатор будет выдавать одинаковое количество TP и FP. Каждая точка на графике соответствует выбору некоторого порога. Площадь под кривой в данном случае показывает качество алгоритма (больше – лучше), кроме этого, важной является крутизна самой кривой – мы хотим максимизировать TPR, минимизируя FPR, а значит, наша кривая в идеале должна стремиться к точке (0,1).

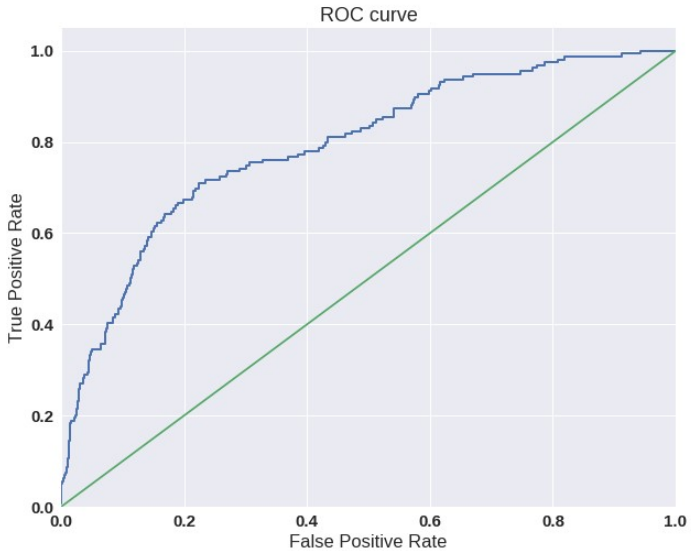

Figure 5: Кривая ROC

Критерий AUC-ROC устойчив к несбалансированным классам (спойлер: увы, не всё так однозначно) и может быть интерпретирован как вероятность того, что случайно выбранный positive объект будет проранжирован классификатором выше (будет иметь более высокую вероятность быть positive), чем случайно выбранный negative объект.

Рассмотрим следующую задачу: нам необходимо выбрать 100 релевантных документов из 1 миллиона документов. Мы намашинлернили два алгоритма:

- **Алгоритм 1** возвращает 100 документов, 90 из которых релевантны. Таким образом,

$$\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{90}{90+10} = 0.9,$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}} = \frac{10}{10+999890} = 0.00001.$$

- **Алгоритм 2** возвращает 2000 документов, 90 из которых релевантны. Таким образом,

$$\text{TPR} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{90}{90+10} = 0.9,$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP}+\text{TN}} = \frac{1910}{1910+997990} = 0.00191.$$

Скорее всего, мы бы выбрали первый алгоритм, который выдает очень мало False Positive на фоне своего конкурента. Но разница в False Positive Rate между этими двумя алгоритмами *крайне* мала – всего 0.0019. Это является следствием того, что AUC-ROC измеряет долю False Positive относительно True Negative и в задачах, где нам не так важен второй (больший) класс, может давать не совсем адекватную картину при сравнении алгоритмов.

Для того чтобы поправить положение, вернемся к полноте и точности:

- **Алгоритм 1**

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{90}{90+10} = 0.9,$$

$$\text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{90}{90+10} = 0.9.$$

- **Алгоритм 2**

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} = \frac{90}{90+1910} = 0.045,$$

$$\text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{90}{90+10} = 0.9.$$

Здесь уже заметна существенная разница между двумя алгоритмами – 0.855 в точности! Precision и recall также используют для построения кривой и, аналогично AUC-ROC, находят площадь под ней.

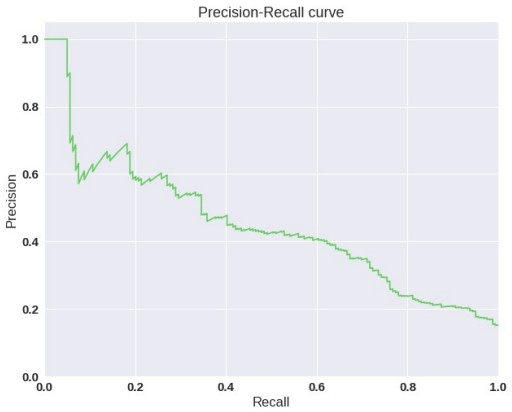

Figure 6: Кривая precision-recall

Здесь можно отметить, что на маленьких датасетах площадь под PR-кривой может быть чересчур оптимистична, потому как вычисляется по методу трапеций, но обычно в таких задачах данных достаточно. За подробностями о взаимоотношениях AUC-ROC и AUC-PR можно обратиться сюда.

Logistic Loss

Особняком стоит логистическая функция потерь, определяемая как:

$$\text{logloss} = -\frac{1}{l} \cdot \sum_{i=1}^{l} \left[y_i \cdot \log\left(\hat{y}_i\right) + \left(1 - y_i\right) \cdot \log\left(1 - \log y_i\right) \right].$$

Здесь $\hat{y}$ – это ответ алгоритма на i-ом объекте, y – истинная метка класса на i-ом объекте, а l – размер выборки.

Подробно про математическую интерпретацию логистической функции потерь уже написано в рамках поста про линейные модели. Данная метрика нечасто выступает в бизнес-требованиях, но часто — в задачах на kaggle. Интуитивно можно представить минимизацию logloss как задачу максимизации accuracy путем штрафа за неверные предсказания. Однако необходимо отметить, что logloss крайне сильно штрафует за уверенность классификатора в неверном ответе.

Выводы

1. В случае многоклассовой классификации нужно внимательно следить за метриками каждого из классов и следовать логике решения **задачи**, а не оптимизации метрики;
2. В случае неравных классов нужно подбирать баланс классов для обучения и метрику, которая будет корректно отражать качество классификации;

3. Выбор метрики нужно делать с фокусом на предметную область, предварительно обрабатывая данные и, возможно, сегментируя (как в случае с делением на богатых и бедных клиентов).