4.1. Постановка задач классификации и примеры таких задач

Постановка задач классификации и примеры таких задач

Постановка задачи классификации

В прошлом лонгриде мы познакомились с задачей регрессии, в которой выходом модели была непрерывная величина из заданного интервала. Теперь рассмотрим задачу, в которой нужно получить номер из заданного дискретного множества. То есть нужно построить алгоритм, соотносящий входной объект с его номером в множестве всех классов объектов, или, иначе говоря, построить **классификатор.**

Теперь приведем формальную постановку задачи классификации:

1. Задана выборка объектов ***X*** и дискретное множество номеров классов ***Y***
2. Существует неизвестное отображение $f : X \to Y$
3. Значения этого отображения известны только для объектов из обучающей выборки: $D = \{X \times Y\}^n = \{(x_i, y_i) | x_i \in X, y_i \in Y\}_{i=1}^n$
4. Нужно построить алгоритм, способный классифицировать произвольный объект

В задаче регрессии мы рассматривали выборку объектов, представляющую из себя признаковое описание. Для задачи классификации можно подобрать гораздо более широкий спектр данных.

Так что в нашей постановке множество X может представляться как:

● Признаковое пространство. Признаки могут быть *бинарными ({0, 1})*, *количественными* (действительные числа), *порядковыми* (конечный упорядоченный набор) и *категориальными* (конечное множество описаний). С последними работают, например, через *one hot encoding* (с ним мы уже работали в прошлом модуле), переводя их в порядковые
● Матрица расстояний — для каждого объекта содержится информация о его расстоянии до всех объектов выборки
● Временной ряд — последовательность измерений во времени, представляющих собой признаковое пространство
●Изображение/видеозапись

Также и предсказываемое значение теперь представляет собой элемент (в зависимости от задачи — подмножество) из Y. Можно выделить следующие типы задач классификации:

- Многоклассовая классификация: $Y = \{1, 2, ..., k\}$

Выход модели — наиболее вероятная метка класса:

$$y = \underset{y_i \in Y}{argmax}\, p((x_i, y_i), f)$$

- Бинарная классификация: $Y = \{0, 1\}$

Частный случай многоклассовый классификации на множестве меток классов {0, 1}. Однако с помощью бинарной классификации решаются более сложные задачи, и, как мы увидим далее в модуле, некоторые метрики в таких задачах сводятся к подсчету метрик в бинарном случае

- Пересекающиеся классы: $Y = \{0, 1\}^k$

При такой постановке задачи объект может относиться к нескольким классам: выходом модели будет вектор размерности *k* с нулем на *i*-м индексе, если объект не принадлежит *i*-му классу, и с единицей, если принадлежит.

Примеры задач классификации

Классификация применяется практически во всех прикладных задачах машинного обучения. В обработке естественного языка классифицируют тексты (например, художественную литературу по жанрам), в компьютерном зрении — изображения (вспомните ту же капчу), в банковской сфере с помощью классификации оценивают кредитоспособность заемщиков. Даже рассмотренную в прошлом модуле задачу регрессии можно свести к классификации с помощью квантилей.

4.2. Методы решения задач классификации: логистическая регрессия, решающие деревья, ансамбли моделей

В предыдущем модуле была введена постановка задачи классификации и примеры таких задач. Традиционно методы решения задач делят на классические и нейросетевые. О нейросетевых решениях будет рассказано позже, а сейчас речь пойдет о наиболее известных классических моделях.

Логистическая регрессия

Под ***логистической регрессией*** понимается метод обучения с учителем, решающий задачу бинарной классификации и имеющий формульное представление:

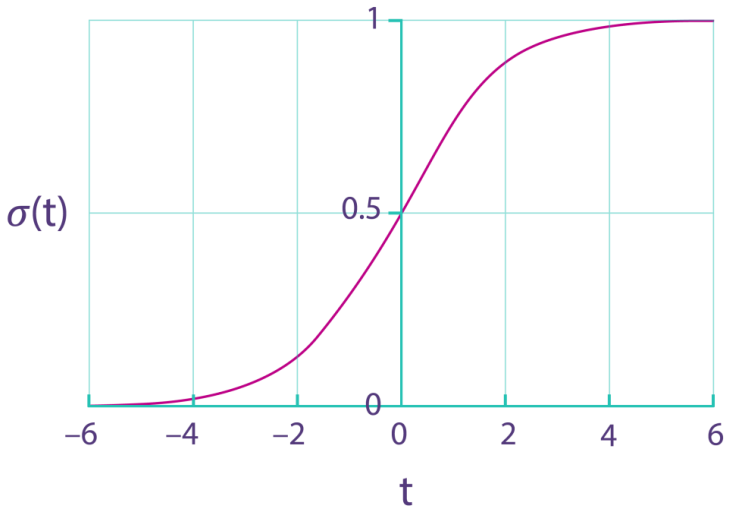
$$p = \frac{1}{1 + e^{-\langle w, x \rangle}}$$

Где:

- *p* — вероятность того, что объект, имеющий признаковое описание x, принадлежит первому из двух классов
- **w** — вектор параметров модели

- **<w,x>** — скалярное произведение векторов весов и признаков, которое нам уже знакомо по линейной регрессии

Нелинейность поверх скалярного произведения нам нужна, чтобы получившееся число было вероятностью — лежало на отрезке от 0 до 1. Функция зависимости вероятности p от **<w,x>** называется ***сигмоидой*** и выглядит так:



Как видно из графика, сигмоидная функция $\sigma(t)$ преобразует число в вероятность из диапазона от 0 до 1. При стремлении аргумента функции в бесконечность значение сигмоиды выходит на константу (1 или 0 в зависимости от знака аргумента). Это свойство сигмоиды еще пригодится в дальнейшем при изучении проблемы затухания градиента.

Обучение логистической регрессии происходит с помощью ***метода максимального правдоподобия (ММП)*** — пожалуй, основного способа оценивания неизвестных параметров, который пришел в машинное обучение из математической статистики. Суть ММП заключается в следующем.

Пусть:

- $(x_1, x_2, \ldots, x_n)$ — обучающая выборка, которая является реализацией независимых и одинаково распределенных случайных величин $(\xi_1, \xi_2, \ldots, \xi_n)$
- $P(x_1, x_2, \ldots, x_n; w) = \prod_{i=1}^{n} p(x_i, w)$ — функция правдоподобия, где $p(x_i, w)$ — вероятность того, что случайная величина ξ_i принимает значение x_i. Вектор весов w — это набор параметров распределения случайной величины, которые и нужно найти

Цель — найти такой вектор весов, при котором правдоподобие будет максимальным. Интуиция такого подхода заключается в том, что мы хотим максимизировать вероятность получить в n экспериментах текущую выборку $(x^1, x^2, \ldots, x^n)$. Формально задача звучит так:

$$\omega_{opt} = arg \max_{\omega} P(x_1, x_2, \ldots, x_n; \omega)$$

Поскольку логарифм — монотонная функция, то положение максимума не изменится. Следовательно, формальную постановку задачи можно переписать по-другому:

$$\omega_{opt} = arg\max_{\omega} \sum_{i=1}^{n} (\ln p\,(x_i; \omega))$$

Такой формальный переход (от произведения к сумме) — достаточно частый прием ввиду свойств дифференцирования (производная суммы равна сумме производных).

Переходя от теории к практике, сразу замечаем, что $p(x_i, w)$ — это предсказание модели с вектором весов для объекта с векторным описанием x_i. Например, этим предсказанием может быть логистическая регрессия. Тогда можно вместо $p(x_i, w)$ подставить формулу для логистической регрессии и найти такие веса, при которых правдоподобие будет максимальным.

Для линейной регрессии метод максимального правдоподобия эквивалентен минимизации среднеквадратичной ошибки. Получаем явную формулу, по которой можно считать параметры. Для логистической регрессии все сложнее, но все еще неплохо. Можно показать, что максимизация правдоподобия эквивалентна минимизации ***кросс-энтропийной*** функции потерь. Для бинарной классификации формула для кросс-энтропии выглядит так:

$$L = -\sum_{i=1}^{n} y_i \ln p(x_i, w) + (1 - y_i) \ln(1 - p(x_i, w)))$$

Где:

- n — размер обучающей выборки (количество пар $(x_i y_i)$)
- $p(x_i; w)$ — решающее правило, где w — веса модели, по которым происходит минимизация функции потерь

Таким образом, необходимо найти такие веса модели, при которых кросс-энтропийная функция будет принимать минимальное значение. Обычно для решения поставленной задачи применяют ***метод градиентного спуска***, а также его вариации (стохастический градиентный спуск (SGD), импульсный градиентный спуск и т. д.).

Если говорить кратко, то общая суть метода градиентного спуска заключается в следующем. Из математического анализа известно, что направление наибольшего роста функции совпадает с направлением градиента. Поскольку мы хотим минимизировать функцию потерь, то мы должны двигаться в сторону антиградиента, т. е. в обратном направлении. Таким образом, на каждой итерации вычисляется антиградиент, и веса модели обновляются по следующему правилу:

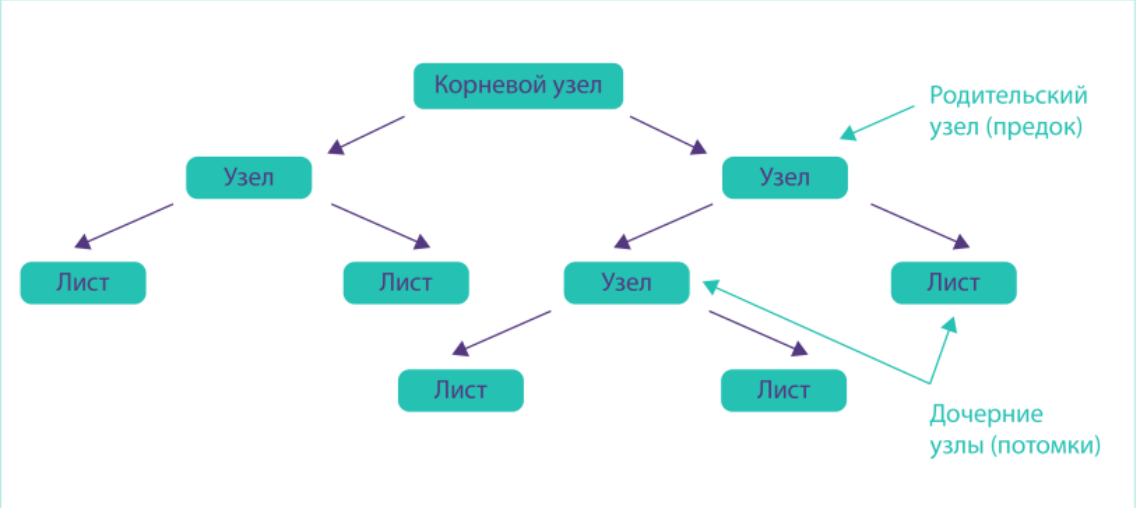
$$w_{t+1} = w_t - \alpha \frac{\partial L}{\partial w}$$

Где:

- w_t — веса модели на текущей итерации
- w_{t+1} — обновленные веса модели
- α — вещественный параметр, отвечающий за скорость движения вдоль антиградиента: $-\frac{\partial L}{\partial w}$

Решающие деревья

Под ***решающим деревом*** понимается (чаще всего) бинарное дерево, в каждой внутренней вершине которого записано условие, а в каждом листе дерева — прогноз. В общем структура дерева выглядит так:



Мы начинаем с корня дерева. В каждом узле дерева производятся проверки свойств объекта. Например, болит ли горло у пациента? Если да, идем в левый узел, если нет — в правый. Когда дошли до листа, смотрим на число, которое в нем записано. Это и будет прогнозом решающего дерева.

Для построения дерева используются так называемые ***бинарные предикаты***. Под бинарным предикатом понимается функция, которой на вход подается некоторый признак и порог. Если значение признака больше порога, то предикат возвращает 1, иначе — 0.
Принцип построения дерева заключается в следующем. Оно строится от корня к листьям. Вначале выбирается корень дерева, который разбивает исходную выборку на две части наилучшим образом с точки зрения выбранного ***функционала качества***, который в качестве параметров принимает выборку в данном узле, признак и порог. Разбиение на две части происходит с помощью бинарных предикатов, о которых было сказано ранее. Затем для каждой части рекурсивно повторяем процедуру разбиения, описанную выше, до тех пор, пока не выполнится ***критерий останова***.

Для завершения описания процесса построения решающего дерева рассмотрим введенные выше понятия, а именно функционал качества и критерий останова. Начнем с функционала качества.

$$Q(X_m, j, t) \to \min_{j,t}$$

Здесь первый аргумент — это выборка в данном узле, а j и t — это признак и порог соответственно. Поскольку признаков конечное число, то параметры j и t можно перебирать. Для фиксированных j и t функционал качества выглядит так:

$$Q(X_m, j, t) = \frac{|X_\ell|}{|X_m|} H(X_\ell) + \frac{|X_r|}{|X_m|} H(X_r)$$

где

$$X_\ell = \{x \in X_m | [x^j \le t]\}, \qquad X_r = \{x \in X_m | [x^j > t]\},$$

Функция *H* называется ***критерием информативности***. Ее значение будет тем меньше, чем меньше разброс ответов в X. Для задачи классификации обычно выбирают два критерия: ***критерий Джини*** и ***энтропийный критерий***.

- **Критерий Джини:**

$$H(X) = \sum_{k=1}^{K} p_k(1 - p_k)$$

$$p_k = \frac{1}{X} \sum_{i \in X} [y_i = k]$$

где K — количество классов.

- **Энтропийный критерий:**

$$H(X) = -\sum_{k=1}^{K} p_k \ln p_k$$

Рассмотрим критерии останова. Существуют много различных критериев останова. Рассмотрим некоторые из них:

- Если в вершину попал один объект или все объекты принадлежат одному классу, тогда дальнейшее разбиение бессмысленно
- Глубина дерева достигла определенного значения

Отметим, что решающее дерево, как и логистическая регрессия, является нелинейной моделью. Одно из свойств решающего дерева заключается в том, что оно легко переобучается. Одним из способов борьбы с переобучением является прунинг, или ***стрижка***. Идея стрижки заключается в построении дерева максимальной глубины до тех пор, пока в каждой вершине не останется по одному объекту обучающей выборки. Затем удаляются листья по определенному критерию. Например, можно удалять листья до тех пор, пока улучшается качество на отложенной выборке.

Ансамбли моделей

Деревья обладают низким смещением, но большим разбросом. Если немного поменять обучающую выборку, дерево может получиться совсем другим. Отсюда возникает идея использования не одного, а нескольких деревьев, т. е. ***композиции алгоритмов или ансамбля моделей***. Под алгоритмом понимается одно решающее дерево. Например, в качестве композиции можно взять усреднение по всем алгоритмам. Из теории вероятностей мы знаем, что для среднего значения независимых случайных величин с одинаковым математическим ожиданием и одинаковой дисперсией справедливы следующие равенства:

$$\xi = \frac{1}{n}(\xi_1 + \ldots + \xi_n)$$

$$\mathbf{E}\xi = \frac{1}{n}(\mathbf{E}\xi_1 + \ldots + \mathbf{E}\xi_n) = \mathbf{E}\xi_i$$

$$\mathbf{D}\xi = \frac{1}{n^2}(\mathbf{D}\xi_1 + \ldots + \mathbf{D}\xi_n) = \frac{\mathbf{D}\xi_i}{n}$$

Из формул видно, что разброс среднего в n раз меньше, чем разброс одной модели, при этом смещение остается тем же. Отсюда видна интуиция по взятию среднего в качестве ансамбля. На практике, конечно, предположение о независимости является очень сильным и далеко не всегда соблюдается. Тем не менее существуют различные способы уменьшения корреляции между базовыми алгоритмами:

●Бэггинг

Обучение базовых алгоритмов происходит на случайных подвыборках обучающей выборки. Итоговый класс определяется голосованием базовых алгоритмов.

●Метод случайных подпространств

Аналогично бэггингу, только обучение каждого базового алгоритма происходит на случайном подмножестве признаков.
К еще одному типу ансамблей можно отнести ***бустинг***. Под бустингом понимается ансамбль моделей, которые обучаются последовательно, при этом каждый последующий алгоритм обучается на ошибках предыдущего алгоритма. Наиболее известным вариантом бустинга является ***градиентный бустинг***, в котором для корректировки весов следующего алгоритма используется градиент функции потерь. Про него будет рассказано дальше в курсе.

4.3. Выбор порога для классификатора. Метрики PR AUC, ROC AUC

Типичный бинарный классификатор состоит из двух частей:
- Часть, которая предсказывает вероятность принадлежности объекта положительному классу. Назовем ее ***регрессор***
- Некоторое вещественное число, которое называется ***порогом***. Если вероятность больше порога, то классификатор предсказывает положительный класс. Иначе — отрицательный

Формально предсказание классификатора можно записать так:

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

Где:
- *b(x)* —регрессор
- *t* —порог
- *[ ]* — скобки Айверсона, которые возвращают 1 в случае верности логического утверждения внутри них. Иначе — возвращается 0

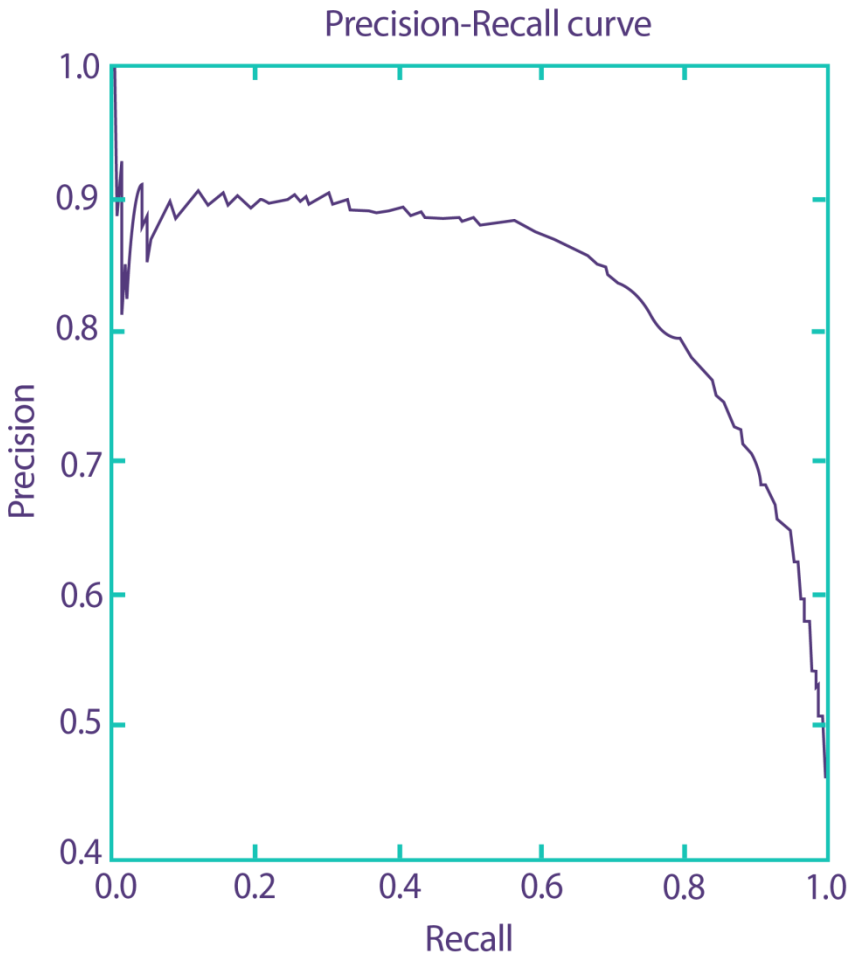
Таким образом качество классификатора зависит как от регрессора, так и от выбранного порога. Рассмотрим сначала метрики качества регрессора, а затем процедуру выбора оптимального порога.

Оценка качества регрессора. PR-AUC

Рассмотрим кривую «точность — полнота» (precision — recall (PR)), которая строится по следующему принципу:

- Фиксируется набор порогов для классификатора
- Для каждого значения из набора порогов производится подсчет precision и recall, которые наносится на график с координатами precision и recall

Когда объектов в выборке достаточно много, то график PR-кривой выглядит так:



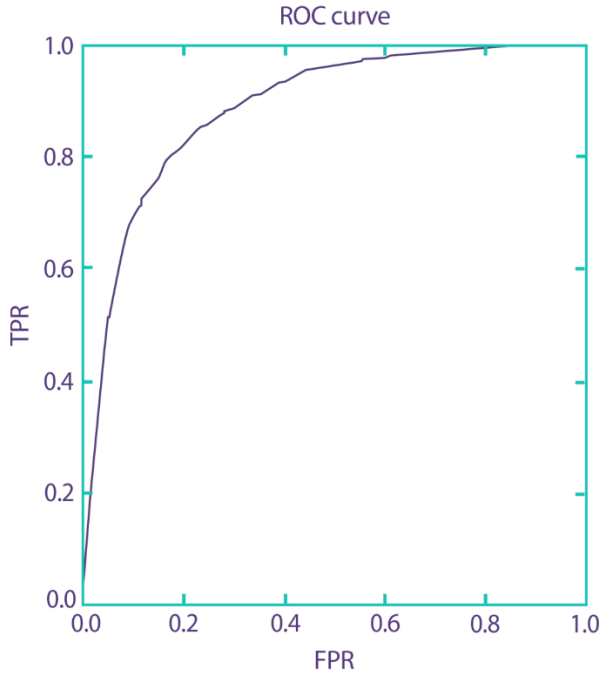
Мы стремимся улучшить одновременно и precision, и recall. Поэтому идеальный классификатор проходит через точку (1, 1). Получается, что чем ближе график PR-кривой проходит к точке (1, 1), тем он лучше. Как следствие, хорошей оценкой качества регрессора является площадь под графиком, которая называется **PR-AUC**.

Оценка качества регрессора. ROC-AUC

Рассмотрим другой важный способ оценки качества — **ROC**-кривую и площадь под ней. Обычно именно на них смотрят в первую очередь. У этой кривой другие оси — False Positive Rate(FPR) и True Positive Rate (TPR). FPR является осью X, а TPR — осью Y. По существу, FPR — это доля ложно-положительных объектов, а TPR — доля верно-положительных объектов.

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

Принцип построения ROC-кривой полностью аналогичен PR-кривой. Для большой выборки объектов график ROC-кривой выглядит примерно следующим образом:



Из определений TPR и FPR становится ясно, что идеальный классификатор проходит через точку (0, 1). Аналогично PR-кривой, чем ближе график проходит к этой точке, тем лучше качество модели. Точно так же площадь под графиком ROC-кривой, которая называется ***ROC-AUC***, является хорошей мерой качества. Лучший алгоритм будет иметь ROC-AUC = 1. У алгоритма, который принимает решения случайно, ROC-AUC = ½, т. к. в таком случае доля ложно-положительных и правильно-положительных объектов одинакова, т. е. происходит случайное угадывание ответа.

Выбор порога с помощью PR-кривой

Существуют различные способы выбора порога. Традиционно подход в поиске подходящего порога заключается в следующем:

- Модель обучается на тренировочной выборке
- Предсказываются вероятности на тестовой выборке
- Далее, используя предсказанные вероятности и тестовые метки, с помощью некоторой метрики производится поиск подходящего порога

Рассмотрим различные метрики, которые используются для поиска оптимального порога.

При нахождении PR-кривой для каждого порога считаются precision и recall. Как по этим значениям определить наилучший порог? Один из способов — использовать F-меру. Тогда для каждого порога у нас будет одна метрика, объединяющая в себе precision и recall. В таком случае оптимальным порогом будет порог, соответствующий максимальному значению F-меры.

Выбор порога с помощью ROC-кривой

Аналогичный принцип можно применить для ROC-кривой. Так, для построения ROC-кривой нужно произвести подсчет ***True Positive Rate (TPR)*** и ***False Positive Rate (FPR)***. На основании TPR и FPR можно ввести одну метрику, объединяющую оба этих значения:

$$Gmean = \sqrt{sensitivity \cdot specifity}$$

где

$$sensitivity = TPR$$
$$specifity = 1 - FPR$$

Таким образом, понятие специфичности мы выразили через FPR. В качестве итоговой метрики используется геометрическое среднее между чувствительностью и специфичностью. Аналогично работе с PR-кривой, оптимальным порогом будет такой, при котором геометрическое среднее будет максимальным.

4.4. Метрики качества для задач классификации: точность и специфичность, F1

В задаче классификации существуют разные подходы к оцениванию качества модели. Дело в том, что метрика должна учитывать специфику решаемой задачи. Например, рассмотрим классическую задачу классификации спама. Задача модели — по входящему письму понять, является ли оно спамом или нет, т. е. задача бинарной классификации. Допустим, в качестве метрики взята доля правильных ответов (***accuracy***). Тогда, если алгоритм на каждое письмо будет утверждать, что оно не является спамом, можно получить достаточно высокое значение метрики, т. к. с точки зрения статистики большинство писем не является спамом.

Получается, что с точки зрения метрики алгоритм хороший, но из здравого смысла ясно, что модель не решает поставленную задачу. Следовательно, для каждой задачи нужно выбирать свою метрику. Так, в текущей задаче явно не хватало более серьезного учета ошибки, когда модель для спама предсказывала не спам. Таким образом, мы переходим к рассмотрению разных типов ошибок предсказаний.

Типы ошибок предсказаний

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

В приведенной выше таблице представлены термины, которые будут неоднократно использованы в дальнейшем. Предположим, что решается задача бинарной классификации. Под a(x) понимаются предсказания алгоритма, y — корректные ответы.

- Если алгоритм предсказал класс 1 и корректный ответ тоже равен 1, то такое событие называют ***True Positive (TP)***. То есть алгоритм корректно предсказал объект класса 1
- Аналогично, если предсказание и корректный ответ совпадают для класса –1, то такое событие называют ***True Negative (TN)***.

В математической статистике существуют такие понятия, как ошибка 1 и 2 рода. Ошибка 1 рода заключается в том, что отвергнута верная нулевая гипотеза. Ошибка 2 рода заключается в принятии неверной нулевой гипотезы. Так же и в нашей задаче:

- Если модель предсказала принадлежность объекта к 1 классу, а на самом деле он принадлежит к другому классу, то такая ошибка называется ***False Positive (FP), ложная тревога*** или ***ошибка 1 рода***
- Аналогично в ситуации, когда алгоритм предсказывает класс –1, а реально объект принадлежит к 1 классу, то ошибка будет называться ***False Negative (FN), пропуск цели*** или ***ошибка 2 рода***

Можно смотреть на все эти 4 метрики сразу и сравнивать их для разных алгоритмов, но такой подход крайне неудобен. Поэтому рассматривают метрики, учитывающие оба типа ошибок.

Точность, полнота, специфичность и F1-score

Точность (precision) показывает, как сильно мы можем доверять классификатору, если он предсказывает объект положительного класса.

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

Полнота (recall) отображает, какую долю положительных объектов распознает классификатор.

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

Специфичность (specificity) является полным аналогом точности, но только при предсказании объекта отрицательного класса:

$$specifity(a, X) = \frac{TN}{TN + FP}$$

Часто хочется следить одновременно за точностью и полнотой. Для этого нужно обе метрики объединить в одну. Так, можно использовать гармоническое среднее, которое называется ***F1-мерой.***

$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

- Выводы
  Если набор данных сбалансирован, т. е. распределение по классам примерно равномерное, то в качестве метрики обычно используют accuracy. Иначе нужно

учитывать разбалансированность классов. Так как на практике часто хочется учесть и точность, и полноту, то можно использовать F1-меру. Кроме того, полезно посмотреть на площади под графиками «точность — полнота» (PR-кривая) и ROC-кривой. О них будет рассказано в следующем лонгриде.