Метрики качества. Несбалансированность классов.

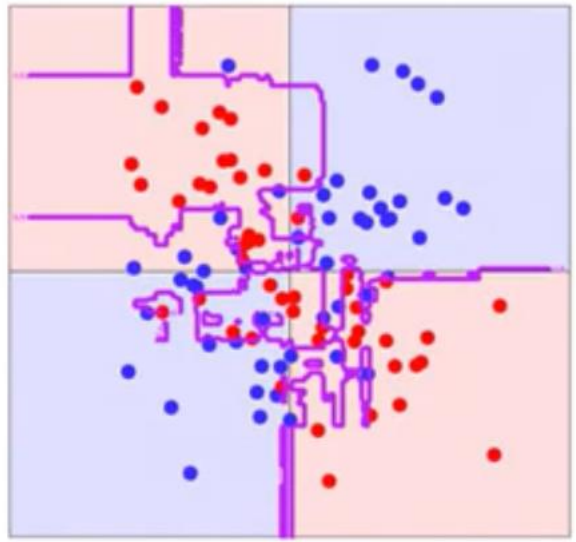
Дерево решений. Что наблюдаем?

- Признаки в вершины дерева выбираются автоматически из набора признаков. Поэтому можно составить произвольный набор признаков, а в процессе обучения автоматически выберутся информативные и проигнорируются неинформативные.
- Если некоторые примеры классифицируются неправильно, можно заново обучить только те вершины дерева, из-за которых это происходит, что очень удобно, когда объем обучающих данных большой и обучение занимает много времени. Кроме того, при тренировке разных поддеревьев могут оказаться более эффективными разные алгоритмы обучения. Обучение заново только части дерева позволяет изменить результат классификации одних объектов, не затрагивая классификацию других объектов.

- при обучении дерево уделяет повышенное внимание классам с большим числом обучающих примеров, и может полностью проигнорировать классы с малым числом обучающих примеров. Поэтому сбалансированность обучающего множества при обучении деревьев очень важна.

- Требуются специальные методы предотвращения переобучения.

ПЕРЕОБУЧЕНИЕ ДЕРЕВЬЕВ



- Построим дерево по части исходных данных
- Тестировать будем на оставшейся части
- Для каждой вершины:
  - Обрежем ветку с корнем в этой вершине
  - Если обрезанное дерево будет лучше справляться с тестами, так и оставим обрезанную ветку, иначе вернём как было

Причины переобучения:

- обучающих данных недостаточно для того, чтобы восстановить по ним информативную закономерность.
- При нехватке тренировочных данных высока вероятность выбрать закономерность, которая выполняется только на этих данных, но не будет верна для других объектов.
- Для деревьев решений сложность модели – это глубина дерева. Но в разные вершины в процессе обучения попадает разное число тренировочных примеров, из-за чего в разных ветвях оптимальной будет разная глубина дерева. Поэтому для деревьев решений требуются специальные методы контроля переобучения.
- **pruning** – удаление тех вершин дерева, которые ухудшают качество классификации данных, не входящих в обучающее множество.

- После обучения каждой вершины дерева происходит разделение ее тренировочного множества на два подмножества.
- На каждом следующем уровне дерева обучающее множество вершины содержит все меньше и меньше примеров. В идеальном случае, если в каждой вершине ее обучающая выборка делится пополам, на уровне дерева h размер выборки в вершине будет в 2^h раз меньше исходного размера выборки.
- А чем меньше размер обучающего множества, тем выше вероятность переобучения.
- **Поэтому для обучения деревьев решений требуются выборки большого размера.**

Качество классификатора

Исходная выборка разбивается на тестовую и тренировочную.

После обучения алгоритма на тренировочной выборке вычисляется число ошибок классификации на контрольной выборке, которое используется как мера качества алгоритма.

Для задачи классификации с конечным числом классов вводится функция :

$I(a,x) = [a(x) \neq y(x)]$, где $y(x)$ – класс объекта x, $a(x)$ – ответ алгоритма для объекта x.

Меры(метрики) качества классификации

Confusion matrix (матрица ошибок).

	$y = 1$	$y = 0$
$\hat{y} = 1$	True Positive (TP)	False Positive (FP)
$\hat{y} = 0$	False Negative (FN)	True Negative (TN)

$$\underbrace{\text{Классификатор ответил верно?}}_{\text{True или False}}\ \underbrace{\text{К какому классу алгоритм отнёс ответ?}}_{\text{Positive или Negative}}$$

Здесь y^ — это ответ алгоритма на объекте, а y — истинная метка класса на этом объекте.

Объекты, которые алгоритм относит к положительному классу, называются **положительными (Positive)**, те из них, которые на самом деле принадлежат к этому классу – **истинно положительными (True Positive)**, остальные – **ложно положительными (False Positive)**.

False Negative - число ложноотрицательных ,ошибочно отнесенных к некоторому другому классу.
True Negative – число истинноотрицательных, верно не отнесенных к классу.

Точность
(Accuracy или Mean Consequential Error)

$$\mathrm{MCE} = \frac{1}{m}\sum_{i=1}^{m} I[a_i = y_i],$$

доля (процент) объектов, на которых алгоритм выдал правильные ответы.

плох в случае дисбаланса классов, когда представителей одного из класса существенно больше, чем другого.

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

$$precision = \frac{TP}{TP + FP}$$

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

Precision(точность) можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall(полнота) показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.
Recall демонстрирует способность алгоритма обнаруживать данный класс вообще, а precision — способность отличать этот класс от других классов.

Применимы в условиях несбалансированных выборок.

Часто в реальной практике стоит задача найти оптимальный (для заказчика) баланс между этими двумя метриками.

Пример

- Оценить работу спам-фильтра почты. У нас есть 100 не-спам писем, 90 из которых наш классификатор определил верно (True Negative = 90, False Positive = 10), и 10 спам-писем, 5 из которых классификатор также определил верно (True Positive = 5, False Negative = 5).

$$accuracy = \frac{5 + 90}{5 + 90 + 10 + 5} = 86,4$$

- Однако если мы просто будем предсказывать все письма как не-спам, то получим более высокую accuracy:

$$accuracy = \frac{0 + 100}{0 + 100 + 0 + 10} = 90,9$$

При этом, наша модель совершенно не обладает никакой предсказательной силой, так как изначально мы хотели определять письма со спамом. В этом случае необходим переход с общей для всех классов метрики к отдельным показателям качества классов.

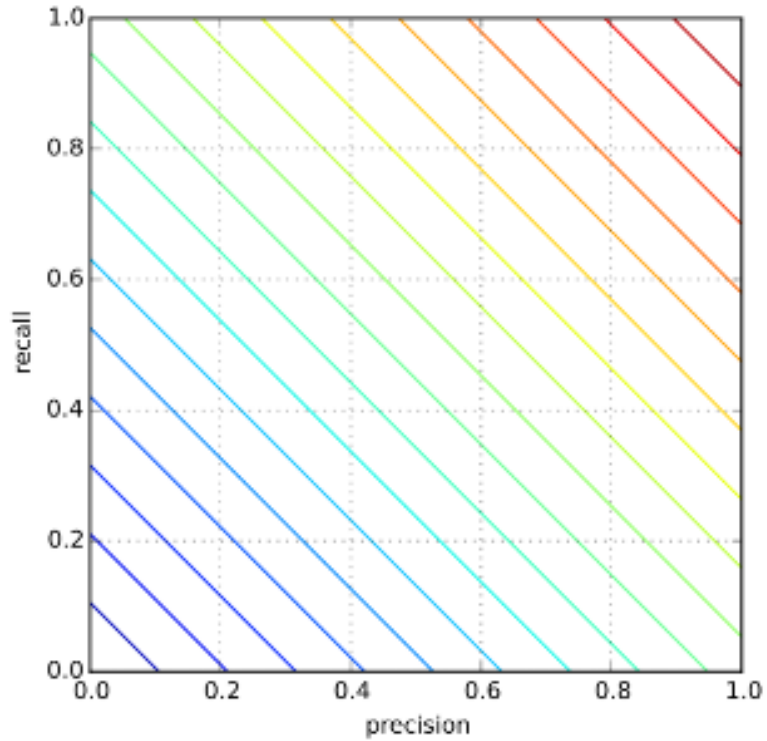
Точность и полнота

- Точность — можно ли доверять классификатору при $a(x) = 1$?
- Полнота — как много положительных объектов находит $a(x)$?

- Оптимизировать две метрики одновременно очень неудобно
- Как объединить?

Арифметическое среднее

$$A = \frac{1}{2}(\text{precision} + \text{recall})$$

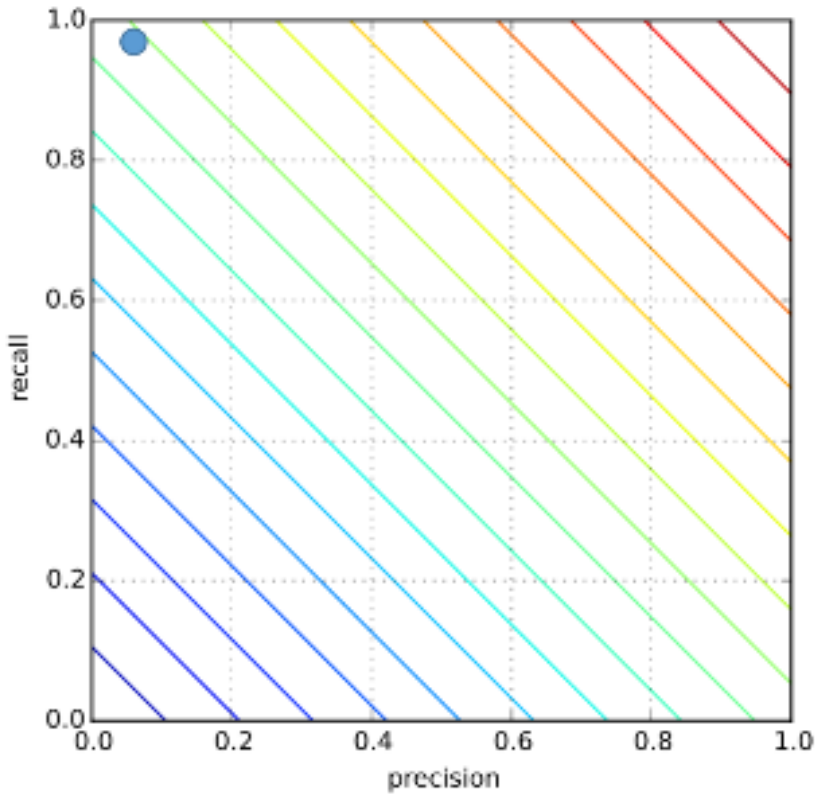


Арифметическое среднее

$$A = \frac{1}{2}(\text{precision} + \text{recall})$$

- $\text{precision} = 0.1$
- $\text{recall} = 1$
- $A = 0.55$

- Плохой алгоритм

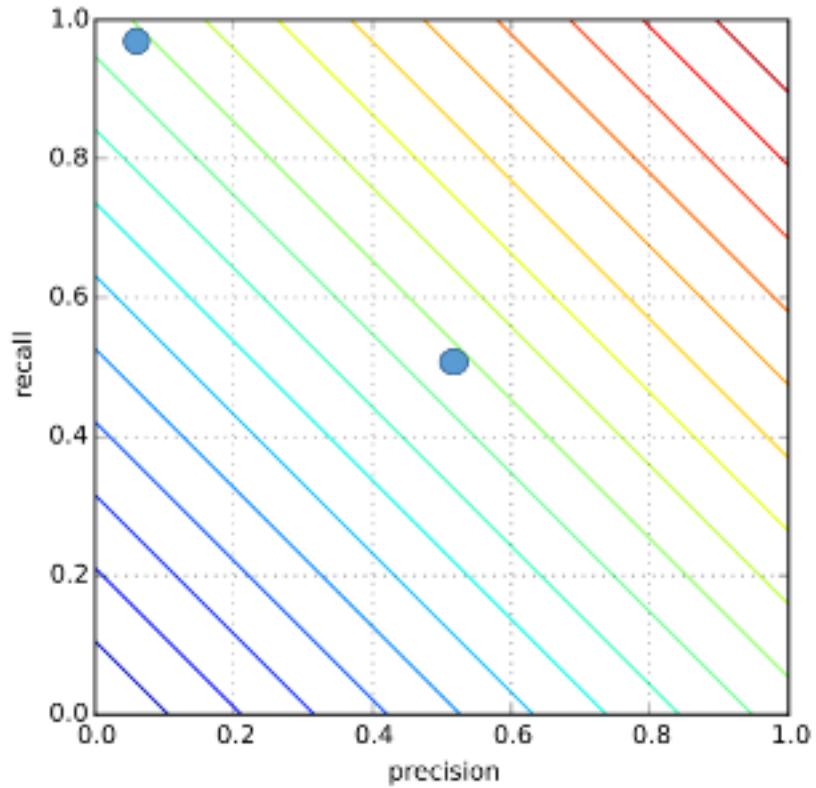


Арифметическое среднее

$$A = \frac{1}{2}(\text{precision} + \text{recall})$$

- $\text{precision} = 0.55$
- $\text{recall} = 0.55$
- $A = 0.55$

- Нормальный алгоритм
- Но качество такое же, как у плохого

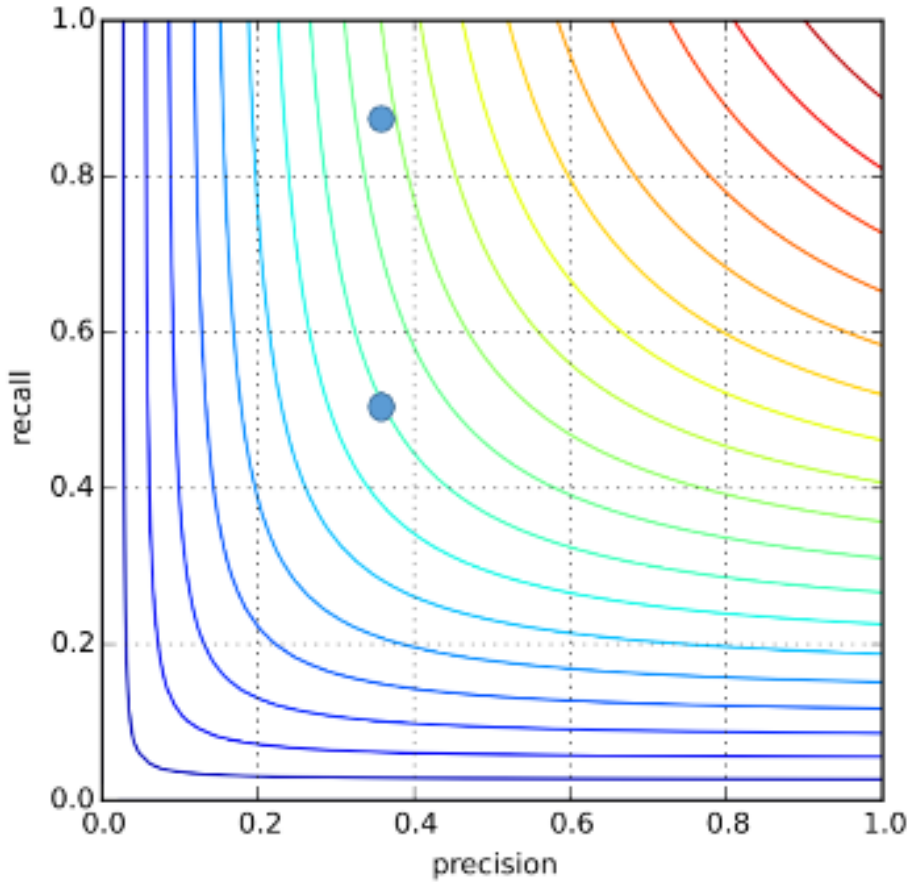


F-мера

$$F = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

- $\text{precision} = 0.4, \text{recall} = 0.5$
- $F = 0.44$

- $\text{precision} = 0.4, \text{recall} = 0.9$
- $M = 0.55$



F-мера (в общем случае Fβ) — среднее гармоническое precision и recall

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

β

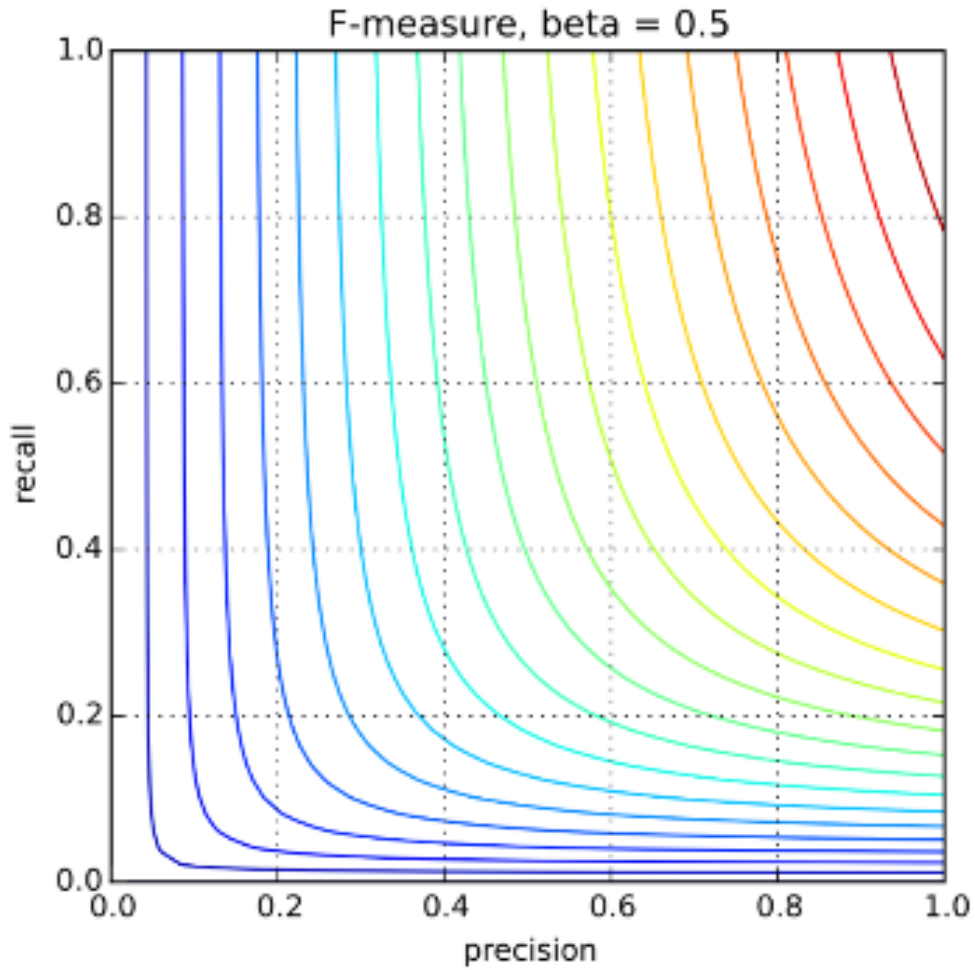
где Fβ принимает значения в диапазоне 0<β<1 если вы хотите отдать приоритет точности, а при β>1 приоритет отдается полноте. При β=1 формула сводится к предыдущей и вы получаете сбалансированную F-меру (также ее называют F_1).

Пример-→

F-мера

$$F_\beta = (1 + \beta^2) \frac{\text{precision} * \text{recall}}{\beta^2 * \text{precision} + \text{recall}}$$

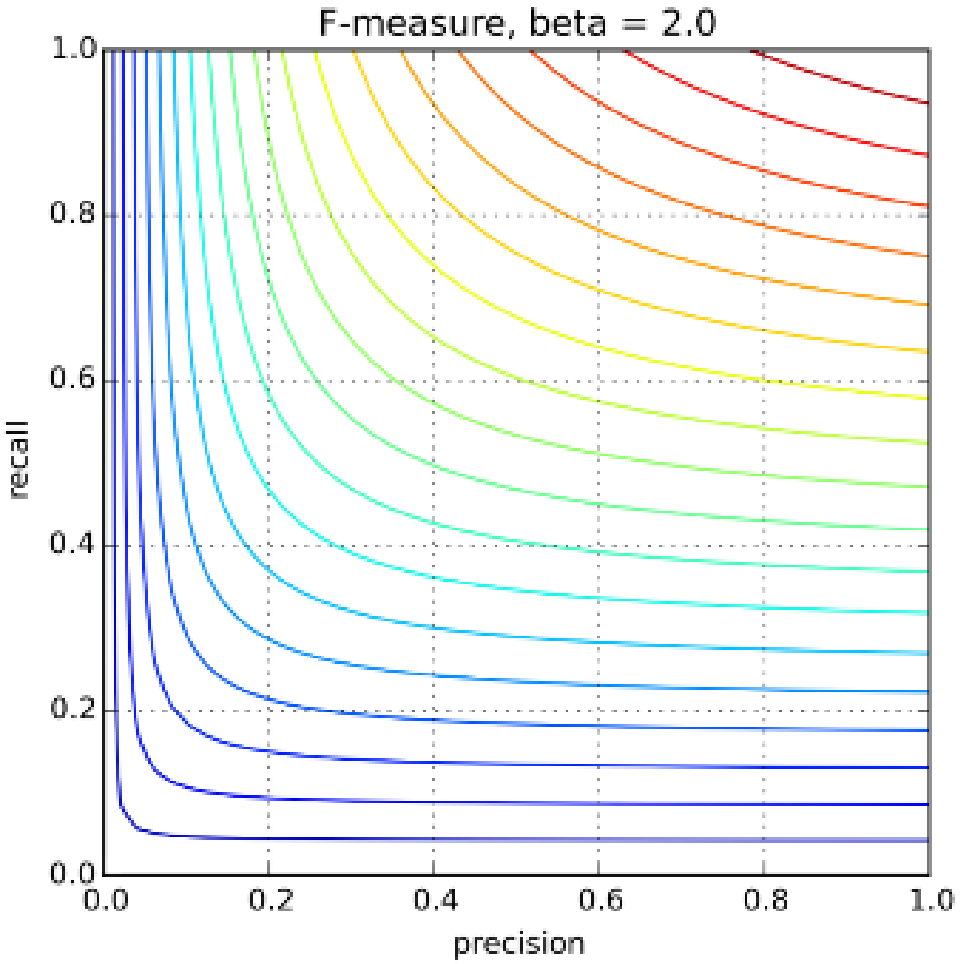
- $\beta = 0.5$
- Важнее полнота



F-мера

$$F_\beta = (1 + \beta^2) \frac{\text{precision} * \text{recall}}{\beta^2 * \text{precision} + \text{recall}}$$

- $\beta = 2$
- Важнее точность



True negative rate| False positive Rate

$$\mathrm{TNR} = \mathrm{Specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

процент правильно классифицированных объектов негативного класса.

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{TNR}$$

доля объектов негативного класса, которых мы ошибочно отнесли к положительному

Несбалансированность классов

- Классификаторы, построенные на основе выборки, в которой репрезентативность классов несбалансирована, имеют в процессе практического использования склонность с большей вероятностью относить новые наблюдения к классам, представленным большим числом обучающих примеров.

- Внимательно и осознанно выбирайте метрику под ваш датасет.
- В случае сильно несбалансированных классов можно получить бесполезную метрику
- Пример: Поиск мошеннических транзакций
- Датасет: 1 млн примеров, из них мошеннических транзакций 1% •Метрика: Аккуратность (Accuracy)
- •Классификатор: всех относим к классу “Честные” •Получаем качество 99%

- проблемы, возникающие при вычислении качества алгоритмов классификации на выборках с несбалансированными классами—

→ введение сбалансированных мер

Коэффициент Мэттьюса (MCC – Matthews correlation coefficient) для несбалансированных выборок

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \in [-1, +1],$$

Каппа Коэна (Cohen's Kappa)

$$\kappa = \frac{\text{Accuracy} - \text{Accuracy}_{\text{chance}}}{1 - \text{Accuracy}_{\text{chance}}}$$

	$a = 0$	$a = 1$
$y = 0$	m_{00}	m_{01}
$y = 1$	m_{10}	m_{11}

$$\text{Accuracy} = \frac{m_{00} + m_{11}}{m}$$

$$\text{Accuracy}_{\text{chance}} = \frac{m_{00} + m_{01}}{m}\frac{m_{00} + m_{10}}{m} + \frac{m_{10} + m_{11}}{m}\frac{m_{01} + m_{11}}{m}$$

красным выделена вероятность угадать класс 0, а синим – класс 1.

Сбалансированная точность (Balanced Accuracy)

$$\mathrm{BA} = \frac{R_1 + R_0}{2} = \frac{1}{2}\left(\frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} + \frac{\mathrm{TN}}{\mathrm{TN}+\mathrm{FP}}\right)$$

Все указанные функционалы реализованы в библиотеке scikit-learn:

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import matthews_corrcoef
from sklearn.metrics import cohen_kappa_score
```

AUC-ROC и AUC-PR

- При конвертации вещественного ответа алгоритма в бинарную метку, мы должны выбрать какой-либо порог, при котором 0 становится 1. Естественным и близким кажется порог, равный 0.5, но он не всегда оказывается оптимальным, например, при вышеупомянутом отсутствии баланса классов.
- Одним из способов оценить модель в целом, не привязываясь к конкретному порогу, является AUC-ROC (или ROC AUC) — площадь (*A*rea *U*nder *C*urve) под кривой ошибок (*R*eceiver *O*perating *C*haracteristic curve).

- Данная кривая представляет из себя линию от (0,0) до (1,1) в координатах True Positive Rate (TPR) и False Positive Rate (FPR)

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

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

ROC-кривая

- Receiver Operating Characteristic
- Ось X — False Positive Rate

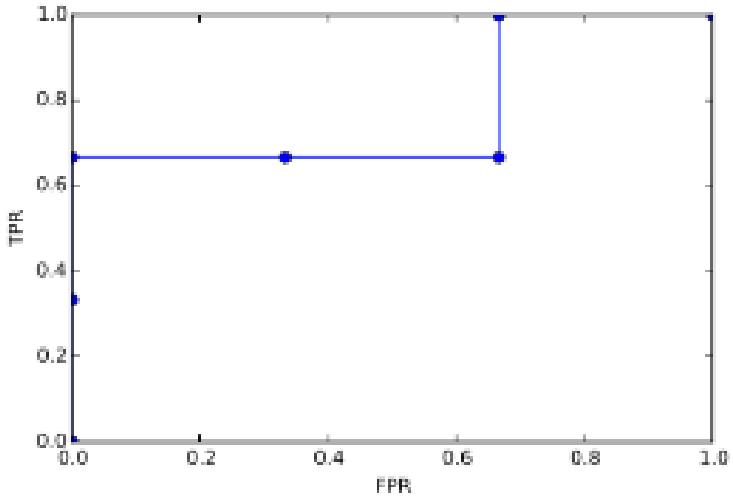
$$FPR = \frac{FP}{FP + TN}$$

Число отрицательных объектов

- Ось Y — True Positive Rate

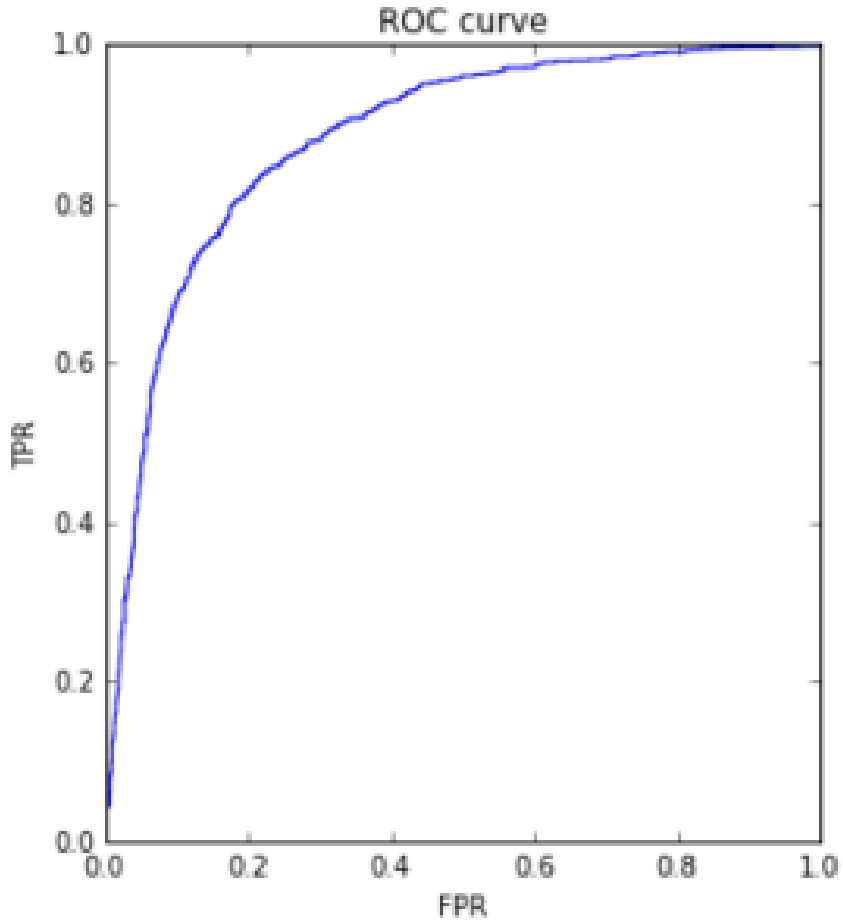
$$TPR = \frac{TP}{TP + FN}$$

Число положительных объектов

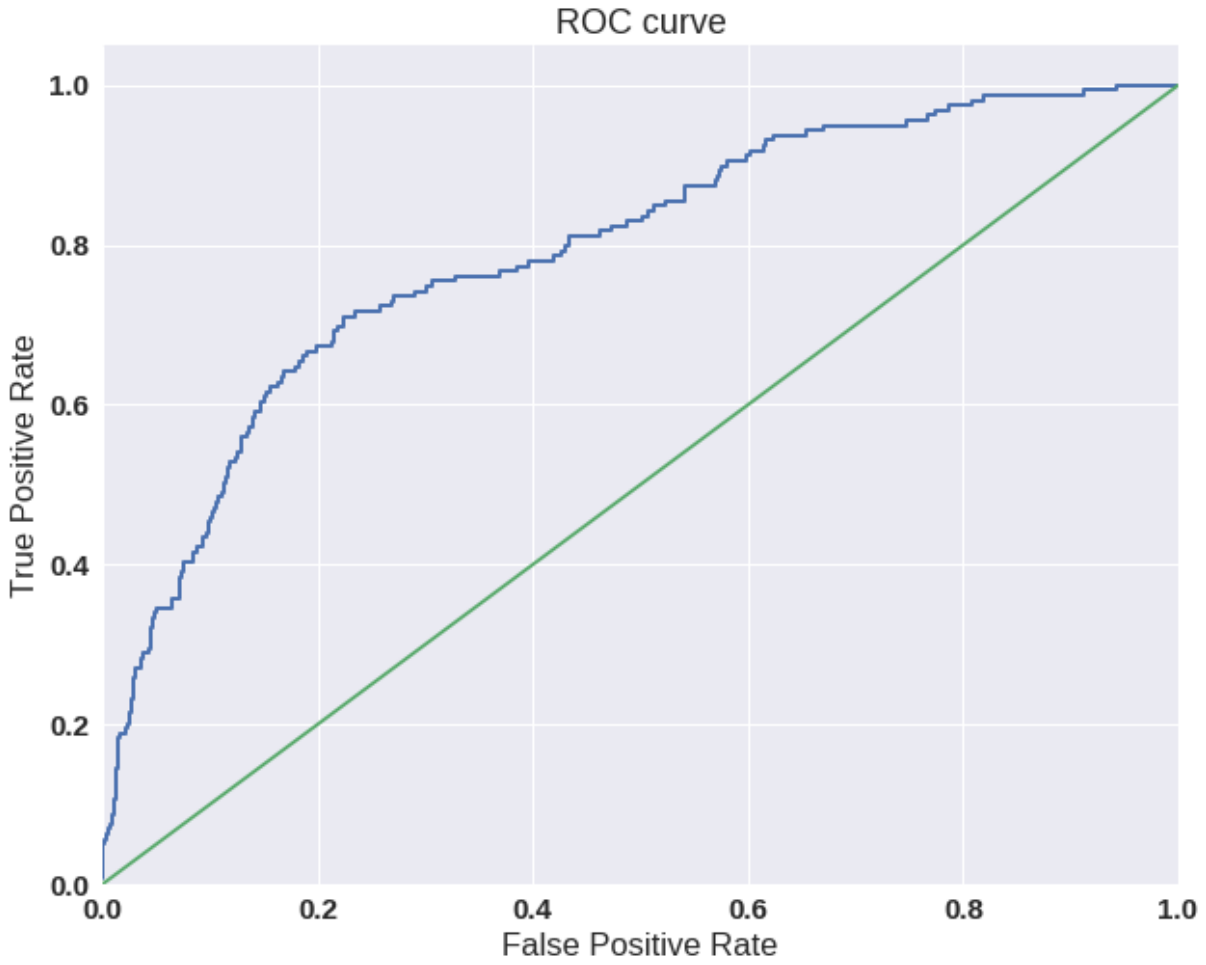


ROC-кривая

- Левая точка: (0, 0)
- Правая точка: $(1, 1)$
- Для идеального классификатора проходит через (0, 1)
- AUC-ROC — площадь под ROC-кривой



- В идеальном случае, когда классификатор не делает ошибок (FPR = 0, TPR = 1) мы получим площадь под кривой, равную единице; в противном случае, когда классификатор случайно выдает вероятности классов, AUC-ROC будет стремиться к 0.5, так как классификатор будет выдавать одинаковое количество TP и FP.
- Каждая точка на графике соответствует выбору некоторого порога.
- Площадь под кривой в данном случае показывает качество алгоритма (больше — лучше), кроме этого, важной является крутизна самой кривой — мы хотим максимизировать TPR, минимизируя FPR, а значит, наша кривая в идеале должна стремиться к точке (0,1).



Критерий AUC-ROC устойчив к несбалансированным классам

AUC-ROC

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

- FPR и TPR нормируются на размеры классов
- AUC-ROC не поменяется при изменении баланса классов
- Идеальный алгоритм: $\mathrm{AUC{-}ROC} = 1$
- Худший алгоритм: $\mathrm{AUC{-}ROC} \approx 0.5$

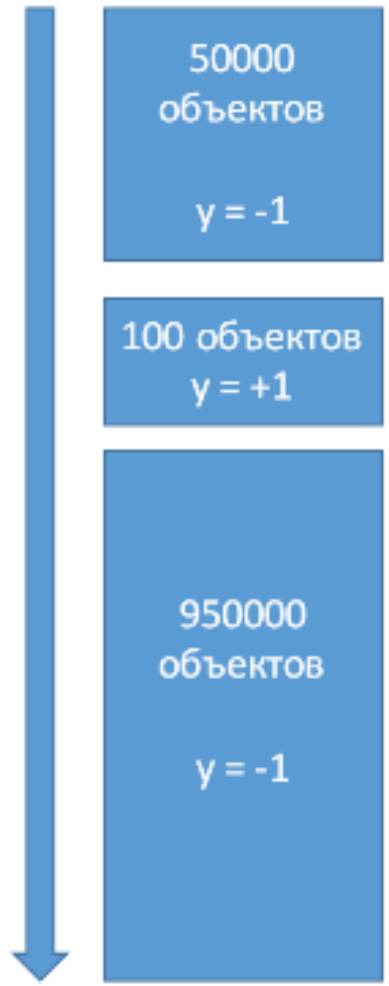
AUC-PRC

$$\text{precision} = \frac{TP}{TP+FP}; \qquad \text{recall} = \frac{TP}{TP+FN}$$

- Точность поменяется при изменении баланса классов
- AUC-PRC идеального алгоритма зависит от баланса классов
- Проще интерпретировать, если выборка несбалансированная
- Лучше, если задачу надо решать в терминах точности и полноты

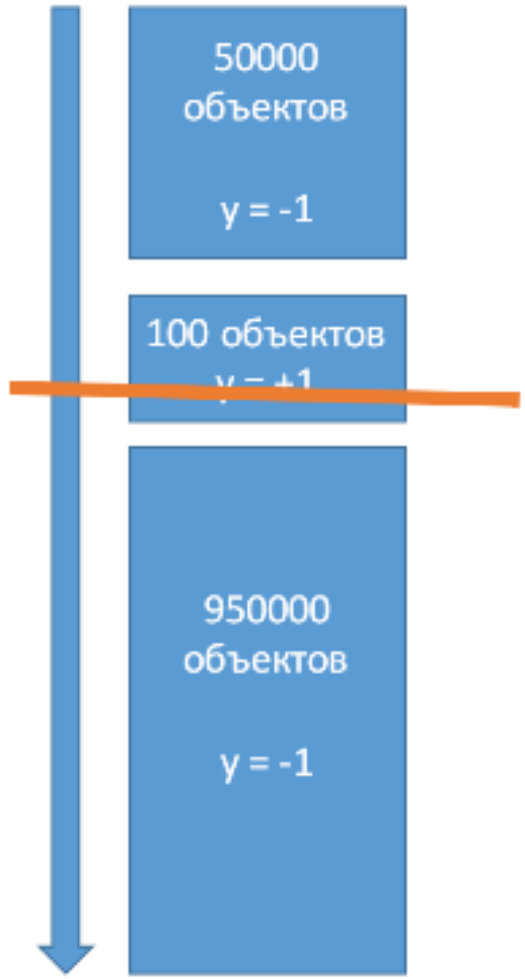
Пример

- AUC-ROC = 0.95
- AUC-PRC = 0.001



Пример

- Выберем конкретный классификатор
- $a(x) = 1$ — 50095 объектов
- Из них FP = 50000, TP = 95
- TPR = 0.95, FPR = 0.05
- precision = 0.0019, recall = 0.95



Методы борьбы с несбалансированностью

- Различия в мощностях классов могут привести и к проблемам с обучением
- Считается, что выборка несбалансированная, если количество объектов в одном из классов превосходит количество объектов в другом в 10 раз или больше.
- При этом меньший класс называется минорным, больший — доминирующим.
- Undersampling - удаляет случайные объекты доминирующего класса до тех пор, пока соотношение классов не станет приемлемым;
- oversampling- дублирует случайные объекты минорного класса

данные методы применяются лишь к обучающей выборке, а контрольная выборка остается без изменений.

- SMOTE - заключается в дополнении минорного класса синтетическими объектами.
- Генерация нового объекта производится следующим образом:
- Выбирается случайный объект x1 минорного класса,
- Для него выделяются k ближайших соседей из этого же класса (k — настраиваемый параметр),
- из этих соседей выбирается один случайный x2.
- Новый объект вычисляется как точка на отрезке между x1 и x2: $\alpha x1 + (1 - \alpha)x2$, для случайного $\alpha \in (0, 1)$.

Задание 1: Получить все рассмотренные оценки качества классификации.

Построить Roc Кривые для классификаторов Organics

- https://www.youtube.com/watch?v=9q7Am6ZMrvs

sklearn.metrics.**roc_curve**(*y_true, y_score, pos_label=None, sample_weight=None, drop_intermediate=True*)

y_true : ***массив, форма = [n_samples]***
Истинные двоичные метки. Если метки не являются {-1, 1} или {0, 1}, тогда явно указывается pos_label.
y_score : ***массив, форма = [n_samples]***
Целевые оценки могут быть либо оценками вероятности положительного класса, доверительными значениями, либо мерой решения без пороговых значений (как во
pos_label : ***int или str, по умолчанию = None***
Метка позитивного класса. Когда `pos_label=None`, если y_true находится в {-1, 1} или {0, 1}, `pos_label`установлено в 1, в противном случае возникнет ошибка.
sample_weight : ***массив формы shape = [n_samples], необязательно***
Образцы весов.
drop_intermediate : ***логический, необязательный (по умолчанию = True)***
Стоит ли отбрасывать некоторые неоптимальные пороги, которые не появляются на кривой ROC. Это полезно для создания более легких кривых ROC.

Задание 2

1) Рассчитать оценки на основе несбалансированных и сблансированных мер для задачи классификации Organics. Используем полную несбалансированную выборку.

2) Organics – сбалансировать выборку. Обучить, сравнить. Обучить модель на сбалансированной выборке ONLINEADS. Использовать сбалансированные и несбалансированные метрики.

Влияет ли дисбаланс выборки на качество классификатора для деревьев решений? Проанализировать метрики на ошибки. Сделать выводы.