Статистический анализ данных (Data Mining)

Лекция 5. Оценка качества алгоритмов.

Смещение и разброс. Функционалы качества в задачах классификации и кластеризации

Московский авиационный институт
«МАИ»

22 сентября 2021 г.

Смещение и разброс

В машинном обучении присутствуют два главных источника ошибок: **смещение (bias)** и **разброс (variance)**

Эндрю Ын https://habr.com/ru/post/420591/

Александр Дьяконов https://dyakonov.org/2018/04/25/

Педро Домингос
https://homes.cs.washington.edu/~pedrod/papers/mlc00a.pdf

Смещение и разброс

Пусть целевая зависимость имеет вид

$$y(x) = f(x) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0; \sigma^2)$$

Мы строим алгоритм (в нашем случае полином фиксированной степени) $a(x)$ минимизируюший средний квадрат ошибки

$$\mathrm{M}[y(x) - a(x)]^2 \to \min_a$$

Матожидание берётся по всем данным (обучающим выборкам) и настройкам алгоритма.
Разбросом будем назвать дисперсию ответов алгоритмов $\mathrm{D}[a]$, а смещением — матожидание разности между истинным ответом и выданным алгоритмом: $\mathrm{M}[f - a]$.
Ошибку можно представить в виде:

$$\mathrm{M}[y(x) - a(x)]^2 = \sigma^2 + \mathrm{D}[a] + (\mathrm{M}[f - a])^2$$

Смещение и разброс

В статистике есть более точное определение для смещения и разброса

Пусть имеется выборка $Z_m = \{X_k, k = 1, \ldots, m\}$ и параметр θ, который требуется оценить.
Среднеквадратической погрешностью точечной оценки $\hat{\theta}_m$ называется величина

$$\Delta_m = \mathrm{M}\left[|\hat{\theta}_m - \theta|^2\right]$$

Теорема

Пусть $\theta \in \mathbb{R}^1$ и $\Delta_m < \infty$, тогда

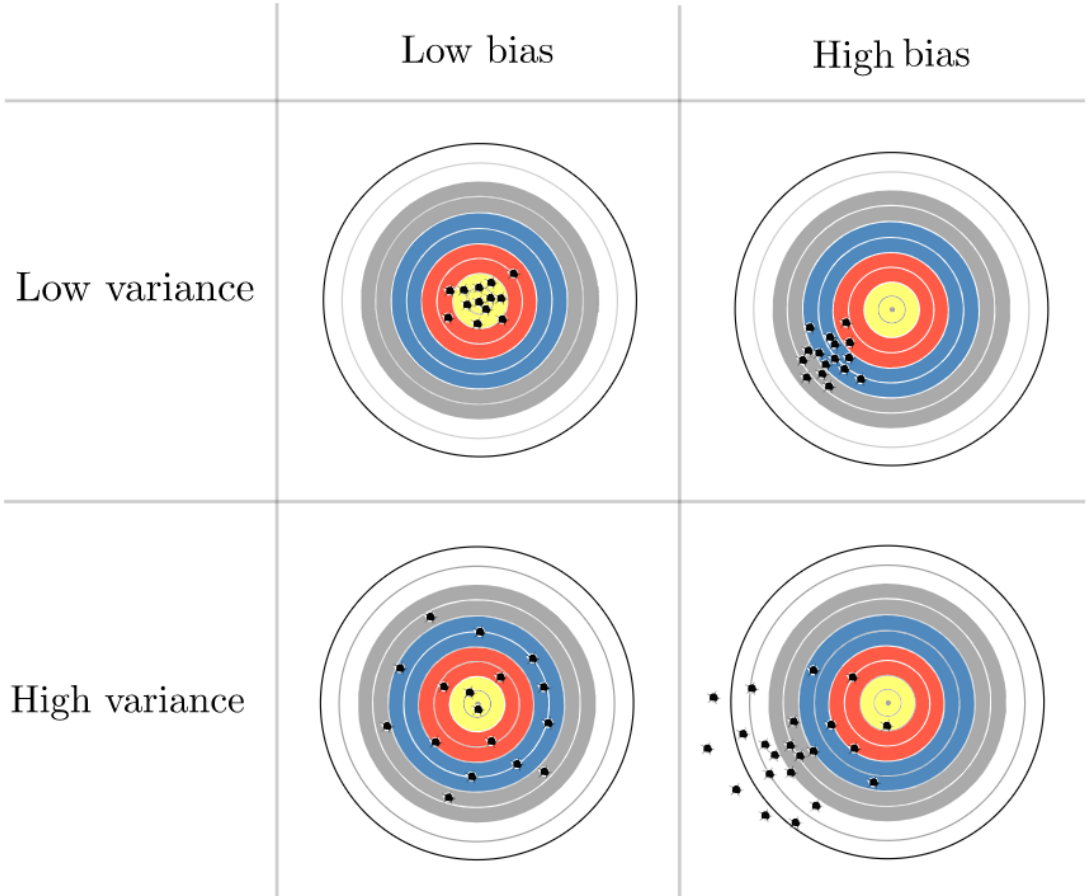
$$\Delta_m = \ell_m^2 + d_m,$$

где $\ell_m = \mathrm{M}[\hat{\theta}_m - \theta]$ — смещение оценки, а $d_m = \mathrm{D}[\hat{\theta}_m - \theta]$ — дисперсия её ошибки.

Смещение и разброс

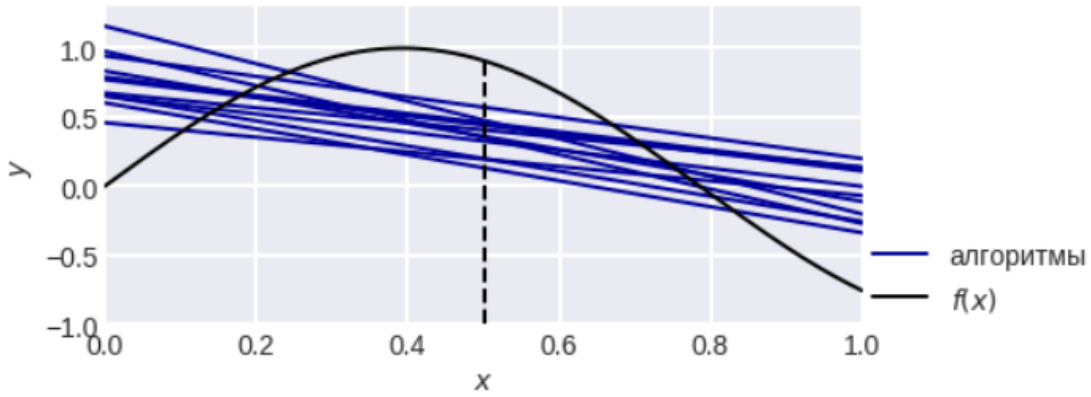
Разброс характеризует разнообразие алгоритмов (из-за случайности обучающей выборки, в том числе шума, и стохастической природы настройки). Смещение — способность модели алгоритмов настраиваться на целевую зависимость.

Можно сказать, что смещение — это ошибка алгоритма на обучающей выборке, а разброс — разница между ошибкой на обучающей и тестовой выборке.

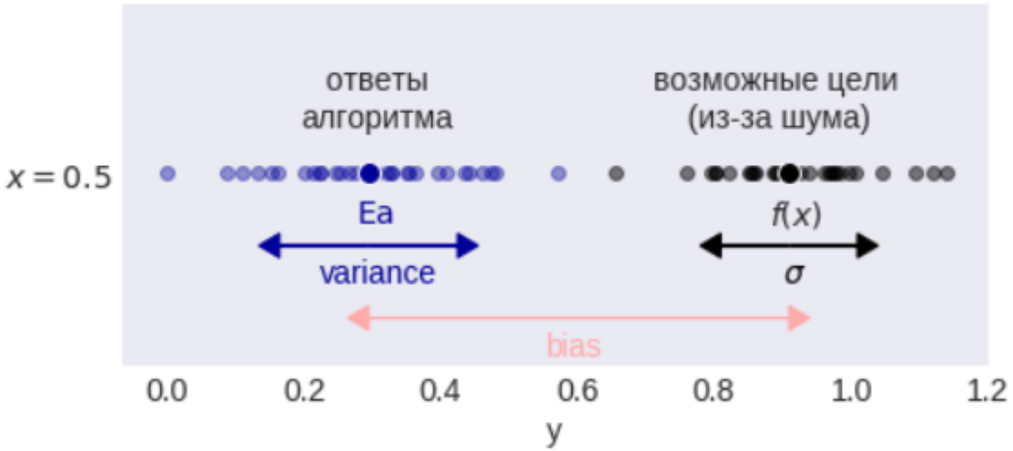


Смещение и разброс

Рассмотрим разные полиномы первой степени в точке $x = 0.5$

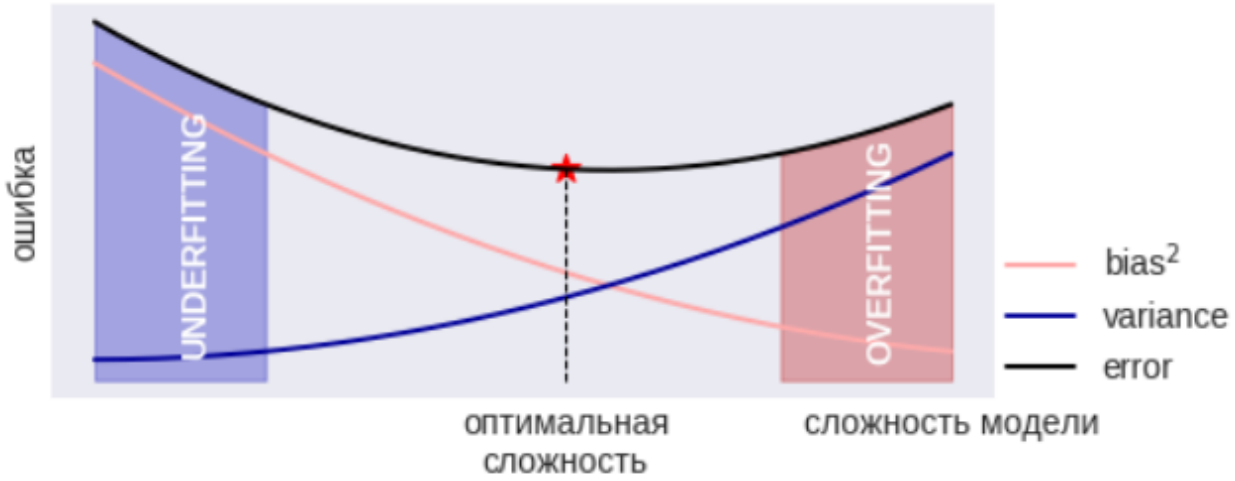


В точке $x = 0.5$ ответы алгоритмов являются случайными величинами, они немного «разбросаны» (есть variance), а также они сильно смещены (есть bias) относительно правильного ответа (который, известен с точностью до шума).



Смещение и разброс

Классическая иллюстрация изменения разброса и смещения.



При увеличении сложности модели (например, степени полинома) ошибка на независимом контроле сначала падает, потом начинает увеличиваться.

Обычно это связывают с уменьшением смещения (в сложных моделях очень много алгоритмов, поэтому наверняка найдутся те, которые хорошо описывают целевую зависимость) и увеличением разброса (в сложных моделях больше алгоритмов, а следовательно, и больше разброс).

Смещение и разброс

Предположим, что качество работы нашего алгоритма следующее:

- Ошибка на тренировочной выборке $= 1\%$
- Ошибка на валидационной выборке $= 11\%$

У алгоритма маленькое смещение и большой разброс, мы имеем дело с **переобучением**.

Теперь рассмотрим такую ситуацию:

- Ошибка на тренировочной выборке $= 15\%$
- Ошибка на валидационной выборке $= 16\%$

Алгоритм имеет большое смещение, но маленький разброс. Этот алгоритм **недообучился**.

Оценивая точность алгоритмов не стоит забывать и о шуме!

Смещение = Оптимальное смещение + Устранимое смещение

Смещение и разброс

Для простых моделей характерно недообучение (они слишком простые, не могут описать целевую зависимость и имеют большое смещение), для сложных — переобучение (алгоритмов в модели слишком много, при настройке мы выбираем ту, которая хорошо описывает обучающую выборку, но из-за сильного разброса она может допускать большую ошибку на тесте).

Реальность может отличаться от приведённых графиков:

- общая ошибка может быть не совсем унимодальна от сложности
- смещение и разброс могут не быть строго монотонными
- смещение может возрастать при увеличении сложности

Смещение и разброс

Приведем простую формулу устранения смещения и разброса:

- Если у вас большое устранимое смещение (avoidable bias), увеличьте сложность вашей модели (например, увеличьте вашу нейронную сеть, добавив слоев или (и) нейронов). При увеличении разброса, используйте регуляризацию.
- Если у вас большой разброс, добавьте примеров в вашу тренировочную выборку
- Если первые два пункта не помогают, то нужно менять подход к построению алгоритма:
  - модификация/отбор признаков
  - уменьшение или отказ от регуляризации
  - модификация архитектуры модели

Функционалы качества

Логистическая функция потерь

Рассматриваем задачу классификации с двумя классами: 0 и 1.

Пусть алгоритм выдаёт не метки классов, а оценку принадлежности к классу 1

$$y \in \{0,1\}$$
$$a \in [0,1]$$

«Раздельная» форма записи функции ошибки

$$L(a,y) = \begin{cases} L(a,1), & y = 1, \\ L(a,0), & y = 0. \end{cases}$$

Часто $L(a,1) = L(1-a,0)$
«Совместная» форма записи функции ошибки

$$L(a,y) = yL(a,1) + (1-y)L(a,0)$$

Логистическая функция потерь

Скоринговые ошибки
— ошибки в задаче бинарной классификации, для которых оптимальный ответ на каждом объекте — вероятность его принадлежности к классу 1.

$$L(y,a):$$

$$p = \arg\min_a \mathrm{M}_y[L(a,y)] \quad \text{для} \quad y \sim \mathrm{Be}(p)$$

- **Log Loss**
- **MSE**
- Exploss
- Misclassification Loss

Логистическая функция потерь

Log Loss

Эту функцию называют «логлосс», перекрёстной / кросс-энтропией (Cross Entropy)

Обучающую выборку $\mathcal{X}$ можно рассматривать, как реализацию обобщённой схемы Бернулли: для каждого объекта генерируется случайная величина, которая с вероятностью p (своей для каждого объекта) принимает значение 1 и с вероятностью $(1-p)$ — 0

$$y_i = \begin{cases} 1, & p_i \\ 0, & 1 - p_i \end{cases}$$

Предположим, что мы строим нашу модель так, чтобы она генерировала правильные вероятности, тогда можно записать функцию правдоподобия

Логистическая функция потерь

Обозначим $a_i = a(x_i|w)$ — ответы алгоритма на i-м объекте $x_i \in \mathcal{X}$, w — параметры алгоритма. Тогда

$$L(y,\mathcal{X},w) = \prod_i p(y_i,x_i,w) = \prod_i a_i^{y_i}(1-a_i)^{1-y_i} \to \max_{a_i};$$

$$\log L(y,\mathcal{X},w) = \sum_i (-y_i \log a_i - (1-y_i)\log(1-a_i)) \to \min_{a_i}.$$

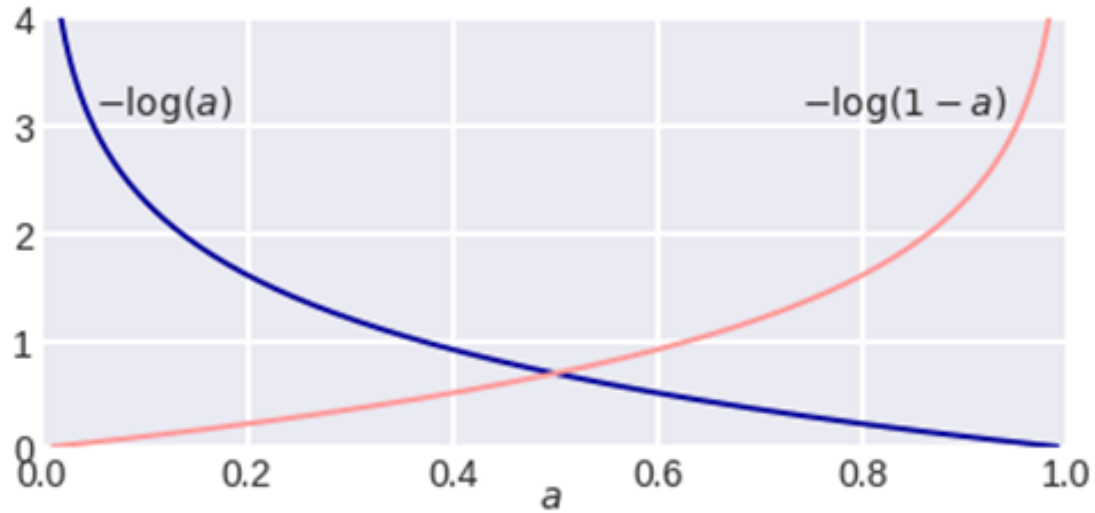
Выражение стоящее под суммой и называют «логистической функцией потерь».

Логистическая функция потерь

Для задачи бинарной классификации, в которой алгоритм должен выдать вероятность принадлежности классу 1, logloss логична ровно настолько, насколько логична MSE в задаче линейной регрессии с гауссовским шумом (обе функции потерь выводятся из метода максимального правдоподобия)

Иногда logloss удобнее представлять как

$$-\begin{cases} \log a_i, & y_i = 1, \\ \log(1 - a_i), & y_i = 0. \end{cases}$$



Логистическая функция потерь

Неудобное свойство logloss: если для объекта 1-го класса мы предсказываем нулевую вероятность принадлежности к этому классу или, наоборот, для объекта 0-го — единичную вероятность принадлежности к классу 1, то ошибка равна бесконечности!
Таким образом, **грубая ошибка на одном объекте сразу делает алгоритм бесполезным**.

Рассмотрим тривиальную модель (алгоритм в виде константы), т.е. $a_i \equiv a$. Пусть выборка состоит из m_1 представителя класса 1 и m_0 представителей класс 0, $m_1 + m_2 = m$, тогда

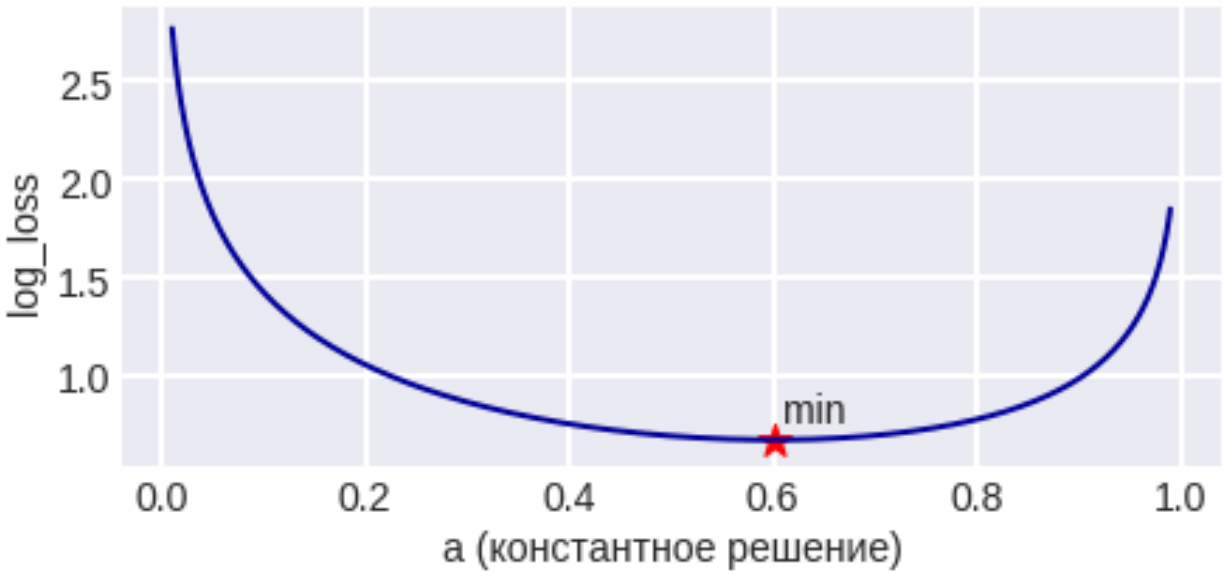
$$-\frac{1}{m}\sum_{i=1}^{m}(y_i \log a + (1 - y_i)\log(1 - a) \min_a$$

$$-\frac{m_1}{m}\log a - \frac{m_0}{m}\log(1 - a) \to \min_a \quad \Rightarrow \quad a = \frac{m_1}{m}$$

Логистическая функция потерь

Описанную задачу приходится решать, например, при построении решающих деревьев (какую метку приписывать листу, если в него попали представители разных классов).

На рисунке изображён график logloss ошибки константного алгоритма для выборки из четырёх объектов класса 0 и 6 объектов класса 1.



Логистическая функция потерь

Представим теперь, что мы знаем, что объект принадлежит к классу 1 с вероятностью p, посмотрим, какой ответ оптимален на этом объекте с точки зрения logloss:

$$-p \log a_i - (1-p) \log(1-a_i) \min_{a_i}$$

$$\frac{p}{a_i} - \frac{1-p}{1-a_i} = 0 \quad \Rightarrow \quad a_i = p.$$

Мы получили, что оптимально для каждого объекта выдавать его вероятность принадлежности к классу 1!
Если подставить полученное оптимальное решение в минимизируемый функционал, то получим энтропию:

$$-p \log p - (1-p) \log(1-p).$$

Это объясняет, почему при построении решающих деревьев в задачах классификации применяют энтропийный критерий расщепления. Энтропия в листе будет примерно равна logloss-ошибке константного решения.

Логистическая функция потерь

В каких пределах может варьироваться logloss?
Ясно, что минимальное значение 0, максимальное — $+\infty$, но реально максимальным можно считать ошибку при использовании константного алгоритма (вряд же мы придумаем алгоритм хуже константы), т.е.

$$\left[0, -\frac{m_1}{m}\log\frac{m_1}{m} - \frac{m_0}{m}\log\frac{m_0}{m}\right]$$

Если брать логарифм по основанию 2, то на сбалансированной выборке это отрезок $[0, 1]$.

Логистическая функция потерь

Связь с логистической регрессией

Логистическая регрессия — метод для решения задачи бинарной классификации, который получает вероятность принадлежности к классу 1.
Посмотрим, как настраивается логистическая регрессия (т.е. сигмоида от линейной комбинации) на функцию logloss методом стохастического градиентного спуска (SGD)

$$logloss(a,y) = -y \log a - (1-y)\log(1-a)$$

$$a = \text{sigmoid}(w^T x) \equiv \frac{1}{1+e^{-w^T x}}$$

$$\frac{\partial logloss}{\partial w} = (a-y)x$$

$$w_{k+1} = w_k - \alpha(a-y)x$$

Логистическая функция потерь

Как видим, корректировка весов точно такая же, как и при настройке линейной регрессии

Это говорит о родстве разных регрессий: линейной и логистической, а точнее, о родстве распределений: нормального и Бернулли (вспоминаем ЦПТ, частоту случайного события и её свойства)

Логистическая функция потерь

Связь с расхождением Кульбака-Лейблера (KL, Kullback–Leibler divergence)

Расхождение KL часто используют для вычисления непохожести двух распределений
Оно определяется по следующей формуле:

$$D_{KL}(P\|Q) = \int p(z) \log \frac{p(z)}{q(z)} \partial z = \underbrace{H(p,q)}_{\text{кросс-энтропия}} - \underbrace{H(p)}_{\text{энтропия}}$$

где P и Q — распределения, $p(z)$, $q(z)$ — плотности этих распределений
Для $y \sim \mathrm{Be(p)}$ энтропия $H(p) = 0$
Для дискретных распределений формулу записывают так:

$$D_{KL}(P\|Q) = \sum_i P_i \log \frac{P_i}{Q_i},$$

где P_i, Q_i — вероятности значений дискретных СВ

Логистическая функция потерь

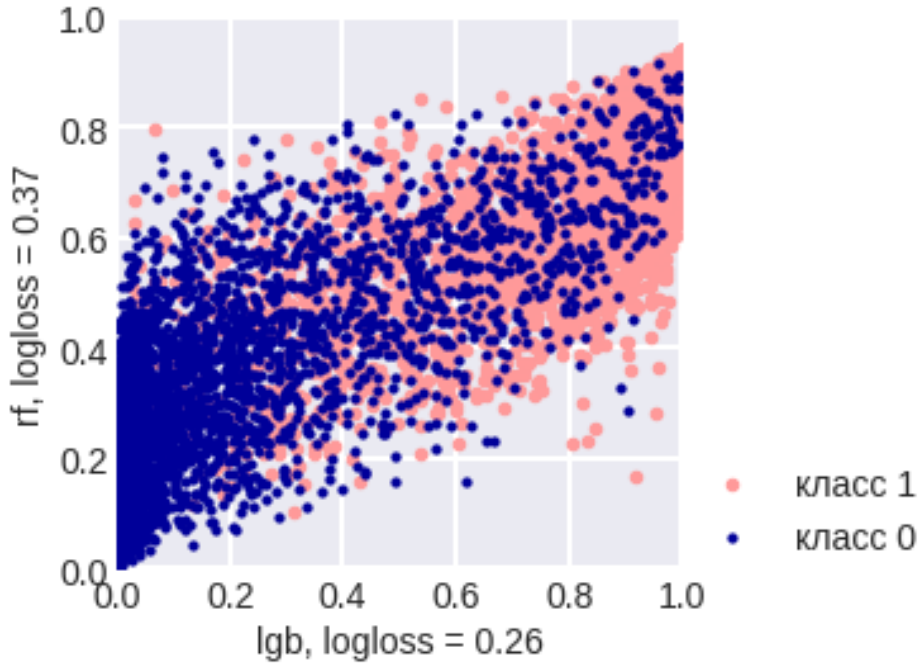
Рассмотрим объект x с меткой y. Если алгоритм выдаёт вероятность принадлежности первому классу — a, то предполагаемое распределение на событиях «класс 0», «класс 1» — $(1–a, a)$, а истинное — $(1–y, y)$, поэтому расхождение Кульбака-Лейблера между ними

$$(1-y)\log\frac{1-y}{1-a} + y\log\frac{y}{a} = -(1-y)\log(1-a) - y\log a,$$

что в точности совпадает с logloss

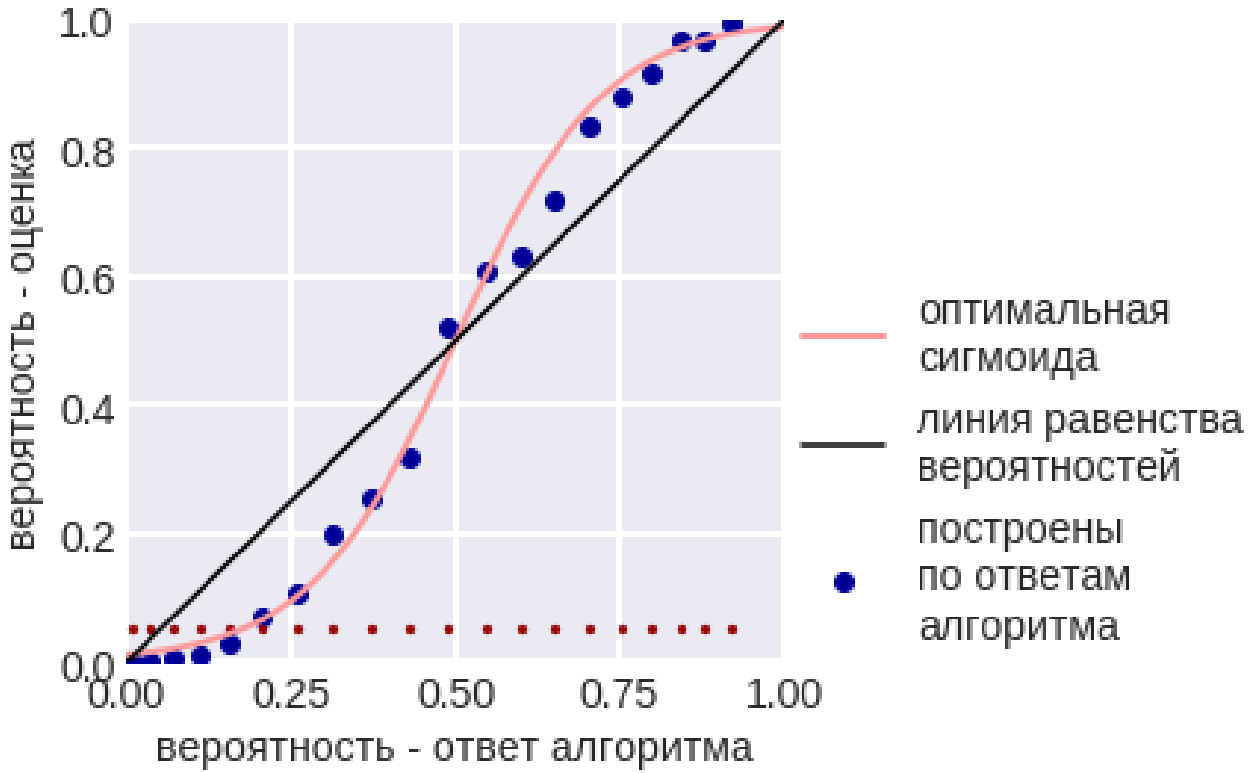
Настройка на logloss

Один из методов «подгонки» ответов алгоритма под logloss — калибровка Платта (Platt calibration)
Пусть алгоритм порождает некоторые оценки принадлежности к классу ‘1‘ — $a(x)$ Проиллюстрируем применение метода на реальной задаче. На рис. показаны ответы двух алгоритмов: градиентного бустинга (LightGbm) и случайного леса (RF)



Логистическая функция потерь

Качество леса намного ниже и он довольно осторожен: занижает вероятности у объектов класса 1 и завышает у объектов класса 0. Упорядочим все объекты по возрастанию вероятностей (RF), разобьем на k равных частей и для каждой части вычислим среднее всех ответов алгоритма и среднее всех правильных ответов. Результат показан на рисунке.



Логистическая функция потерь

Нетрудно видеть, что точки располагаются на линии, похожей на сигмоиду — можно оценить параметр сжатия-растяжения в ней на отложенной выборке (calibration set). Оптимальная сигмоида показана на рисунке. Новые ответы алгоритма вычисляются по формуле

$$\widetilde{a}(x) = \mathrm{sigmoid}(\alpha r(x) + \beta),$$

где $r(x)$ — расстояния от ответов $a(x)$ до линии равенства вероятностей.
Если подвергать ответы такой сигмоидной деформации, то logloss случайного леса снижается с 0.37 до 0.33.

Логистическая функция потерь

Многоклассовый logloss

logloss обобщается и на случай нескольких классов естественным образом:

$$\text{logloss} = -\frac{1}{q}\sum_{i=1} q \sum_{j=1}^{l} y_{ij} \log a_{ij},$$

где q — число элементов в выборке, l — число классов, a_{ij} — ответ (вероятность) алгоритма на i-м объекте на вопрос принадлежности его к j-му классу, $y_{ij} = 1$ если i-й объект принадлежит j-му классу, в противном случае $y_{ij} = 0$.

Коэффициент Джини

Основной недостаток logloss — невозможность интерпретации его значений для заказчика

Если использовать **MSE** в задаче классификации

$$L(a,y) = (y-1)^2 = y(1-a)^2 + (1-y)a^2$$

Пусть объект x с вероятностью p принадлежит классу 1, тогда матожидание ошибки

$$p(1-a)^2 + (1-p)a^2$$

оптимальный ответ тоже $a = p$.
Если в минимум матожидания подставить $a = p$, то получим

$$p(1-p)^2 + (1-p)p^2 = p(1-p)$$

критерий расщепления Джини минимизирует эту функцию ошибки!
В sklearn это называется «Brier score» (brier_score_loss)

Коэффициент Джини

Коэффициент Джини (Gini coefficient) — метрика качества, которая часто используется в задачах бинарной классификации в условиях сильной несбалансированности классов целевой переменной

- банковское кредитование
- страхование
- целевой маркетинг

Не путать с коэффициентом Джини (Gini Impurity), который используется в деревьях решений как критерий расщепления!

Коэффициент Джини

Экономика

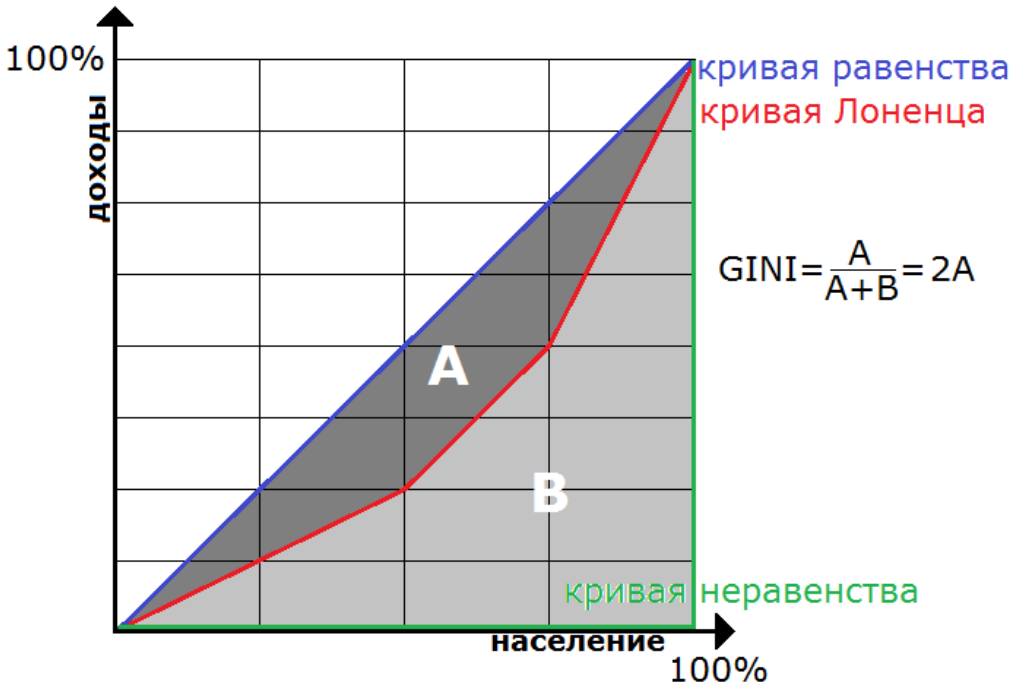
Коэффициент Джини — статистический показатель степени расслоения общества относительно какого-либо экономического признака, например, дохода

Коэффициент показывает отклонение фактического распределения доходов в обществе от абсолютно равного их распределения

Допустим, в компании работают 4 человека с суммарным доходом 8000\$.
Равномерное распределение дохода —
$2000\$ + 2000\$ + 2000\$ + 2000\$$,
неравномерное — $0\$ + 0\$ + 0\$ + 8000\$$
А как оценить неравномерность, скажем, для случая
$1000\$ + 1000\$ + 2000\$ + 4000\$$?
Упорядочим сотрудников по возрастанию дохода

Коэффициент Джини

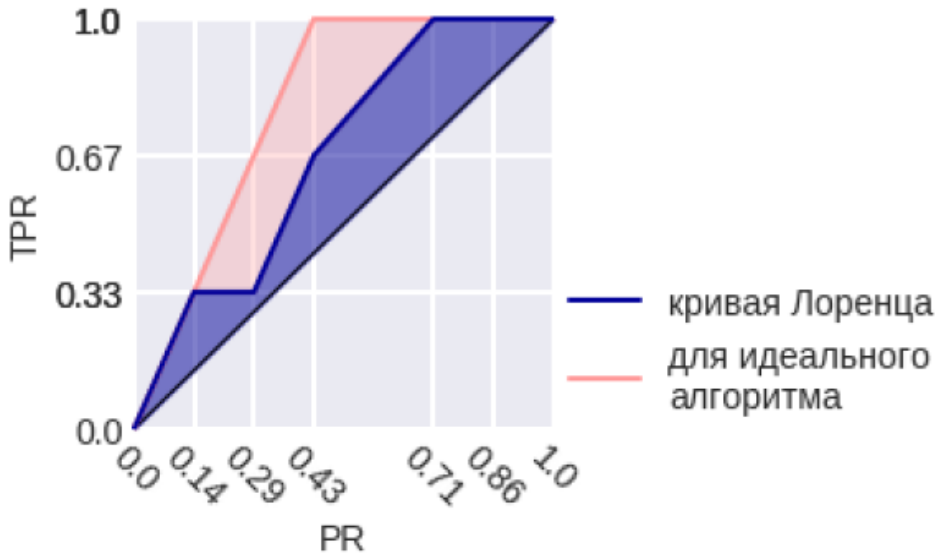
Построим кривую (Лоренца) в координатах [процент населения, процент дохода этого населения] — идём по всем сотрудникам и откладывает точки. Для первого — $[25\%, 12.5\%]$ — сколько он составляет процентов от всего штата и сколько процентов составляет его доход, для первого и второго — [50%, 25%] — это сколько они составляют процентов и сколько процентов их доход, для первых трёх — [75%, 50%], для всех — [100%, 100%]



Коэффициент Джини

В машинном обучении

Кривая Лоренца (Cumulative Accuracy Profile Curve или Lift Curve) является зеркальным отображением экономической кривой Лоренца относительно линии абсолютного равенства



PR — Positive Rate, процент объектов, которые при определённом выборе порога, отнесены к классу 1

Коэффициент Джини — отношение площадей $\blacksquare / (\blacksquare + \square)$

Коэффициент Джини

Численно коэффициент равен площади фигуры, образованной линией абсолютного равенства и кривой Лоренца

Связь с AUC ROC:

$$\mathrm{GINI} = 2 \cdot \mathrm{AUCROC} - 1 \qquad (1).$$

Меняется от -1 до $+1$

$$0.9\mathrm{AUC} = 0.8\mathrm{GINI}$$

Доказательство формулы (1) и примеры расчётов можно найти в статье Д. Петухова
https://habr.com/ru/company/ods/blog/350440/

Коэффициент Джини

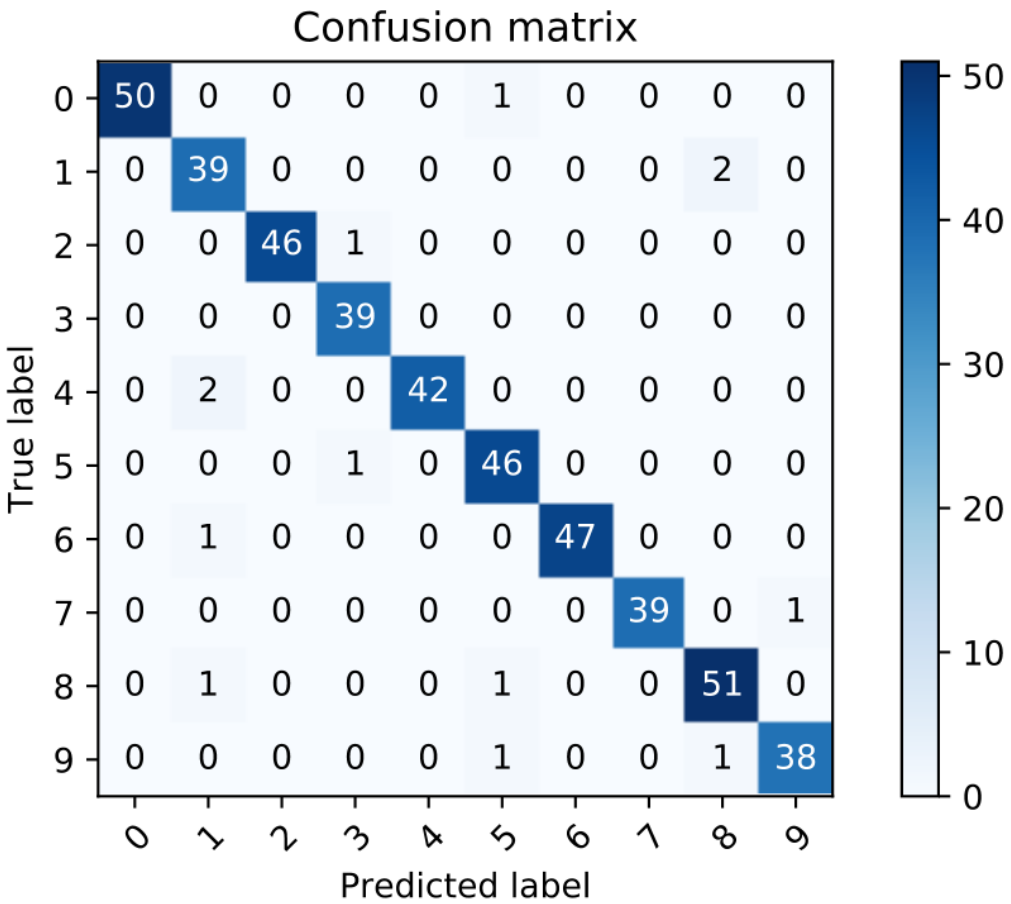
- Предсказание идеального алгоритма является максимальным коэффициентом Джини для текущего набора данных и зависит только от истинного распределения классов в задаче
- При равномерном распределении классов целевой переменной коэффициент Джини идеального алгоритма всегда будет равен 0.25
- Для идеального алгоритма форма фигуры, образуемой Lift Curve и линией абсолютного равенства, всегда будет треугольником
- Коэффициент Джини случайного алгоритма равен 0, а Lift Curve совпадает с линией абсолютного равенства
- Коэффициент Джини является метрикой качества, которую необходимо максимизировать

Практическое применение

- **Кредитный скоринг.** Разрабатываются предиктивные модели, оценивающие по признаковому пространству вероятность того, что клиент не выплатит кредит
  https://www.kaggle.com/c/home-credit-default-risk
- **Страхование.** Необходимо разделить клиентов на тех, кто подаст страховое требование и на тех, кто этого не сделает
  https://www.kaggle.com/c/porto-seguro-safe-driver-prediction
- **Целевой маркетинг.** Например, у нас есть база данных пользователей когда-то игравших в компьютерную игру и по каким-то причинам бросивших. Мы хотим их вернуть в наш игровой проект
  https://www.kaggle.com/rodsaldanha/arketing-campaign
  https://www.kaggle.com/c/springleaf-marketing-response
  https://www.kaggle.com/c/avito-context-ad-clicks

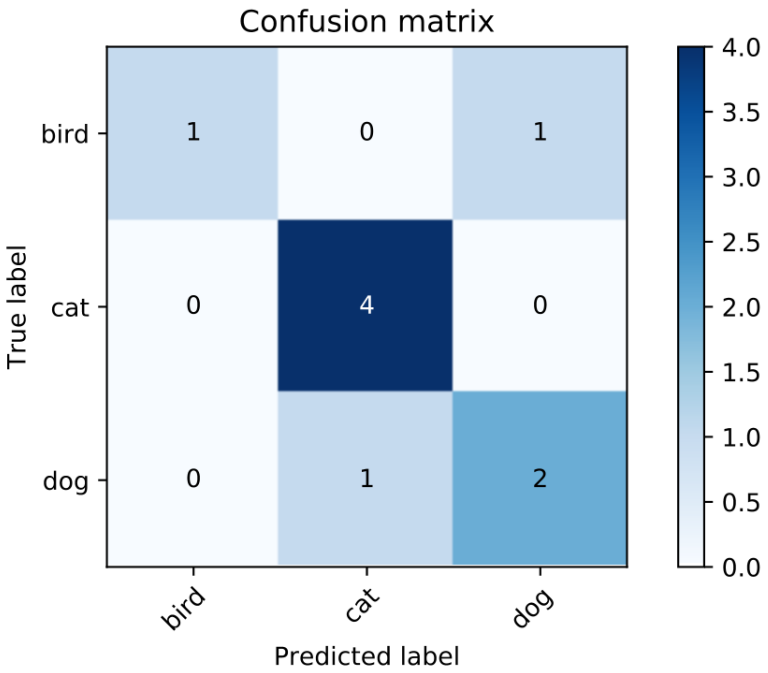
Многоклассовая классификация

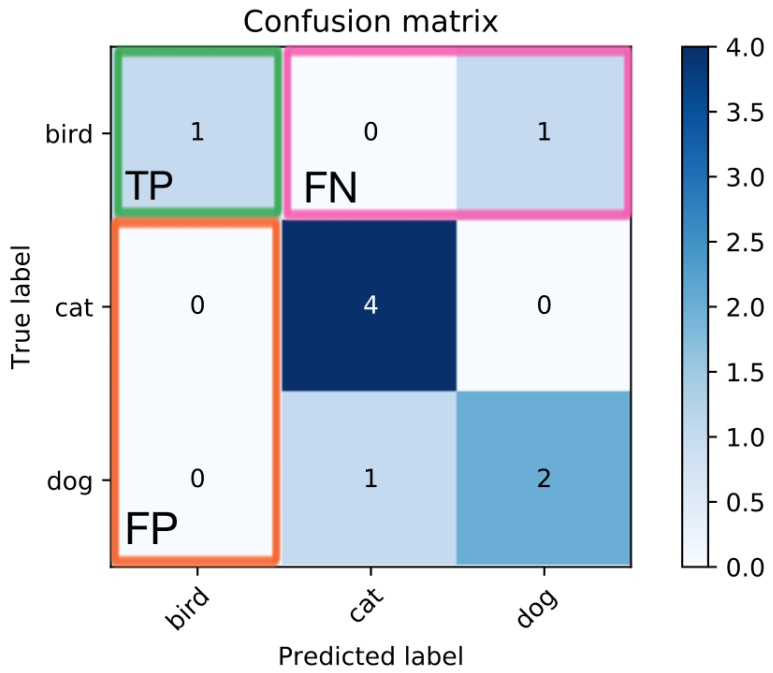
Рассмотрим задачу классификации для $\ell > 2$ классов.



Обозначим элемент confusion matrix m_{ij}, $i,j = 1,\ldots,\ell$

Многоклассовая классификация





Многоклассовая классификация

матрица классификаций $\{y_{ij}\}_{m\times\ell}$

class 1	class 2	class 3
1	0	0
0	1	0
0	0	1
1	1	0

матрица ответов $\{a_{ij}\}_{m\times\ell}$

class 1	class 3	class 3
0.75	0.00	0.25
0.00	0.50	0.25
0.25	0.50	0.25
0.00	0.25	0.75

Нужно сравнить матрицы на схожесть

микро-подход	сравнить матрицы как векторы, далее как в бинарной задаче
макро-подход	сравнить столбцы матриц и усреднить
по объектам	сравнить строки матриц и усреднить

Многоклассовая классификация

Многоклассовая задача: Hamming Loss

Число ошибок
$a_{ij} \in \{0,1\}$

$$\mathrm{HL} = \frac{1}{m\ell} \sum_{i=1}^{m} \sum_{j=1}^{\ell} \mathrm{I}\left(y_{ij} \neq a_{ij}\right)$$

(«1» — точность)

Многоклассовая задача: Log Loss (cross-entropy)

Естественное обобщение
логистической ошибки
$a_{ij} \in [0,1]$

$$\mathrm{logloss} = -\frac{1}{m} \sum_{i=1}^{m} \sum_{j=1}^{\ell} y_{ij} \log a_{ij}$$

(лучше использовать для непересекающихся классов)

Многоклассовый AUCROC: Макро-усреднение

$$\mathrm{AUC} = \frac{1}{\ell}\sum_{j=1}^{\ell}\mathrm{AUC}_j$$

AUC_j — значение функционала в задаче бинарной классификации «j-й класс / не j-й класс»

$$\left\{\left(x_i, \mathrm{I}\left[y(x_i)_{[j]} = 1\right]\right\}_{i=1}^{m}\right.$$

Многоклассовый AUCROC: Весовое макро-усреднение

$$\mathrm{AUC} = \frac{\frac{1}{\ell}\sum_{j=1}^{\ell}P_j\mathrm{AUC}_j}{\sum_{j=1}^{\ell}P_j}$$

P_j — вероятность j-го класса (процент «1» в столбце матрицы классификации)

Многоклассовый AUCROC: Микро-усреднение

«вытягиваем матрицу ответов в вектор»

$$\big((x_i,j), \mathrm{I}[y(x_i)_{[j]} = 1]\big)\,, \quad i = 1,\ldots,m,\ j = 1,\ldots,\ell$$

Многоклассовый AUCROC: Усреднение по объектам

$$\mathrm{AUC} = \frac{1}{m}\sum_{i=1}^{m} \mathrm{AUC}'_i$$

AUC'_i — значение функционала в задаче

$$\big((x_i,j), \mathrm{I}[y(x_i)_{[j]} = 1]\big)\,, \quad j = 1,\ldots,\ell$$

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html

Многоклассовая классификация

матрица классификаций

class 1	class 2	class 3
1	0	0
0	1	0
0	0	1
1	1	0

матрица ответов

class 1	class 2	class 3
0.75	0.00	0.25
0.00	0.50	0.25
0.25	0.50	0.25
0.00	0.25	0.75

macro	**micro**	**weighted**	**samples**
0.49	0.53	0.52	0.56

	class 1	class 2	class 3
AUC_per_class	0.62	0.50	0.33
P_per_class	0.50	0.50	0.25

	sample 0	sample 1	sample 2	sample 3
AUC_per_instance	1.0	1.0	0.25	0

Точность: сравнение макро- и микро- усреднения

$$\mathrm{Pr}_j = \frac{\mathrm{TP}_j}{\mathrm{TP}_j + \mathrm{FP}_j}$$

$$\mathrm{Pr}_{\text{macro-mean}} = \frac{1}{\ell} \sum_{j=1}^{\ell} \mathrm{Pr}_j$$

Макро-усреднение можно делать по-разному, например, среднее геометрическое

$$\mathrm{Pr}_{\text{micro-mean}} = \frac{\sum_{j=1}^{\ell} \mathrm{TP}_j}{\sum_{j=1}^{\ell} \mathrm{TP}_j + \sum_{j=1}^{\ell} \mathrm{FP}_j}$$

Многоклассовая классификация

	TP	FP	Pr
class 1	2	2	0.50
class 2	5	10	0.33
class 3	10	40	0.20

	TP	FP	Pr
class 1	2	2	0.50
class 2	5	10	0.33
class 3	100	400	0.20

$P_{\text{macro}} = 0.34,\ P_{\text{micro}} = 0.246;\quad P_{\text{macro}} = 0.344,\ P_{\text{micro}} = 0.206$

микро-точность смещается в сторону «большего» класса: 0.206

F-мера — ещё больше вариантов усреднения

Макро F-мера:

$$\mathrm{F}_{\text{macro-mean}} = \frac{1}{\ell}\sum_{j=1}^{\ell}\mathrm{F}_j;$$

на основе других макро-параметров:

$$\mathrm{F} = \frac{2}{\dfrac{1}{\mathrm{P}_{\text{macro-mean}}} + \dfrac{1}{\mathrm{R}_{\text{macro-mean}}}},$$

P — точность, R — полнота

Использовалась в задаче
Large Scale Hierarchical Text Classification
Классификация документов Wikipedia в одну из 325056 категорий https://www.kaggle.com/c/lshtc/

Многоклассовая классификация

В качестве исходного решения организаторы предложили подход на основе **решающего правила с отсечкой**.
Пусть c — порог бинаризации ответов алгоритма a_{ij}, тогда метки классов:

$$\alpha_{ij} = \begin{cases} 1, & a_{ij} \geq \min\left(c, \max\limits_{j=1,\ldots,\ell}\{a_{ij}\}\right), \\ 0, & \text{иначе} \end{cases}$$

Это правило гарантирует, что каждому объекту будет присвоена хотя бы одна 1, т.е. он будет отнесен к какому-нибудь классу.

Основная идея лучшего решения: раз класс существует, значит он нужен и у него должны быть представители. Следовательно нужно модифицировать решающее правило так, чтобы в каждом классе (столбце) тоже была хотя бы одна 1 или процент 1 как в обучающей выборке.

Сбалансированная точность «Balanced accuracy» — макро-усреднение полноты

$$\mathrm{BA} = \frac{1}{\ell}\sum_{j=1}^{\ell}\mathrm{R}_j$$

Другие (неэквивалентные) определения:

$$\mathrm{BA} = \frac{1}{\ell}\sum_{j=1}^{\ell}\min[\mathrm{P}_j,\mathrm{R}_j], \qquad \mathrm{BA} = \frac{1}{\ell}\sum_{j=1}^{\ell}\min[\mathrm{TPR}_j,\mathrm{TNR}_j]$$

Weighted kappa в задачах ранжирования (поисковые выдачи)

Пусть каким-то образом заданы разумные веса ошибок за конкретные несогласованности w_{ij}, тогда

$$\varkappa = 1 - \frac{\sum_{i=1}^{l}\sum_{j=1}^{l} w_{ij} m_{ij}}{\sum_{i=1}^{l}\sum_{j=1}^{l} w_{ij} s_{ij}} \in [-1, +1],$$

где веса, например, квадратичные (м.б. любая весовая схема)

$$w_{ij} = \frac{(i-j)^2}{(l-1)^2}$$

$S = \{s_{ij}\}$ — матрица случайных ответов

$$s_{ij} = \frac{1}{m} \sum_{j} m_{ij} \sum_{i} m_{ij}$$

Вычисление Quadratic Weighted Kappa

y	a
1	1
1	1
1	2
2	1
2	3
3	2
3	3
3	3
1	2
2	2

матрица ошибок

	a=1	a=2	a=3
$y = 1$	2	2	0
$y = 2$	1	1	1
$y = 3$	0	1	2

матрица случайных ответов

	a=1	a=2	a=3
$y = 1$	1.2	1.6	1.2
$y = 2$	0.9	1.2	0.9
$y = 3$	0.9	1.2	0.9

Многоклассовая классификация

матрица весов

	a=1	a=2	a=3
$y = 1$	0.00	0.25	1.00
$y = 2$	0.25	0.00	0.25
$y = 3$	1.00	0.25	0.00

$$\varkappa = 0.615$$

```
from sklearn.metrics import cohen_kappa_score
cohen_kappa_score(df.y, df.a, weights='quadratic')
```

Каппа сложна для оптимизации

Многоклассовая классификация

Пример вычисления Карра для упорядоченных классов при различных ответах алгоритма

y	a_1	a_2	a_3	a_4	a_5	a_6	a_7
0	0	0	0	0	0	0	2
0	0	0	0	0	0	1	2
0	0	1	0	2	0	2	2
1	1	1	1	1	0	0	1
1	1	1	1	1	0	1	1
1	1	0	2	1	0	2	1
2	2	2	2	2	2	0	0
2	2	2	2	2	2	1	0
2	2	2	1	0	2	2	0
$\varkappa$	**1.0**	**0.83**	**0.83**	**0.33**	**0.8**	**0.0**	**-1.0**

Функционалы качества в задачах кластеризации

Оценка результатов кластеризации

Если знаем верную кластеризацию, то можно вычислить **внешнюю оценку** (External evaluation)

Ничего не знаем $\Rightarrow$ согласованность с данными и **внутренняя оценка** (Internal evaluation)

Функционалы качества в задачах кластеризации

«Internal evaluation»

Пусть построена чёткая (нет пересечений) кластеризация $U = U_1 \cup \ldots \cup U_K$ для множества $\mathcal{X} = \{x_1, \ldots, x_m\}$

Davies–Bouldin index, использует центроиды c_i и дисперсии σ_i

$$DB = \frac{1}{K} \sum_{i=1}^{K} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{\rho(c_i, c_j)} \right),$$

$$\sigma_i = \left(\frac{1}{|U_i|} \sum_{j=1}^{|U_i|} |x_i^j - c_i|^p \right)^{1/p}$$

Dunn index, min между кластерами и max внутри кластеров

$$D = \frac{\min\limits_{1 \le i \le j \le K} \rho(U_1, U_2)}{\max\limits_{1 \le i \le K} \rho_{in}(U_i)}$$

Silhouette, среднее расстояние до всех точек кластера

$$S(x_i) = \frac{\rho_1(x_i) - \rho_2(x_i)}{\max(\rho_1(x_i), \rho_2(x_i))}, \text{ если } |U_i| > 1$$

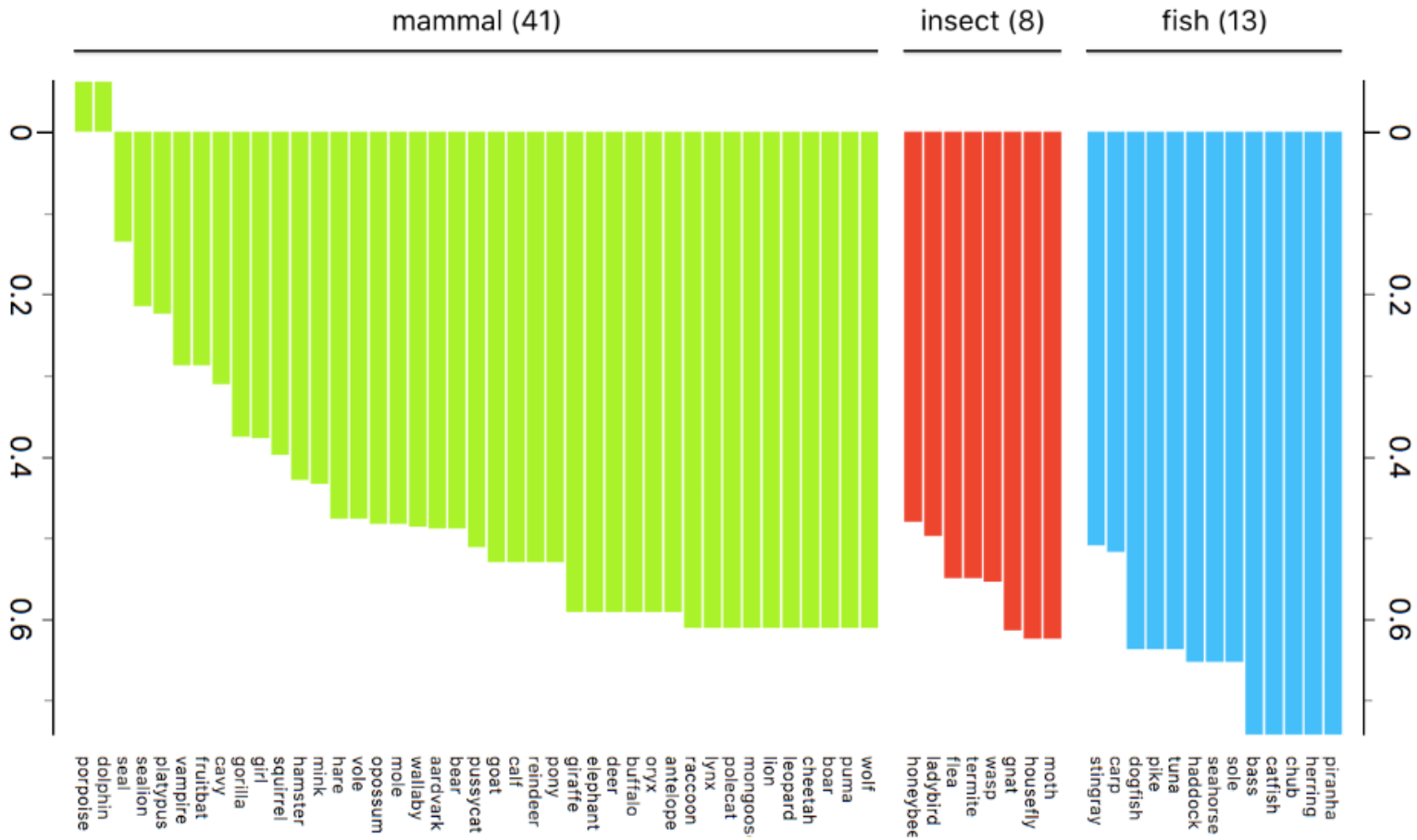
$$\rho_1(x_i) = \frac{1}{|U_i| - 1} \sum_{x_j \in U_i,\ i \neq j} \rho(x_i, x_j)$$

— среднее расстояние внутри кластера, которому x_i принадлежит

$$\rho_2(x_i) = \min_{k \neq i} \frac{1}{|U_k|} \sum_{j \in U_k} \rho(x_i, x_j)$$

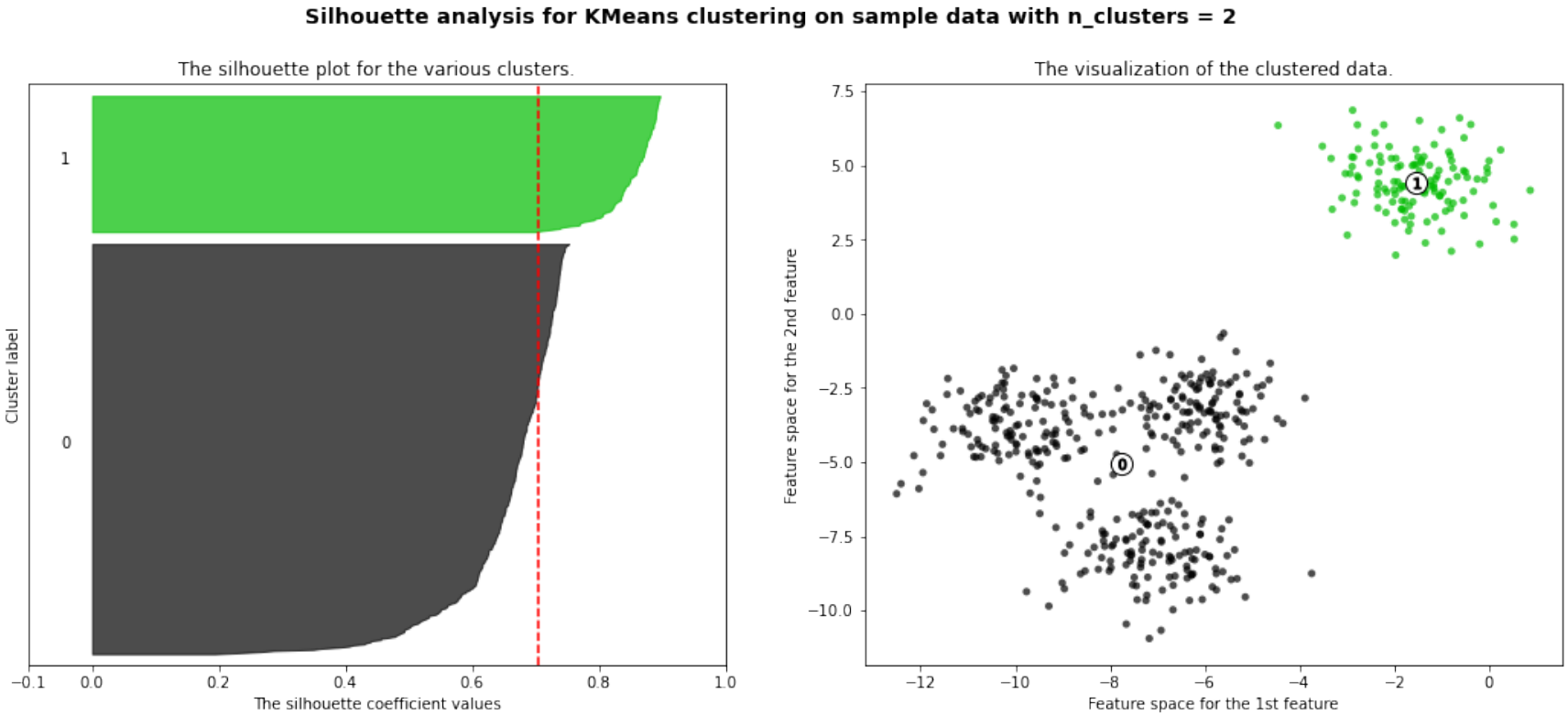
— минимальное среднее расстояние до точек кластера, которому x_i не принадлежит

Функционалы качества в задачах кластеризации

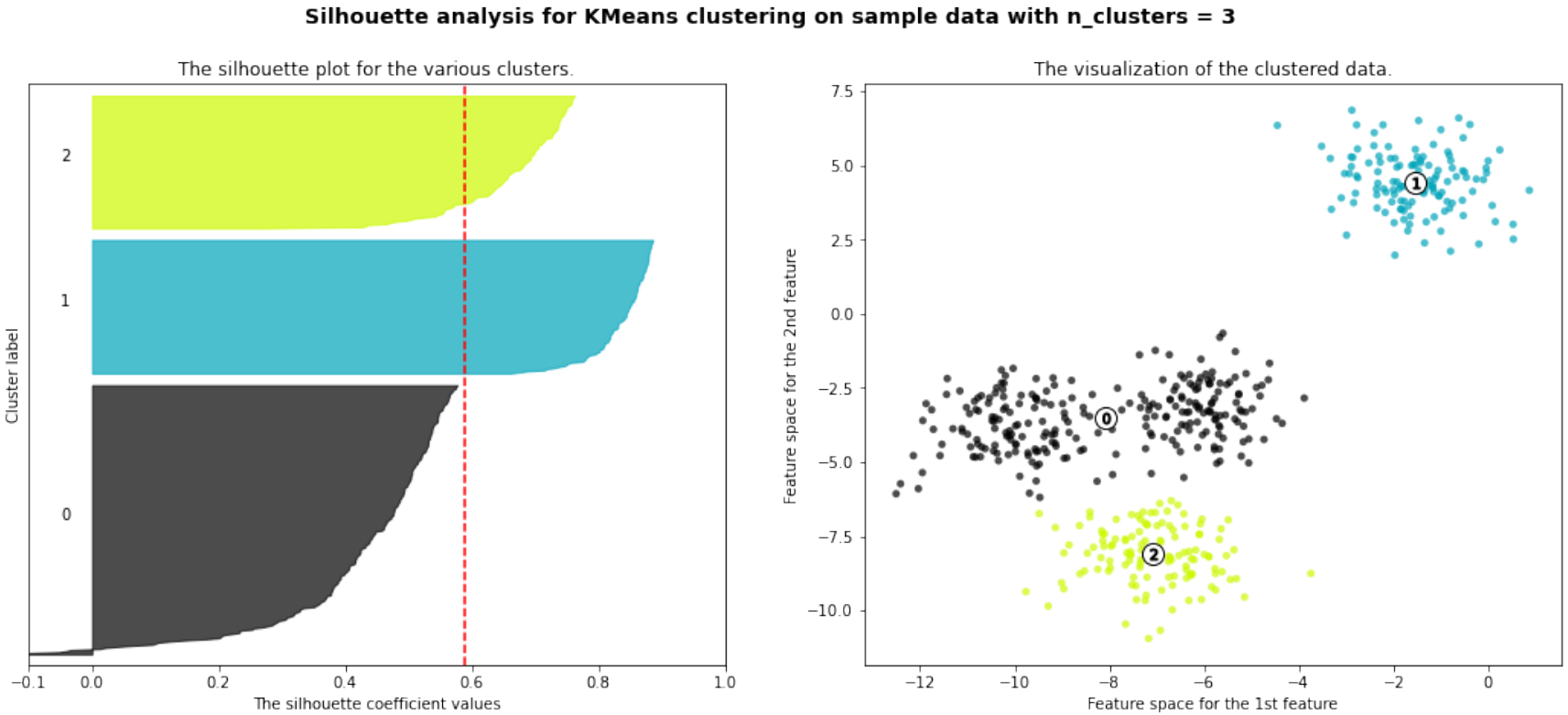


https://en.wikipedia.org/wiki/Silhouette_(clustering)

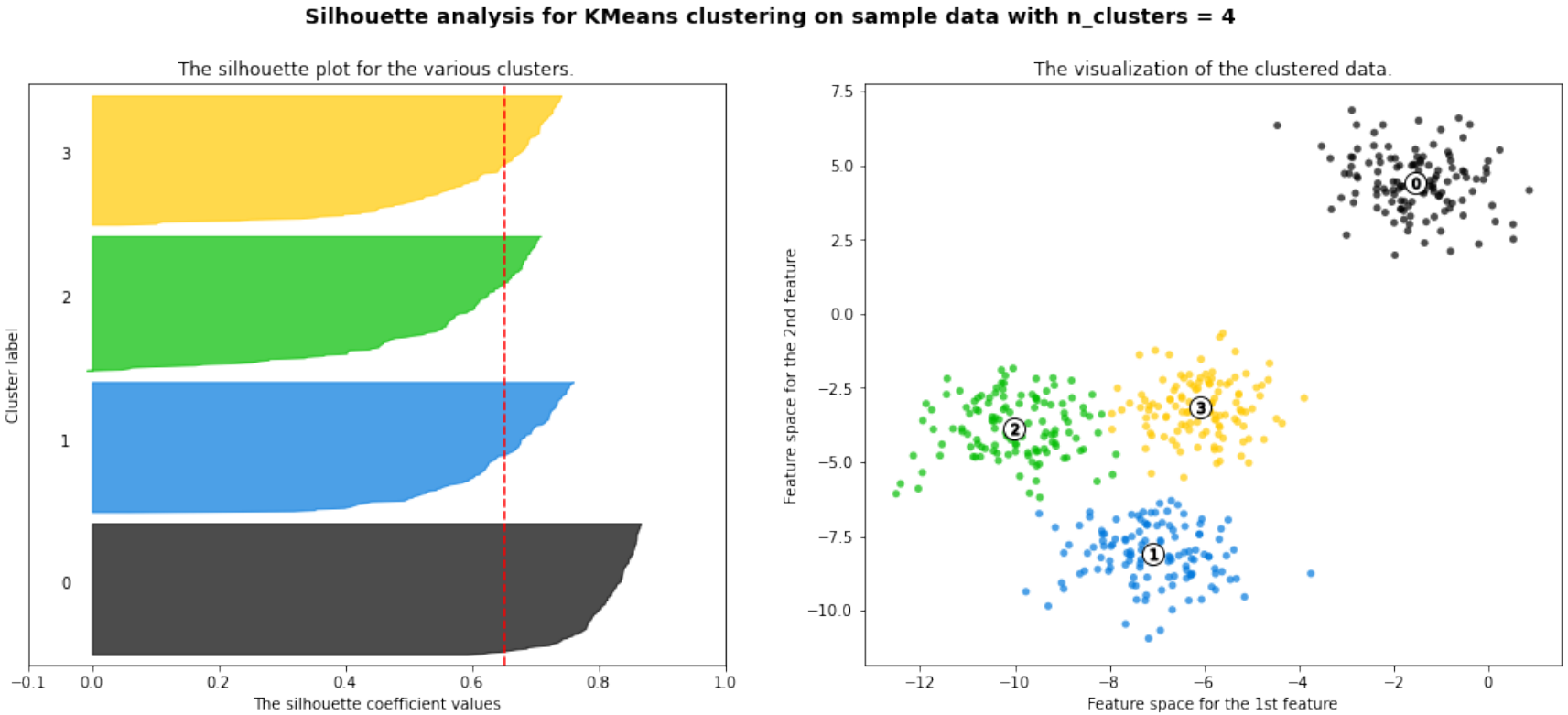
Функционалы качества в задачах кластеризации



Функционалы качества в задачах кластеризации



Функционалы качества в задачах кластеризации



Функционалы качества в задачах кластеризации

Плюсы Silhouette

- Меняется от -1 за неправильную кластеризацию до $+1$ за высокоплотную кластеризацию.
- Значения около нуля указывают на перекрывающиеся кластеры.
- Оценка выше, когда кластеры плотные и хорошо разделены

Минусы

- Значение больше для выпуклых кластеров, чем для других концепций кластеров, таких как кластеры получены с помощью DBSCAN.
- Высокая вычислительная сложность: $O(n^2)$

Функционалы качества в задачах кластеризации

Calinski-Harabasz Index (Variance Ratio Criterion)

$$\frac{\operatorname{trace}\left(\frac{1}{K-1}\sum_{i=1}^{K}|U_i|(x-c_i)(x-c_i)^T\right)}{\operatorname{trace}\left(\frac{1}{m-K}\sum_{i=1}^{K}\sum_{x\in U_i}(x-c_i)(x-c_i)^T\right)}$$

— отношение средней дисперсии между кластерами и средней дисперсии, вычисленной внутри кластеров

- чем больше, тем лучше кластеризация
- вычисляется быстро
- остальное как у Silhouette

Функционалы качества в задачах кластеризации

«External evaluation»

Пусть есть чёткие (без пересечений) кластеризации множества $\mathcal{X} = \{x_1, \ldots, x_m\}$:

$$U = U_1 \cup \ldots \cup U_{|U|}$$

$$V = V_1 \cup \ldots \cup V_{|V|}$$

Оценка качества алгоритма кластеризации не так тривиальна как вычисление точности алгоритма классификации.
В частности, любой функционал качества не должен только ориентироваться на истинные значения меток классов, а скорее, оценивать как эта кластеризация определяет разделения данных, аналогичное набору истинных классов.
Недостаток: знания истинных классов почти никогда не доступны на практике или требуют ручного назначения людьми

Взаимная информация (Mutual Information)

Обозначим $p_i = \frac{|U_i|}{m}, \quad p'_j = \frac{|V_j|}{m}, \quad p_{ij} = \frac{U_i \cap V_j}{m}$, тогда

$$H(U) = -\sum_{i=1}^{|U|} p_i \log p_i, \quad H(V) = -\sum_{j=1}^{|V|} p'_j \log p'_j$$

$$\mathrm{MI}(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} p_{ij} \log \frac{p_{ij}}{p_i p'_j}$$

Взаимная информация — это функция, которая измеряет согласованность двух назначений, игнорируя перестановки.

Функционалы качества в задачах кластеризации

Доступны две различные нормализованные версии этой меры: нормализованная взаимная информация (NMI) и скорректированная взаимная информация (AMI)

AMI (Adjusted mutual information)

$$\mathrm{AMI}(U,V) = \frac{\mathrm{MI}(U,V) - \mathrm{E}[\mathrm{MI}(U,V)]}{\max\{H(U),H(V)\} - \mathrm{E}[\mathrm{MI}(U,V)]}$$

Матожидание можно вычислить аналитически, см. sklearn

Clustering: https://scikit-learn.org/stable/modules/clustering.html

или https://en.wikipedia.org/wiki/Adjusted_mutual_information

Функционалы качества в задачах кластеризации

$\mathrm{AMI} \approx 0$ — случайное присвоение меток для любого числа кластеров и наблюдений.
$\mathrm{AMI} = 1$ — два присвоения меток равны.

```
from sklearn.metrics import mutual_info_score

from sklearn.metrics import normalized_mutual_info_score

from sklearn.metrics.cluster import adjusted_mutual_info_score

adjusted_mutual_info_score([0, 0, 1, 1], [0, 0, 1, 1])
```

Функционалы качества в задачах кластеризации

V-мера — среднее гармоническое homogeneity и completeness:

- однородность: каждый кластер содержит только объекты одного класса
- полнота: все объекты конкретного класса отнесены в один кластер

$$V = \frac{(1+\beta) \times \text{homogeneity} \times \text{completeness}}{\beta \times \text{homogeneity} + \text{completeness}}$$

По умолчанию $\beta = 1$, чем больше β, тем важнее однородность.

Замечание. V-мера симметрична:

$$\text{homogeneity\_score}(a, b) = \text{completeness\_score}(b, a)$$

```
from sklearn.metrics.cluster import v_measure_score
```

Rand index

Пусть a — количество пар элементов, которые одновременно находятся в одном кластере в U и в одном классе в V
b — количество пар элементов, которые находятся в разных кластерах в U и в разных классах в V

$$\mathrm{RI} = \frac{a+b}{C_m^2} \in [0,1],$$

где C_m^2 — общее количество возможных пар в $\mathcal{X}$
Однако RI не гарантирует, что случайные присвоения меток получат значение, близкое к нулю, поэтому требуется калибровка

$$\mathrm{ARI} = \frac{\mathrm{RI} - \mathrm{M}[\mathrm{RI}]}{\max \mathrm{RI} - \mathrm{M}[\mathrm{RI}]}$$

Функционалы качества в задачах кластеризации

$U \backslash V$	V_1	V_2	$\cdots$	V_s	$\sum$
U_1	n_{11}	n_{12}	$\cdots$	n_{1s}	a_1
U_2	n_{21}	n_{22}	$\cdots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U_r	n_{r1}	n_{r2}	$\cdots$	n_{rs}	a_r
$\sum$	b_1	b_2	$\cdots$	b_s	

$n_{ij} = |U_i \cap V_j|$ — количество общих объектов.

$$\mathrm{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{m}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_i}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{m}{2}}$$

ARI может принимать отрицательные значение, если $\mathrm{RI} < \mathrm{M}[\mathrm{RI}]$

Функционалы качества в задачах кластеризации

Можно рассматривать кластеризацию как классификацию пар

$$\{x_1,\ldots,x_m\} \to \{(1,1),\ldots,(i,j),\ldots,(m,m)\}$$

$a_U(i,j) = 1 \Leftrightarrow i,j \in U$

Можно сравнить a_U и a_V, тогда

$$\mathrm{RI} = \frac{TP + TN}{TP + FP + FN + TN}$$

Fowlkes-Mallows index (FMI) (есть и другие)
— среднее геометрическое точности и полноты

$$\mathrm{FMI} = \frac{TP}{\sqrt{(TP + FP)(TP + FN)}}$$

```
from sklearn.metrics.cluster import fowlkes_mallows_score
```