**Метрики в машинном обучении (https://dzen.ru/a/YVgbRCGCpTrNKTph)**

О метриках можно услышать не только в мире машинного обучения. Они позволяют численно или в виде графиков отобразить качество работы той или иной системы. Например, вы подняли веб-сервер, и вам скорее всего будет интересно знать, сколько запросов обрабатывает ваш сервер в течение некоторого отрезка времени, чтобы понимать далека ли нагрузка от предельной, при которой ваш сервер упадет.

В машинном обучении метрики возникают в тот момент, когда вы думаете о том, как вы будете отвечать перед начальством на вопрос: “Насколько качественно работает моя новенькая модель?”. Например, при разработке модели, делающей предсказания банковского скоринга идеальнее всего понимать перед выпуском модели сколько же модель вам принесет денег. Или хотя бы оценить, насколько больше вы бы заработали в случае, если перейдете на новую модель. Однако, это обычно очень сложно. Поэтому есть некоторый набор достаточно популярных метрик. И, конечно, чем метрика выше - тем скорее всего больше денег эта модель принесет.

В этой статье я хотел бы поделиться наиболее известными метриками. Ко всему прочему есть некоторая проблема, или точнее путаница. В машинном обучении и в вычислительной математике есть известная проблема - hell of dimension. Здесь хочется провести аналогию и сказать, что в метриках есть проблема hell of naming - когда одна и та же вещь имеет несколько названий.

Итак, список наиболее популярных метрик:

* Accuracy
* Precision
* Recall
* ROC AUC
* F1 score
* Confusion matrix
* Precision\_recall\_curve

Итак, у вас некоторая задача бинарной классификации. И вы уже обучили некоторый классификатор. Классификатор по закону жанра отдает вероятность от 0 до 1 принадлежности одному из классов. Чтобы превратить эту вероятность в класс, нужно установить некоторый порог, выше которого будем считать, что объект 1-го класса, а ниже которого - объект нулевого класса. Такой процесс называется *отсечением по порогу*или*thresholding.*

**Accuracy**

Что первое придет вам в голову, когда вы захотите измерить качество модели сразу же после ее обучения? Скорее всего это будет желание узнать долю правильно предсказанных объектов. И так мы придем с вами к метрике *accuracy.*Давайте за T (*от* *англ. True)*обозначим количество объектов на которых наша модель верно предсказала ответ, а за F (*от англ. False)* - количество объектов, на которых наша модель ошиблась. Тогда accuracy = T / (T+F).

Достоинство этой метрики в ее простоте, зато недостатков полно.

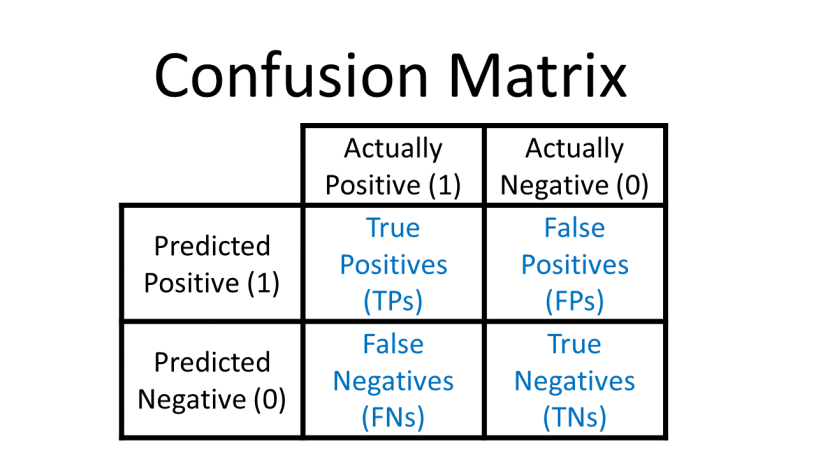
* Вы не знаете, как качественно модель предсказывает каждый из классов
* Метрика получается практически бесполезной в случае несбалансированной выборки. Например, вы решаете задачу “Есть ли на фотографии кошка”, и общий объем датасета у вас 1000, а кошек в нем только на 10 фотографиях. С этой точки зрения классификатор, выдающий на любую картинку “Нет, кошки нет”, отработает с accuracy = 99%.

**Несбалансированные данные**

Как же оценивать классификатор в случае несбалансированных данных? Прежде всего предлагаю поговорить о 4-х часто используемых в статистике величинах: True Positive (TP), False Positive (FP), True Negative (TN) и False Negative (FN).

* Первая буква означает верно ли вы предсказали класс
* Вторая буква означает предсказанный вами класс. То есть FP - (False Positive) говорит о том, что ваш классификатор на входной объект сказал, что это “Кошка” (Positive), но это оказалось не кошка - классификатор ошибся (False).
* Картинка будет относиться к TP, если на картинке была кошка и классификатор так же сказал, что на картинке есть кошка
* Картина будет относиться к TN, если на картинке не было кошки и наш классификатор так же сказал, что кошки на картинке нет.
* Будет относиться к FP, если на картинке кошки не было, а классификатор сказал, что она там есть
* Будет относиться к FN, если на картинке кошка была, а классификатор сказал, что её там нет.

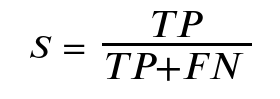
Таблица, которая состоит из этих 4-х чисел называется **confusion matrix.**



В принцпе, чем это не метрика? Глядя на неё, можно в голове представить пропорции разбиения и понять, насколько классификатор хорош. Идеальная *confusion-matrix* - с единицами на главной диагонали и нулями на другой диагонали.

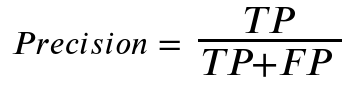
**Recall. Sensitivity. True Positive Rate (TPR). Полнота**

Довольно тривиальная метрика, которая может прийти на ум, это то насколько наш классификатор хорошо находит экземпляры класса. Иными словами насколько наш классификатор чувствителен к экземплярам класса кошка. Такая метрика имеет три названия на английском *Recall,* *Sensitivity, True Positive Rate*. На русском одно название - *полнота*. Формула этой метрики следующая:



**Точность (Precision)**

Чувствительность - это хорошая метрика, но она не показывает, насколько наш классификатор лажает и классифицирует объекты на которых нет кошек, как объекты с кошками (FP). Например, у классификатора, который будет говорить, что на всех картинках есть кошка (*назовем его бесполезный классификатор 2*), чувствительность будет 1. Но как увидеть численно, что классификатор - полная лажа? Да тоже очень просто - посмотреть на точность предсказаний класса кошки. Так появляется метрика Precision:

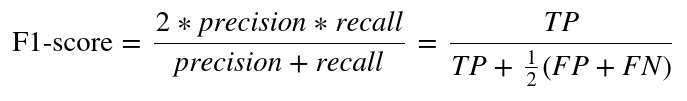


Если как в случае с несбалансированным датасетом на 1000 картинок, только 10 будут с кошками, то наш *бесполезный классификатор 2*, будет иметь precision 0.1%.

В целом, балансируя между precision и recall уже можно добиться неплохих результатов для решения бизнесовой задачи. Например, посмотрим на задачу прогнозирования результатов спортивных матчей. В этой ситуации для вас precision будет играть ключевую роль, а sensitivity будет играть роль второстепенную. Пусть ваш классификатор будет находить только 30% побед, но вам важно будет, что бы в этих 30% он имел точность (в смысле precision) не ниже 90%. Понижая порог, вы увеличиваете чувствительность, но неизбежно будете терять в точности предсказаний.

**F1-Score**

Итак, precision показывает точность предсказаний некоторого класса, а recall - охват этого класса. Как в одной метрике вместить эти две? Взять гармоническое среднее, и получим метрику *f1-score.*

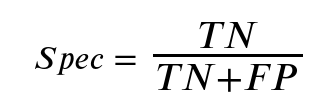


**ROC-AUC**

Но давайте погрузимся в мир темной лошадки по имени ROC-AUC. Прежде чем о нем говорить - поговорим о еще парочке метрик.

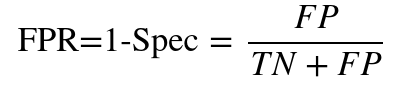
**Specificity и FPR**

Похожей на sensitivity есть метрика specificity. Она определяется как



По большому счету это полнота для негативного класса N. На эту метрику удобно посмотреть в рамках решения медицинских задач. Допустим у нас есть 1000 больных: 200 больны и 800 здоровы. Некоторый диагностический метод из 200 больных верно классифицировал 180(**TP**)человек, как больных, а 20 (**FN**) ошибочно классифицировал здоровыми. Из 800 здоровых, 700 (**TN)** он отнес к здоровым, а 100(**FP**) ошибочно классифицировал как больных. И специфичность показывает по сути, как хорошо метод диагностирует то, что человек здоров. В приведенном примере специфичность равна 700/800=7/8

Ещё есть метрика FPR, которая по сути равна FPR = 1 - Specificity. В нашем случае FPR=1/8. О чем это нам говорит? Это говорит о том, что в случае, если классификатор говорит, что среди всех здоровых, 12.5% будут помечены как больные.



**ROC-AUC собственной персоной**

В чем можно найти недостаток предыдущих метрик? В том, что мы уже оперируем оттрешхолженными числами. Но, что если есть два классификатора. Для того чтобы их сравнить, нужно для каждого подобрать некоторый оптимальный трешхолд, а после сравнивать выше перечисленные метрики. Но зависимость от трешхолда как бы намекает на то, что он может быть найден не оптимально. В общем-то метрика ROC-AUC решает эту проблему и позволяет сравнить качество работы двух классификаторов без необходимости отсекать по порогу.

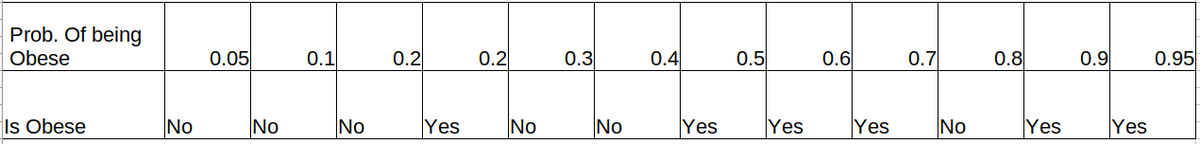
ROC-AUC является аббревиатурой двух вещей:

* ROC - Receiving Operating Characteristic
* AUC - Area Under the Curve.

ROC - это некоторая кривая, принцип построения которой я изложу ниже. AUC - это площадь под этой кривой.

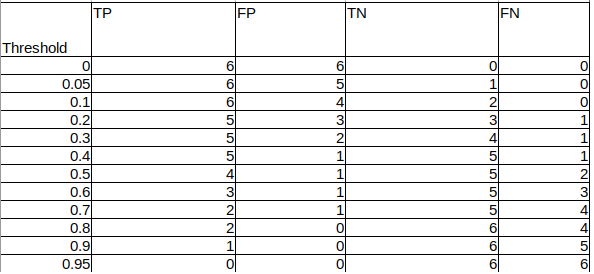
Итак, как же строится ROC кривая? Если коротко - то это график, на оси X которого расположена FPR, а на оси Y - TPR.

Давайте, на примере. Пусть у нас есть 12 мышей. В таблице ниже выбраны эти 12 мышей разбиваются на две группы по 6 мышей. Одна группа страдает ожирением, а вторая - нет. В верхнем ряду таблицы приведены вероятности, который дал некоторый классификатор, что мышь страдает ожирением. Видим, что наш классификатор оценил мышку, не страдающую ожирением, как высоковероятно страдающую ожирением - 80% (третья с конца).



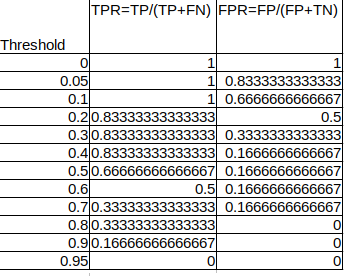
Классификация мышей, страдающую ожирением и не страдающую.

Теперь выпишем следующие трешхолды: 0; 0.05; 0.1; 0.2; 0.3; 0.4; 0.5; 0.6; 0.7; 0.8; 0.9; 0.95. Будем считать, что мышка была классифицирована, как страдающая ожирением, если классификатор выдал вероятность p и p >= t, где t-threshold. И далее для каждого из перечисленных трешхолдов находим две метрики TPR и FPR. Для удобства я сделал таблицу, вписав туда числа TP, FP, TN, FN.



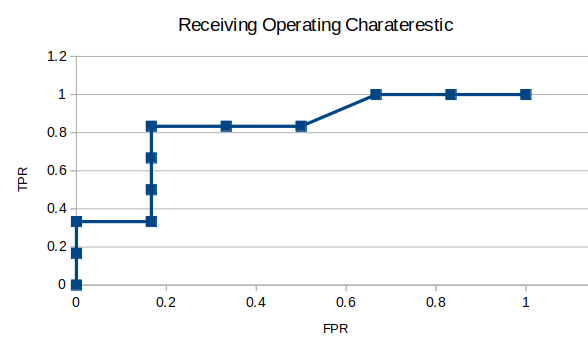
TP, TN, FP, FN

Теперь можно посчитать TPR и FPR на основе этой таблицы.



TPR, FPR

А теперь полученные точки просто разместим на графике и получим ROC-кривую.

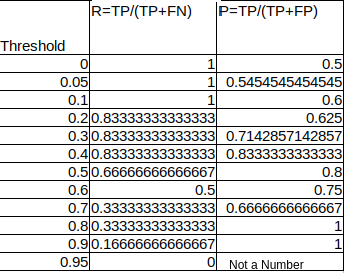


ROC кривая

Как сравнивать такие кривые? Какая кривая будет лучше? Все очень просто - та под которой площадь будет больше. И именно это кроется в названии AUC - Area Under the Curve.

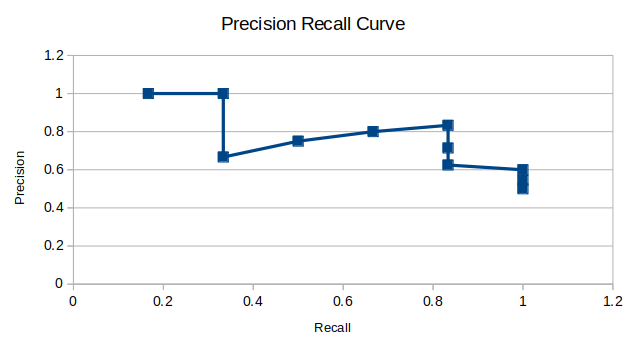
**Precision Recall Curve**

Помимо построения ROC кривой можно построить еще Precision Recall Curve. Аналогично для каждого трешхолда находим precision и recall и отмечаем точки на графике.



Precision и Recall для разных трешхолдов

В таблице напротив трешхолда в 0.95 для колонки "Precision" стоит выражение "Not a Number". Это происходит по тому, что при таком трешхолде у нас нет не одной мыши, для которой предсказание будет "Страдает ожирением".



PR Curve

**Заключение**

На этом об основных метриках все. К сожалению, нет метрики, которая позволит перед запуском модели предсказать прибыль или хотя оценить будет ли она больше/меньше. Но есть некоторая интуиция чем выше метрика новой модели, тем больше денег она принесет.