Лекция 10. Метрики качества классификации и регрессии

Мы довольно детально обсудили какие существуют алгоритмы для решения задачи обучения с учителем, рассмотрели какие у них есть параметры и на что стоит обращать внимание при обучении той или иной модели. Но при этом мы пока только вскользь затронули тему того, как оценивать качество работы модели. Давайте исправим это!

И в этот раз начнем с задачи регрессии.

Регрессия

Как мы помним, в данном случае X- множество описаний объектов, Y - множество вещественных. Существует неизвестная *целевая зависимость* - отображение $y^*: X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X_m = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$. Требуется построить алгоритм $f: X \to Y$, способный посчитать y на произвольном объекте $x \in X$.

Абсолютная ошибка

R^2, RMSE, MSE

Начнем с одной из самых популярных метрик для задачи регрессии - MSE. Мы её уже неоднократно рассматривали и вот её общая формула:

$$MSE(y_{true}, y_{model}) = \frac{1}{N}\sum_{i=1}^{N}(y_{true_i} - y_{model_i})^2$$

Как мы можем заметить, мы возводим в квадрат отклонение прогноза модели от фактического значения, что может приводить к переполнениям памяти в случае больших значений признаков или наличия аномалий в данных (т.е. объектов, которые сильно отличаются от общего тренда), поэтому, для того, чтобы привести к размерности самих данных, часто используют метрику RMSE, которая является квадратным корнем из MSE

$$RMSE(y_{true}, y_{model}) = \sqrt{MSE(y_{true}, y_{model})} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_{true_i} - y_{model_i})^2}$$

Недостатком этих метрик является неограниченность сверху и как следствие, невозможность указать границ для “хорошего” и “плохого” алгоритмов. Но есть некоторые подходы, которые позволяют получить понимание по качеству:

- Сравнивать ошибку с ошибкой baseline-модели. В данном случае в роли простой модели используется прогнозирование константой - средним или медианой, в зависимости от характеристик распределения целевой переменной
- Перейти к показателю R^2:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_{true_i} - y_{model_i})^2}{\sum_{i=1}^{N}(y_{true_i} - \bar{y})^2}$$

MSE наиболее применим, когда большие ошибки для нас действительно неприемлемы, т.е. нам важно настроить модель на все точки (и аномальные, если вдруг они присутствуют в наборе данных), т.е. квадратичный штраф за них позволяет тонко настраивать модель под такие объекты. Но с другой стороны такое свойство этой метрики не позволяет напрямую сравнивать 2 модели, т.к. невысокая ошибка одной модели может быть за счет хорошей подстройки под малую долю аномальных точек, при этом модель с бОльшей ошибкой более точно улавливает паттерны на “обычных” данных.

MAE

Для исключения квадратичной зависимости, можно также использовать абсолютную ошибку:

$$MAE(y_{true}, y_{model}) = \frac{1}{N}\sum_{i=1}^{N}|y_{true} - y_{model}|$$

Однако в таком случае мы лишаемся превосходного свойства MSE - дифференцируемости этого функционала на $\mathbb{R}$.

Относительная ошибка

Как мы уже убедились, предыдущие метрики хороши, когда мы хотим как можно ближе подогнать модельные значения под исходные данных, вне зависимости от того, аномальная это точка или нет, но сложность интерпретации такого подхода в невозможности понять, какая из моделей лучше при одинаковой метрике.

Проще говоря, ошибиться на 1 при абсолютном значении целевой переменной 10 это не то же самое, что ошибиться на ту же 1 при фактическом значении 100. И тут мы плавно подходим к относительным ошибкам.

MAPE, SMAPE

И начнем с самой популярной метрики - **MAPE** (**mean absolute percentage error**). Т.к. мы говорим про относительные ошибки, то очевидно, что в данном случае мы должны посчитать отношение чего-то к чему-то. MAPE - это дельта между фактом и прогнозом, деленная на целевое значение.

$$MAPE(y_{true}, y_{model}) = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_{true_i} - y_{model_i}|}{|y_{true_i}|}$$

В данном случае возникает вполне закономерный вопрос: а что делать, когда целевое значение =0? Можно придумать некоторые ухищрения в виде 100% отклонения, но это может сильно помешать модели, поэтому есть ещё одна метрика, название которой **SMAPE** (**symmetric mean absolute percentage error**):

$$SMAPE(y_{true}, y_{model}) = \frac{1}{N}\sum_{i=1}^{N}\frac{2|y_{true_i} - y_{model_i}|}{y_{true_i} + y_{model_i}}$$

WAPE

Но и метрика MAPE/SMAPE также неидеальна и это проявляется при работе с временными рядами. Например, модель не учитывает некоторую сезонность и в некоторые месяцы внутри года дает очень плохой прогноз, при этом в остальные MAPE на уровне 95%. Но при этом когда мы посчитаем среднее отклонение на периоде всего года, то метрика будет сильно скошена. Поэтому лучше использовать ещё один вариант относительной ошибки - **WAPE (weighted average percentage error)**:

$$WAPE(y_{true}, y_{model}) = \frac{\sum_{i=1}^{N} |y_{true_i} - y_{model_i}|}{\sum_{i=1}^{N} |y_{true_i}|}$$

Оптимизация метрик

С общими описаниями метрик понятно, но как же оптимизировать эти метрики? Ведь основная задача при обучении модели - свести это отклонение к 0. Пусть мы выбрали в качестве метрики качества модели некоторый функционал $L(b(X), y)$. Тогда при обучении модели мы минимизируем функционал L. Как мы уже отметили, **MAE** не дифференцируема, так как там присутствует модуль. Но это ограничение очень легко можно пройти - возвращать в точке 0 значение производной 0.

Для минимизации прочих метрик мы можем использовать любые техники по решении оптимизационной задачи, МНК до генетических алгоритмов.

Классификация

Пусть X- множество описаний объектов, Y - конечное множество меток классов. Существует неизвестная *целевая зависимость* - отображение $y^* : X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X_m = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$. Требуется построить алгоритм $f : X \to Y$, способный классифицировать произвольный объект $x \in X$.

Бинарная классификация

Accuracy

Начнем с бинарной классификации, т.е. предполагается, что ответ модели может быть 0 и 1, или -1 и +1. И конечно же у нас есть некоторый алгоритм, качество которого мы и хотим оценить на выборке. В предыдущих лекциях мы часто пользовались метрикой accuracy (точность) - это доля объектов, на которых модель дает правильный ответ.

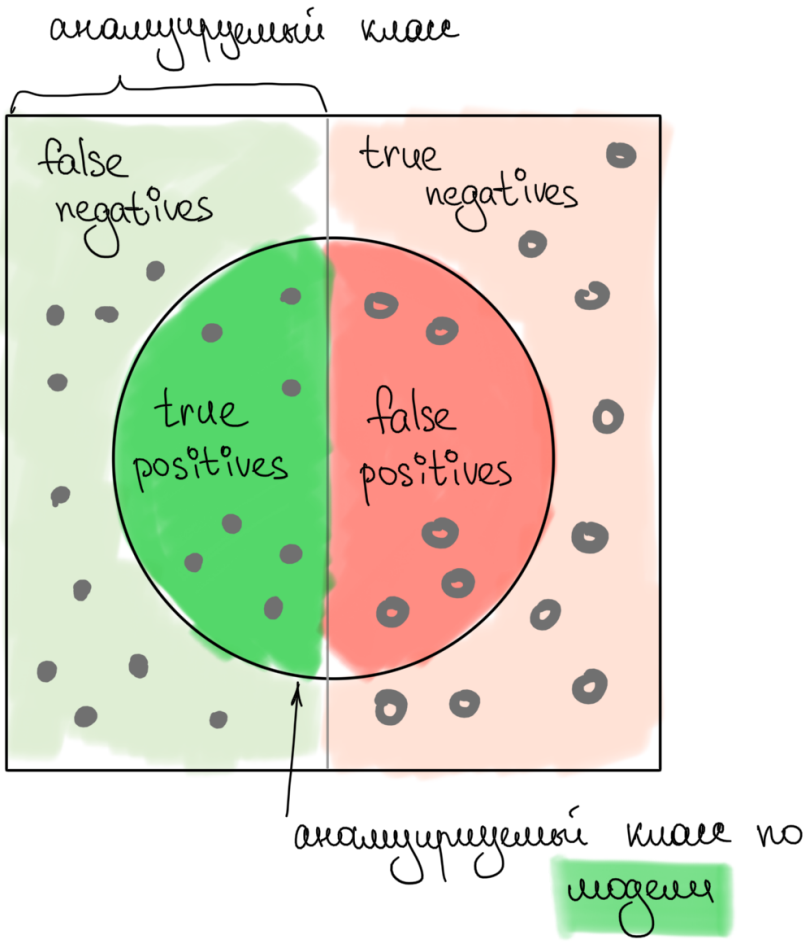
$$Accuracy(y_{true}, y_{model}) = \frac{1}{N} \sum_{i=1}^{N} I[y_{true_i} = y_{model_i}]$$

Но ограничиваться только этой метрикой конечно же не стоит. Например, эта метрика будет давать некорректные результаты на несбалансированных выборках,

т.е. выборках, где количество одного класса сильно больше другого. Пусть в выборке 100 объектов, из них 90 относятся к классу 0, 10 относятся к классу 1. Подобный дисбаланс можно встретить в банковском секторе (количество тех, кто не вернул кредит обычно меньше вернувших), медицине (количество больных пациентов обычно сильно меньше здоровых) и т.д. И как альтернатива алгоритму машинного обучения давайте выдавать прогноз как константу и будем возвращать 0. Безусловно, такое решение не имеет никакой ценности, но давайте посмотрим на метрики: accuracy в таком случае будет 90%, а если бы дисбаланс классов был более ярко выраженным, то и метрика была бы ещё больше!

Confusion Matrix

Бинарная классификация удобна тем, что здесь очень просто расписать возможные ситуации по соотношению исходной метки класса и прогноза модели



TP — **true positive:** классификатор верно отнёс объект к *положительному* классу.

TN — **true negative:** классификатор верно отнес объект не к *положительному* классу.

FP — **false positive:** классификатор неверно отнёс объект к *положительному* классу.

FN — **false negative:** классификатор неверно отнес объект не к рассматриваемому классу.

Таким образом, при помощи этих показателей можно посчитать ту самую долю правильных ответов. Она равна отношению числа верных ответов в нашей выборке, то есть сумме верных срабатываний и верных пропусков, к размеру всей выборки, то есть сумме всех четырех показателей.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision, Recall

Как же мы теперь можем использовать знания о возможных случаях соотношения фактической и модельной меток?

Очень важно понимать, что в разных задачах на первое место выходят разные метрики. Чаще всего этого зависит от редкости прогнозируемого события и возможных последствий, если мы неверно отнесем к *положительному* классу или же наоборот, неверно не отнесем. Например, в задаче медицины ложное срабатывание, т.е. отнесение здорового к классу больных не так критично как совершить ложный пропуск - мы не увидим больного человека и отнесем его к классу здоровых. Конечно, когда заболевание столь редкое, что наблюдается у очень малого процента людей, то и решение не будет приниматься только на основании одной модели - в любом случае, после прогноза будут назначены дополнительные исследования, однако несмотря на это, нам крайне важно находить всех больных людей на первом этапе. В подобных случаях предпочтительнее измерять две другие метрики качества вместо одной: точность и полноту.

$$Precision\ (\text{точность}) = \frac{TP}{TP + FP}$$

Точность показывает, насколько мы можем доверять классификатору, если он выдает ответ 1, т.е. *положительный* класс. Более формально она равна отношению числа верных срабатываний к количеству объектов, на которых алгоритм выдал ответ 1. Рассмотрим на примере. Исходно у нас 1000 здоровых пациентов (0) и 40 больных (1). Пусть классификатор выдает ответ 1 на 80 объектах. При этом 30 из них действительно относятся к классу 1, к классу «больные». Значит, точность равна 30 / 80, или 27,5 %. При этом точность константного классификатора, который все объекты относит к классу 0, равна нулю.

$$Recall\ (\text{полнота}) = \frac{TP}{TP + FN}$$

Вторая метрика называется «полнотой» или recall. Она показывает, как много объектов *положительного* класса алгоритм находит. Более формально она равна отношению числа верных срабатываний к общему размеру класса 1. В нашем примере в классе 1 – 40. Из них 30 мы отнесли к классу 1. Значит, полнота равняется 30 / 40, или 75 %. В данном случае довольно неплохая метрика! При этом полнота константного классификатора снова равна 0%, потому что он не находит ни один объект положительного класса. Если мы хотим минимизировать ложные срабатывания, то необходимо акцентироваться на precision. Если же приоритетнее минимизировать ложные пропуски, то максимизируем полноту.

F-мера

Как мы уже поняли, precision и reacll довольно неплохие метрики и позволяют нам оптимизировать работу алгоритма в зависимости от того, что нам необходимо максимизировать. Но для задачи оптимизации удобнее работать с одной метрикой. Тут по аналогии с ансамблями алгоритмов хочется каким-то образом агрегировать эти метрики. Самое просто - найти арифметическое среднее полноты и точности. Но использовать такой подход неоптимально: пусть у нас есть некоторый очень простой алгоритм, который для всех объектов дает ответ 1. В этом случае его полнота равна 100 %, а точность – 10 % в случае, если положительных примеров в выборке – 10 %. Среднее арифметическое в таком случае – 55 %. А теперь предположим, что мы построили ещё один алгоритм, у которого и точность, и полнота составляют 54 %. Этот алгоритм намного сильнее предыдущего, но при этом среднее арифметическое меньше! Решить эту проблему можно при помощи гармонического среднего или F_1 – мера. Чтобы посчитать F_1 – мера нужно

найти отношение произведения точности и полноты к их сумме и умножить это на 2.

$$F_1 = \frac{Recall * Precision}{Recall + Precision} = \frac{TP}{TP + \frac{FP+FN}{2}}$$

Как это поменяет наши метрики? При таком подходе константный получает метрику 18%, разумный – 54 %. Теперь это существенная разница и ровно то, чего мы и добивались! F-мера является стандартом в машинном обучении для усреднения точности и полноты.

F_1 − мера является частным случаем F_β − меры, для которой можно варьировать важноть *Precision* и *Recall*.

$$F_\beta = (\beta^2 + 1)\frac{Recall * Precision}{Recall + \beta^2 Precision}$$

PR-кривая

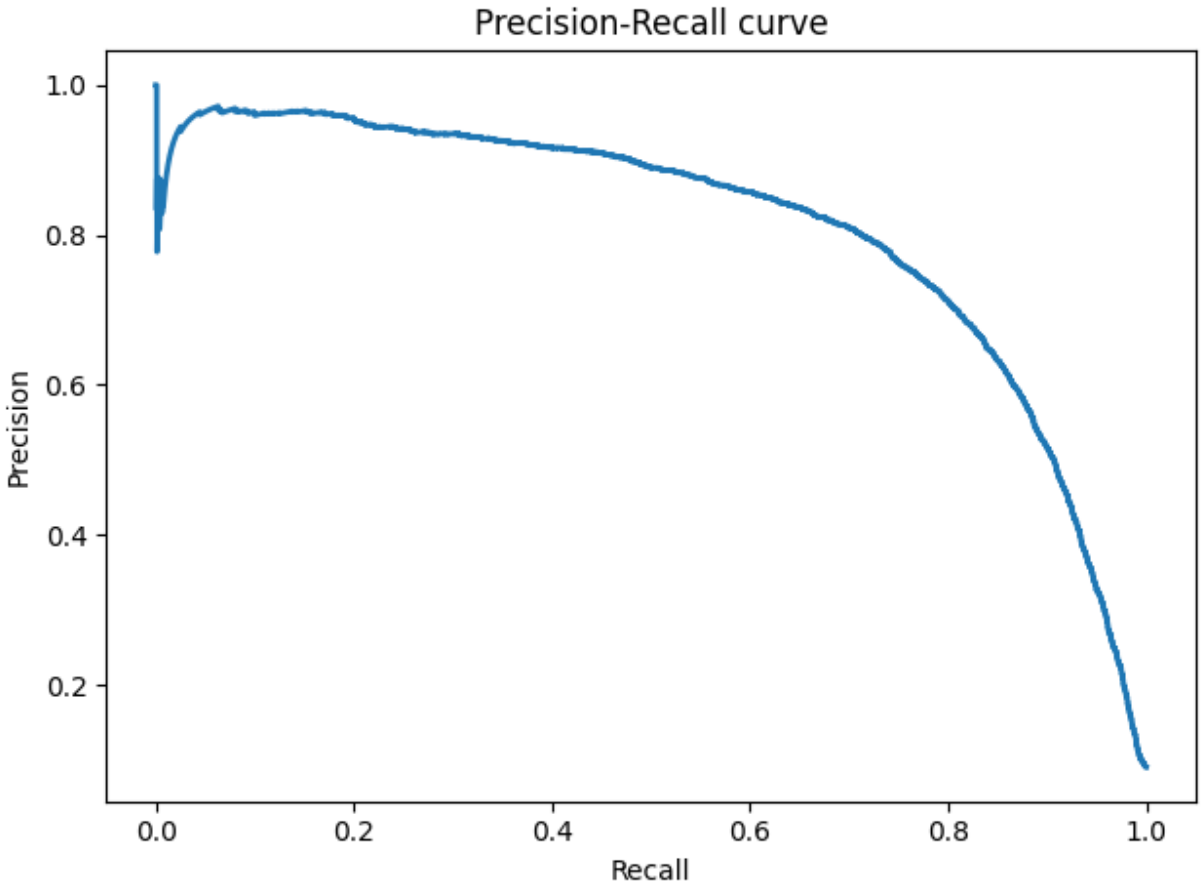
Но не только F-мера позволяет оценивать качество классификации!

Многие модели в качестве результата выдают не просто метку конкретного класса, а некоторую уверенность, что объект принадлежит тому или иному классу в виде вероятности. В случае бинарной классификации, модель может выдавать одно значение, которое на основании порогового значения для вероятности делает вывод о принадлежности конкретному классу.

$$f(x; \omega; t) = I[g(x, \omega) > t]$$

Как же теперь оценивать качество алгоритма?

Первый подход основан на кривой точности и полноты или $PR-$кривой. Чтобы построить ее, нам нужно отсортировать объекты по возрастанию их оценки принадлежности к нужному классу. После этого, варьируя порог, считаем долю и полноту классифицированных с этим порогом объектов.

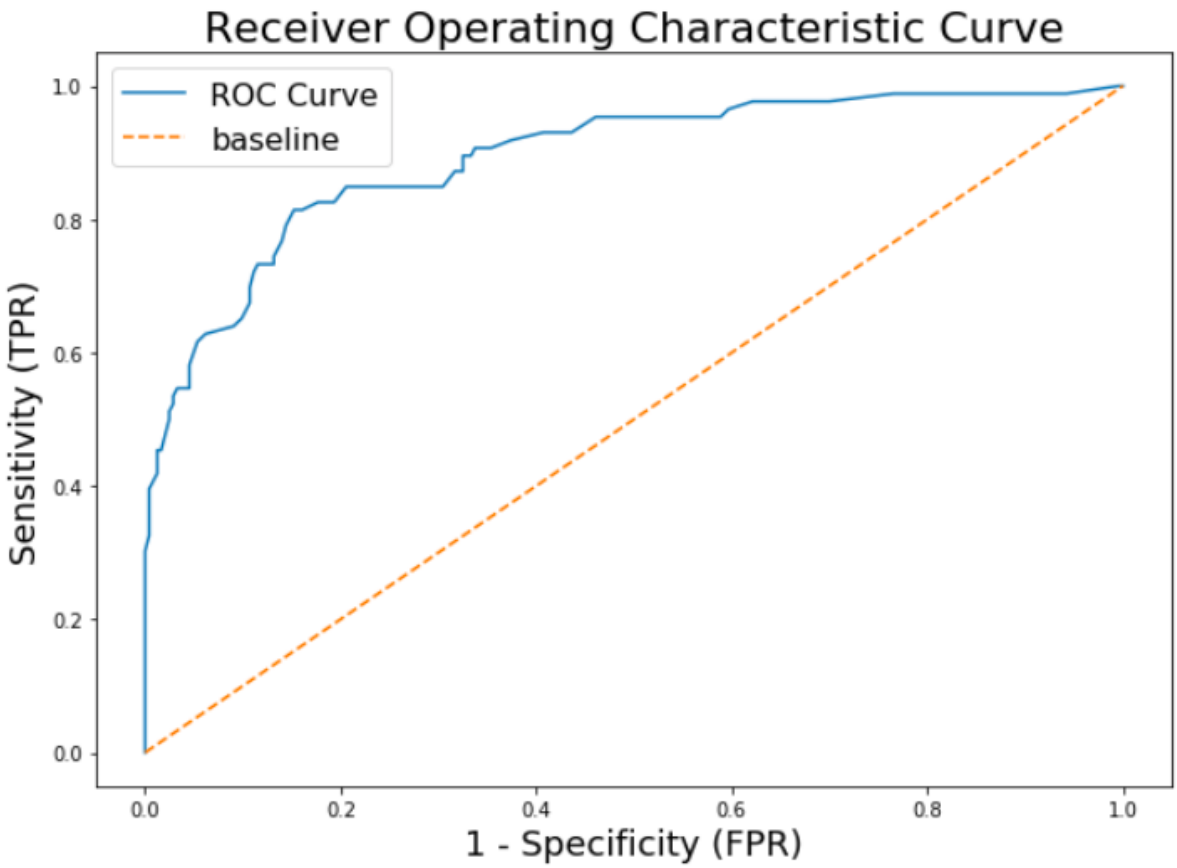


Соединив полученные точки, мы получаем кривую точность и полноты или PR-кривую. Левая точка этой PR-кривой всегда находится на координатах (0, 0), поскольку, когда мы ни один объект не относим к классу один, то и точность, и полнота равны 0. Единого правила для правой точки нет, полнота (ось X) всегда равна единице, а вот точность равна отношению числа объектов первого класса выборки к общему размеру выборки и это значение очень зависит от самого датасета. В случае линейно разделимой выборки существует вероятность найти такой порог, при котором, и точность, и полнота, равны 100 %. В этом случае кривая пройдет через точку (1, 1).

ROC-AUC

Но как и в случае с Precision и Recall, нам хотелось бы иметь одно число, которое характеризовало бы точность классификатора. В данном случае напрашивается использовать площадь по PR-кривой, ведь чем больше площадь, тем лучше работает алгоритм.

Логичным развитием PR-кривой является ROC-кривая.



По оси X отложена $FPR(FalsePositiveRate)$ или **Specificity** - доля ошибочных положительных классификаций - отношение числа ложных срабатываний к мощности нулевого класса.

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

Вторая ось - $TPR(TruePositiveRate)$ или **Sensitivity** - доля правильных положительных классификаций - отношение числа верных срабатываний к размеру первого класса.

$$FPR = \frac{FP}{FP + TN}$$

График всегда стартует с точки (0,0), когда мы ни одну точку не относим к классу 1. Дальше отнесем первый объект с максимальной оценкой к классу один. Это будет верная классификация, поэтому доля положительных срабатываний увеличится, доля ложных срабатываний останется нулевой.

Обе эти величины растут при уменьшении порога. Кривая в осях FPR/TPR, которая получается при варьировании порога, и называется **ROC-кривой** (**receiver operating characteristics curve**, сокращённо **ROC curve**).

И таким образом, если выборка идеально разделимая, то площадь под ROC-кривой будет равна единице. Если классификация по качеству совпадает со случайной, то кривая пройдет по диагонали, и площадь под ней будет равна примерно одной второй. Площадь под ROC-кривой тоже является хорошей метрикой качества, так и называется AUC-ROC или площадь под ROC-кривой.

AUC-ROC равен доле пар объектов вида (объект класса 1, объект класса 0), которые алгоритм верно упорядочил, т.е. предсказание классификатора на первом объекте больше:

$$AUC = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N} I[y_{true_i} < y_{true_j}] I^*[y_{model_i} < y_{model_j}]}{\sum_{i=1}^{N}\sum_{j=1}^{N} I[y_{true_i} < y_{true_j}]}$$

$$I[y_{true_i} < y_{true_j}] = \begin{cases} 0 & y_{true_i} \geq y_{true_j} \\ 1 & y_{true_i} < y_{true_j} \end{cases}$$

$$I^*[y_{model_i} < y_{model_j}] = \begin{cases} 0 & y_{model_i} > y_{model_j} \\ 0.5 & y_{model_i} = y_{model_j} \\ 1 & y_{model_i} < y_{model_j} \end{cases}$$

Обратите внимание, что доля ложных срабатываний нормируется на размер нулевого класса, доля верных срабатываний на размер первого класса. Благодаря этому площадь под ROC-кривой не зависит от соотношения классов. А это значит, что площадь под ROC-кривой останется такой же, даже если взять другую выборку с такими же свойствами, но с другим соотношением классов.

❗ **Интерпретации ROC-AUC метрики.**
Эта метрика равна вероятности того, что случайно взятый объект положительного класса получит оценку принадлежности к положительному классу выше, чем случайно взятый объект негативного класса.

❗ **Недостатки ROC-AUC**.
Precision Recall-кривая зависит от точности и полноты. При этом Precision нормируется на количество объектов, которые классификатор отнес к положительному классу. Из-за этого Precision зависит от соотношения классов. Если одного из классов выборки станет больше, то и Precision может измениться.

Мультиклассификация

Помимо бинарной классификации, конечно мы сталкиваемся с задачами мультиклассификации - когда множество меток классов состоит из 3 и более элементов. Для таких случаев необходима трансформация предыдущих метрик. Во-первых, мы по-прежнему можем пользовать Матрицей неточностей, только теперь она будет бОльшей размерности. Но в более простой постановке мы можем разбить задачу мультиклассификации на k задач **один против всех,** т.е. выделение конкретного класса от остальных. В таком случае мы можем построить k матриц ошибок. Но как мы уже привыкли, если есть несколько метрик, то нужно каким-то образом привести их к одной метрике, чтобы в дальнейшем передать как аргумент в оптимизатор. Для данной задачи есть 2 подхода:

❗ **Макроусреднение -** усреднение Precision и Recall, посчитанные отдельно для каждого классификатора

❗ **Микроусреднение -** расчет Precision, Recall, F-меры на усредненной матрице ошибок, которая построена на элементах матрицы ошибок **(TP, FP, TN, FN)**, посчитанные отдельно для каждого классификатора.

$$FP = \frac{1}{N}\sum_{i=1}^{k} FP_i$$

На первый взгляд может показаться, что никакой разницы между двумя этими подходами нет. Действительно это может давать очень похожие результаты, основные отличия будут проявляться при любимом нами дисбалансе классов. Рассмотрим на примере бинарной классификации, когда есть минорный класс, мощность которого обозначим **S**. Тогда метрики **TP** и **FN** этого класса будут не больше **S,** тоже довольно небольшими. Вероятнее всего, **FP** также мало, т.к. при дисбалансе классов модель не будет предсказывать редкий класс слишком часто.

Микроусреднение сделает вклад маленького класса в общую метрику незаметным. А в макроусреднении среднее считается уже для нормированных величин, так что вклад каждого класса будет одинаковым. Поэтому, при выборе способа усреднения результатов, в первую очередь стоит обратить внимание на наличие дисбаланса классов.

Оптимизация метрик

Precision, Recall

Невозможно оптимизировать напрямую, т.к. эти метрики нельзя рассчитать на одном объекте, а затем усреднить. Они зависят от того, какими были правильная метка класса и ответ алгоритма на всех объектах. Пусть модель обучена на стандартную для классификации функцию потерь. Но результатом модели являются не метки классов, а вероятности принадлежности классам. И только после получения вероятностей, используя пороговое значение вероятности для каждого класса, мы можем привести к метке класса. Итого, варьировать мы можем тот самый порог, чтобы максимизировать наши метрики

ROC-AUC

AUC не дифференцируема, поэтому оптимизировать напрямую, используя численные методы, не получится. Напомним, что метрика AUC считает долю верно упорядоченных пар. Значит от исходной выборки можно перейти к выборке

упорядоченных пар объектов. На этой выборке ставится задача классификации: метка класса 1 соответствует правильно упорядоченной паре, 0 — неправильно. Новой метрикой становится accuracy — доля правильно упорядоченных пар. А теперь применяем тот же подход, что и для Precision и Recall: варьируем порог принятия решения для максимизации метрики.

Подытожим

Для задач регрессии существуют абсолютные и относительные ошибки. Более предпочтительно считать относительные, т.к. они дают более полную картину о качестве работы модели, но в то же время абсолютные ошибки позволяют оценить, на сколько в абсолютных единицах ошибается модель.

Для классификации решение в лоб - вычислять долю верных ответов. Но такую метрику сложно интерпретировать на несбалансированных выборках и она не учитывает важность ошибок на малых и больших значениях датасета. Поэтому полезнее вычислять точность и полноту. Они позволят балансировать между ложными срабатываниями и ложными пропусками. Эти метрики можно агрегировать F-меру при помощи расчета гармонического среднего.

При переходе от меток классов к вероятностям принадлежности класса, пользуются ROC или PR-кривыми. AUC - площадь под ROC-кривой имеет разные интерпретации, ее максимум всегда равен единице, но для несбалансированных классов лучше использовать PR-кривую.