Семинар 7: Классификация

В этом семинаре мы подробнее поговорим про классификацию и метрики для неё. План будет таким:

- сформулируем задачу и поймём её специфику;
- поймём с помощью каких метрик можно оценить качество прогнозирования;
- попробуем разобраться какой смысл стоит за этими метриками.

Задача 1 (формулируем задачу)

Вася — меломан. Каждый день он смотрит на youtube музыкальные клипы и читает комментарии к ним. В один прекрасный день ему стало интересно, можно ли определить жанр клипа (рэп или попса) по его характеристикам: число комментов, лайков, характер комментов и тп.

1. К какому типу относится такая задача: классификация или регрессия? Почему?
2. Какие факторы из профилей вы бы использовали, чтобы её решить? Как бы вы оценивали качество итогового прогноза?

Задача 2 (метрики)

Бандерлог оценил три модели: нейросеть, случайный лес и KNN. Он построил на тестовой выборке прогнозы и получил три матрицы ошибок:

	$y = 1$	$y = 0$
$\hat{y} = 1$	80	20
$\hat{y} = 0$	20	80

	$y = 1$	$y = 0$
$\hat{y} = 1$	48	2
$\hat{y} = 0$	52	98

	$y = 1$	$y = 0$
$\hat{y} = 1$	10	20
$\hat{y} = 0$	90	10000

а) Найдите для всех трёх моделей долю правильных ответов. Чем плоха эта метрика?

б) Найдите для всех трёх моделей точность (precision) и полноту (recall)

в) Предположим, что целевая переменная y принимает значение 1, если заемщик вернул кредит и 0, если не вернул. Вы хотите научиться прогнозировать платежеспособность клиента. Какую из первых двух моделей вы бы выбрали в таком случае?

г) Предположим, что целевая переменная y принимает значение 1, если человек болен больной болезнью с болью и 0, если он здоров. Вы хотите спрогнозировать нужно ли человеку обследование. Какую из первых двух моделей вы б выбрали в этом случае?

Решение:

а) Ежели мы совершаем ошибку, то мы делаем это спрогнозировав $\hat{y} = 1$ там, где реально должно быть $y = 0$ либо наоборот, прогнозируя $\hat{y} = 0$ там, где реально должно быть $y = 1$. В остальных случаях всё ок. Наша классная табличка выглядит следущим образом:

	$y = 1$	$y = 0$
$\hat{y} = 1$	TP	FP
$\hat{y} = 0$	FN	TN

Долю правильных ответов можно посчитать как

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

Давайте проделаем эту несложную процедуру для вердиктов всех трёх алгоритмов, описанных выше.

$$\begin{aligned} Accuracy_1 &= \frac{80 + 80}{80 + 80 + 20 + 20} = 0.8 \\ Accuracy_2 &= \frac{48 + 98}{48 + 98 + 2 + 52} = 0.73 \\ Accuracy_3 &= \frac{10 + 10000}{10 + 10000 + 20 + 90} = 0.98 \end{aligned} \tag{1}$$

У этой метрики есть как минимум две существенные проблемы.

Первая проблема связана с несбалансированными выборками. Именно такая ситуация наблюдается в третьей табличке. Нулевой класс заметно перетягивает на себя выборку. Выходит, что если мы просто-напросто спрогнозируем, что все объекты в выборке нулевые, мы получим табличку

	$y = 1$	$y = 0$
$\hat{y} = 1$	0	0
$\hat{y} = 0$	100	10020

и $Accuracy = 0.99$. Наша модель показала более низкое качество, чем простое угадывание. Выходит, что наша модель абсолютно бесполезна.

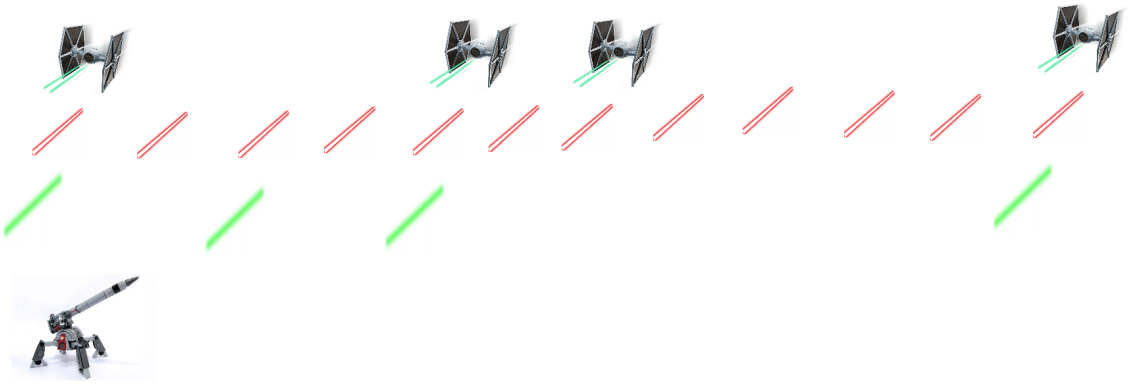
Чтобы «бороться» с этой проблемой, используется следующий факт. Пусть q_0 — доля объектов самого крупного класса, тогда доля правильных ответов для разумных алгоритмов $accuracy \in [q_0, 1]$, а не $[0.5, 1]$, как это можно было бы ожидать. Поэтому, если получается высокий процент правильных ответов, 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, представим себе линейное небо, которое бороздят вражеские самолёты. Мы стоим на земле, у нас есть пушка и снаряды. Сбивать вражеские самолёты можно руководствуясь двумя разными стратегиями:

Путь первый: стрелять по всему небу, куда только можно попасть. В таком случае точность наших выстрелов будет низкой, но зато мы собьём все самолёты, то есть добьёмся высокой полноты. Такая стратегия на картинке прорисована красными лазерными выстрелами из пушки.

Путь второй: стрелять поточнее, пореже. Тогда мы будем сбивать самолёты точно, потратим мало снарядов вхолостую, но собьём не все самолёты. Такая стратегия на картинке прорисована зелёными выстрелами из пушки.



Для разных задач бывают характерны разные стратегии. Если вернуться к нашей исходной таблице ошибок, то за точность классификатора будет отвечать первая строка

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}.$$

За полноту будет отвечать первы столбец

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

Посмотрим ещё один пример: если мы решаем задачу крединтого скоринга, нам нужно получить деньги назад после выдачи кредитов, иначе мы разоримся. Нам нужна модель, которая будет точно определять надёжного заёмщика. В этой ситуации для нас неважно покрыть все вражеские истребители снарядами (выдать кредиты каждому надёжному заёмщику), для нас важно сделать это точно. Поэтому основное внимание мы уделяем ошибке FP.

Если мы пытаемся найти больных больной болезнью с болью и отправить их делать дополнительные анализы, для нас страшнее FN ошибка. Если мы отправим лишнего человека на анализы, ничего страшного с ним не произойдёт. Если мы забудем проверить больного, он умрёт. Тут лучше добиться высокой полноты, при небольшой точности.

В разных ситуациях ошибки имеют разные цены. $\mathrm{Accuracy}$ не видит этого, поэтому на практике обычно используют $\mathrm{Precision}$ и Recall.

б)

$$\begin{aligned} \mathrm{Precision}_1 &= 0.8 & \mathrm{Recall}_1 &= 0.8 \\ \mathrm{Precision}_2 &= 0.96 & \mathrm{Recall}_2 &= 0.48 \\ \mathrm{Precision}_3 &= 0.33 & \mathrm{Recall}_3 &= 0.1 \end{aligned} \tag{2}$$

Вторая модель является очень точной, но в ущерб полноте. Третья модель очень плохая. Эти две метрики, в отличие от $\mathrm{Accuracy}$, позволили заметить этот факт.

в) 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.

Выше мы обсудили, что бизнес-специфика задачи диктует нам необходимость взять модель, где побольше точность.

г) Выше мы обсудили, что нам нужна модель, где побольше полнота.

Задача 3 (ещё немного метрик)

Бандерлог из Лога[1] ведёт блог, любит считать логарифмы и оценивать модели. С помощью нового алгоритма Бандерлог решил задачу классификации по трём наблюдениям и получил $b_i = \hat{P}(y_i = 1|x_i)$.

y_i	b_i
1	0.7
0	0.2
0	0.3
1	0.25

а) Найдите ROC AUC.

б) Постройте ROC-кривую.

в) Постройте PR-кривую (кривая точность-полнота).

г) Найдите площадь под PR-кривой.

д) Как по-английски будет «бревно»?

Решение:

а) В предыдущей задаче мы немного обсудили метрики классификации. Когда мы на практике оцениваем модель, она выплёвывает в нас не принадлежность объекта к классу в явном виде, а вероятности того, что наши объекты — единички.

Например, давайте думать в терминах оттока клиентов. Пусть наш сервис привлёк каких-то ребят в постоянные пользователи. Если они начнут унывать, им захочется свалить. Это называется оттоком. Давайте предположим, что модель выдаёт нам вероятность того, что человек решил приуныть. Если мы сможем понимать кто собрался приуныть, будем одаривать их ништяками и тогда они будут оставаться нашими клиентами.

Как понять, кто собирается приуныть, если модель выплёвывает на нас вероятности? Давайте выберем порог и будем считать, что все, у кого вероятность уныния $\geqslant 0.5$ — относятся к классу 1. Есть вероятность, что они решат свалить. Их и будем одаривать ништяками. В нем случае получатся прогнозы: $1, 0, 0, 0$. Если взять порог 0.3, получим прогнозы $1, 0, 1, 0$.

[1] деревня в Кадуйском районе Вологодской области

Как выбрать порог, кого одаривать ништяками. Если у нас большой бюджет, можно попробовать поставить низкий порог, чтобы добиться большой полноты. Если маленький бюджет, то давайте поставим порог так, чтобы борьба с унынием была поточнее. Выбор порога зависит от специфики бизнеса и от того, что от нас хотят.

Видно, что точность и полнота зависят от выбора порога. А хотелось бы, чтобы та метрика, по которой мы выбираем модель, от порога не зависила. Так рождаются идеи о метриках $\mathrm{roc}_\mathrm{a}\mathrm{uc}$ и $\mathrm{pr}_\mathrm{a}\mathrm{uc}$.

Представим себе два объекта: приунывший и нормальный. Представим себе, что модель предсказала нам вероятность уныния для первого объекта b_1 и для второго b_2.

$$\begin{aligned} y &= 1 \qquad & \widehat{P}(y=1) &= b_1 \\ y &= 0 \qquad & \widehat{P}(y=1) &= b_2 \end{aligned} \tag{3}$$

Если у нас хорошая модель, то явно $b_1 > b_2$. Иначе модель всё путает и говорит, что неунывающие приуныли. Давайте посмотрим как часто на нашей выборке такая путаница происходит и рассмотрим все возможные пары нулей и единичек. Всего будет четыре пары.

$$\begin{aligned} 0.7 &> 0.2 \qquad & &\mathrm{ok} \\ 0.7 &> 0.3 \qquad & &\mathrm{ok} \\ 0.25 &> 0.2 \qquad & &\mathrm{ok} \\ 0.25 &< 0.3 \qquad & &\mathrm{not\ ok} \end{aligned} \tag{4}$$

Видим, что модель ошиблась в упорядочивании один раз. $\mathrm{roc}_\mathrm{a}\mathrm{uc}$ — это доля пар, где модель оказалась права. В нашем случае это 0.75. $\mathrm{roc}_\mathrm{a}\mathrm{uc}$ принимает значения от 0.5 до 1, если её значения близки к 0.5, наш алгоритм ничем не лучше монетки, потому что он упорядочивает пары из унывших и нормальных абы как.

Такая метрика позволяет не привязываться к конкретному значению порога и видеть насколько классно у модели выходит упорядочивать пары объектов.

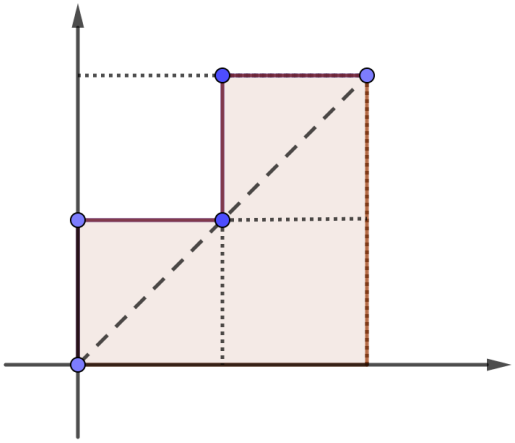
б) Величина, которую мы посчитали выше является площадью под roc-кривой (roc = receiver operating characteristic, иногда говорят «кривая ошибок»), с помощью которой часто визуализируют качество работы алгоритма. Когда говорять про roc-auc имеется в виду area under the curve.

Чтобы нарисовать ROC-кривую, надо взять единичный квадрат на координатной плоскости, разбить его на m равных частей горизонтальными линиями и на n – вертикальными, где m – число 1 среди правильных меток теста (в нашем примере $m = 2$), n – число нулей ($n = 2$). В результате квадрат разбивается сеткой на $m \times n$ блоков.

Отсортируем нашу табличку по значению вероятности, которую предсказала модель.

y_i	b_i
1	0.7
0	0.3
1	0.25
0	0.2

Теперь будем просматривать строки сверху вниз и прорисовывать на сетке линии, переходя их одного узла в другой. Стартуем из точки $(0, 0)$. Если значение метки класса в просматриваемой строке 1, то делаем шаг вверх; если 0, то делаем шаг вправо. Ясно, что в итоге мы попадём в точку $(1, 1)$, т.к. сделаем в сумме m шагов вверх и n шагов вправо.



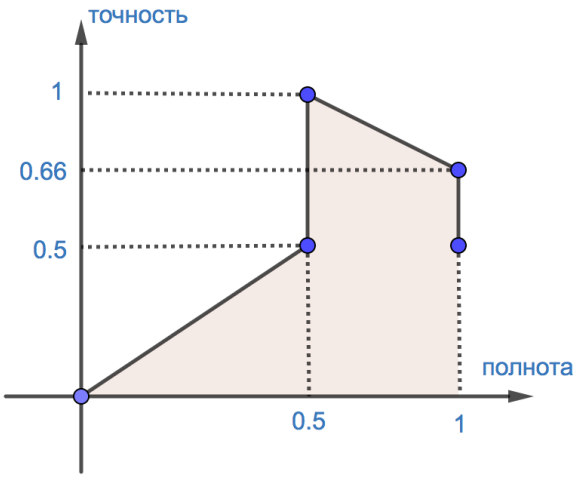
Сетка разбила квадрат на $m \times n$ блоков. Ровно столько же пар вида (объект класса 1, объект класса 0), составленных из объектов тестовой выборки. Каждый закрашенный блок соответствует паре (объект класса 1, объект класса 0), для которой наш алгоритм правильно предсказал порядок (объект класса 1 получил оценку выше, чем объект класса 0), незакрашенный блок – паре, на которой ошибся.

в) Давайте по оси x откладывать полноту, а по оси y точность. Давайте по очереди перебирать разные пороги и считать для них точность и полноту. Нанесём на картинку все полученные точки и соединим их. Это и будет PR-кривая.

Например, если мы возьмём порог $\geqslant 0.2$, тогда все объекты будут принадлежать к классу 1 точность составит 0.5, а полнота 1. Если мы возьмём порог $\geqslant 0.25$, тогда один объект будет нулевым, а остальные три единичными. Точность составит 0.66, а полнота 1. По аналогии получим точность и полноту при порогах $\geqslant 0.3$ и $\geqslant 0.7$.

порог	точность	полнота
0.2	0.5	1
0.25	0.66	1
0.3	0.5	0.5
0.7	1	0.5
0.8	0	0

Строим кривую!



PR- кривая всегда начинается из точки $(0, 0)$ и заканчивается в точке $(1, r)$, где r — это доля объектов первого класса в выборке.

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 pr-auc, или площадь под PR-кривой.

г) Посчитав площадь под кривой получим pr-auc. Будем делать это, считая площади треугольников и прямоугольников.

$$1 - 0.5^2 - 0.5^2 \cdot 0.5 - 0.33 \cdot 0.5 \cdot 0.5 = 0.542$$

Эта метрика подобно roc-auc не зависит от выбора порога и отражает способность модели правильно упорядочивать пары, но немножечко в другом плане. Эта метрика строится в осях точность и полнота. Из-за этого она чувствительна к дисбалансу в классах.

Можно посмотреть на метрику roc-auc немного иначе. Ввести две дроби

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

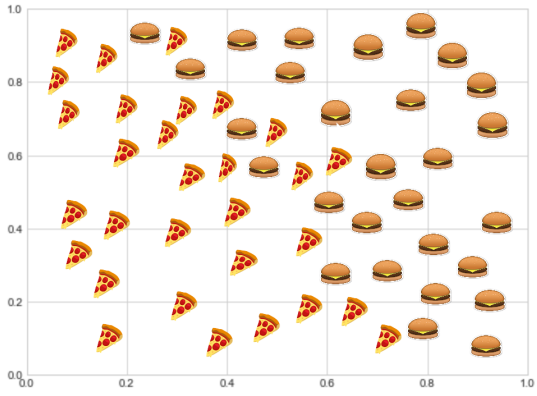
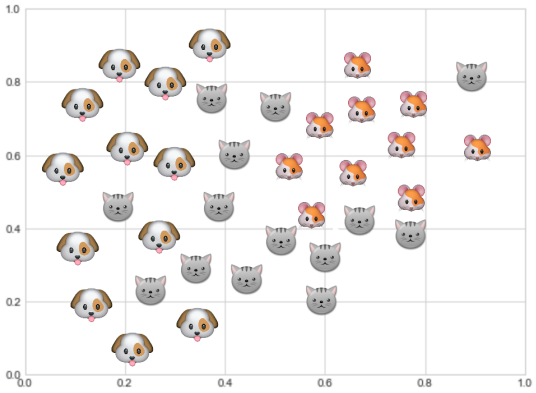
перебирать порог и в осях, соответсвующих TPR и FPR отмечать точки. В итоге получится ровно такая же кривая как у нас. Именно такое определение вы встретите в большинстве курсов по ML. Но любой нормальный человек сразу же забывает что означают эти FPRFRTPRPR, поэтому мы так делать не будем.

Единственный профит от такого определения в том, что сразу же видно, что roc-auc устойчив к дисбалансу в классах.

д) log

Задача 4 (классификация в картинках)

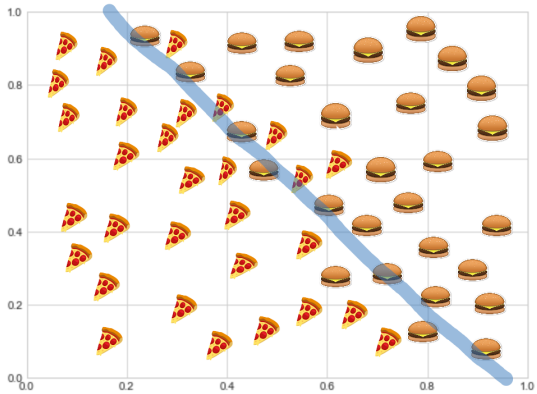
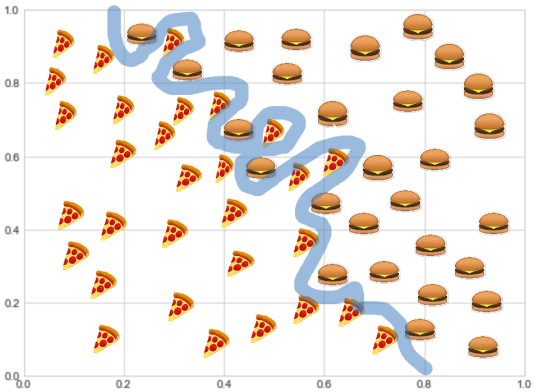
Нам нужно научиться отделять пиццу от бургеров, а также котиков от пёсиков и от мышек. Проведите на картинках линии, которые отделят одни классы от других. Да, это и есть машинное обучение. Но обычно кривые рисуем не мы, а компуктер.

Почему нельзя провести между пиццей и бургерами слишком подробную и извилистую границу? В чём проблема самого правого верхнего котика? Что такое переобучение? Как понять переобучились ли мы?

Решение:

Сначала обсудим бургеры и пиццу. Первый вариант: провести между ними прямую. Тогда мы в части случаев ошибёмся и признаем некотрые бургеры пиццей, а некоторые пиццы бургерами. Второй вариант: провести извилистую разделительную линию, которая чётко разграничит бургеры и пиццу. Вопрос: какой из этих двух вариантов лучше?

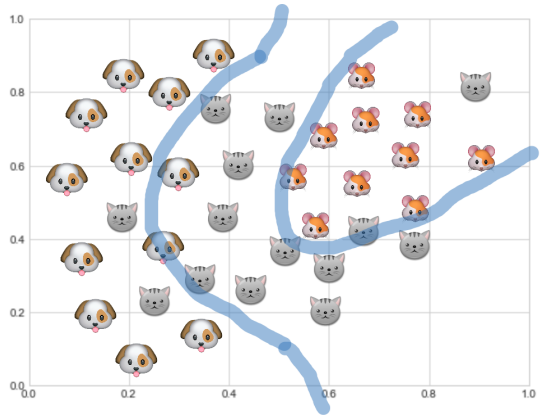
Если у нас в выборке оказались все пиццы и бургеры мира, и других быть не может, вторая граница нам подойдёт. Мы подстроимся под все особенности нашей гениральной пицце-бургерной совокупности и будем всегда чётко и безошибочно отличать одно от другого.

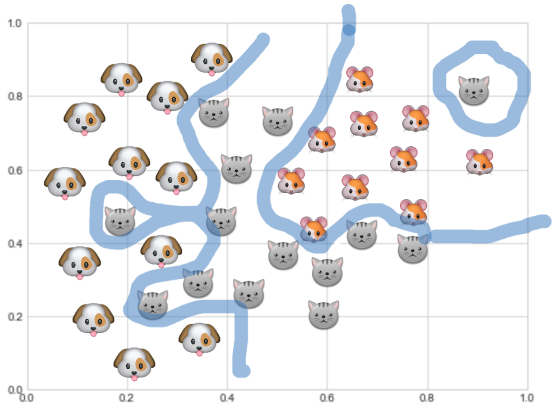
НО в нашем распряжении обычно находится не вся генеральная совокупность, а лишь какая-то её часть. Мы в выборке видим не все возможные варианты, и хотим обучить наш классификатор обощать. Если к нам попадает новая пицуля или бургер, классификатор должен адекватно сработать на них.

Скорее всего, пиццы, проникшие на территорию бургеров, обладают какими-то аномальными особенностями, на детекцию которых затачивать классификатор нет никакого смысла. Если мы попробуем сделать это, мы влезем на территорию бургеров, и на новых объектах, которые оказались обычными бургерами, будем делать ошибки, подумав, что это аномальные пиццы. Из-за этого лучше разграничить бургеры и пиццы простой линией, которая изображена на первой картинке.

Ещё раз, ещё раз. Если мы проведём подробную границу, мы заточим классификатор под особенности выборки, вместо того, чтобы научить его отличать пиццу от бургера в общем случае. Такие ситуации называются переобучением. И это главная головная боль людей, занимающихся машинным обучением. С переобучением у них идёт вечная борьба.

Теперь посмотрим на котиков, пёсиков и мышек. Снова мы можем провести границы между ними разными способами.





Снова мы можем провести более-менее простую границу и иногда ошибаться. Ну знаете, есть такие собаки мелкие, похожие на кошек. Или даже на мышек. И, если мы будем специфицировать границу под этих собак, мы начнём ошибаться на котах, так как подобные аномалии встречаются редко.

Основная проблема верхнего котика в том, что он аномальный. Каким-то образом он попал на территорию мышек. Выделять для него свою зону будет плохой идеей, так как в таком случае мы будем переобучать классификатор под конкретный выброс.

Осталось обсудить главный вопрос: как понять а не переобучились ли мы. Для этого обычно дробят выборку на две части: тренировочную и тестовую. На тренировочной учат алгоритм (в данном случае границу между классами), а на тестовой проверяют насколько хорошо он работает. Насколько часто алгоритм на тестовой части делает ошибку.

Если получается, что на обучающей выборке качество высокое, а на тестовой низкое — мы переобучились и вместо того, чтобы научить модель обобщать закономерности, существующие в данных, обучили его под особенности конкретной выборки. Если на тестовой выборке качество сравнимо с обучающей, значит мы научились извлекать какие-то реальные закономерности.

Бьюсь об заклад, что для простых линий, качество на тесте для бургеров и мышек будет выше, чем для сложных. Конечно же, простые границы оказываются хороши не всегда, но всегда имеет смысл сначала построить простую модель, а после сравнивать с ней сложные.

Ещё задачи!

Тут лежит ещё несколько задач для самостоятельного решения. Возможно, похожие будут в самостоятельной работе...

Задача 5

Бандерлог начинает все определения со слов «это доля правильных ответов»:

а) accuracy — это доля правильных ответов...

б) точность (precision) — это доля правильных ответов...

в) полнота (recall) — это доля правильных ответов...

г) TPR — это доля правильных ответов...

Закончите определения Бандерлога так, чтобы они были, хм, правильными.

Решение:

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

б) $\text{precision} = \frac{TP}{TP+FP}$

в) $\text{recall} = \frac{TP}{TP+FN}$

г) $TPR = \frac{TP}{TP+FN}$

Задача 6

Бандерлог обучил модель для классификации и получил вектор предсказанных вероятностей принадлежностей к классу 1.

y_i	b_i
1	0.9
0	0.1
0	0.75
1	0.56
1	0.2
0	0.37
0	0.25

а) Бинаризуйте ответ по порогу t и посчитайте точность и полноту для $t = 0.3$ и для $t = 0.8$.

б) Какой порог бы вы выбрали?

в) Постройте ROC-кривую и найдите площадь под ней.

Решение:

а) Построим в нашей табличке две колонки с прогнозами. Для удобства.

y_i	b_i	$t = 0.3$	$t = 0.8$
1	0.9	1	1
0	0.1	0	0
0	0.75	1	0
1	0.56	1	0
1	0.2	0	0
0	0.37	1	0
0	0.25	0	0

Построим для обоих порогов матрицы ошибок. Слева матрица, соответствующая порогу 0.3, справа матрица, соответствующая порогу 0.8.

	$y = 1$	$y = 0$
$\hat{y} = 1$	2	2
$\hat{y} = 0$	1	2

	$y = 1$	$y = 0$
$\hat{y} = 1$	1	1
$\hat{y} = 0$	2	3

Считаем для обоих случаев точность и полноту.

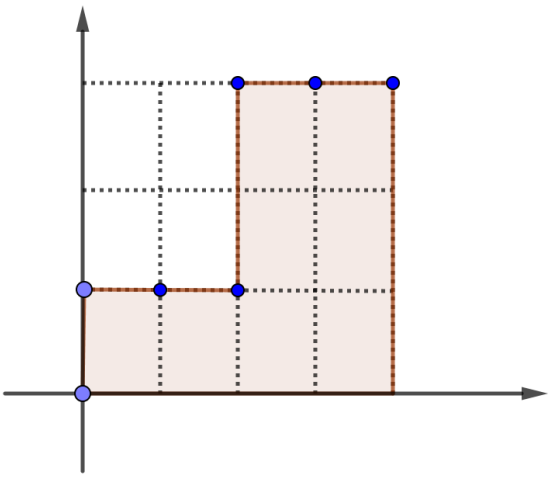
$$\begin{aligned} \text{Precision}_1 &= 0.5 \qquad & \text{Recall}_1 &= 0.66 \\ \text{Precision}_2 &= 0.5 \qquad & \text{Recall}_2 &= 0.33 \end{aligned} \tag{5}$$

б) Обе модели дают одинаковую точность при разной полноте. У первой модели полнота повыше, имеет смысл выбрать её (но это неточно, надо бы это проверить на большем числе данных).

в) Построим roc-кривую. Для этого отсортируем все наблюдения в нашей табличке по возрастанию.

y_i	b_i
1	0.9
0	0.75
1	0.56
0	0.37
0	0.25
1	0.2
0	0.1

Заведём сетку размера 4 на 3 и начнём делать шаги по ней так, как было описано во второй задаче.



Профит, получили roc-кривую. Видим, что площадь под ней равна $\frac{8}{12}$. Иным языком говоря, среди 12 пар ноликов и единичек, вероятности отсортированы так, как нам хотелось бы 8 раз.

Для первого наблюдения $(1, 0.9)$ все четыре пары верны. Для второго $(1, 0.56)$ будет одна ошибочная сортировка, для третьей $(1, 0.2)$ будет три ошибочных сортировки.