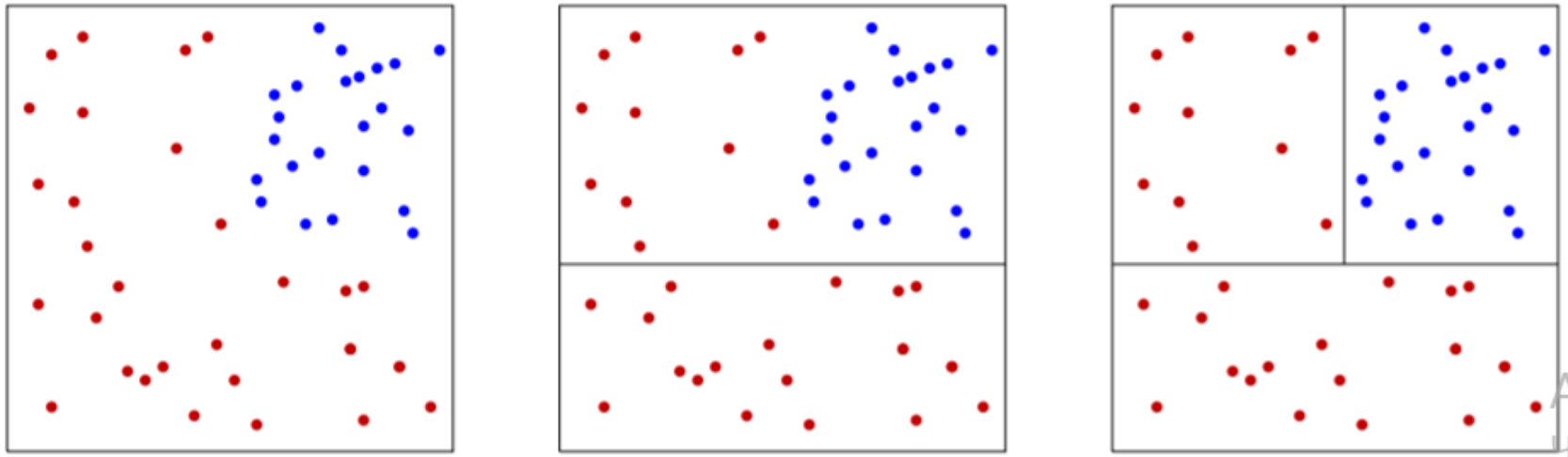
Популярные алгоритмы

- See5/C5.0 [Quinlan et., 1997] ← C4.5 [Quinlan, 1993] ← ID3 [Quinlan, 1979] ← CLS [Hunt & Marin & Stone & 1966]
- CART — Classification and Regression Trees [Breiman & Friedman & Olshen & Stone, 1984] ← CHAID [Kass, 1980] ← THAID [Morgan & Messenger 1973] ← AID [Morgan & Sonquist, 1963]

— это *жадные рекурсивные* алгоритмы, на каждом шаге разбивающие очередной ящик, чтобы добиться максимального уменьшения взвешенной *неоднородности*:



Алгоритмы построения решающего дерева по обучающей выборке

- C4.5- C5.0: использует критерий Gain Ratio (нормированный энтропийный критерий). Критерий останова — ограничение на число объектов в листе. Стрижка производится с помощью метода Error-Based Pruning, который использует оценки обобщающей способности для принятия решения об удалении вершины. Обработка пропущенных значений осуществляется с помощью метода, который игнорирует объекты с пропущенными значениями при вычислении критерия ветвления, а затем переносит такие объекты в оба поддерева с определенными весами. Количество потомков у узла не ограничено. Одним из «плюсов» Алгоритма C5.0 является его способность осуществлять «обрезку» построенного дерева решений, т.е. «post-pruning». Отсекаются те узлы и ветви, использование которых мало влияет на результаты классификации (не сильно уменьшает ошибки классификации). Иногда вместо того, чтобы «отсекать» ветви, их переносят выше по дереву или же заменяют более простыми (менее разветвлёнными). Этот процесс называют «subtree raising» (т.е. «подъём поддерева») и «subtree replacement» (т.е. «замена поддерева»), соответственно. Алгоритм C5.0 использует оба вида post-pruning и показывает хорошие результаты.

Hastie T., Tibshirani R., Friedman J. (2009). The Elements of Statistical Learning.

- CART(Classification and Regression Tree): использует критерий Джини. Стрижка осуществляется с помощью CostComplexity Pruning. Для обработки пропусков используется метод суррогатных предикатов. Это Алгоритм построения бинарного дерева решений – дихотомической классификационной модели. Каждый узел дерева при разбиении имеет только двух потомков.

ML: ожидание

```
clf = XGBClassifier()
clf.fit(X_train, y_train)
labels = clf.predict(X_test)
```

ML: реальность

Выгрузка таблиц

Соединение по ключам

Расчёт агрегатов

Заполнение пропусков

```
clf = XGBClassifier()
clf.fit(X_train, y_train)
labels = clf.predict(X_test)
```

Калибровка

Усреднение моделей

Расчёт целевой функции

Оптимизация

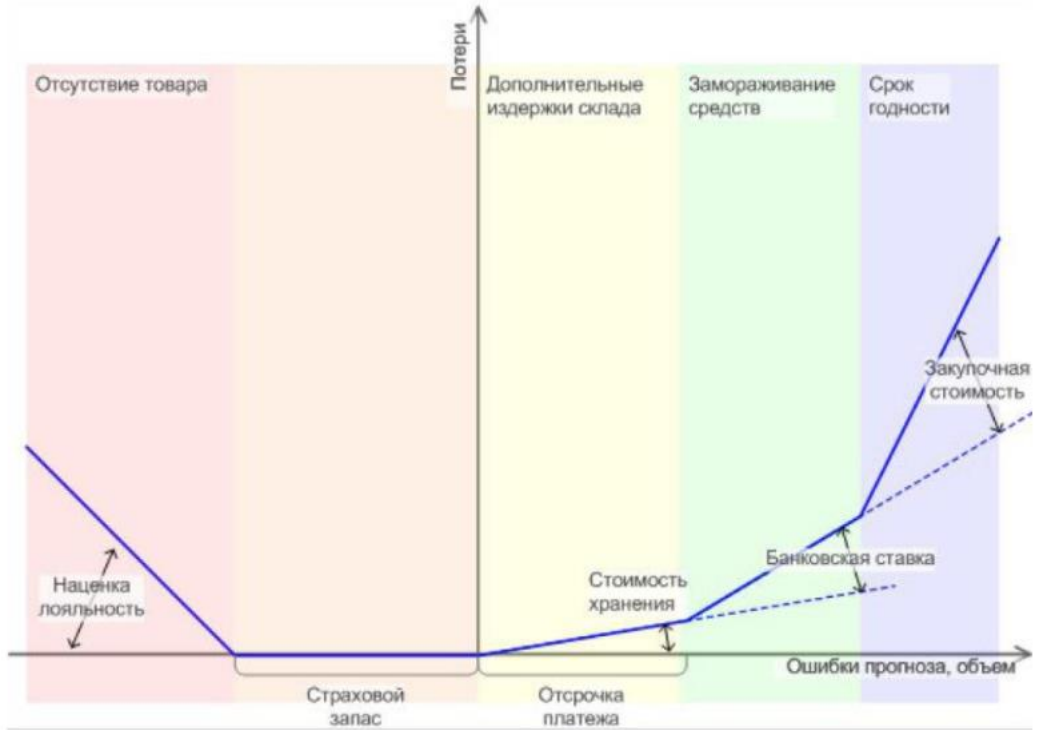
- Мы подали в модель всю релевантную информацию?
- Не подали ли мы туда ничего лишнего?
- Сможет ли модель корректно обработать признаки?

- Что должна выдавать модель?
- Насколько отличается её прогноз от желаемого?
- За счёт чего прогноз можно было бы улучшить?

Целевая функция

- На основе моделей принимаем решение
- Решение приносит пользу
  - Например, увеличение прибыли

- Неправильное решение приводит к потерям
- Хотим прогноз, который минимизирует потери!



Иерархия метрик

- Бизнес-метрики (онлайн)
  - То, ради чего вы работаете (прибыль, счастье пользователя, …)
- Прокси-метрики (онлайн, но быстрее)
  - То, что можно измерить быстро (конверсия, средний чек, …)
- Меры качества (оффлайн, для выбора гиперпараметров)
  - Доля верных/неверных прогнозов, …
- Функции потерь (оффлайн, для подгона основных парамеров)
  - Правдоподобие, MSE, …
  - Должны быть гладкими и быстро вычисляться

Хотелось бы жить ближе к бизнес-метрикам, но это не всегда удаётся

Офлайн-метрики для классификации

Accuracy

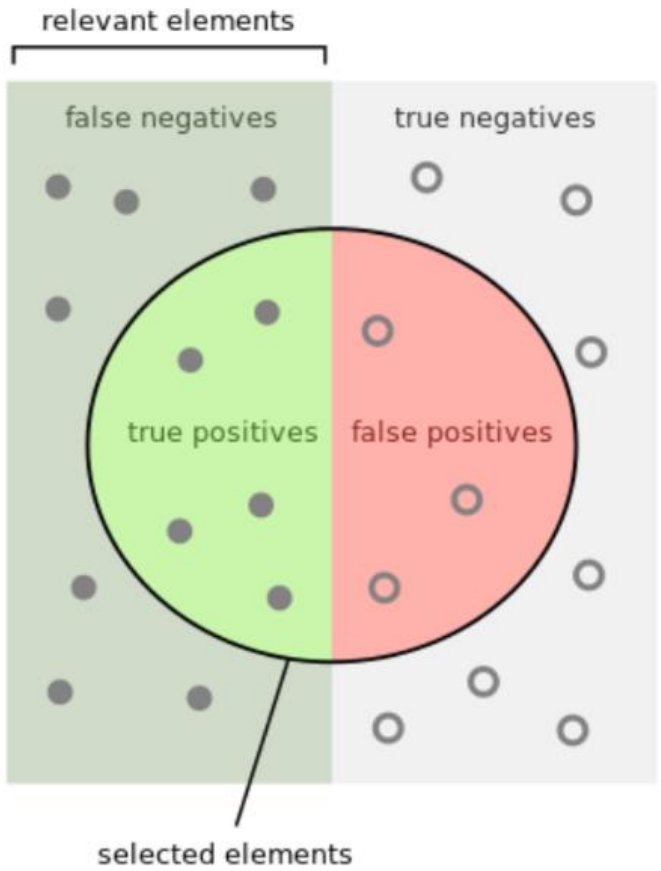
```
Accuracy = np.mean(ytrue == ypred)
```

Лучшее константное решение - самый часто встречающийся класс

Пример с несбалансированными классами:

```
y={0,1}; np.mean(y) == 0.99
Accuracy = np.mean(ytrue == 1) = 0.99
```

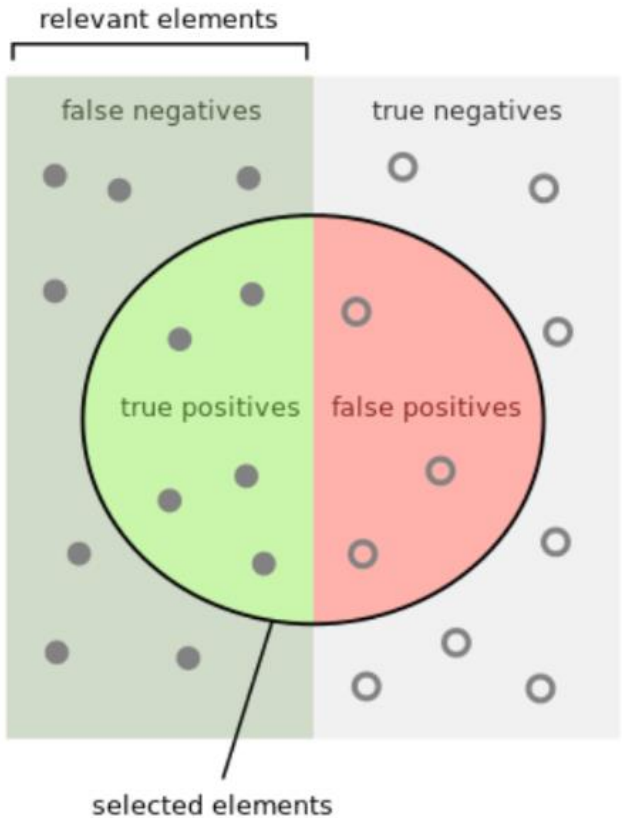
Precision and recall

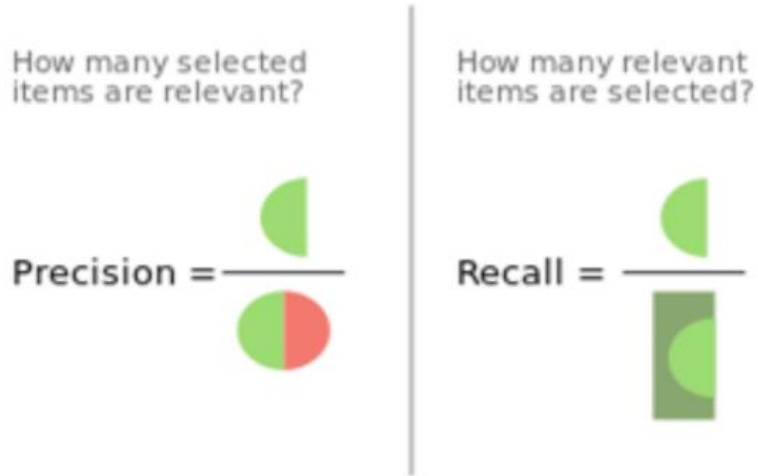


False Positive - ошибка I рода (ложное срабатывание)

False Negative - ошибка II рода (объект пропущен)

Precision and recall





$$\text{Precision} = \frac{tp}{tp + fp}$$

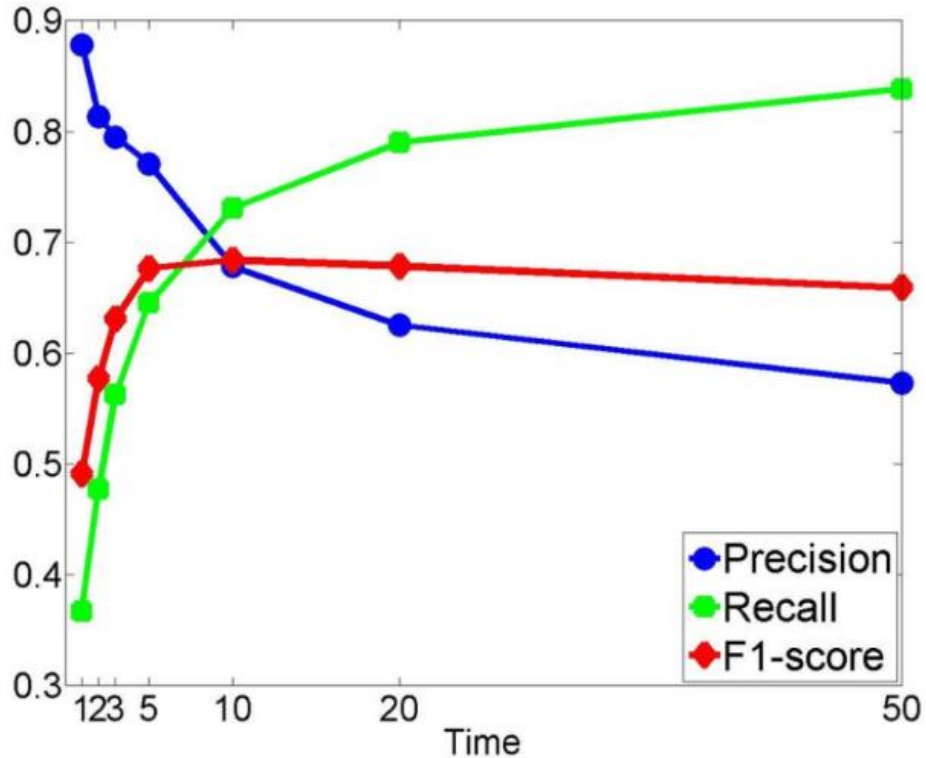
Сколько хороших клиентов среди одобренных?

$$\text{Recall} = \frac{tp}{tp + fn}$$

Какой доле всех хороших мы дали кредит?

F-score

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$



F-score

$$F1 = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

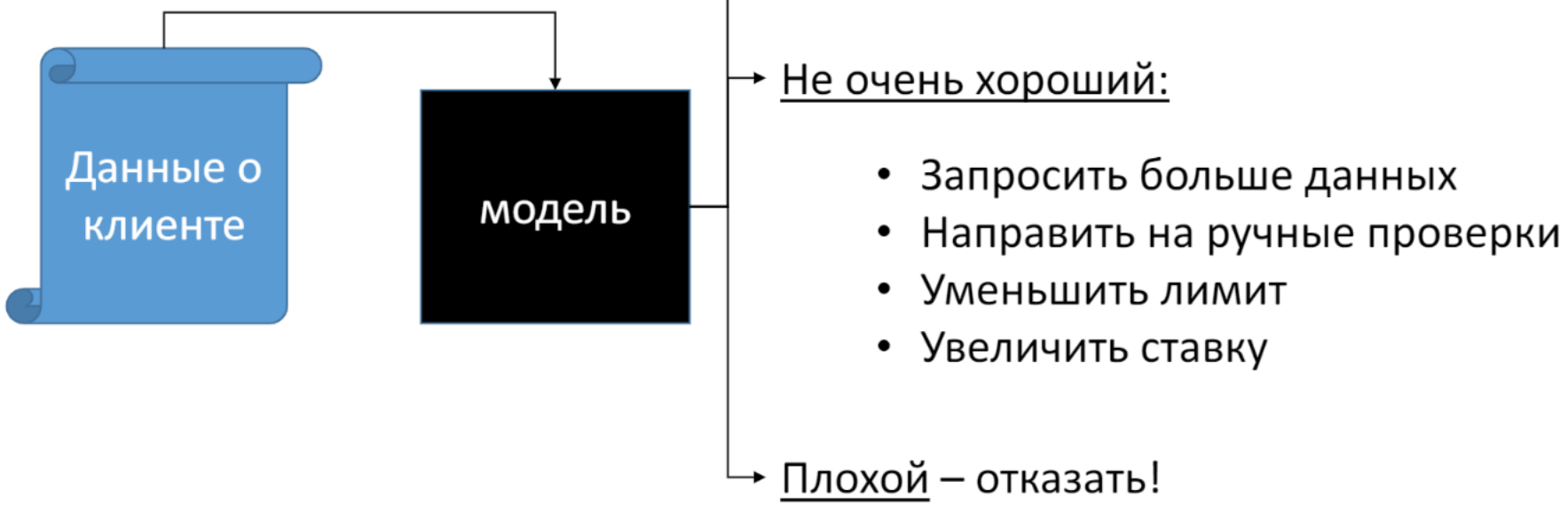
```
y_true = [[1, 2],
          [3, 4, 5],
          [6],
          [7]]

y_pred = [[1, 2, 3, 9],
          [3, 4],
          [6, 12],
          [1]]

mean_f1(y_true, y_pred)
# 0.53333333
```

F-score

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

при $0 < \beta < 1$ предпочтение отдаётся точности
при $\beta > 1$ больший вес приобретает полнота

Кредитный скоринг: бинарной оценки мало

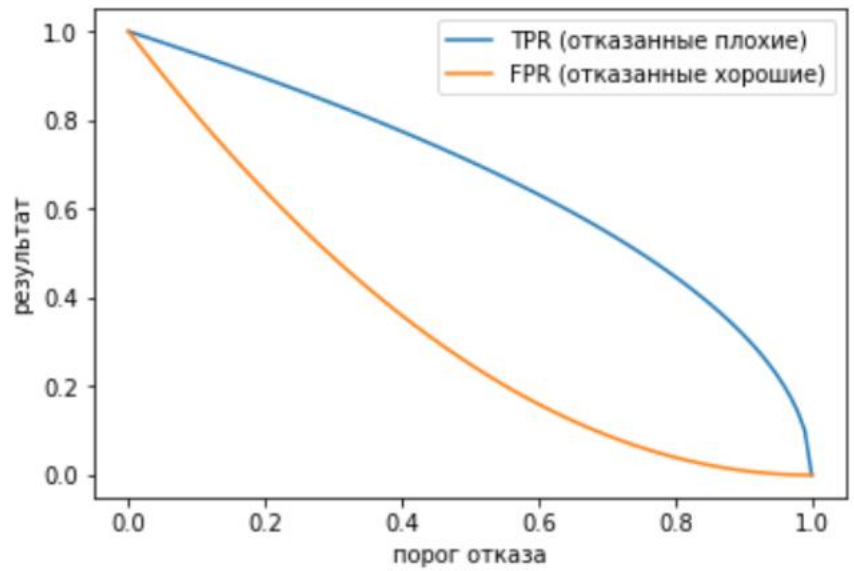


Нужно уметь *ранжировать* клиентов

Хорошее ранжирование + калибровка = точная оценка вероятности

Какой порог выставить?

- $\tilde{p}$ - вероятность дефолта
- Отказывать, если:
- $\tilde{p} > 50\%$?
- $\tilde{p} > 10\%$?
- $\tilde{p} \times \mathbb{E}(profit|bad) + (1 - \tilde{p})\mathbb{E}(profit|good) < 0$
- Как оценить качество, ещё не зная порога отказа?

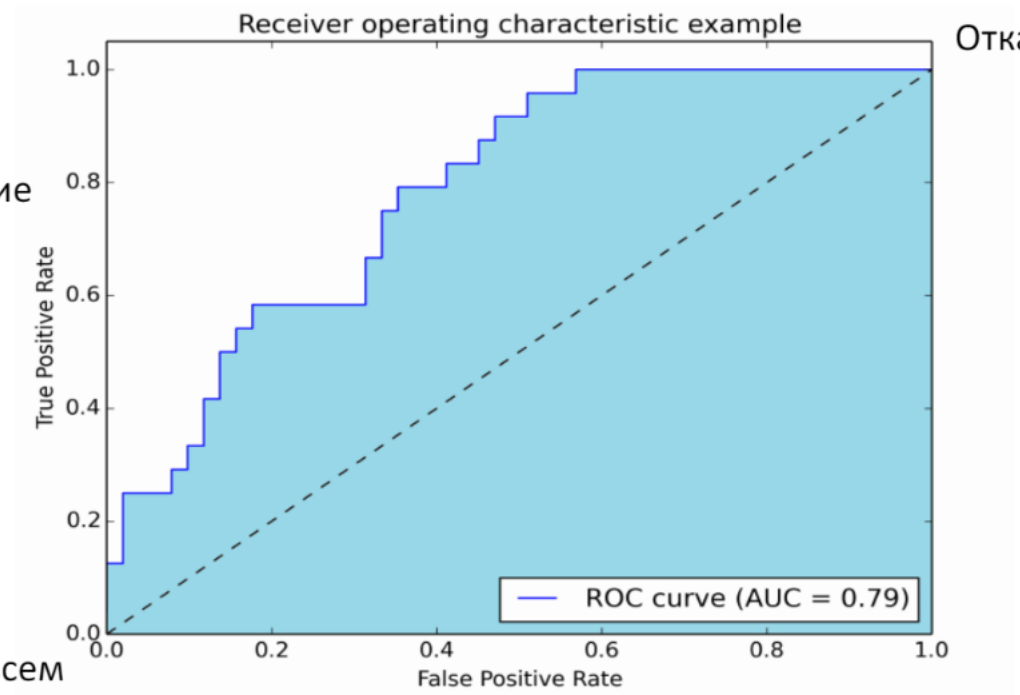


ROC-кривая

Как будет меняться TPR в зависимости от TPR при смене порога?

Отказываем всем

Отказанные плохие

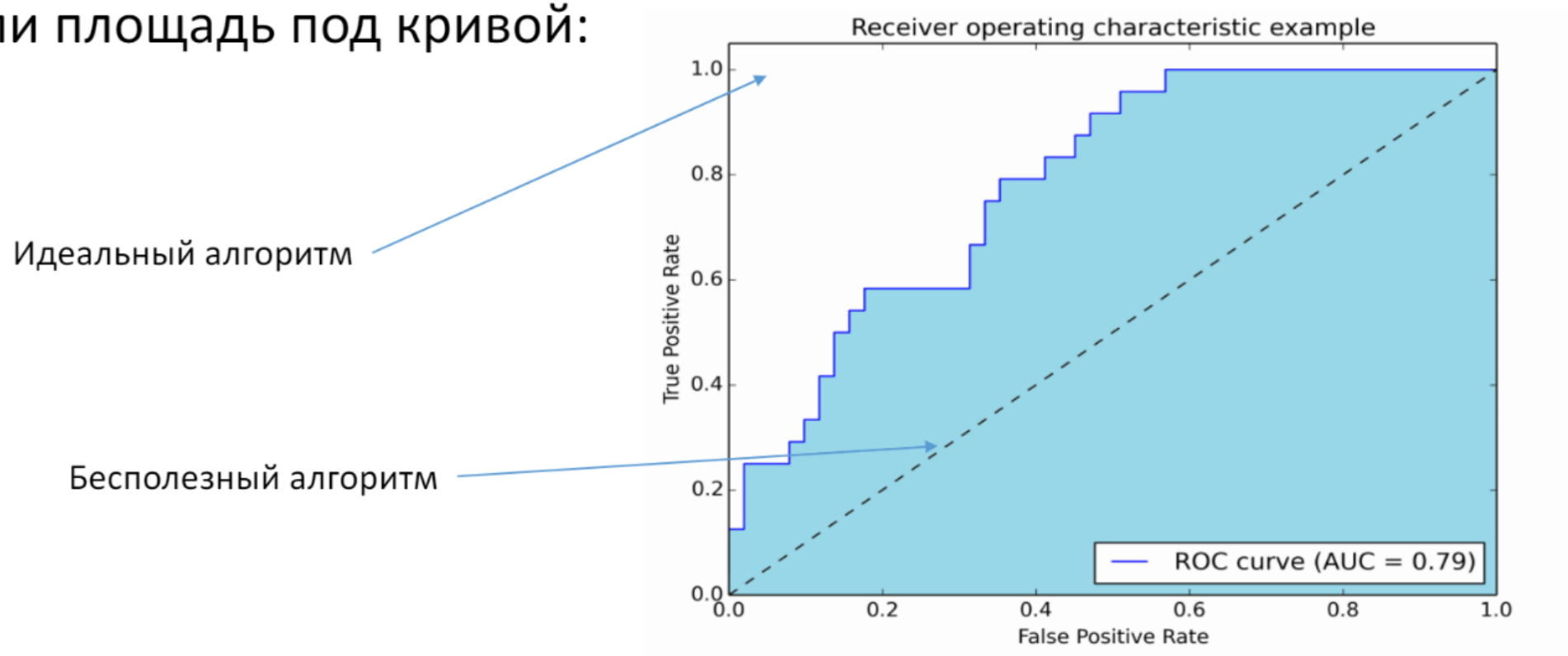


Одобряем всем

Отказанные хорошие

AUC (ROC)

Доля правильно отранжированных пар:
`ypred_i > ypred_j если ytrue_i > ytrue_j`
Или площадь под кривой:



AUC (ROC)

Доля правильно отранжированных пар:

```
ypred_i > ypred_j IF ytrue_i > ytrue_j
```

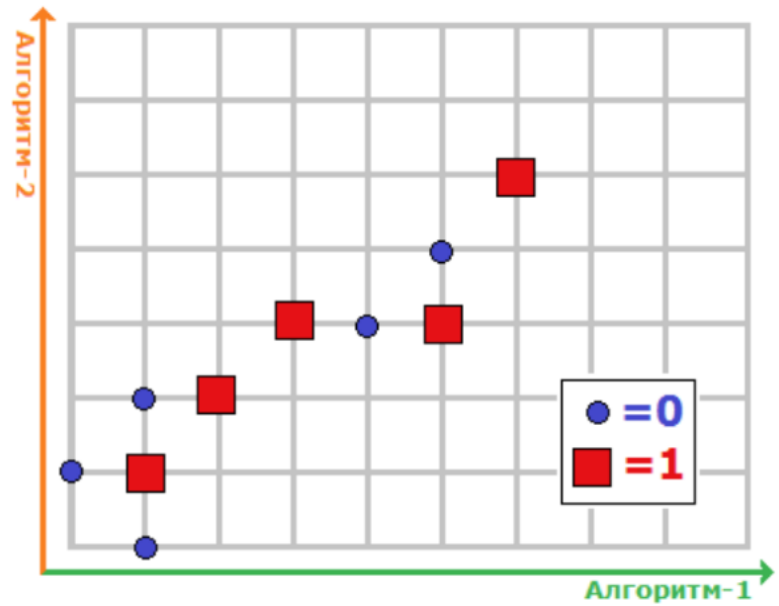
То же самое:

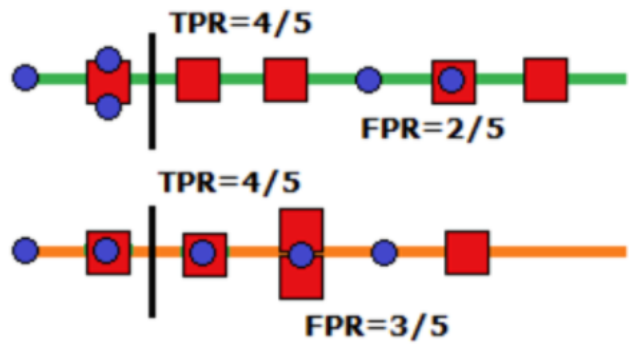
```
ypred_i > ypred_j IF ytrue_i == 1 and ytrue_j == 0
```

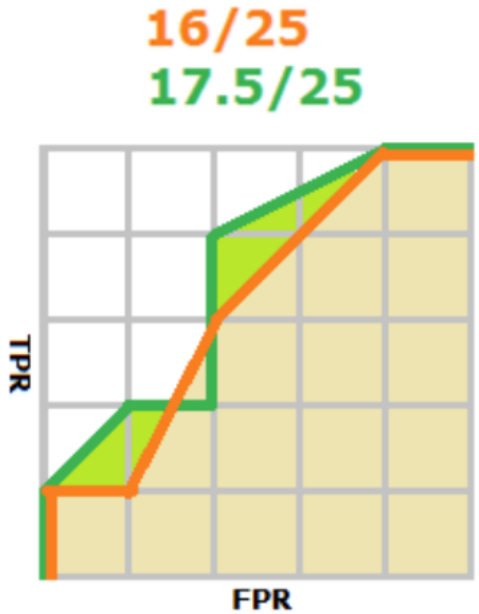
То же самое (с точностью до линейного преобразования):
Средняя по всем порогам доля хороших среди одобренных

AUC (ROC)

Пример: построить ROC-кривую для предсказаний двух алгоритмов







Вероятностный подход

- Мы пытаемся прогнозировать Y, но он случаен
- Будем прогнозировать распределение $p(Y|X)$!
- Правдоподобие: $Likelihood = p(y_1, \dots y_n|x_1, \dots x_n)$
- Если наблюдения независимы: $Likelihood = \prod_{i=1}^{n} p(y_i|x_i)$
- На практике почти всегда работают с суммой логарифмов
- Правдоподобие порождает привычные функции потерь
  - Например, если $p(y|\hat{y}) \sim \mathcal{N}(0, \sigma^2)$, то логарифм правдоподобия равен

$$\log L = -n \log \sqrt{2\pi}\sigma - \sum_{i=1}^{n} \frac{(y - \hat{y})^2}{2\sigma^2} = a - b \times MSE$$

Алгоритм машинного обучения

Модель ML можно разложить на три компоненты

1. Функциональная форма
   - Линейная, дерево, композиция…
2. Целевая функция
   - Разница прогноза и факта, доля ошибок, отступ, правдоподобие…
3. Метод оптимизации
   - Линейное программирование, градиентный спуск, генетические алгоритмы….

Качество тестирования во многом зависит от объема тестовой выборки и 'равномерности' данных, которые попали в обучающую и тестовую выборки.

Проверка качества модели

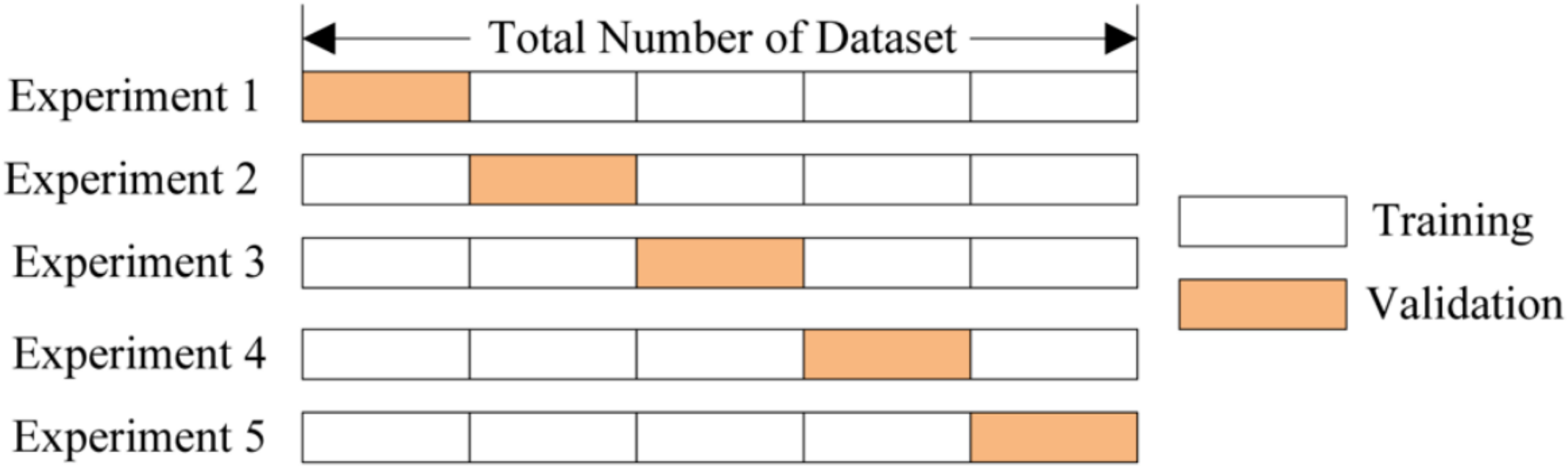
- отложенная выборка (*held-out/hold-out set*). При таком подходе мы оставляем какую-то долю обучающей выборки (как правило от 20% до 40%), обучаем модель на остальных данных (60-80% исходной выборки) и считаем некоторую метрику качества модели (например, самое простое – долю правильных ответов в задаче классификации) на отложенной выборке.
- кросс-валидация (*cross-validation*, на русский еще переводят как скользящий или перекрестный контроль). Тут самый частый случай – K-fold кросс-валидация

Методы ресемплинга (resampling)

Позволяют оценить прогнозную модель, используя обучающий набор данных.

Подобное оценивание выполняется для того, чтобы настроить параметры модели и выяснить ее реальные возможности без участия тестового набора данных.

1. k-блочная кросс-валидация (k-fold cross-validation). Этот метод случайным образом разбивает данные на *k* непересекающихся блоков примерно одинакового размера. Поочередно каждый блок рассматривается, как валидационная выборка, а остальные *k-1* блоков – как обучающая выборка. Модель обучается на *k-1* блоках и прогнозирует валидационный блок. Прогноз модели оценивается с помощью выбранного показателя (правильность (accuracy), среднеквадратическое отклонение (СКО) и т.п.). Процесс повторяется *k* раз, и мы получаем *k* оценок, для которых рассчитывается среднее значение, являющееся итоговой оценкой модели. Обычно *k* выбирают равным 10, иногда 5.



При оценке модели имеющиеся в наличии данные разбиваются на k частей. Затем на k−1 частях данных производится обучение модели, а оставшаяся часть данных используется для тестирования. Процедура повторяется k раз; в итоге каждая из k частей данных используется для тестирования. В результате получается оценка эффективности выбранной модели с наиболее равномерным использованием имеющихся данных. оценивается его средняя ошибка на объектах контрольной подвыборки.

Оценкой скользящего контроля называется средняя по всем разбиениям величина ошибки на контрольных подвыборках.

2. Многократная k-блочная кросс-валидация (repeated k-fold cross-validation). В рамках этого метода k-блочная кросс-валидация выполняется несколько раз. Например, 5-кратная 10-блочная кросс-валидация даст 50 оценок, на основе которых затем будет рассчитана средняя оценка

3. Кросс-валидация на основе метода Монте-Карло (МККВ, Monte Carlo cross-validation, leave-group-out cross-validation). Данный метод заданное количество раз случайным образом разбивает исходный набор данных на обучающую и валидационную выборку в заданной пропорции.

- Кросс-валидация дает лучшую по сравнению с отложенной выборкой оценку качества модели на новых данных.
- Но кросс-валидация вычислительно дорогостоящая, если данных много.

- https://scikit-learn.org/stable/modules/cross_validation.html

- https://code-examples.net/ru/docs/scikit_learn/modules/cross_validation

- https://code-examples.net/ru/docs/scikit_learn/modules/grid_search#grid-search

- Себастьян Рашка

- Цель кросс-валидации в том, чтобы оценить ожидаемый уровень соответствия модели данным, независимым от тех данных, на которых модель тренировалась.

Оценивание качества алгоритмов

Просмотреть урок

https://ru.coursera.org/lecture/supervised-learning/otsienivaniie-kachiestva-alghoritmov-sjbVd

- Задание разбить выборку на k Блоков . Получить оценку посредством кросс-валидации для Organics и найти оптимальные параметры модели

- --→

Соревнование- для всех

- →Kaggle_trees_titanic.ipynb

https://github.com/Rai220/MlSexDetector

Распознавание пола по ФИО