курс «Прикладные задачи анализа данных»

Ансамбли алгоритмов машинного обучения

Александр Дьяконов

26 ноября 2020 года

Ансамбль алгоритмов

Ensemble / Multiple Classifier System

– алгоритм, который состоит из нескольких алгоритмов машинного обучения (базовых алгоритмов – base learners)

простой ансамбль в регрессии:

$$a(x)=\frac{1}{n}\left(b_1(x)+\ldots+b_n(x)\right)$$

простой ансамбль в классификации:

$$a(x)=\mathrm{mode}\left(b_1(x),\ldots,b_n(x)\right)$$

комитет большинства

В чём может быть усложнение?

Ансамбль алгоритмов

$$a(x) = b\left(b_1(x), \ldots, b_n(x)\right)$$

b – **мета-алгоритм (meta-estimator),**

b_i – **базовые алгоритмы (base learners)**

в бустинге – слабые (weak)

Реализация в scikit-learn

`sklearn.ensemble.VotingClassifier`

`estimators`	**Список базовых алгоритмов**
`voting="hard"`	**Голосование по меткам или усреднение вероятностей**
`weights=None`	**Веса**
`n_jobs=None`	**«number of jobs»**
`flatten_transform=True`	**Формат ответа (для soft-ансамбля)**

есть ещё

`ensemble.VotingRegressor`

Ошибка суммы регрессоров

теоретическое обоснование: почему несколько алгоритмов лучше одного...

Если ответы регрессоров на объекте – независимые случайные величины с одинаковым матожиданием и дисперсией

$$\xi = \frac{1}{n}(\xi_1 + \ldots + \xi_n)$$

$$\mathrm{E}\xi = \frac{1}{n}(\mathrm{E}\xi_1 + \ldots + \mathrm{E}\xi_n) = \mathrm{E}\xi_i$$

$$\mathbf{D}\xi = \frac{1}{n^2}(\mathbf{D}\xi_1 + \ldots + \mathbf{D}\xi_n) = \frac{\mathbf{D}\xi_i}{n}$$

ДЗ А если есть корреляция между базовыми алгоритмами?

решите в постановке, что корреляция между любыми двумя алгоритмами равна ρ

Ошибка комитета большинства

Пусть три (независимых) классификатора на два класса с вероятностью ошибки p

Пусть верный ответ – 0

$$\left.\begin{array}{cc}
(0,0,0) & (1-p)(1-p)(1-p) \\
(1,0,0) & p(1-p)(1-p) \\
(0,1,0) & (1-p)p(1-p) \\
(0,0,1) & (1-p)(1-p)p
\end{array}\right\}$$

верный ответ

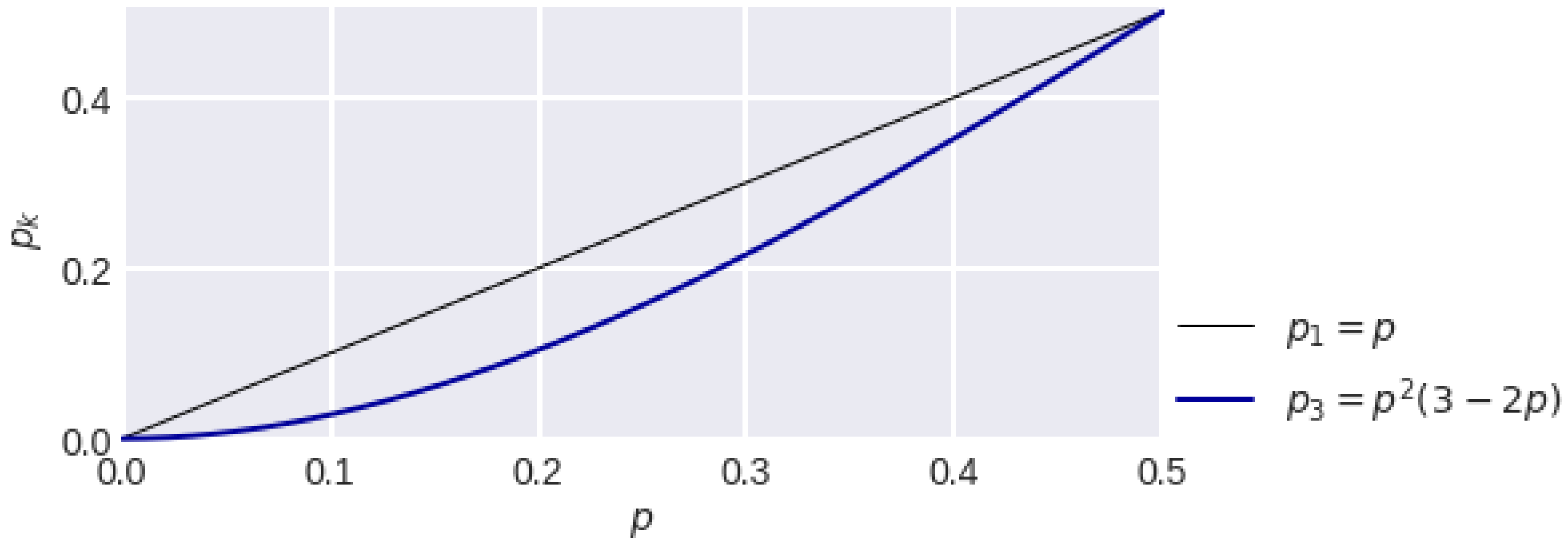
$$\left.\begin{array}{cc}
(1,1,1) & ppp \\
(1,1,0) & pp(1-p) \\
(0,1,1) & (1-p)pp \\
(1,0,1) & p(1-p)p
\end{array}\right\}$$

ошибка

вероятность ошибки

$$p^3 + 3(1-p)p^2 = p^2(3-2p)$$

Ошибка комитета большинства



При малых p ошибка комитета очень мала!

При $p = 0.2$ – почти в два раза меньше

Ошибка комитета большинства

Общий случай:

$$\sum_{t=0}^{\lfloor n/2 \rfloor} C_n^t (1-p)^t p^{n-t} \leq e^{-\frac{1}{2}n(2p-1)^2}$$

неравенство Хёфдинга (Hoeffding)

Ошибка экспоненциально снижается
с увеличением числа базовых алгоритмов…
но это в теории

На практике

Классификаторы / регрессоры точно не являются независимыми

Почему?

На практике

Классификаторы / регрессоры точно не являются независимыми

Почему?

- **Решают одну задачу**
- **Настраиваются на один целевой вектор**
- **Могут быть из одной модели (ну, 2-3 разных)!**

Выход

Пытаться делать алгоритмы разнообразными

Пусть алгоритмы ошибаются, но по-разному:
ошибки одних компенсируются правильными ответами других

Пример: подвыборки и подмножества признаков в RF

Ещё почему алгоритмы в ансамбле должны быть разными

одинаковые	похожие	разные
1111110000 q=0.6	1111110000 q=0.6	1010010111
1111110000 q=0.6	1111101000 q=0.6	1100110011
1111110000 q=0.6	1111100100 q=0.6	1111110000
		1110110011
q_ens = 0.6	q_ens = 0.5	q_ens = 0.7

Нет ли здесь обмана?

Выход

Пытаться делать алгоритмы разнообразными

**Пусть алгоритмы ошибаются, но по-разному:
ошибки одних компенсируются правильными ответами других**

Пример: подвыборки и подмножества признаков в RF

Ещё почему алгоритмы в ансамбле должны быть разными

одинаковые	похожие	разные
1111110000 q=0.6	1110111000 q=0.6	1010010111
1111110000 q=0.6	1101110100 q=0.6	1100110011
1111110000 q=0.6	1011101100 q=0.6	1111110000
		1110110011
q_ens = 0.6	**q_ens = 0.8**	q_ens = 0.7

ДЗ Честный эксперимент, оценивающий зависимость качество(разнообразие)

Повышения разнообразия

«варьируют»…

- **обучающую выборку**
  (бэгинг)

- **признаки**
  (Random Subspaces)

- **целевой вектор**
  (ECOC, f(y))

- **модели**
  (стекинг)

- **алгоритмы в модели**
  (разные гиперпараметры, разные алгоритмы в рамках одной – специальные НС, рандомизация в алгоритме – случайный лес, инициализация в НС)

Варьирование алгоритмов в модели

л/к полиномов разной степени

л/к случайных лесов с разной глубиной

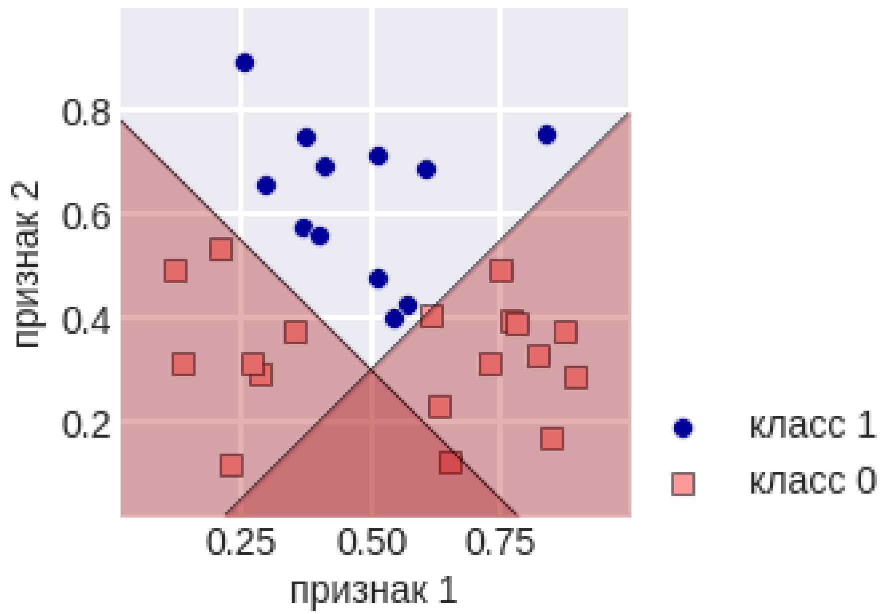
л/к НС с разной инициализацией / после разных эпох

Обоснования применения ансамблей

Статистическое (Statistical) – ошибка может быть меньше

Вычислительное (Computational) – обучение = оптимизация функции, а ансамбль «распараллеливает» процесс

Функциональное (Representational) –можно представить функции, которые нельзя было с помощью базовых алгоритмов



Ансамбли

- **комитеты (голосование) / усреднение**

в том числе, усреднение по Коши, калибровка + усреднение
сюда же **бэгинг (bagging)** – усреднение моделей,
обученных на бутстреп-подвыборках **+ обобщения (RF)**

- **перекодировки ответа**

кодирование целевого вектора
ECOC (error-correcting output coding)

- **стекинг (stacking)**

построение метапризнаков — ответы алгоритмов
на объектах выборки, обучение на них мета-алгоритма

- **бустинг (boosting)**

Построение суммы нескольких алгоритмов.
Каждое следующее слагаемое строится
с учётом ошибок предыдущих

- **«ручные методы»**

Эвристические способы комбинирования
ответов базовых алгоритмов

- **однородные ансамбли**

рекурсия в формуле мета-алгоритм(базовые)
+ общая схема оптимизации (пример: нейросети)

Комитеты (голосование, Voting Ensembles)

голосование по большинству (Majority vote)

$$a(x) = \mathrm{mode}\big(b_1(x), \ldots, b_n(x)\big)$$

комитеты единогласия

в бинарной задаче классификации – $a(x) = \min\big(b_1(x), \ldots, b_n(x)\big)$

обнаружение аномалий:

мата-алгоритм – максимум

«тревога при малейшем подозрении»

$$a(x) = \max\big(b_1(x), \ldots, b_n(x)\big)$$

Усреднение

«среднее арифметическое»

$$a(x) = \frac{1}{n}\left(b_1(x) + \ldots + b_n(x)\right)$$

+ любые другие средние (ex: по Колмогорову)

$$a(x) = \frac{1}{n} f^{-1}\left(f(b_1(x)) + \ldots + f(b_n(x))\right)$$

Ранговое усреднение (Rank Averaging)

$$a(x) = \frac{1}{n}\left(\mathrm{rank}(b_1(x)) + \ldots + \mathrm{rank}(b_n(x))\right)$$

ориентировано на конкретный AUC ROC

Усреднение с весами (weighted averaging)

$$a(x)=\frac{1}{w_1+\ldots+w_n}\left(w_1\cdot b_1(x)+\ldots+w_n\cdot b_n(x)\right)$$

Голосование с весами

$$a(x)=\arg\max_j\left[\sum_{t:b_t(x)=j} w_t\right]$$

Feature-Weighted Linear Stacking

Области компетентности алгоритмов – линейные регресии

$$a(x)=w_1(x)\cdot b_1(x)+\ldots+w_n(x)\cdot b_n(x)=$$

$$=\sum_t\left(\sum_i w_{ti}x_i\right)b_t(x)=\sum_{t,i} w_{ti}x_i b_t(x)$$

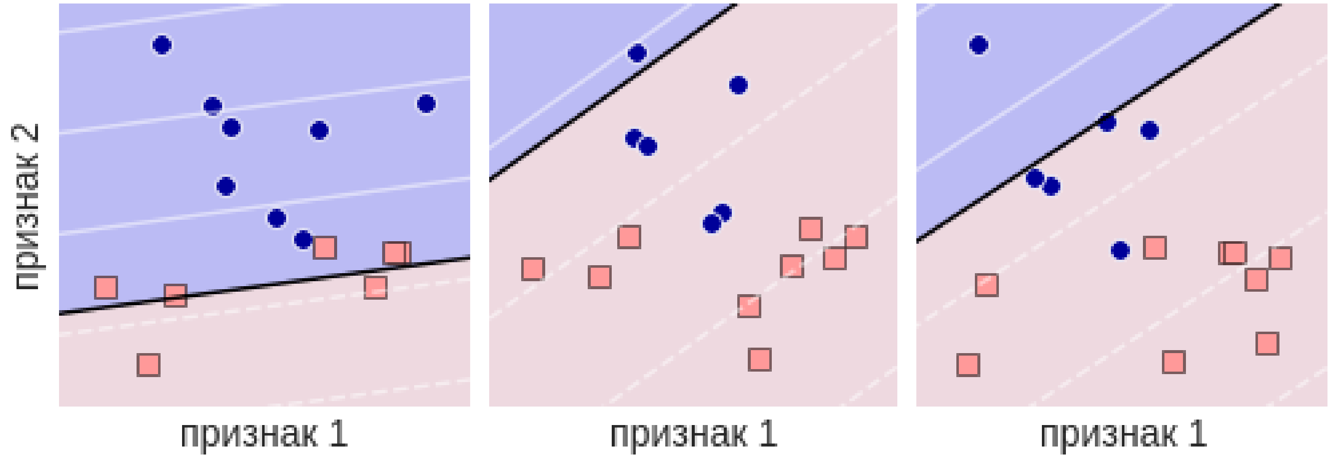
Бэгинг (Bagging)
– bootstrap aggregating

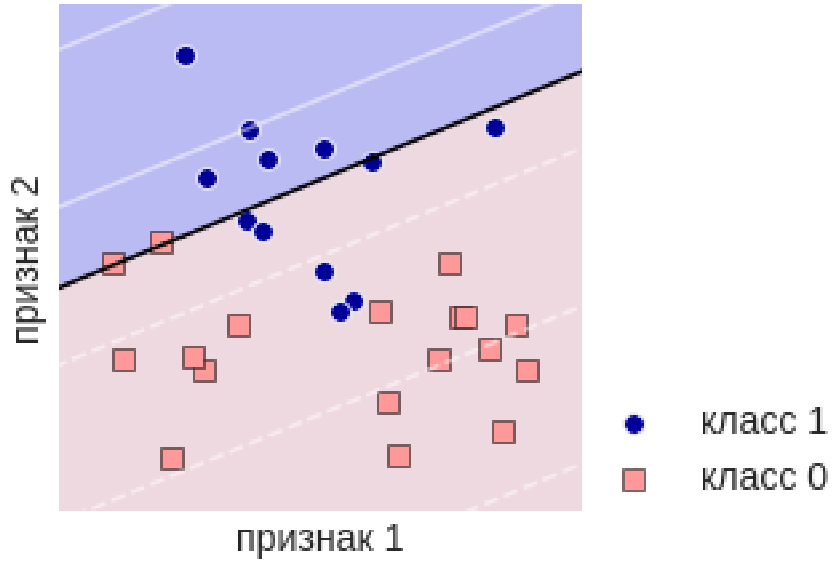
Каждый базовый алгоритм настраивается на случайной подвыборке обучения

Бэгинг	Подвыборка обучающей выборки берётся с помощью бутстрепа
Пэстинг (Pasting)	Случайная обучающая подвыборка.
Случайные подпространства (Random Subspaces)	Случайное подмножество признаков
Случайные патчи (Random Patches)	Одновременно берём случайное подмножество объектов и признаков
cross-validated committees	k обучений на (k-1)-м фолде

Особенности ансамблирования

Не всегда получается «как было задумано»…






```
model = BaggingClassifier(base_estimator=LogisticRegression(),
                          n_estimators=100,
                          max_samples=1.0,
                          max_features=1.0,
                          bootstrap=True,
                          bootstrap_features=False,
                          oob_score=False,
                          warm_start=False,
                          n_jobs=None,
                          random_state=None,
                          verbose=0)
model.fit(X, y)
```

Бэгинг (Bagging)

1. Цикл по t (номер базового алгоритма)

1.1. Взять подвыборку $[X', y']$ обучающей выборки $[X, y]$

1.2. Обучить t-й базовый алгоритм на этой подвыборке:

$$b_t = \mathrm{fit}(X', y')$$

2. Ансамбль

$$a(x) = \frac{1}{n}\left(b_1(x) + \ldots + b_n(x)\right)$$

(для задач регрессии).

Вероятность отбора при бутстрепе (при $m \to \infty$): $1 - \frac{1}{e} \approx 0.632$

Каждый базовый алгоритм обучается ~ на 63% данных

Остальные называются – out-of-bag-наблюдениями (OOB)

~ процедура снижения variance в статистическом обучении

OOB-prediction

На OOB-части выборки можно получить ответы алгоритма

Пусть на i-й итерации это часть: OOB_i

и мы построили алгоритм b_i

OOB-ответы бэгинга (OOB-prediction)

$$a_{\mathrm{OOB}}(x_j) = \frac{1}{|\{i : x_j \in \mathrm{OOB}_i\}|} \sum_{i:\, x_j \in \mathrm{OOB}_i} b_i(x_j)$$

Можно вычислить

OOB-ошибку бэгинга

хорошая оценка ошибки на тесте

похожа на CV-ошибку…

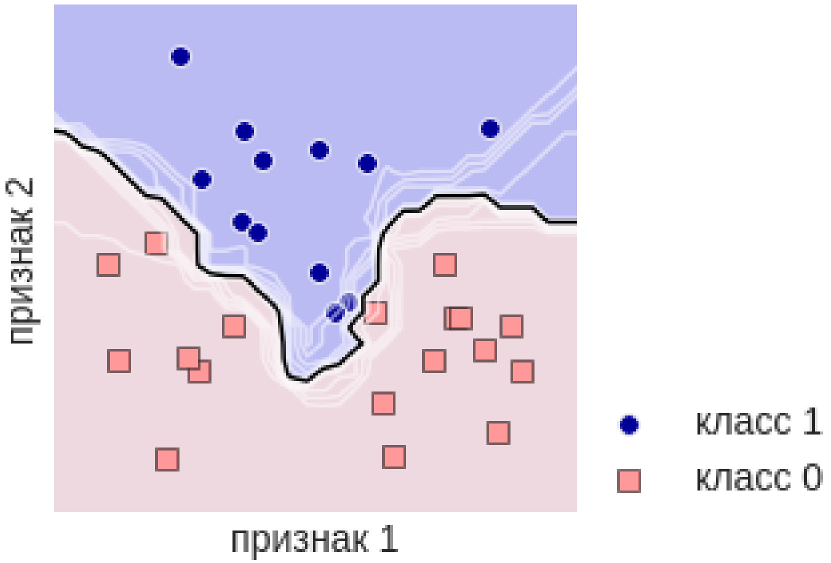
Д/З Сравнить OOB-ошибку и ошибку на OOB-ответе (экспериментально)

Устойчивость (stable learners) модели –

незначительное изменение оптимальных параметров при взятии подвыборки (SVM, kNN k>3)

В бэгинге используются неустойчивые модели (high variance)!
Но несмещённые (small bias)!

Но тут нет хороших теоретических результатов…



Пример – если выбрать правильную базовую модель для бэгинга

Здесь – kNN(1)

Реализация в scikit-learn

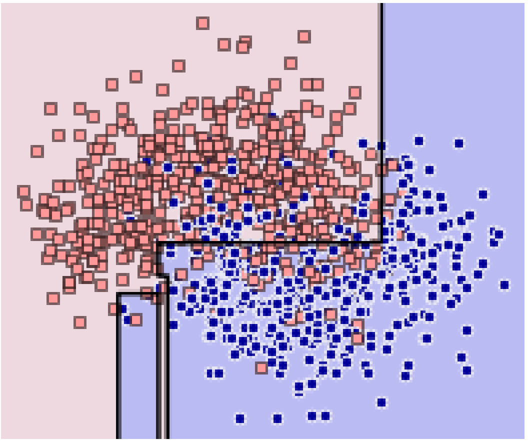
`sklearn.ensemble.BaggingClassifier`

`base_estimator`	**Базовая модель**
`n_estimators=10`	**Число алгоритмов в ансамбле**
`max_samples=1.0`	**Размер подобучения (доля или число)**
`max_features=1.0`	**Число / доля признаков для обучения базового алгоритма**
`bootstrap=True`	**Выбирать ли подобучение с возвращением**
`bootstrap_features=False`	**Аналогичная опция для признаков**
`oob_score=False`	**Вычислять ли OOB-ошибку**
`warm_start=False`	**Использовать ли в качестве начальных приближений старые веса**

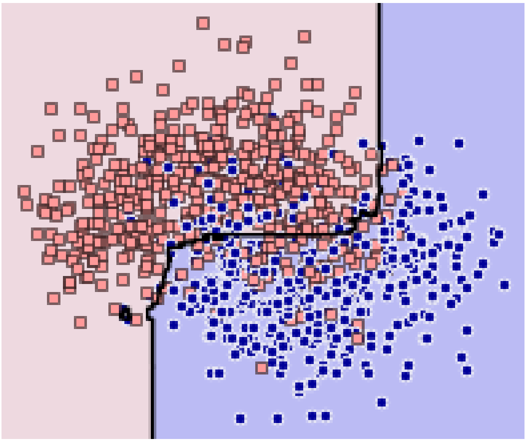
`n_jobs=None, random_state=None, verbose=0`

есть ещё
`ensemble.BaggingRegressor`

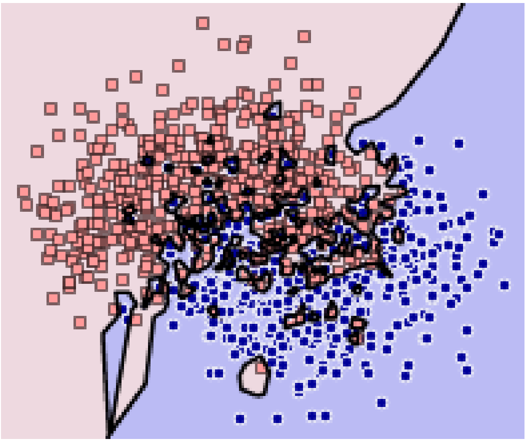
Примеры бэгинга



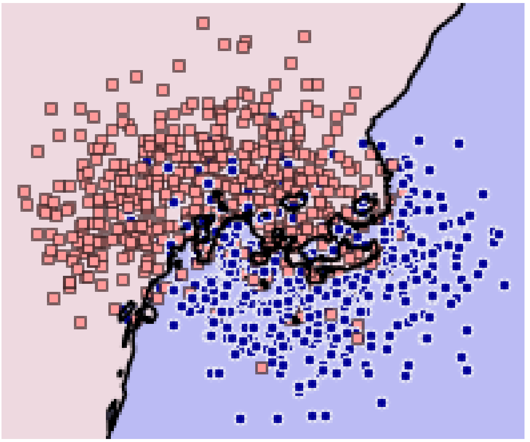
Одно дерево



Бэгинг 100 деревьев



Ближайший сосед



Бэгинг 100 ближайших соседей

Идеи из бэгинга, RS и т.п.

Часто признаки делятся / можно разделить на группы:

- **по источнику данных (БКИ1, БКИ2, …)**
- **по типу признака (вещественный, категориальный, …)**
- **по кодированию (OHE, hash, label, …)**
- **по способу агрегирования (PCA, t-SNE, кластеризация,…)**

Иногда объекты:

- **по источнику данных**
- **по времени**
- **по значениям каких-то признаков (в том числе по кластерам)**

– эти деления можно использовать при формировании подвыборок…

как?

Случайный лес (Random Forest)

дальнейшие улучшения независимости базовых классификаторов

бэгинг + случайности при построении деревьев

отдельная лекция

Стекинг (stacking)

Идея: хорошо усреднять алгоритмы, но почему именно усреднять?

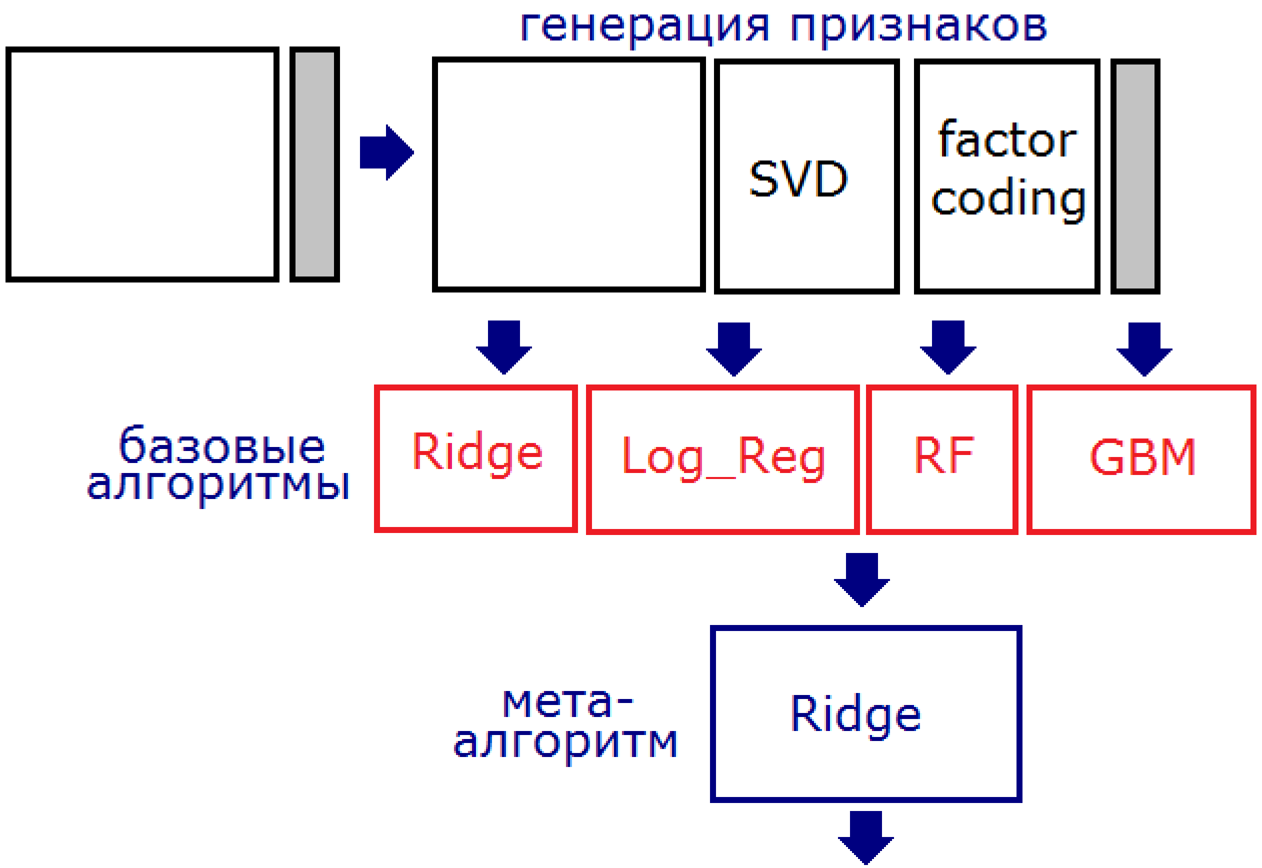
приходит в голову всем…

$$a(x) = b\left(b_1(x), \ldots, b_n(x)\right)$$

b **– мета-алгоритм,**
который нужно отдельно настроить!

Д. Волпертом, автором серии теорем «No free lunch…» в 1992 году

Стекинг

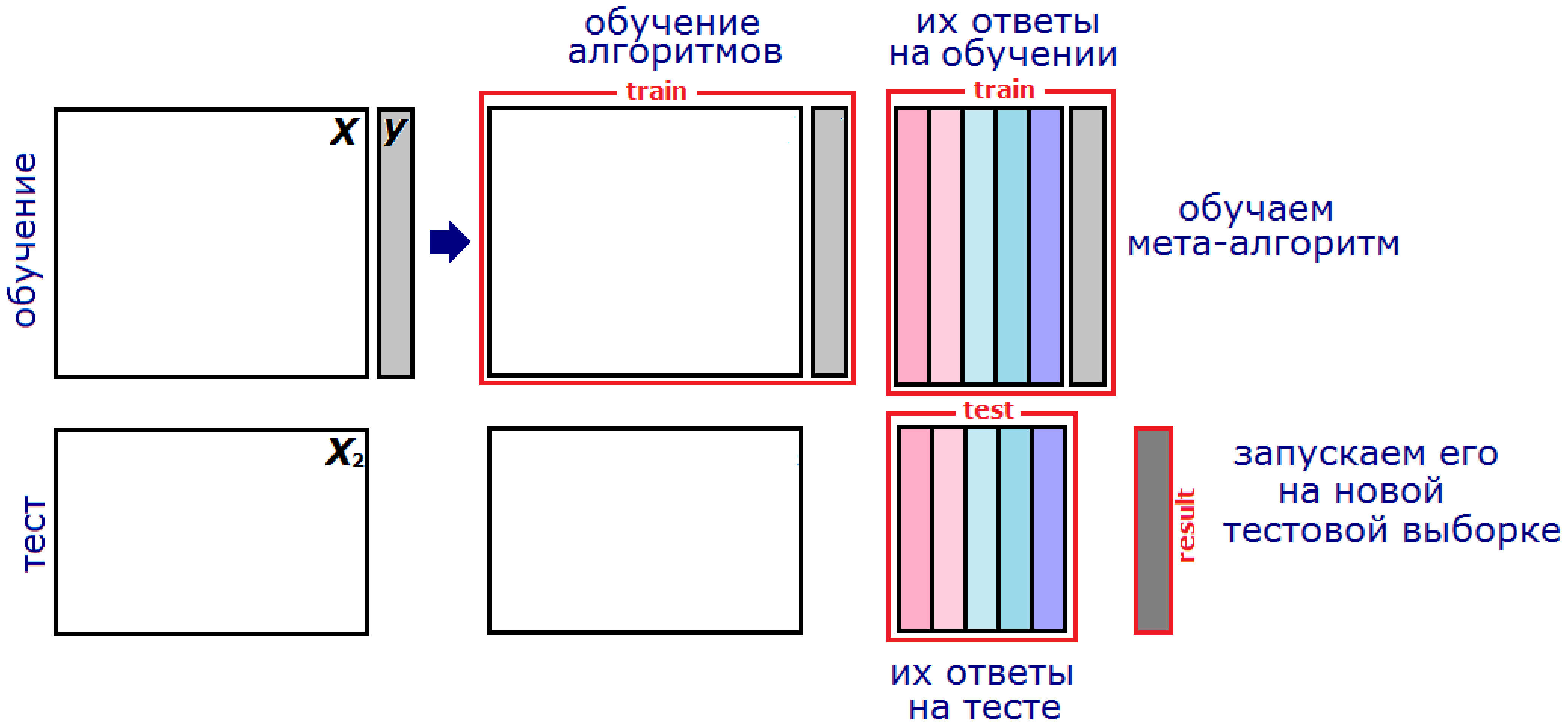


Используем ответы алгоритмов как признаки для нового мета-алгоритма машинного обучения

уже есть реализация в scikit-learn!

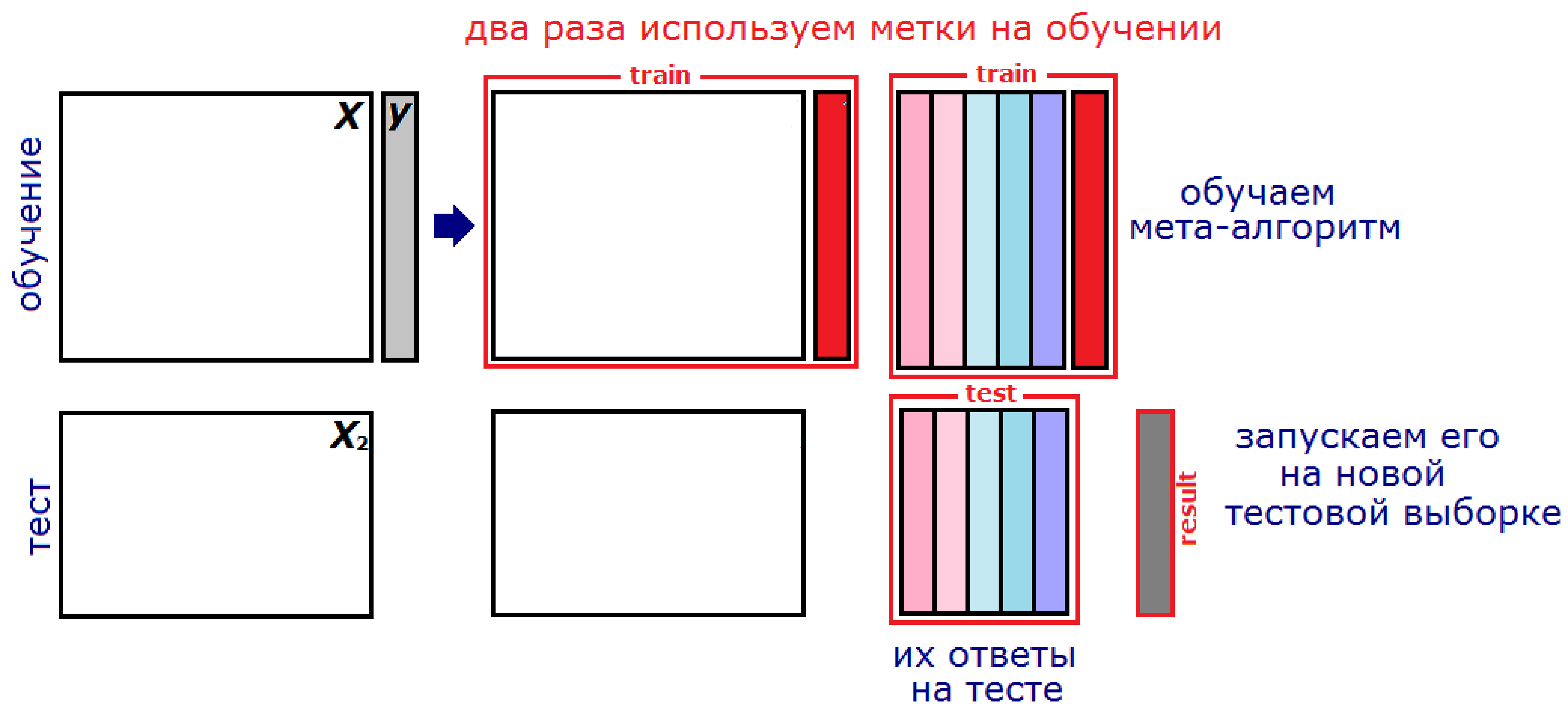
ДЗ Сравнить с другими…

Наивная форма стекинга

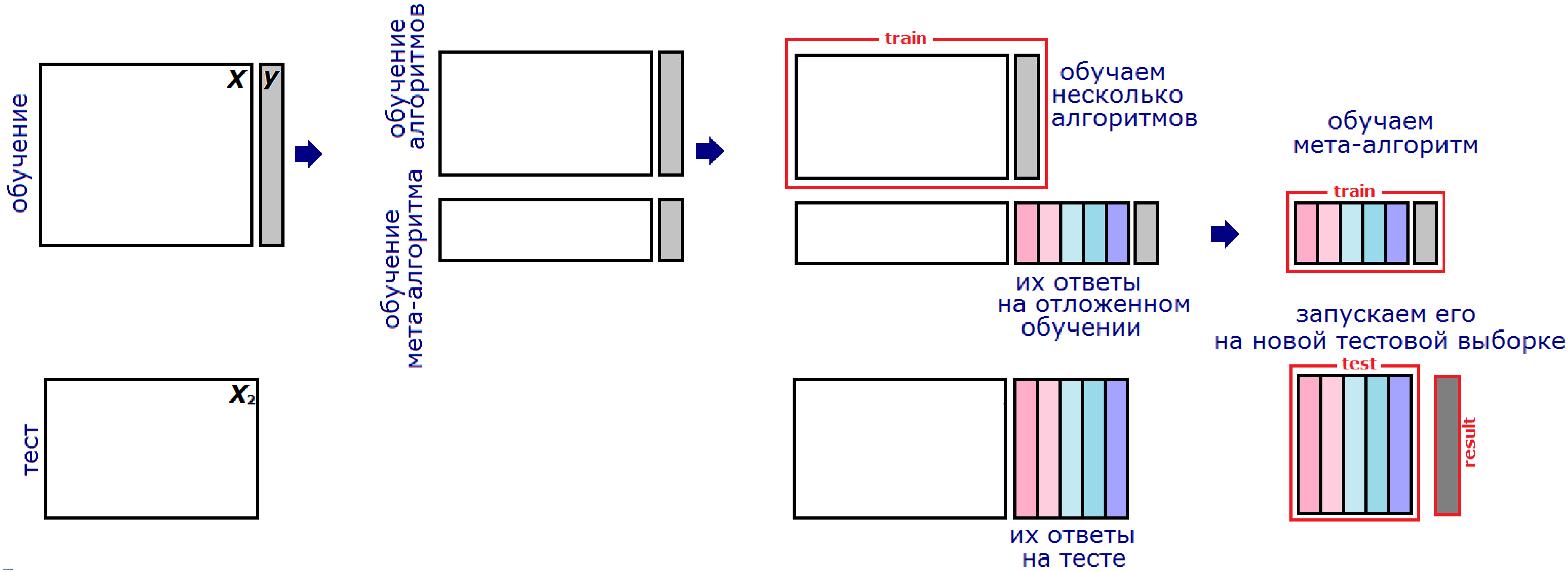


что здесь неправильно?

Наивная форма стекинга



Блендинг / Blending (простейшая форма стекинга)



Блендинг

– термин введён победителями конкурса Netflix

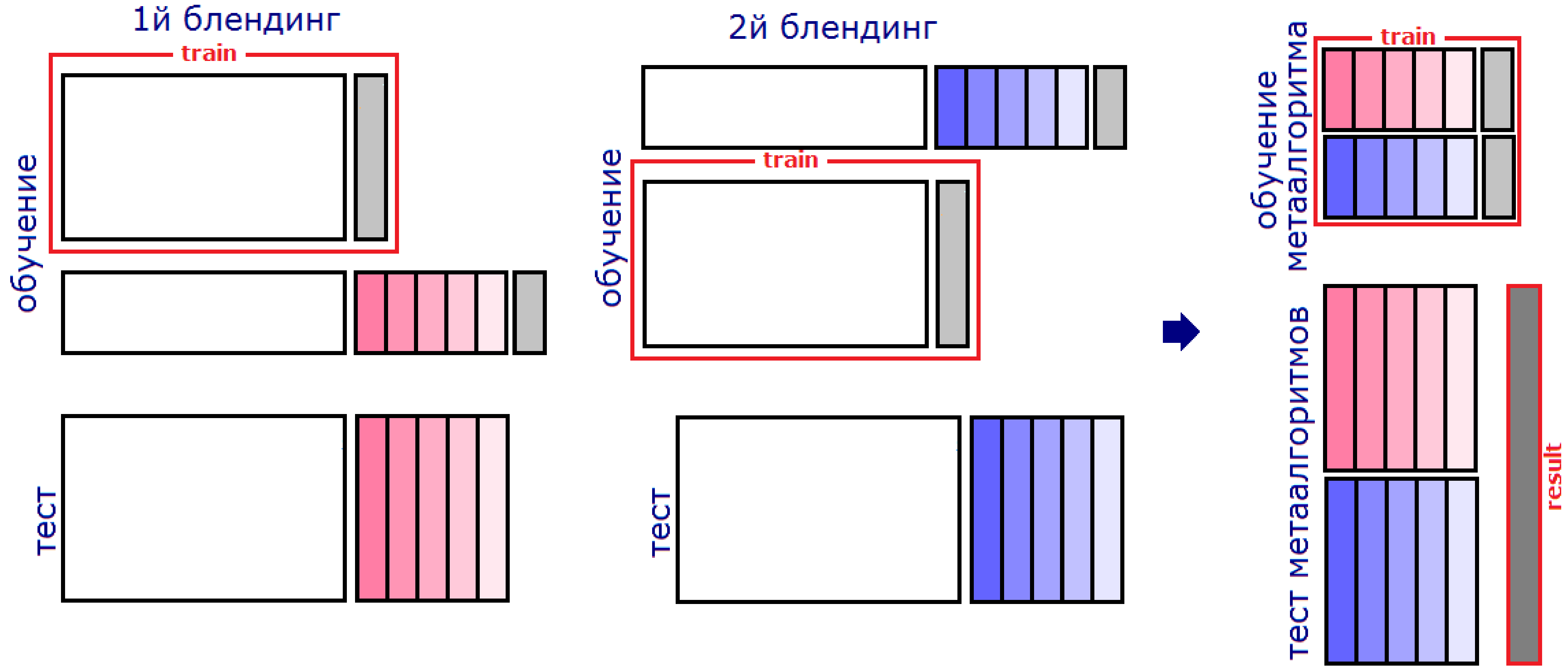
Сейчас блендингом называются простейшие формы стекинга

Недостатки

Используется не вся обучающая выборка

- **можно усреднить несколько блендингов**
- **можно «состыковать»**
  - долго и не всегда лучше по качеству
  - ответы всё равно надо будет усреднить

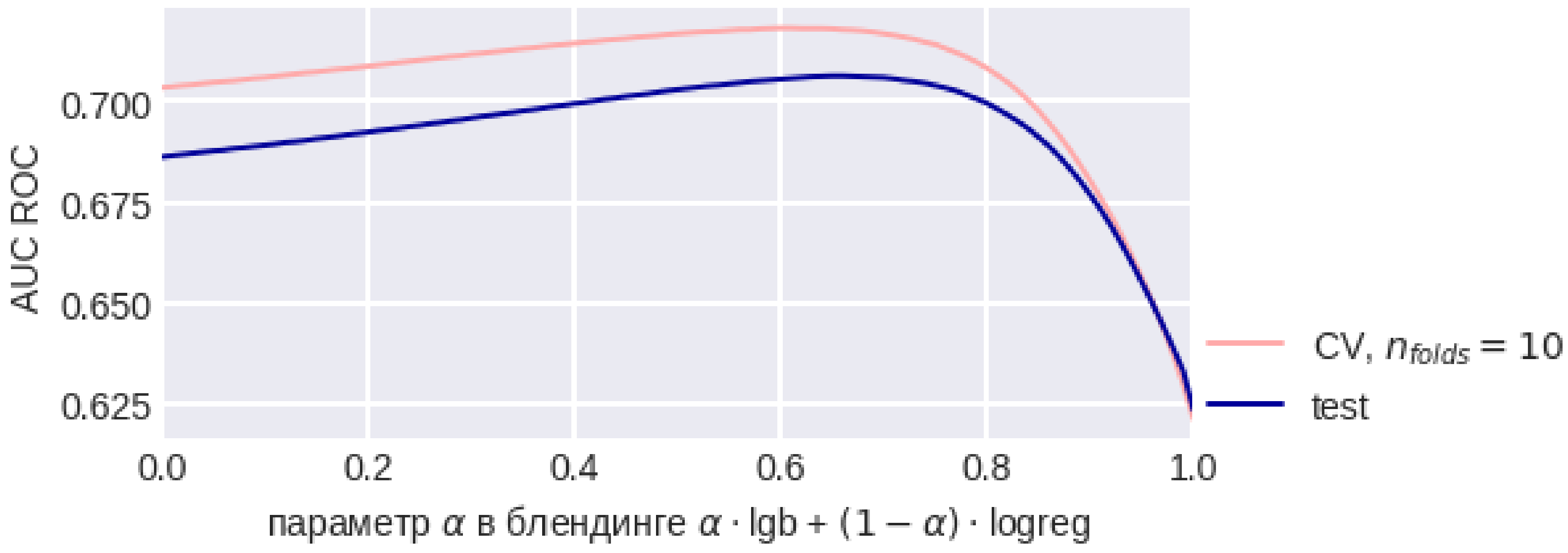
Блендинг: состыковка таблиц



ещё можно усреднить значения мета-признаков на тесте

но это меняет распределения

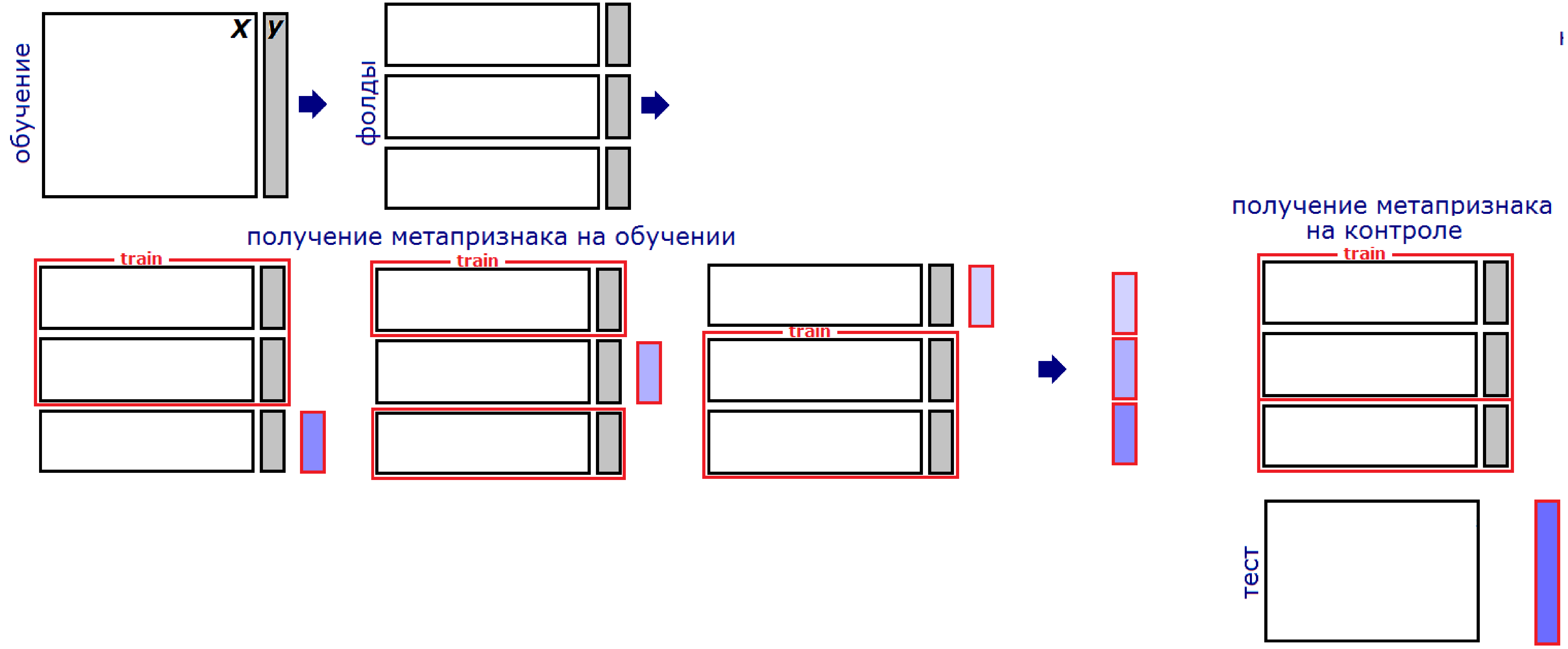
Настройка параметров блендинга



задача Boosters

Стекинг

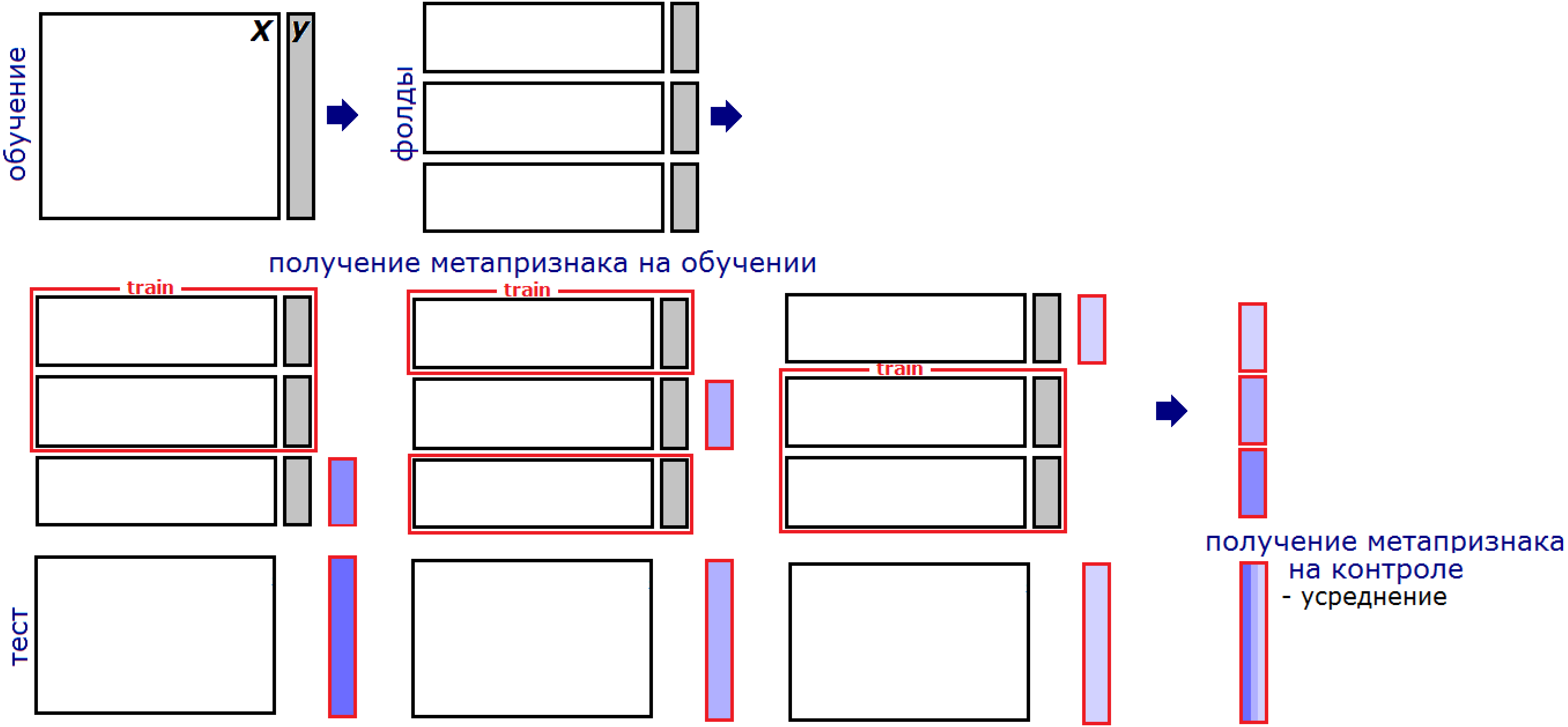
Хотим использовать всю обучающую выборку



м.б. разные разбиения на фолды и усреднить ответы базовых алгоритмов или стекингов

Стекинг

Другой способ получения метапризнаков на контроле



Стекинг

также можно брать разные разбиения и усреднять

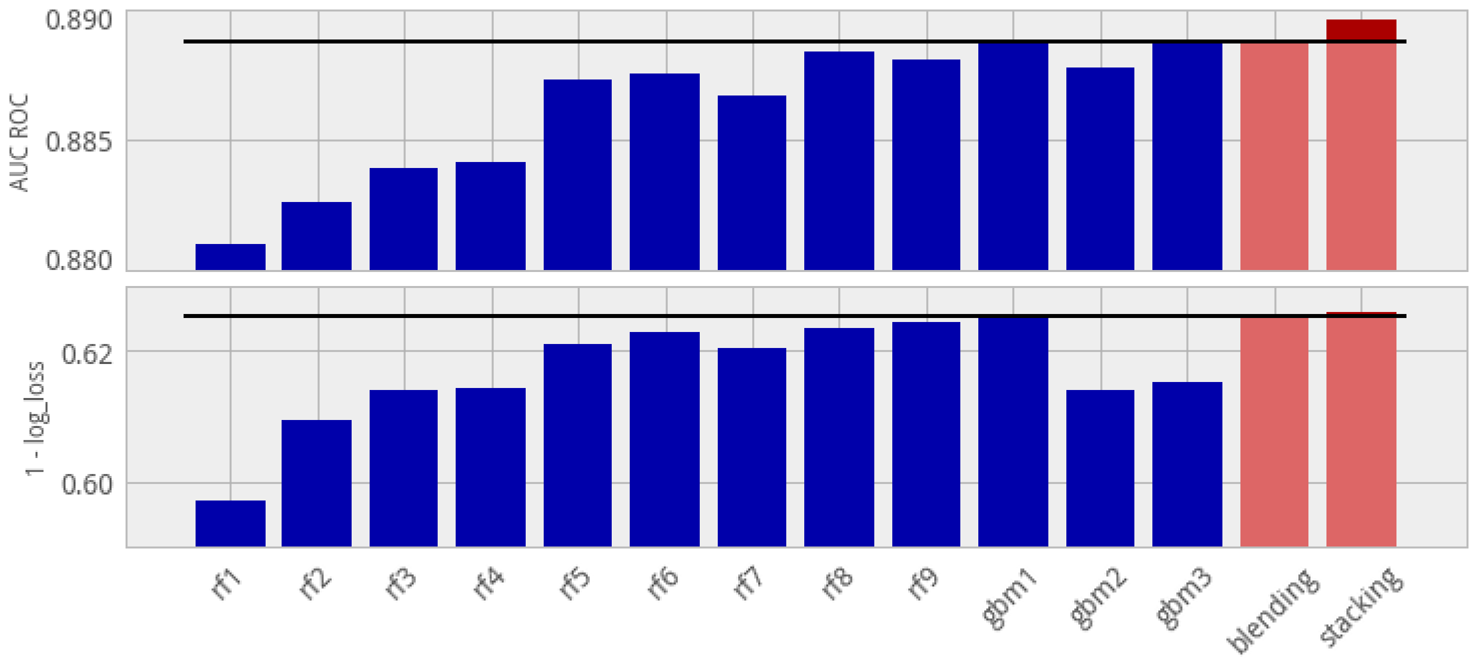
Недостаток

Метапризнаки на обучении и тесте разные!

- регуляризация
- нормальный шум к метапризнакам

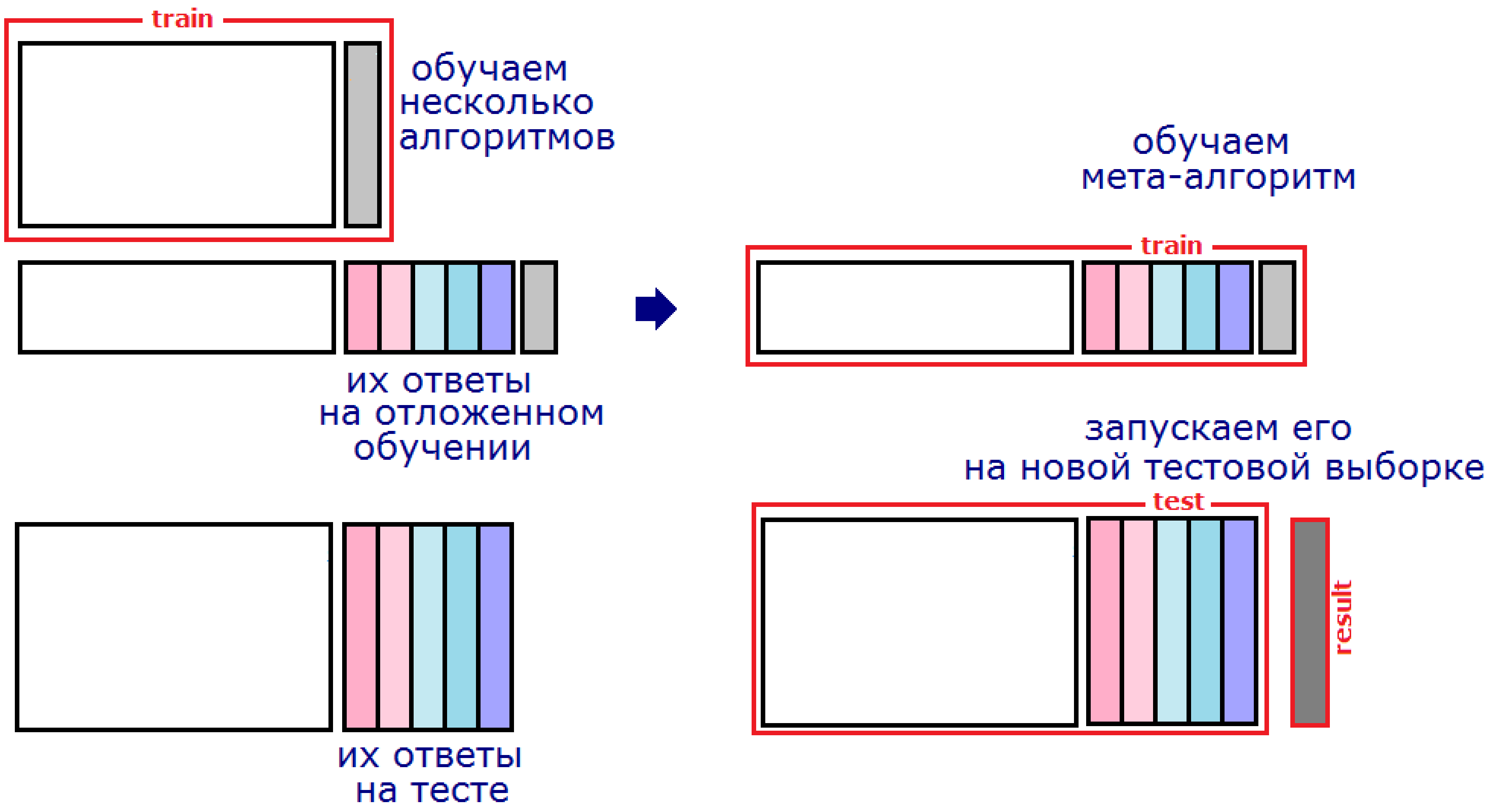
ДЗ Реализовать и сравнить разные виды стекинга

Стекинг



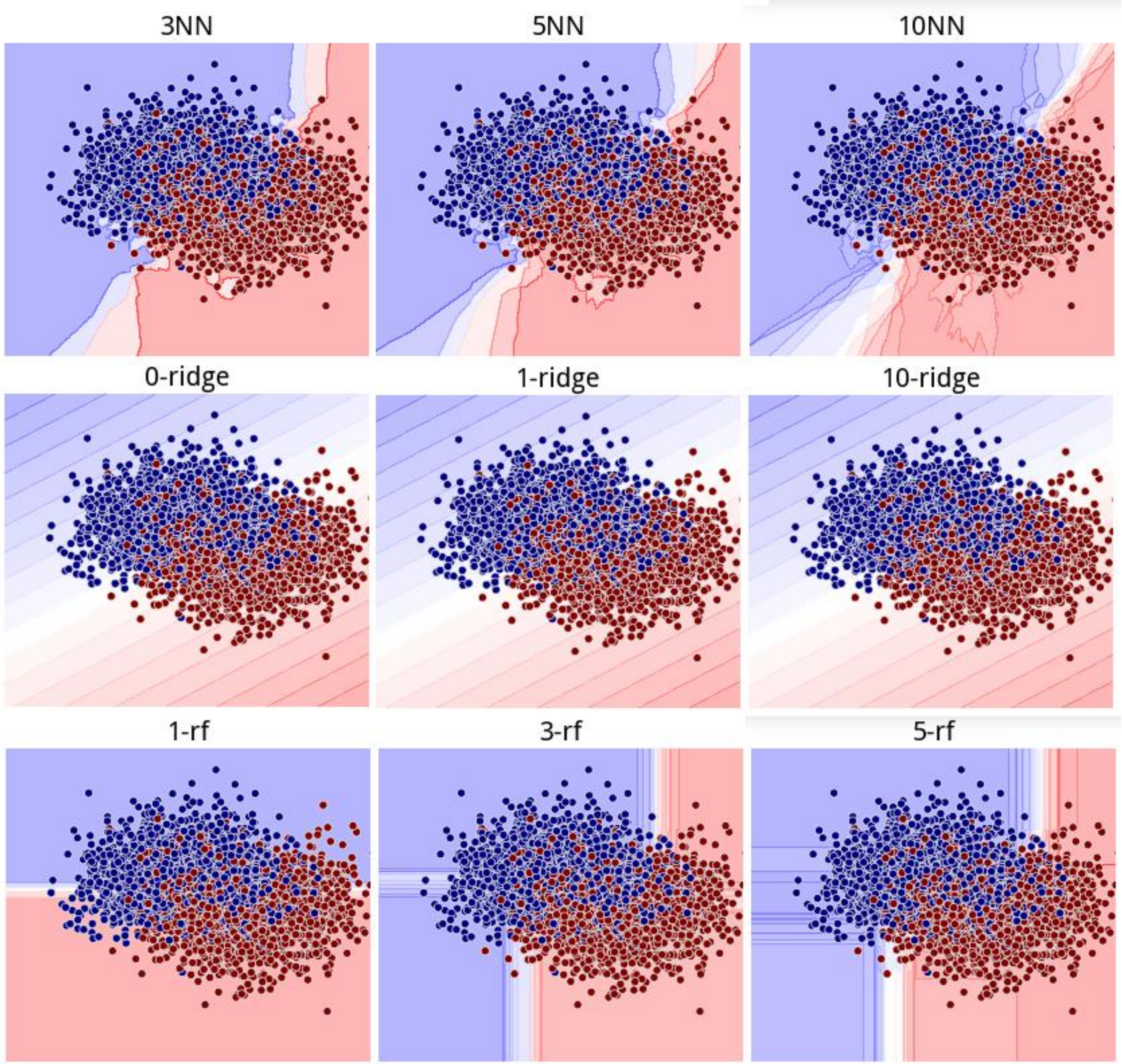
На данных реальной задачи mlbootcamp

Использование признаков с мета-признаками

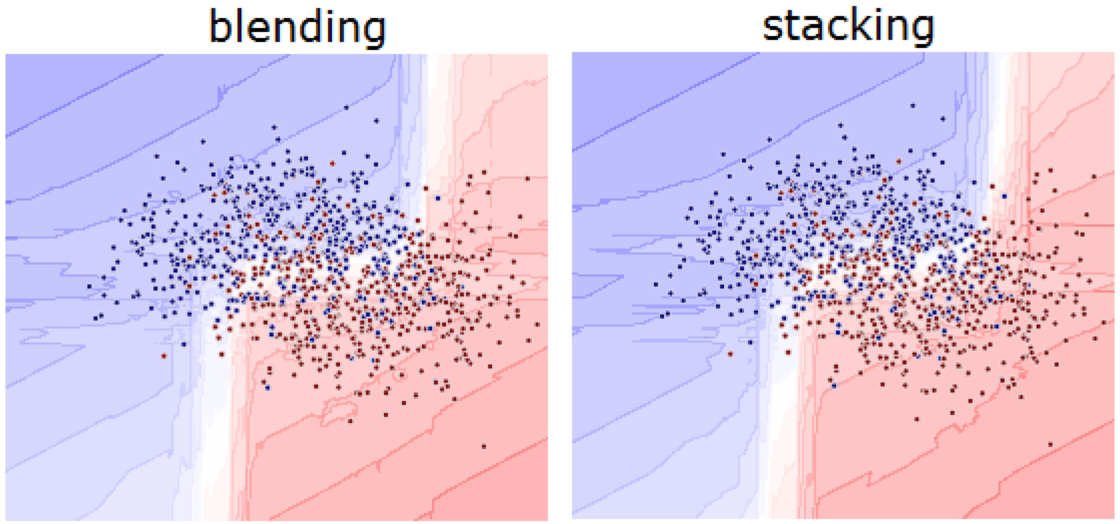


можно добавлять результаты обучения без учителя…

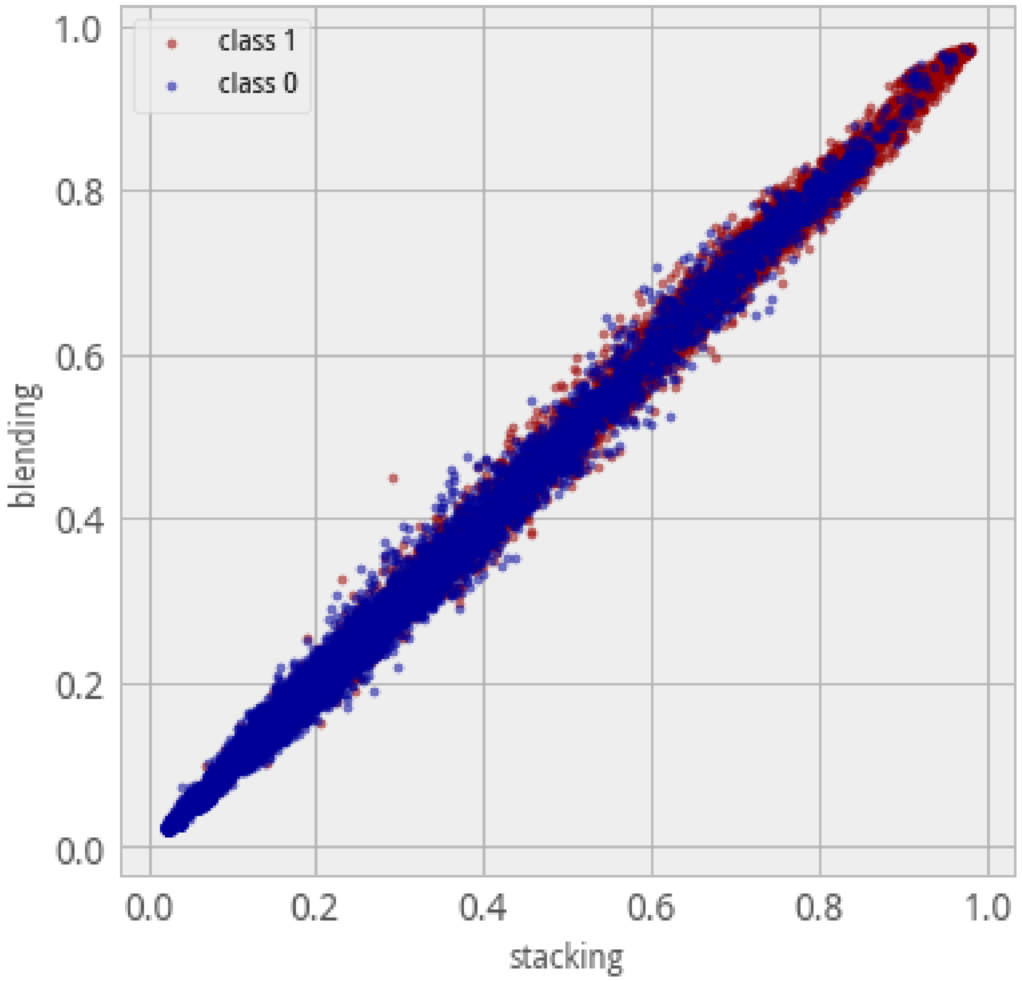
Геометрия стекинга

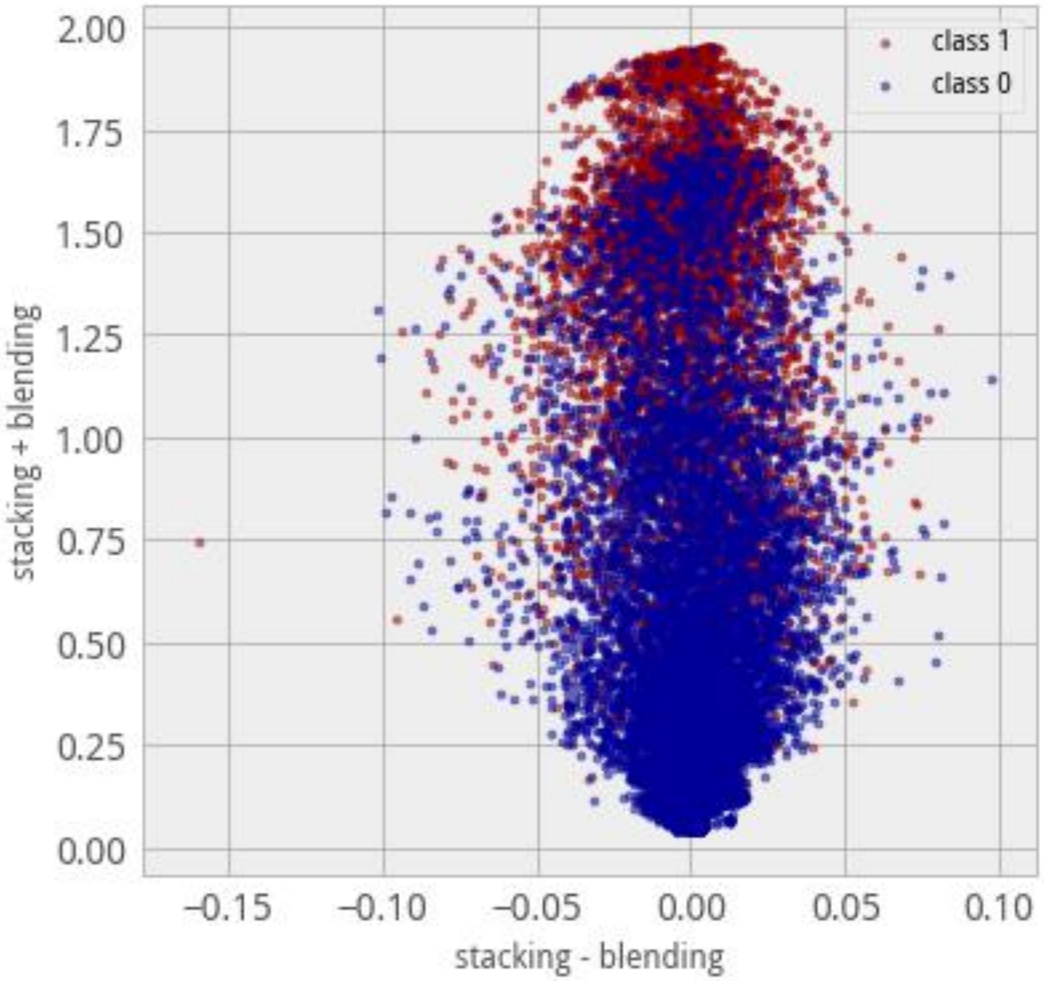


Геометрия стекинга



Стекинг vs Блендинг





Результаты очень похожи…

Стекинг

- **Нужны достаточно большие выборки**

- **Заточен на работу алгоритмов разной природы**

Но для каждого м.б. своё признаковое пространство

- **Хорош на практике (бизнес-задачи)**

Пример: регрессоры + RF =
устойчивость к аномальным значениям признаков

- **Метаалгоритм должен минимизировать целевую функцию**

Не всё так просто... log_regs + log_reg может не справится с Log_loss

- **Многоуровневый стекинг**

Оправдан только в спортивном анализе данных

Стекинг

- **Пространство метапризнаков удобнее признакового, но признаки сильно коррелированны**

Но нет хорошей теории на эту тему

- используют, как правило, регрессоры

базовые алгоритмы не сильно оптимизируют,

- настраиваются не на целевой признак (на его квадрат, на разницу между каким-то признаком и целевым),
- используют модели ориентированные на разные функционалы качества,
- пополняют множество базовых алгоритмов алгоритмами, которые решают другую задачу (например кластеризаторами)

- **Появляются дополнительные параметры**

количество фолдов, уровень шума

Простейший стекинг – кодирование категорий

mean-target-encoding

~

использование байесовского алгоритма
для формирования мета-признака

ECOC = Error-Correcting Output Code

Пусть есть задача с L классами, а у нас классификаторы на 2 класса

1. One-vs-All – каждый класс отделяем от остальных

```
0 – 1000
1 – 0100
2 – 0010
3 – 0001
```

2. One-vs-One – попарно классы друг от друга

```
0 – 111---
1 – 0--11-
2 – -0-0-1
3 – --0-00
```

прочерк – объекты соответствующего класса не участвуют в задаче

ECOC

3. Допустима произвольная кодировка классов:

```
0 – 00
1 – 10
2 – 01
3 – 11
```

это минимальная кодировка, но тут высока цена ошибки бинарного классификатора

4. В том числе, с помощью ECOC

```
0 – 000111
1 – 011100
2 – 101010
3 – 110001
```

Бустинг: Forward stagewise additive modeling (FSAM)

Центральная идея бустинга

Задача регрессии

$$(x_i, y_i)_{i=1}^m$$

функция ошибки

$$L(y, a)$$

уже есть алгоритм $a(x)$ **строим** $b(x)$**:**

$$a(x_i) + b(x_i) = y_i,\ i \in \{1, 2, \ldots, m\}.$$

Надо:

$$\sum_{i=1}^m L(y_i, a(x_i) + b(x_i)) \to \min,$$

Бустинг: Forward stagewise additive modeling (FSAM)

0. Начать с $a_0(x) \equiv 0$

1. Цикл

$$(b,\eta) = \arg\min_{b,\eta} \sum_{i=1}^{m} L(y_i, a_{k-1}(x_i) + \eta b(x_i))$$

$$a_k = a_{k-1} + \eta b$$

Пример: L_2–бустинг

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

$$\sum_{i=1}^{m} (y_i - a_{k-1}(x_i) - b(x_i))^2 \to \min$$

тут м.б. обычная регрессия

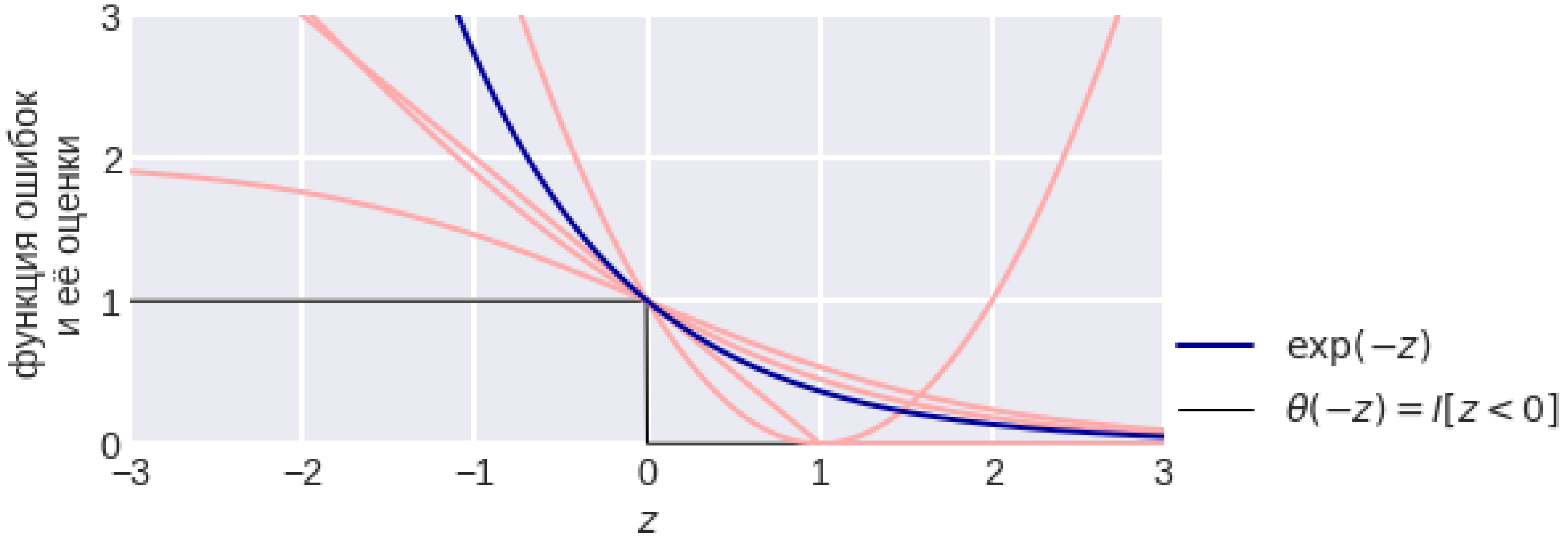
градиентный бустинг – отдельная лекция…

AdaBoost: постановка задачи

– FSAM для бинарной задачи классификации $Y=\{+1,-1\}$
базовые классификаторы генерируют классы $b(x)\in\{+1,-1\}$

Ансамбль

$$a(x)=\operatorname{sgn}\left(\sum_{j=1}^{s}\alpha_j b_j(x)\right)$$



exponential loss

$$L(y,a)=\operatorname{exploss}(y,a)\equiv$$
$$\equiv\exp\left(-y\sum_{j=1}^{s}\alpha_j b_j(x)\right)$$

заметим, что

$$I[y\neq a]\leq\operatorname{exploss}(y,a)$$

AdaBoost: весовая схема

У каждого объекта – вес (распределение!)

$$\sum_{t=1}^{m} w^t = 1$$

«ошибка, порождённая распределением»
(не имеет общего с экспоненциальной ошибкой – просто обозначение) –

$$e(a) = \sum_{\substack{t=1,\\ a(x^t)\neq y(x^t)}}^{m} w^t = \sum_{t=1}^{m} w^t I[a(x^t) \neq y(x^t)]$$

AdaBoost: алгоритм

Цикл

- **перевзвешиваем выборку**
  (чем больше ошибок раньше на объекте, тем больше вес)
- **обучаем новый слабый (week) классификатор на взвешенной выборке**
- **добавляем классификатор в ансамбль**

уменьшается смещение,
т.к. фокусируемся на «плохо классифицируемых» объектах

AdaBoost: алгоритм

0. Зададим начальное вероятностное распределение (веса)

$$W = \left(\frac{1}{m}, \ldots, \frac{1}{m}\right)$$

1. Цикл по j от 1 до s

1.1. Построить классификатор b_j, (вычисляется по распределению W)

который допускает ошибку $e(b_j)$ предполагаем, что $0 < e(b_j) < 0.5$

1.2. Пусть $\alpha_j = \frac{1}{2} \ln\left(\frac{1 - e(b_j)}{e(b_j)}\right)$

«Перестроить» распределение $W = (w^1, \ldots, w^m)$**:**

$$w^t = \frac{w^t \exp(-\alpha_j y(x^t) b_j(x^t))}{\sum_{i=1}^{m} w^i \exp(-\alpha_j y(x^i) b_j(x^i))}$$

вариант: перенастраивать веса только объектов, на которых ошибки…

AdaBoost: вывод формул для экспоненциальной ошибки

экспоненциальная ошибка:

$$L(y,a(x)) = \exp\left(-y\sum_{j=1}^{s}\alpha_j b_j(x)\right)$$

распишем (выделим последний «только что построенный» базовый)

$$\sum_{t=1}^{m} w_t I[y_t \neq a(x_t)] \leq \sum_{t=1}^{m} w_t L(y_t, a(x_t)) = \sum_{t=1}^{m} w_t \exp\left(\underbrace{-y_t\sum_{j=1}^{s-1}\alpha_j b_j(x_t)}_{H} - y_t\alpha_s b_s(x_t)\right) =$$

$$= \exp(H)\sum_{t=1}^{m} w_t \exp(-y_t\alpha_s b_s(x_t))$$

можно множитель не учитывать

AdaBoost: вывод формул для экспоненциальной ошибки

$$\sum_{t=1}^{m} w_t \exp(-y_t \alpha_s b_s(x_t)) =$$

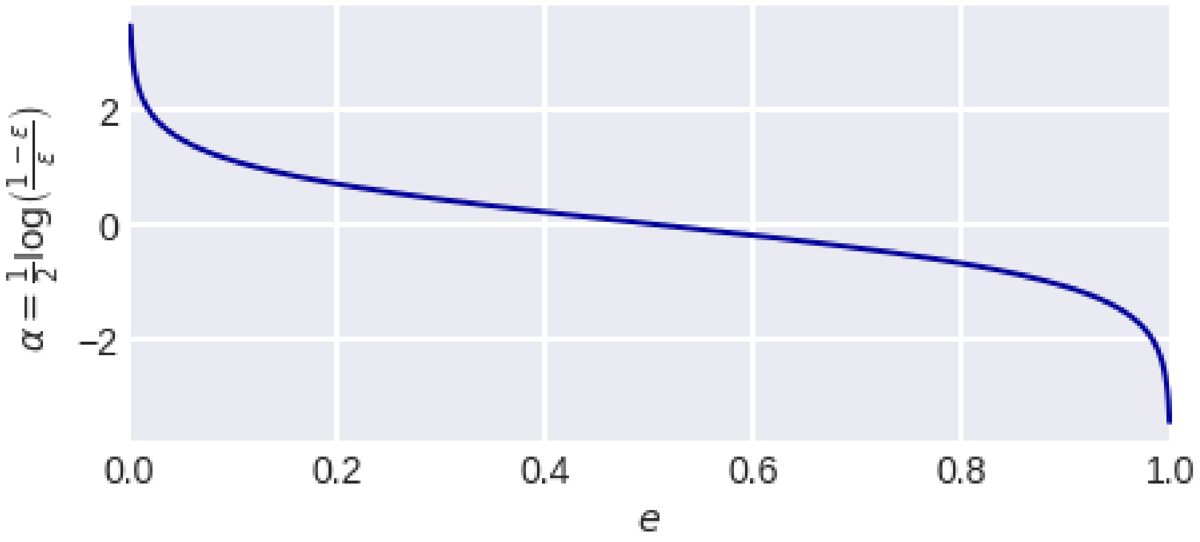
$$= \sum_{t:y_t = a_s(x_t)} w_t \exp(-\alpha_s) + \sum_{t:y_t \neq a_s(x_t)} w_t \exp(\alpha_s) =$$

$$= (1-e)\exp(-\alpha_s) + e\exp(\alpha_s)$$

если хотим найти оптимальный множитель, продифференцируем и приравняем к нулю

$$\alpha_s = \frac{1}{2}\log\frac{1-e}{e}$$

вот откуда та формула!



Зависимость коэффициента от ошибки

AdaBoost: вывод формул для экспоненциальной ошибки

если подставить в формулу…

$$\propto (1-e)\exp\left(-\log\sqrt{\frac{1-e}{e}}\right) + e\exp\left(\log\sqrt{\frac{1-e}{e}}\right) =$$

$$= \frac{(1-e)\sqrt{e}}{\sqrt{1-e}} + \frac{e\sqrt{1-e}}{\sqrt{e}} = 2\sqrt{e(1-e)} \le \exp(-2(0.5-e)^2)$$

т.е. верхняя оценка ошибки экспоненциально уменьшается

AdaBoost: вывод формул для экспоненциальной ошибки

теперь смотрим на формулу ошибки

$$\exp(H)\sum_{t=1}^{m} w_t \exp\left(-y_t \alpha_s b_s(x_t)\right)$$

и пересчёта весов… это просто учёт в весах новых ответов

всегда происходит умножение…

график добавки в зав-ти от ошибка / нет

AdaBoost: вывод формул для экспоненциальной ошибки

теперь смотрим на формулу ошибки

$$\exp(H)\sum_{t=1}^{m} w_t \exp\left(-y_t\alpha_s b_s(x_t)\right)$$

и пересчёта весов... это просто учёт в весах новых ответов

всегда происходит умножение...

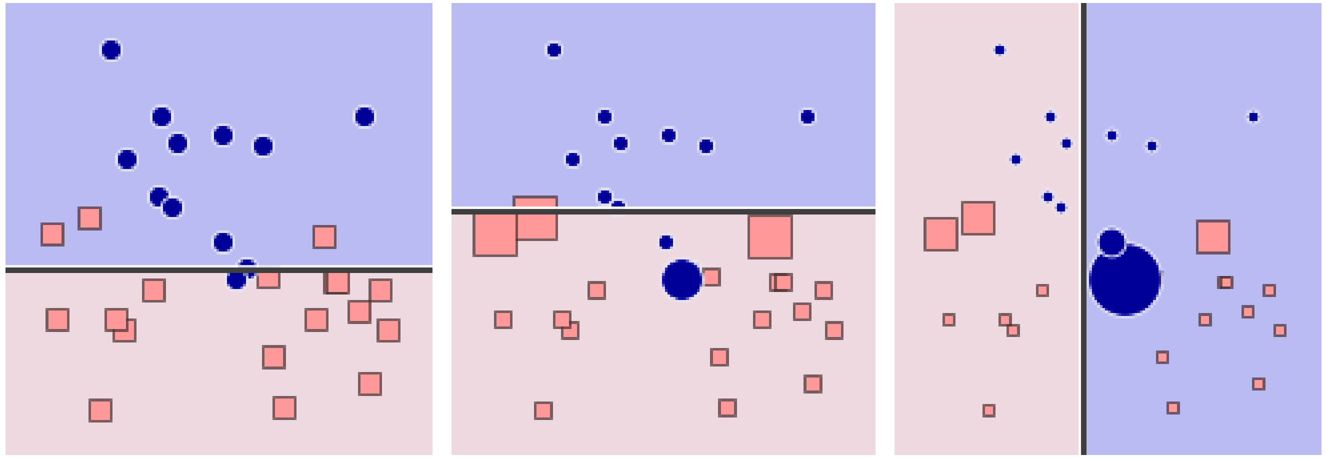
если рекурсивно пересчитать...

$$\sum_{t=1}^{m} w_t\,|_{s=0} \exp\left(-y_t\alpha_1 b_1(x_t)\right)\ldots\exp\left(-y_t\alpha_{s-1} b_{s-1}(x_t)\right)\exp\left(-y_t\alpha_s b_s(x_t)\right)=$$

$$\sum_{t=1}^{m} w_t\,|_{s=0} \exp\left(-y_t(\alpha_1 b_1(x_t)+\ldots+\alpha_{s-1} b_{s-1}(x_t)+\alpha_s b_s(x_t))\right)=$$

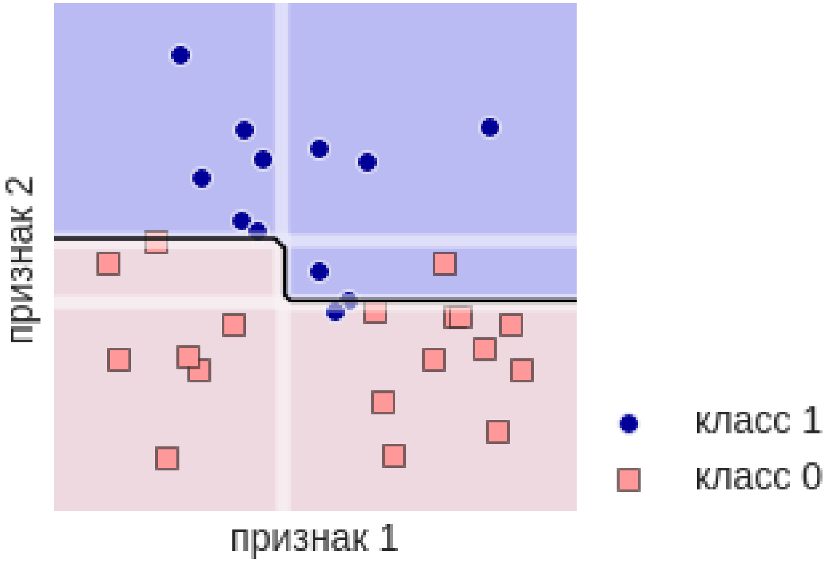
$$=\sum_{t=1}^{m} \operatorname{exploss}(y_t, a_t)$$

AdaBoost: пример



объектов пропорционален весу…

– полученная комбинация классификаторов



AdaBoost: теория

Если на каждом шаге мы можем построить слабый (weak) классификатор:
$e(b_j) \leq 0.5 - \varepsilon$**,** $\varepsilon > 0$**, то ошибка ансамбля**

$$a(x) = \operatorname{sgn}\left(\sum_{j=1}^{s} \alpha_j b_j(x)\right)$$

на обучении оценивается как

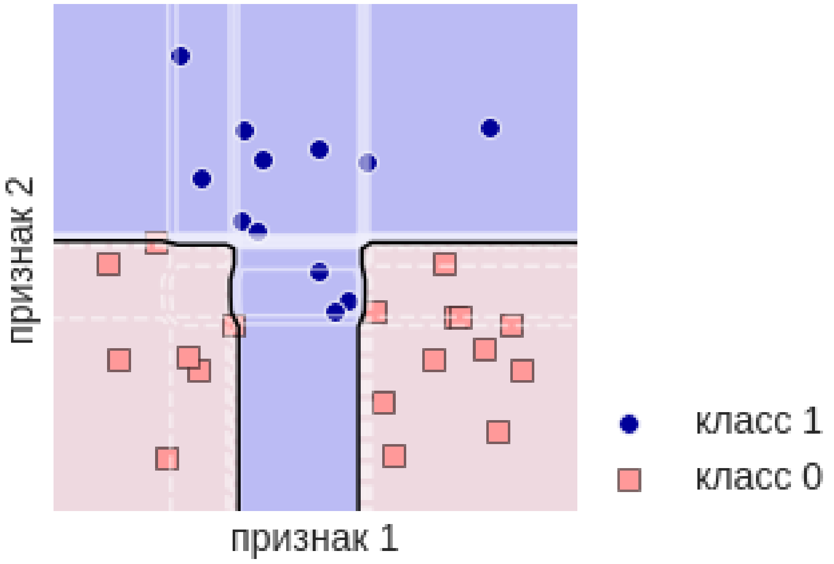
$$e(a) = \frac{1}{m}\sum_{t=1}^{m} I[a(x^t) \neq y(x^t)] \leq \exp(-2\varepsilon^2 s)$$

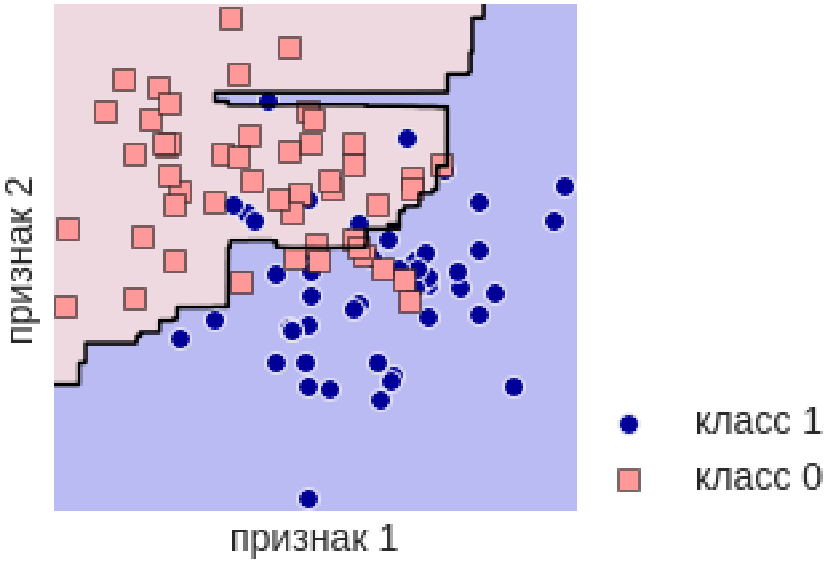
т.е. всего лишь из предположения, что слабый классификатор на ε **лучше случайного**

В чём небольшая некорректность в этой фразе?

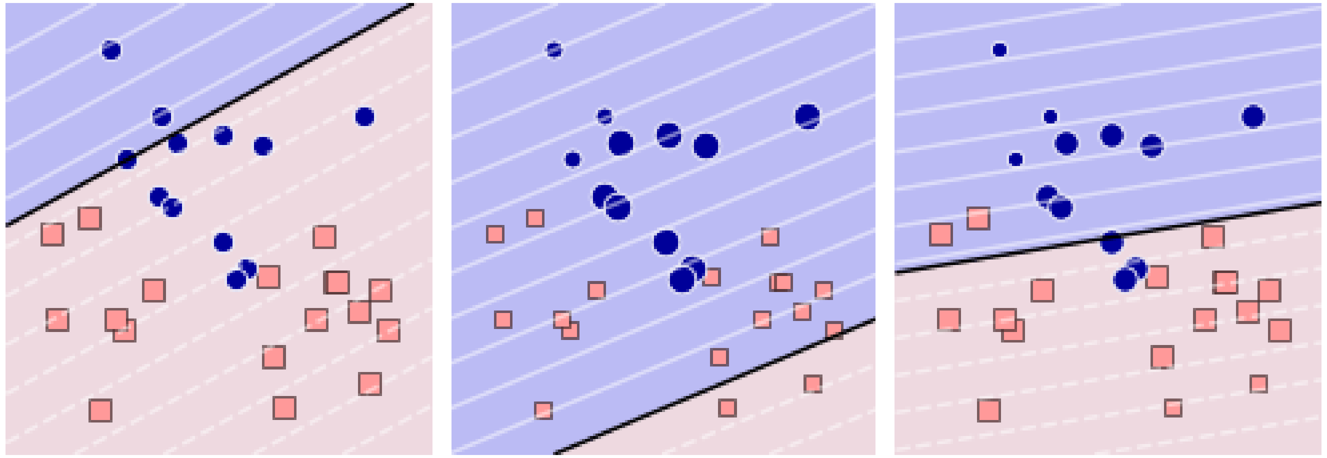
AdaBoost: минутка кода

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier
model = AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1),
                           n_estimators=20, learning_rate=1.0,
                           algorithm='SAMME.R', random_state=1)
model.fit(X, y)
```





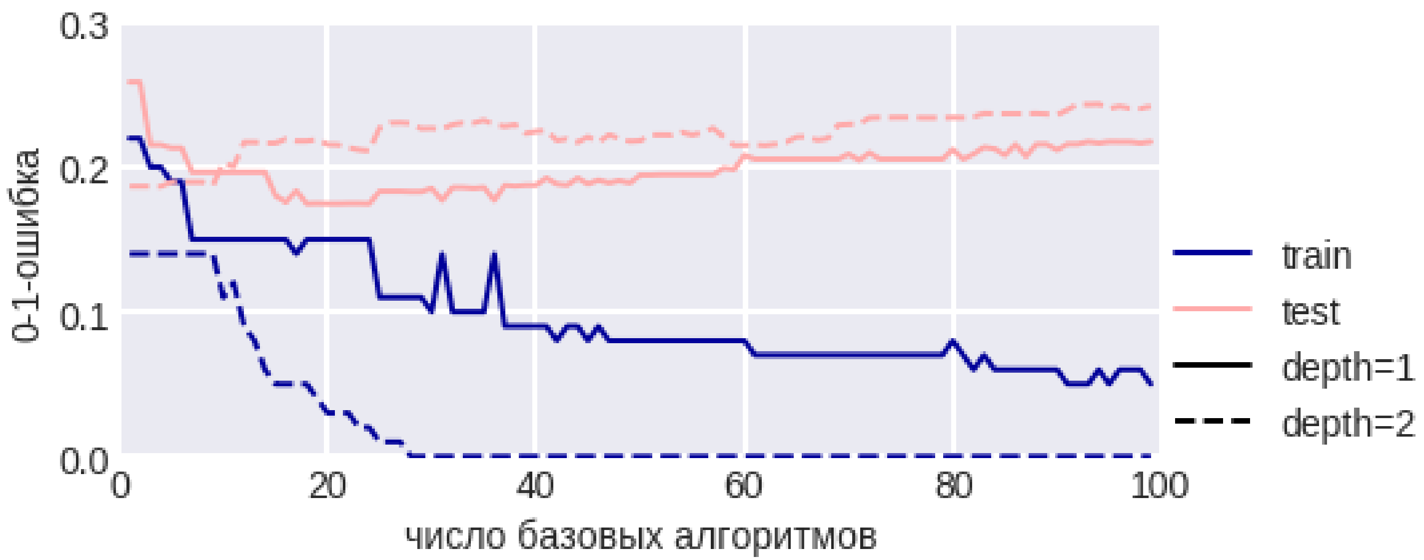
AdaBoost: недостатки

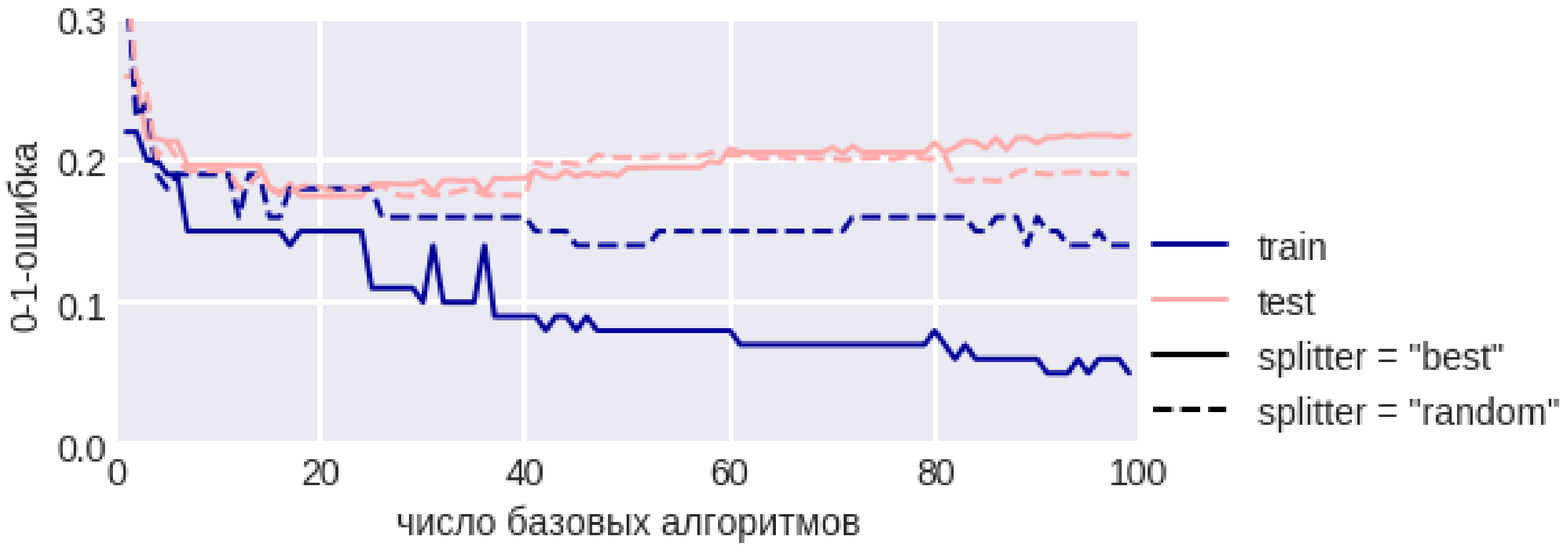


Бустинг плох, когда есть выбросы.

В приведённом примере бустинг плох над логистической регрессией (над стабильными алгоритмами)!

AdaBoost: переобучение / уменьшение ошибки





иногда ошибка на тесте уменьшается даже после обнуления на обучении

Теория без доказательства

Теорема. Чем больше зазор (margin), тем лучше обобщение.

идея: если большой, то алгоритм можно аппроксимировать простым

Теорема. При бустинге зазор увеличивается.

идея: аналогично, как смотрели на ошибку

Ручные методы ансамблирования

Метод Ефимова

$$f(a_1, a_2)$$

	$a_1 \le 0.1$	$0.1 < a_1 < 0.9$	$a_1 \ge 0.9$
$a_2 \le 0.1$	$\min(a_1, a_2)$	$\min(a_1, a_2)$	$0.55a_1 + 0.45a_2$
$0.1 < a_2 < 0.9$	$0.1a_1 + 0.9a_2$	$\mathrm{mean}(a_1, a_2)$	$0.9a_1 + 0.1a_2$
$a_2 \ge 0.9$	$0.75a_1 + 0.25a_2$	$\max(a_1, a_2)$	$\max(a_1, a_2)$

Amazon Employee Access Challenge

Итог: ключевые идеи ансамблирования

1. Объединение ответов разных алгоритмов

усреднение / голосование / стекинг …

2. Повышения разнообразия / независимости базовых алгоритмов

«варьирование» признаков, объектов, моделей, в модели и т.п.

Использование подвыборок / весов

3. Ансамблирование: параллельное и последовательное

Parallel ensembles – все алгоритмы строятся независимо

Идея: усреднить (high complexity, low bias)-модели, для снижения variance

Sequential ensembles – алгоритмы строятся последовательно

Общая классификация главных мета-алгоритмов

	разброс (model's variance)	смещение (model's bias)	функциональная выразимость	основа техники
Bagging «среднее»	уменьшает			bootstrap
Boosting «взвешенное среднее»		уменьшает	(увеличивает)	градиентный спуск (сейчас)
Stacking Мета-алгоритм	(уменьшает)	(уменьшает)	увеличивает	суперпозиция алгоритмов

Некоторые библиотеки

ML-Ensemble http://ml-ensemble.com/ General ensemble learning

mlxtend http://rasbt.github.io/mlxtend/ Regression and Classification ensembles

H20 http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/stacked-ensembles.html
Distributed stacked ensemble learning. Limited to estimators in the H20 library

Литература

Статья про ансамбли

Dietterich, T. G. (2000). «Ensemble Methods in Machine Learning» // First International Workshop on Multiple Classifier Systems, Lecture Notes in Computer Science (pp. 1-15). New York: Springer Verlag.

Предложен Feature-Weighted Linear Stacking

Sill, J.; Takacs, G.; Mackey, L.; Lin, D. (2009). «Feature-Weighted Linear Stacking». arXiv:0911.0460.

Бэгинг и аналогичные идеи:

L. Breiman, Pasting small votes for classification in large databases and on-line, Machine Learning, 36(1), 85-103, 1999.

L. Breiman, Bagging predictors, Machine Learning, 24(2), 123-140, 1996.

T. Ho, The random subspace method for constructing decision forests, Pattern Analysis and Machine Intelligence, 20(8), 832-844, 1998.

G. Louppe and P. Geurts, Ensembles on Random Patches, Machine Learning and Knowledge Discovery in Databases, 346-361, 2012.

Ансамбли в машинном обучении

https://dyakonov.org/2019/04/19/ансамбли-в-машинном-обучении/

AdaBoost – немного другой вывод:

http://www.cs.toronto.edu/~rgrosse/courses/csc411_f18/slides/lec09-slides.pdf