курс «Машинное обучение»

Случайные леса и AdaBoost

Александр Дьяконов

23 ноября 2021 года

План

Случайный лес (RF)

Параметры метода

ExtraTrees

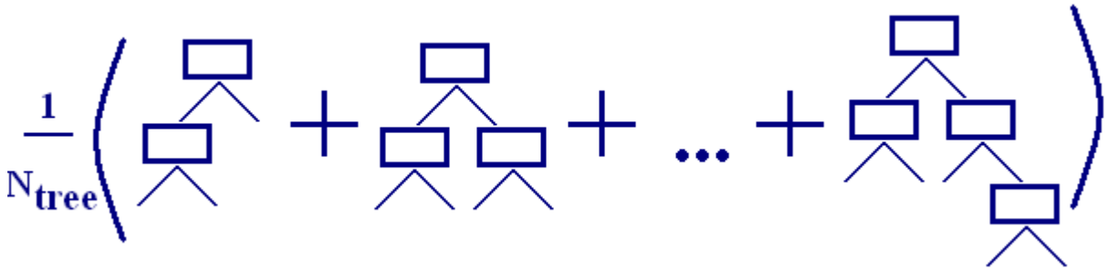
AdaBoost

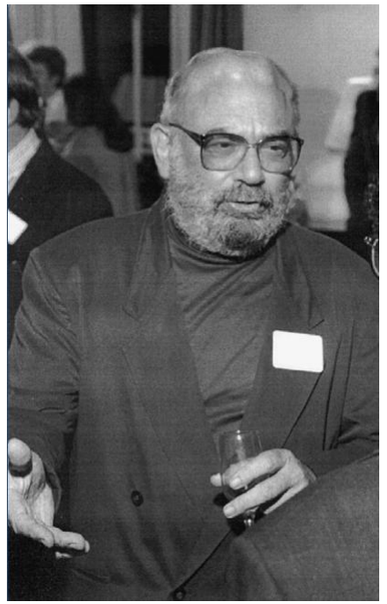
Случайный лес (Random Forest)

– специальный метод ансамблирования

= бэггинг + специальное построение деревьев
(подмножество признаков при расщеплении)

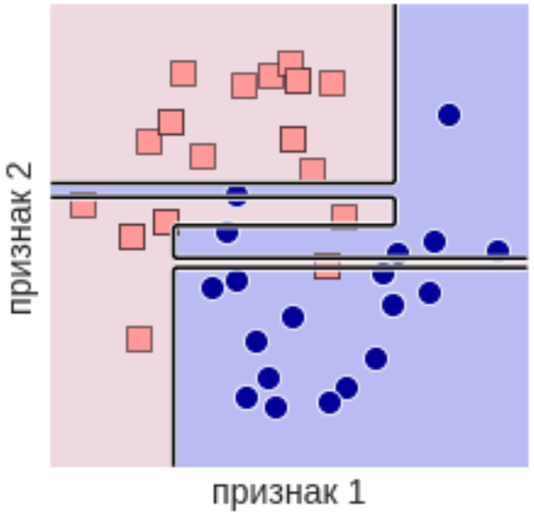
Качество одного дерева очень низкое!



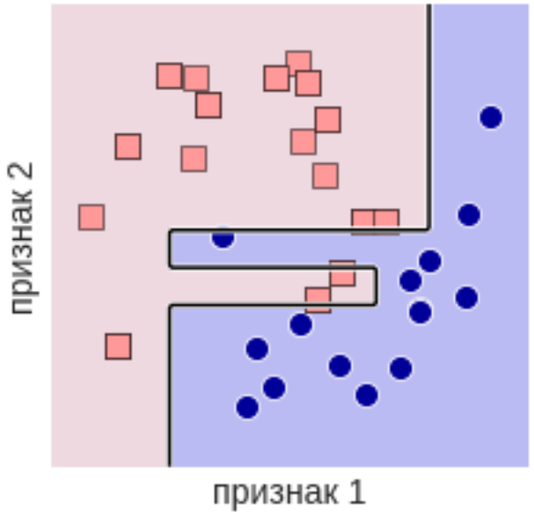


Лео Брейман, 1928 – 2005

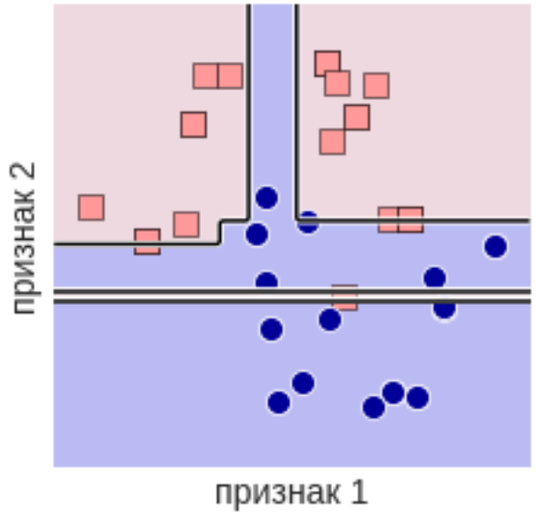
Случайный лес (Random Forest)



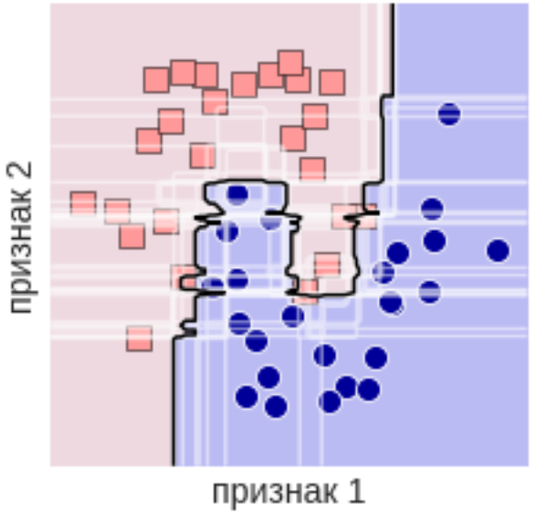
дерево № 1



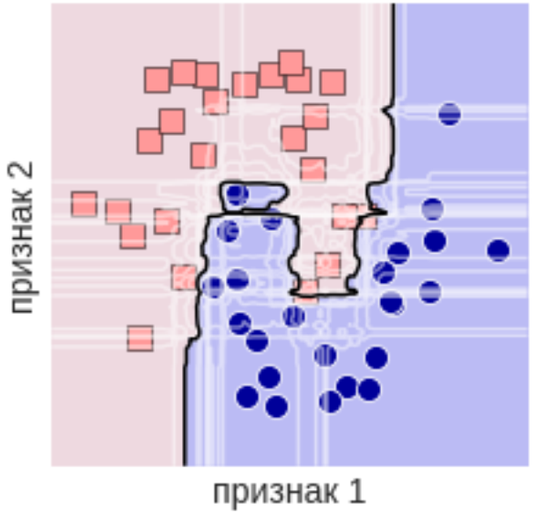
дерево № 2



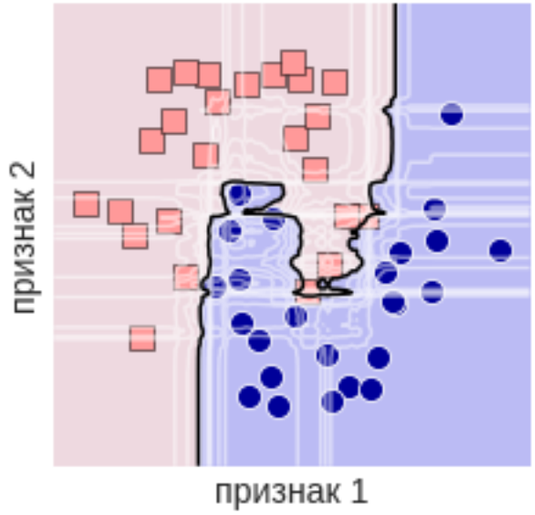
дерево № 3



RF, число деревьев=10



RF, число деревьев=100



RF, число деревьев=1000

Построение случайного леса

1. Выбирается подвыборка `samplesize` (м.б. с повторением) – на ней строится дерево

2. Строим дерево
   2.1. Для построения каждого расщепления просматриваем `mtry` / `max_features` случайных признаков
   2.2. Как правило, дерево строится до исчерпания выборки (без прунинга)

Ответ леса в задачах классификации:

по большинству, вероятность = процент деревьев (R)

сравниваем вероятность с порогом, вероятность = среднее арифметическое вероятностей в листьях деревьев ансамбля (sklearn)

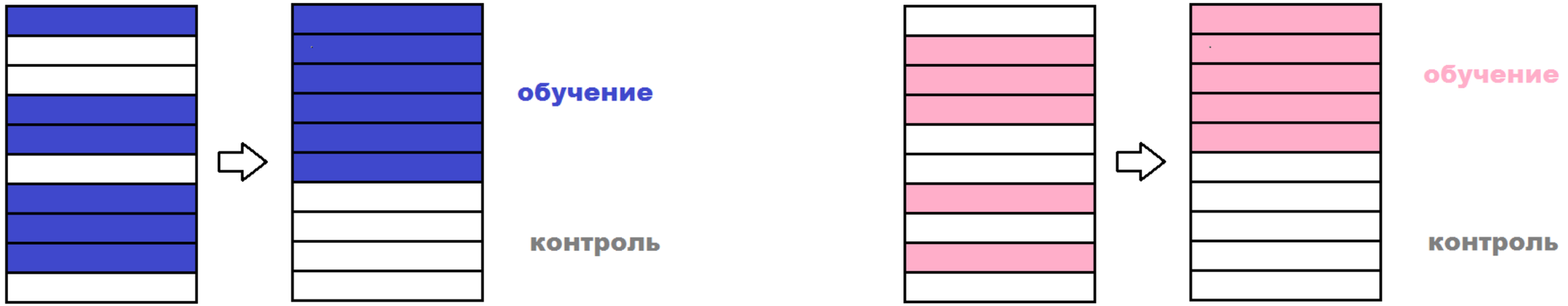
Ответ леса в задачах регрессии:

среднее арифметическое (в задачах регрессии)

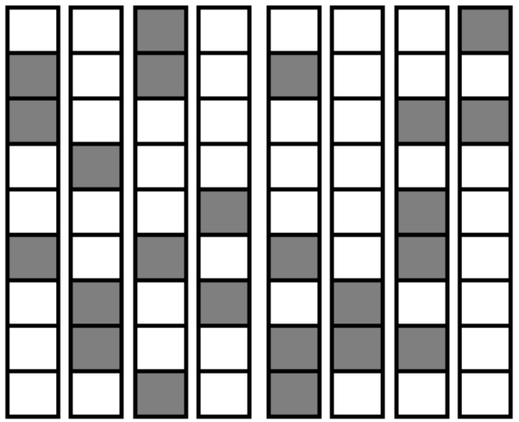
Автоматически при построении RF получаем

- **рейтинг признаков –**
  `importance(model)/ .feature_importances_`
- **уверенность в ответе – дисперсия ответов деревьев**

Бэггинг и OOB (out of bag)



Выбор объектов для обучения (с помощью бутстрепа), остальные – локальный контроль…



Ответы разных деревьев – можно усреднить и вычислить качество

Реализация решающего дерева: `sklearn.tree import DecisionTreeClassifier`

criterion	критерий расщепления «gini» / «entropy»
splitter	разбиение «best» / «random»
max_depth	допустимая глубина
min_samples_split	минимальная выборка для разбиения
min_samples_leaf	минимальная мощность листа
min_weight_fraction_leaf	аналогично с весом
max_features	число признаков, которые смотрим для нахождения разбиения
random_state	инициализация генератора случайных чисел
max_leaf_nodes	допустимое число листьев
min_impurity_decrease	порог «зашумлённости» для разбиения
min_impurity_split	порог «зашумлённости» для останова
class_weight	веса классов («balanced» или словарь, список словарей)
ccp_alpha	для автоматической подрезки (0.0)

Реализация случайного леса: sklearn.ensemble import RandomForestClassifier

n_estimators=100	число деревьев
criterion='gini'	критерий расщепления «gini» / «entropy»
splitter	разбиение «best» / «random»
max_depth=None	допустимая глубина
min_samples_split=2	минимальная выборка для разбиения
min_samples_leaf=1	минимальная мощность листа
min_weight_fraction_leaf=0.0	аналогично с весом
max_features='auto'	число признаков для нахождения разбиения
max_leaf_nodes=None	допустимое число листьев
min_impurity_decrease=0.0	порог изменения «зашумлённости» для разбиения
~~min_impurity_split=None~~	~~порог «зашумлённости» для останова~~
class_weight	веса классов («balanced» или словарь, список словарей)
ccp_alpha	для автоматической подрезки (0.0)
bootstrap=True	делать ли бутстреп
max_samples	объём выборки для сэмплирования
oob_score	вычислять ли OOB-ошибку
warm_start	дополнять ли существующий лес

n_jobs, random_state, verbose

Особенности

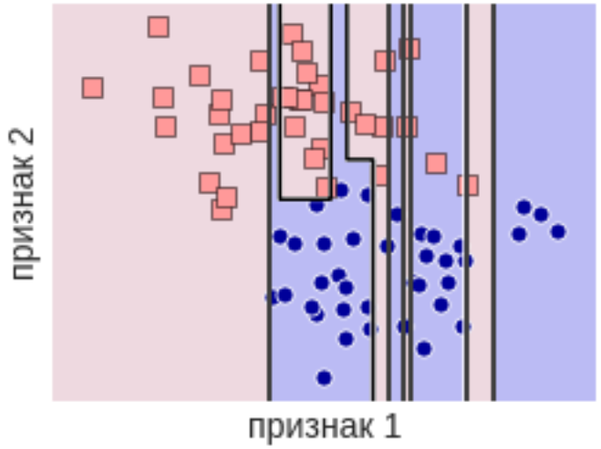
Изменение impurity – порог на

$$\frac{|R|}{m}\left(H(R) - \frac{|R_{\text{left}}|}{|R|} H(R_{\text{left}}) - \frac{|R_{\text{right}}|}{|R|} H(R_{\text{right}}) \right)$$

«Случайный лес»

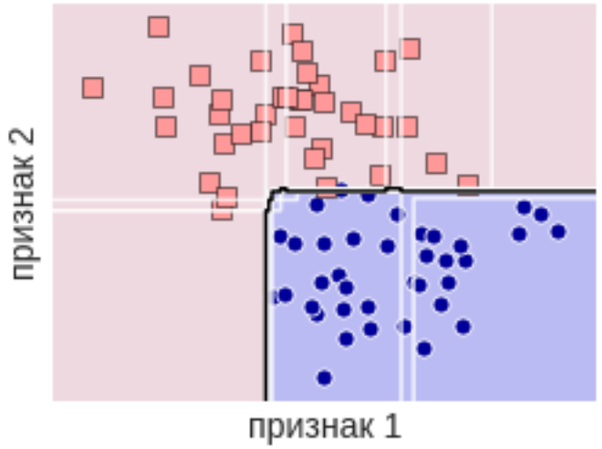
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=1)
rf.fit(X_train, y_train)
```

n_estimators=1



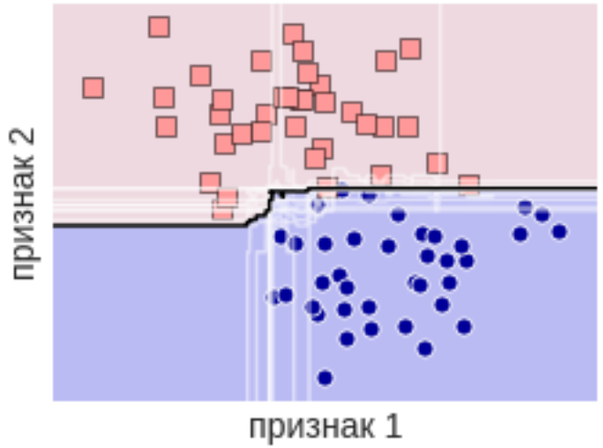
86%

n_estimators=5



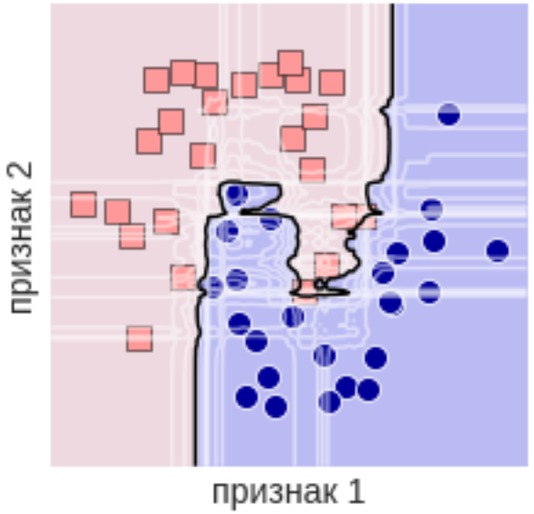
84%

n_estimators=100

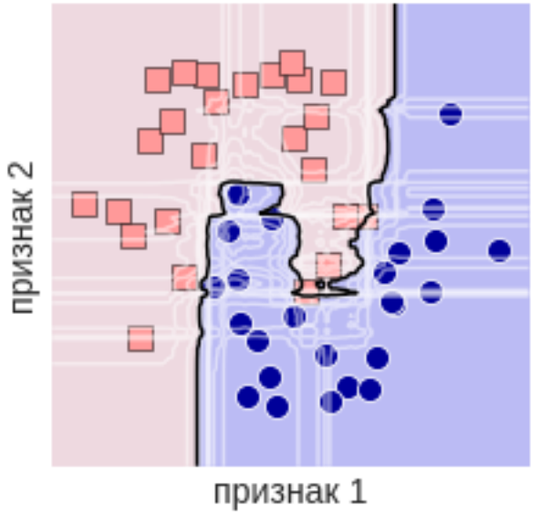


88%

Различные критерии расщепления



`criterion='gini'`



`criterion='entropy'`

в модельных задачах «на глаз» разницы не видно

в авторском коде был реализован Джини…

Настройка параметров: размер подвыборки sampsize / max_samples

1. Определиться с типом выбора

   с возвратом / без возврата

2. Настройка по объёму

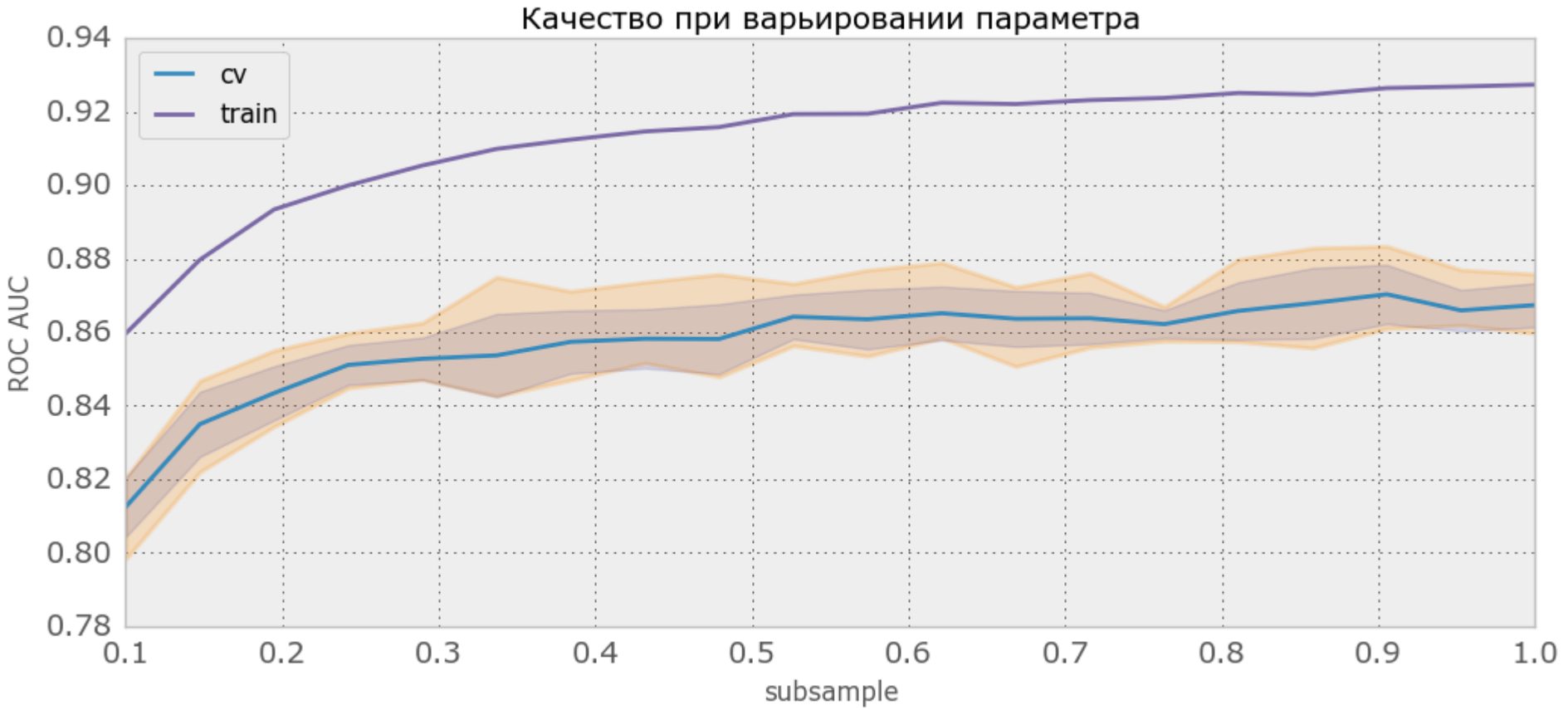
   – не в первую очередь

Часто «нужны все объекты»

Чем больше – тем однотипнее деревья

Что из этого следует?

Настройка параметров: размер подвыборки sampsize (СберБанк)



Всю выборку надо использовать по максимуму!

Настройка параметров: число признаков mtry / max_features

Самый серьёзный параметр

По умолчанию часто: $\sqrt{n}$ **– классификация**

$n/3$ **– регрессия**

Зависимость унимодальная

Настраивается в первую очередь

Зависит от числа шумовых признаков

Надо перенастраивать при добавлении новых признаков

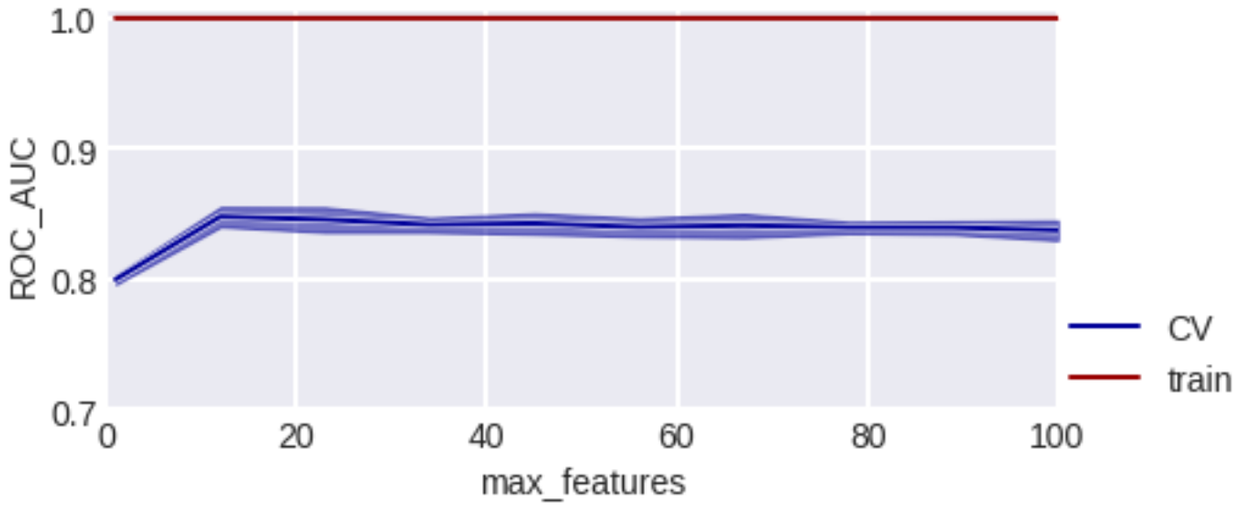
Чем больше – тем однотипнее деревья.

Чем больше – тем медленнее настройка!

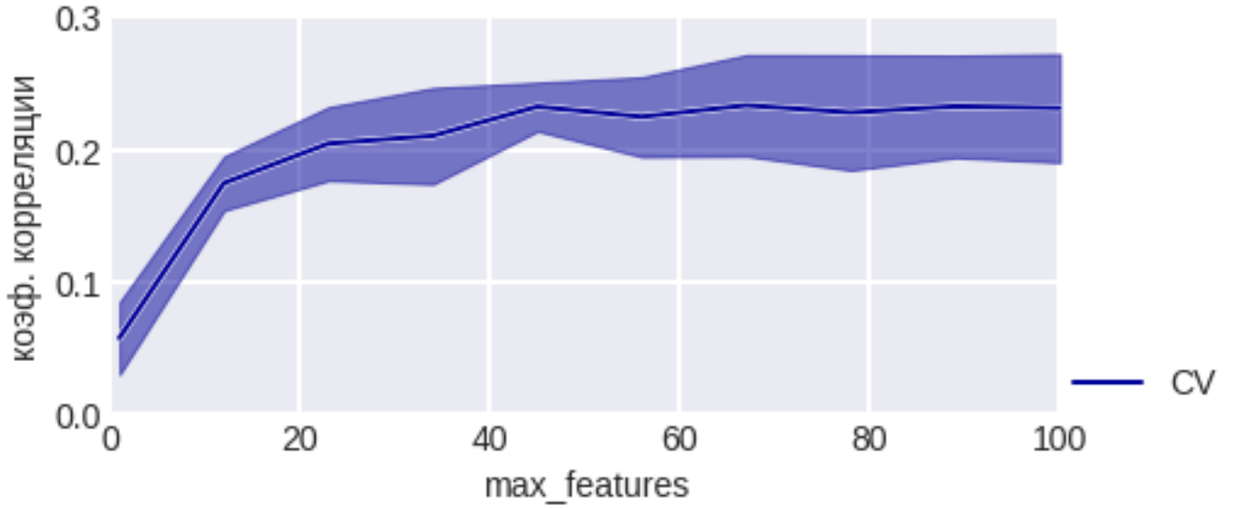
Kaggle: часто суммируют алгоритмы с разными mtry.

Настройка mtry / max_features (СберБанк)

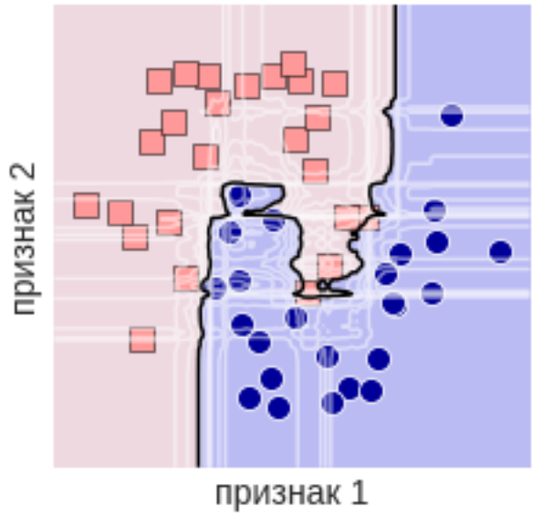
Качество от числа признаков, как правило, унимодально



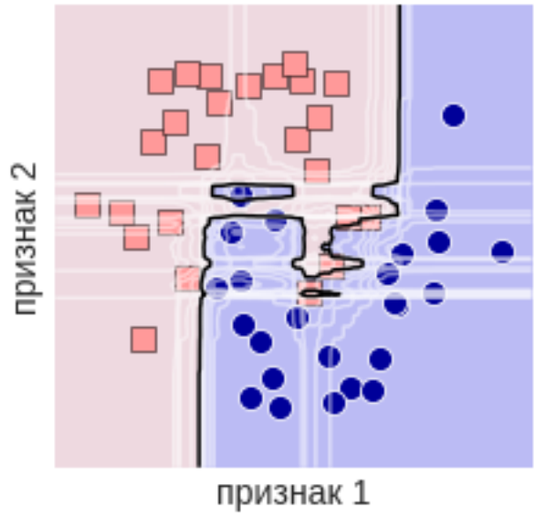
Коэффициент корреляции между ответами деревьев леса



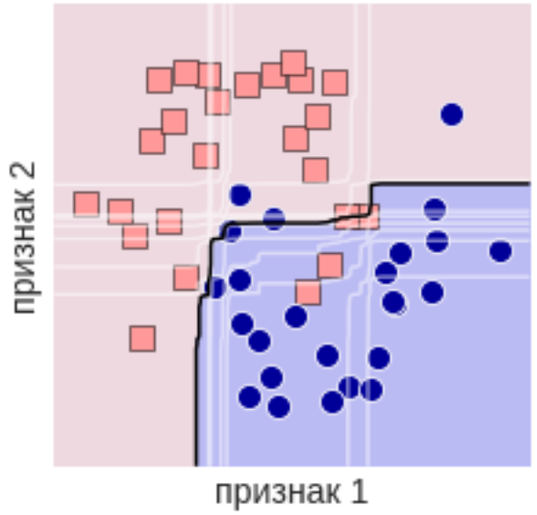
Геометрия mtry / max_features что можно сказать?



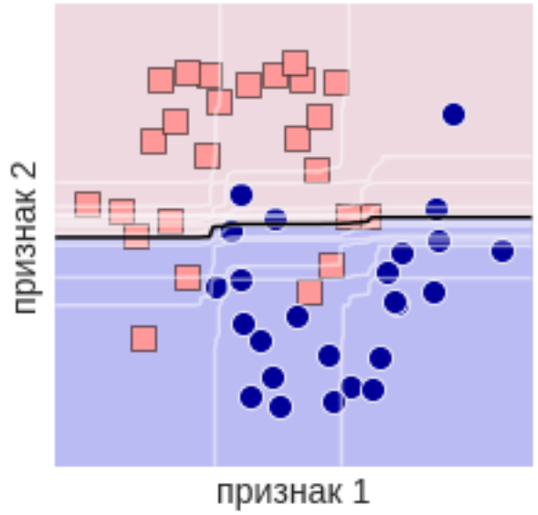
max_features=1



max_features=2

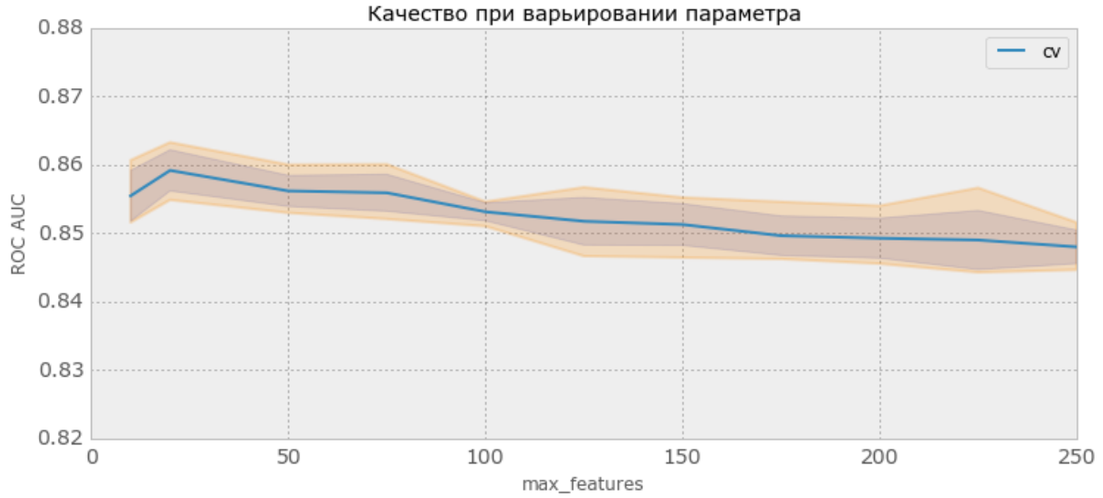


max_depth=1, max_features=1

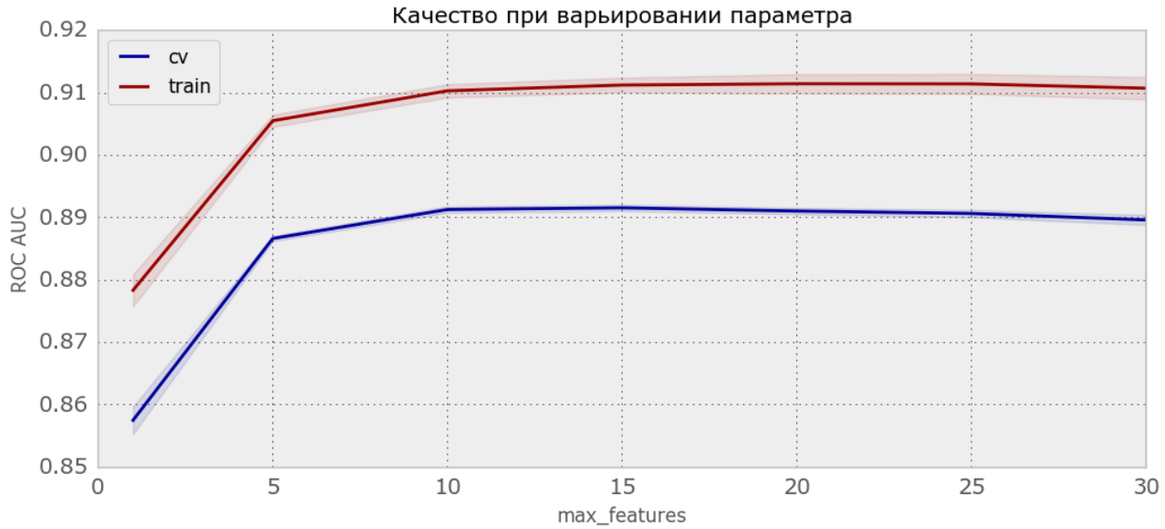


max_depth=1, max_features=2

Настройка mtry / max_features

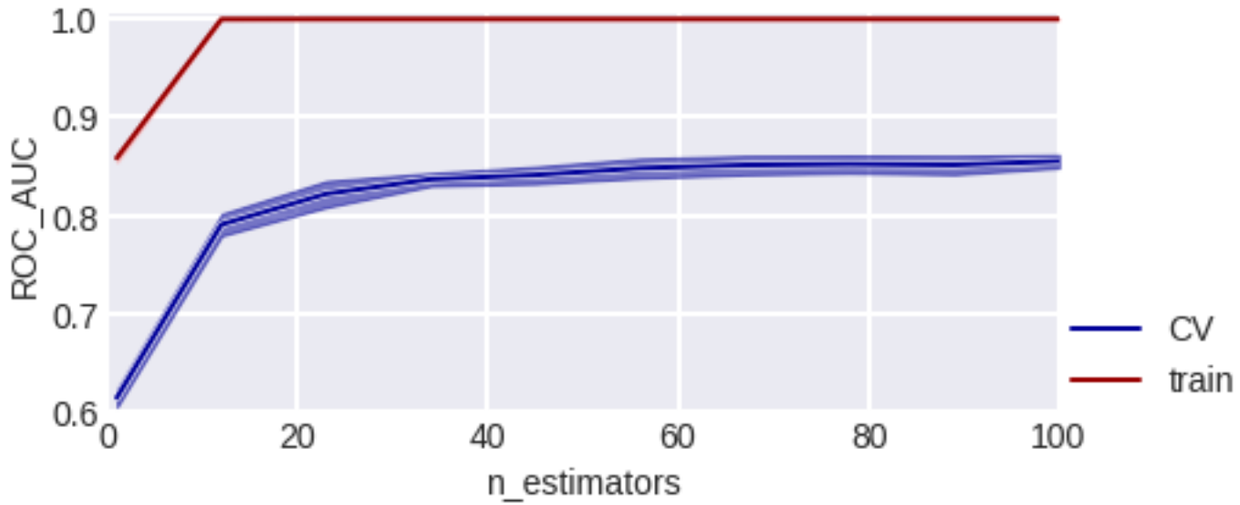


(ed Бозон) в задаче ~ 33 признака

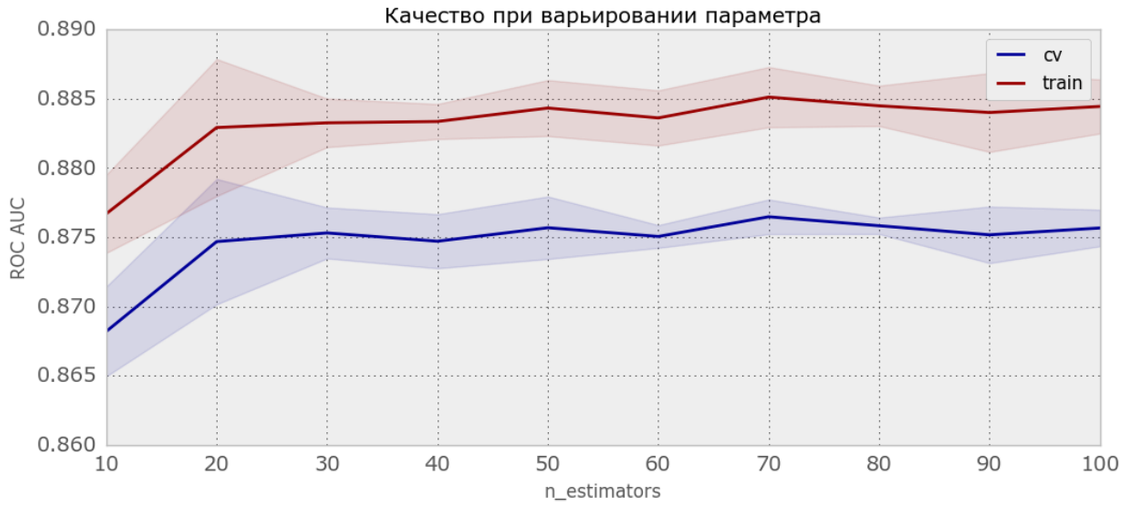


Настройка параметров ntree / n_estimators (СберБанк)

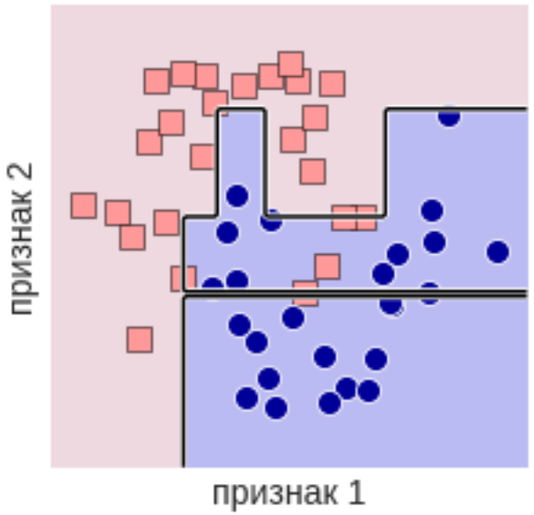
Чем больше деревьев – тем лучше!



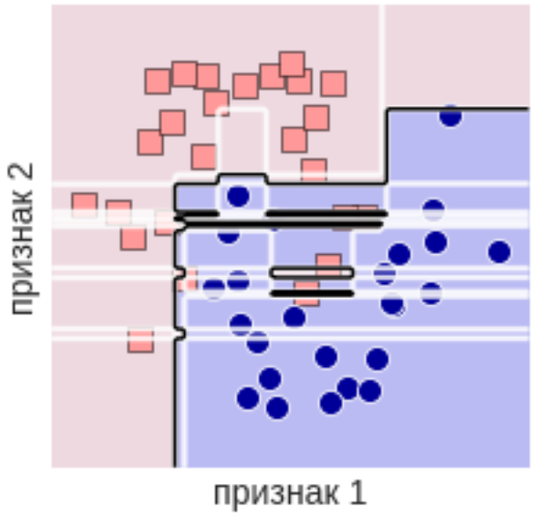
(ed Бозон)



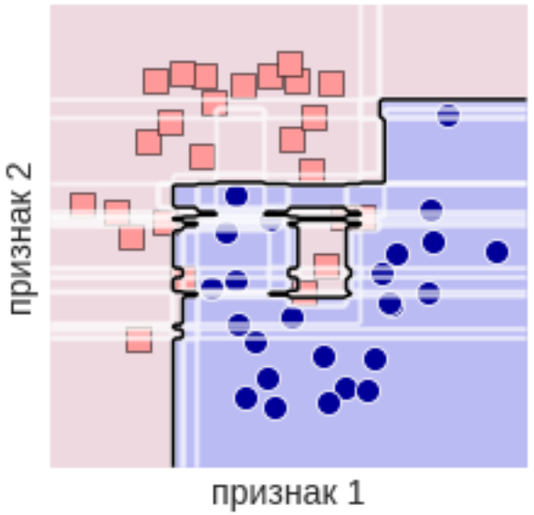
Настройка параметров ntree / n_estimators



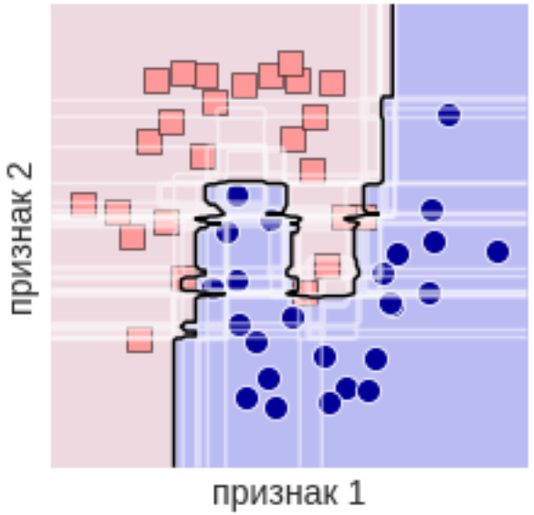
RF, число деревьев=1



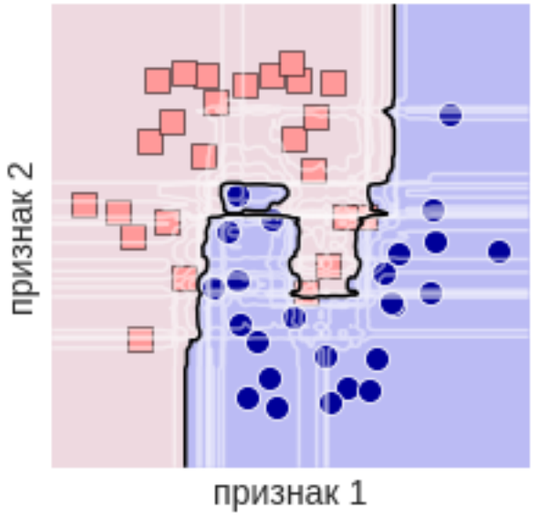
RF, число деревьев=3



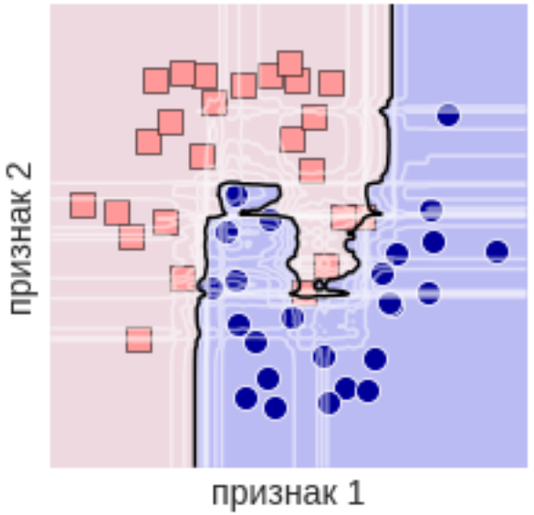
RF, число деревьев=5



RF, число деревьев=10



RF, число деревьев=100



RF, число деревьев=1000

Настройка параметров ntree / n_estimators (СберБанк)

Чем больше – тем лучше!

Проблемы:

- **как использовать при настройке параметров очень большое число деревьев**
- **что делать, если не помещаются в память… (например, в R)**
  можно строить по одному, получать ответ на тесте, не хранить в памяти

Настройка параметров: ограничения в листьях

число объектов в листе,
число объектов для расщепления,
максимальная глубина дерева

От параметров существенно зависит скорость построения леса

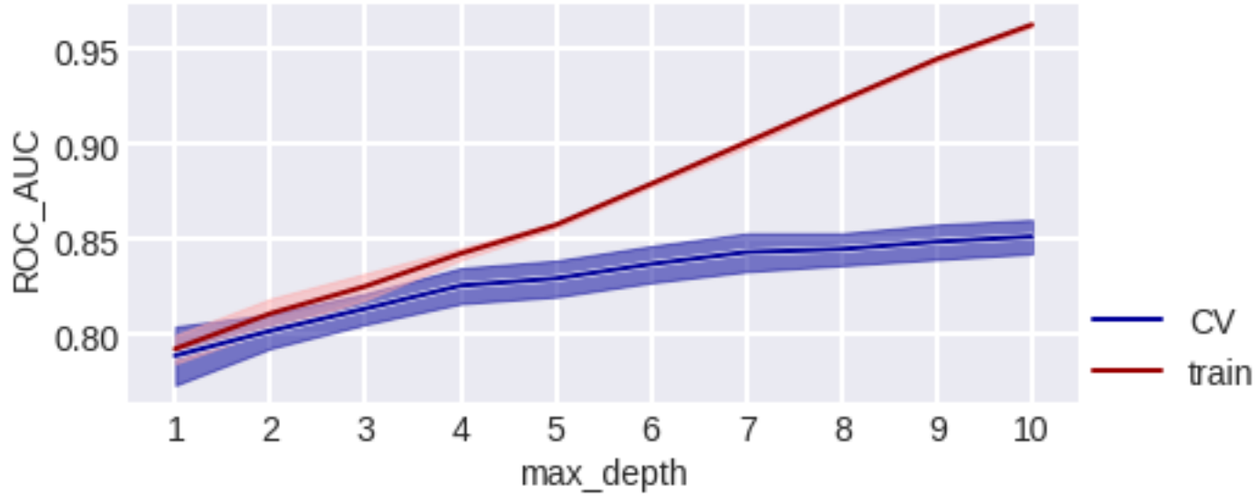
Оптимальные значения, как правило, – несколько объектов в листе.

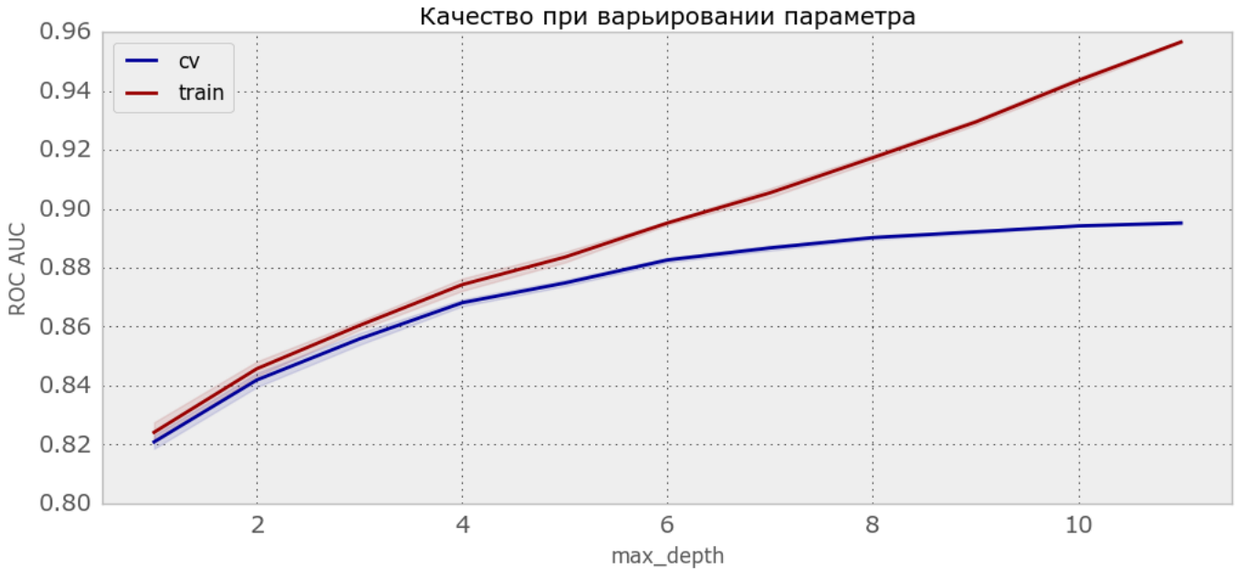
Настраиваются не в первую очередь

В классическом случайном лесе деревья строятся до исчерпания выборки…

«Good results are often achieved when setting `max_depth=None` in combination with `min_samples_split=1`»

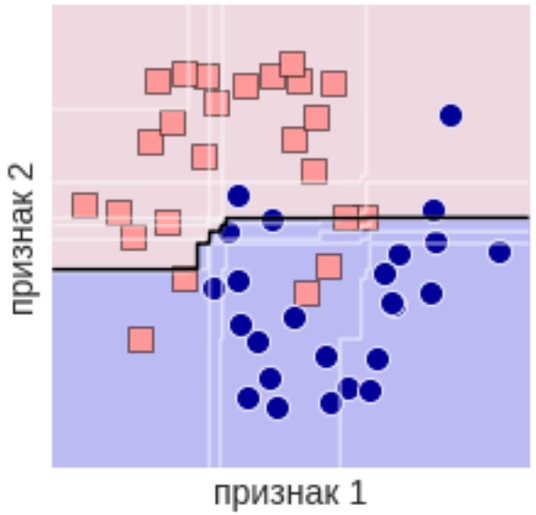
Глубина дерева: max_depth (СберБанк)



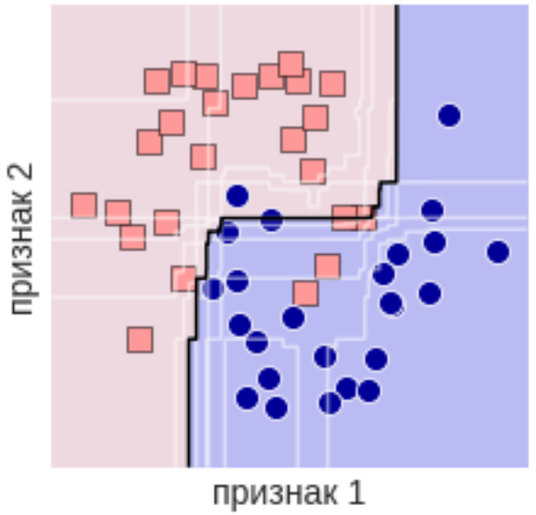


Как правило, чем больше, тем лучше!

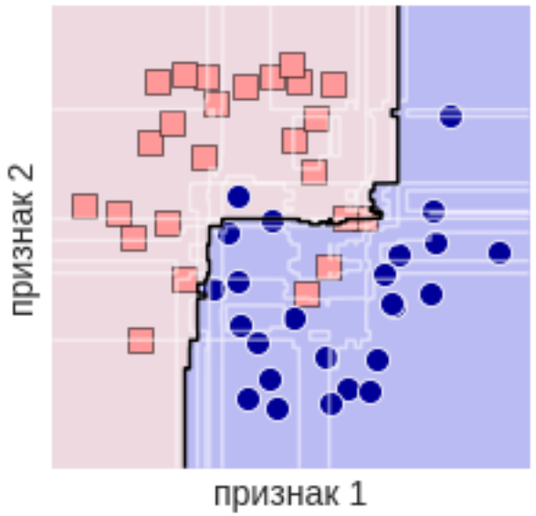
Глубина дерева: max_depth



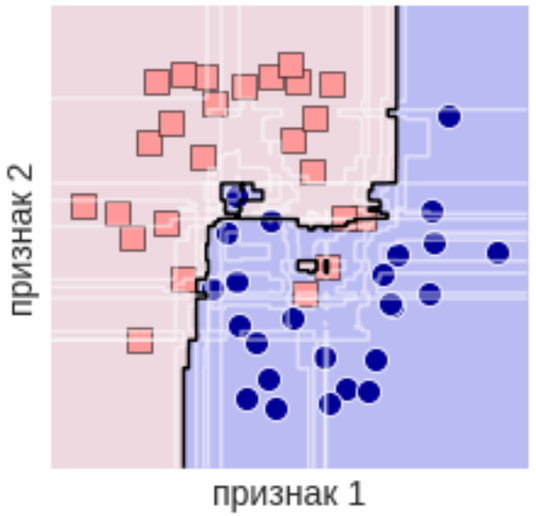
max_depth=1



max_depth=2



max_depth=3



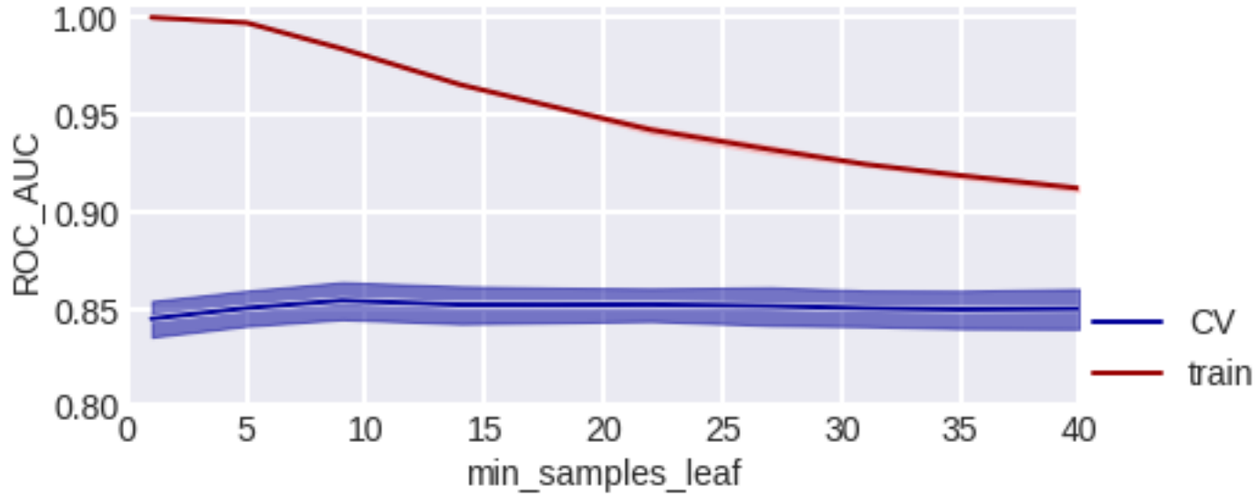
max_depth=4

Глубина дерева: max_depth

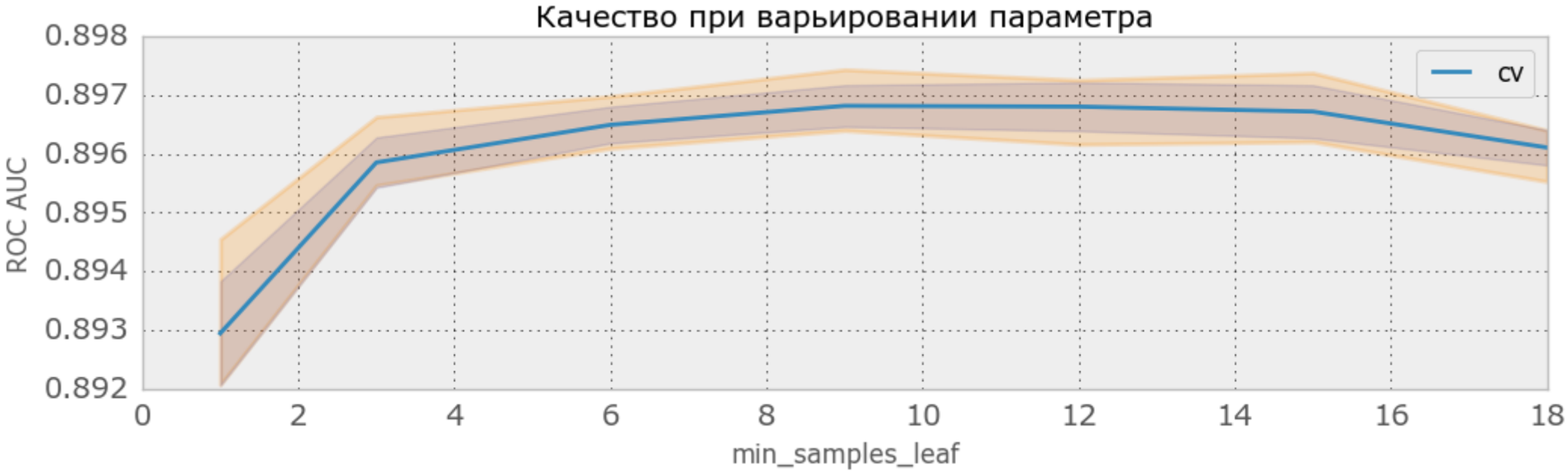
Неглубокие деревья:

- **в задачах с выбросами**
- **когда много объектов**
  (деревья большие и долго строятся)
- **настройка некоторых других (каких?) параметров не имеет смысла**

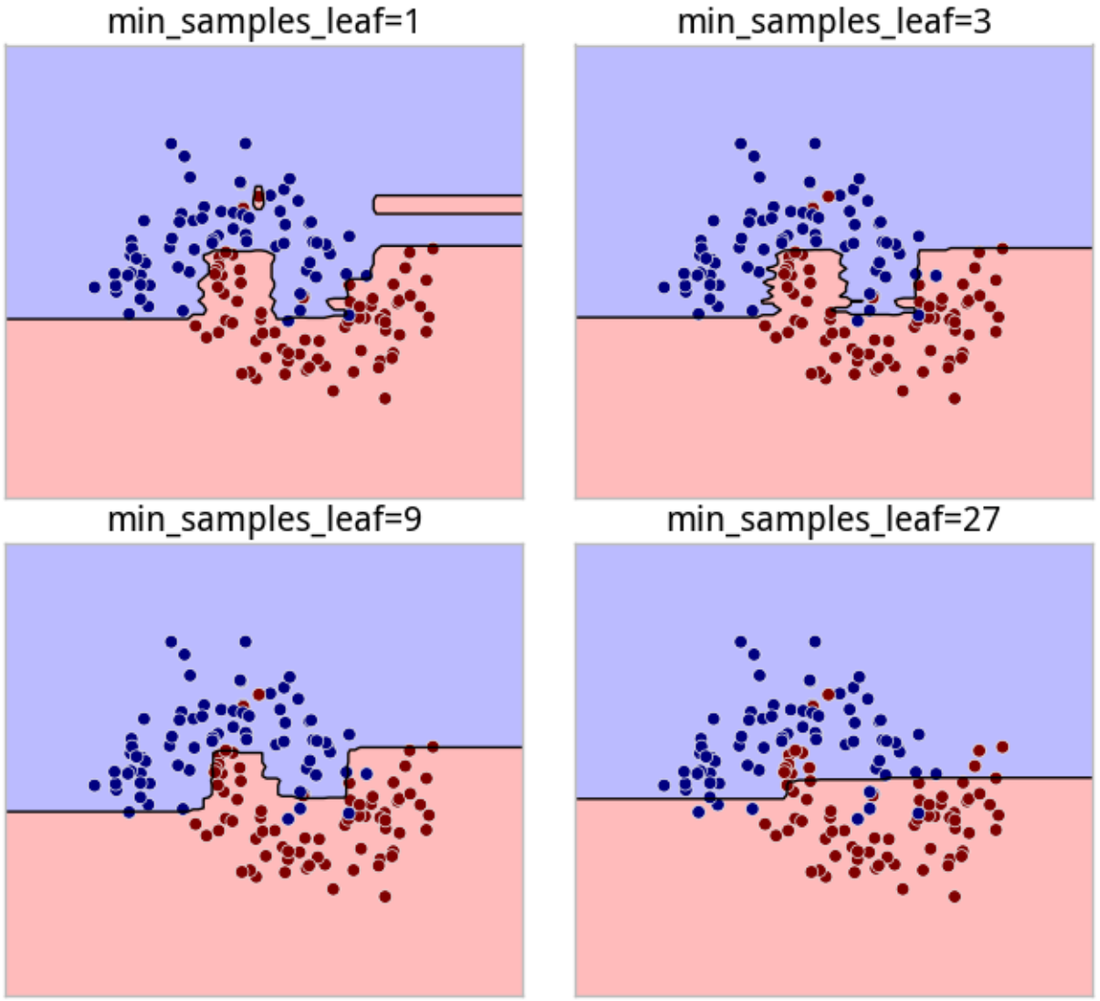
RandomForestClassifier: min_samples_leaf



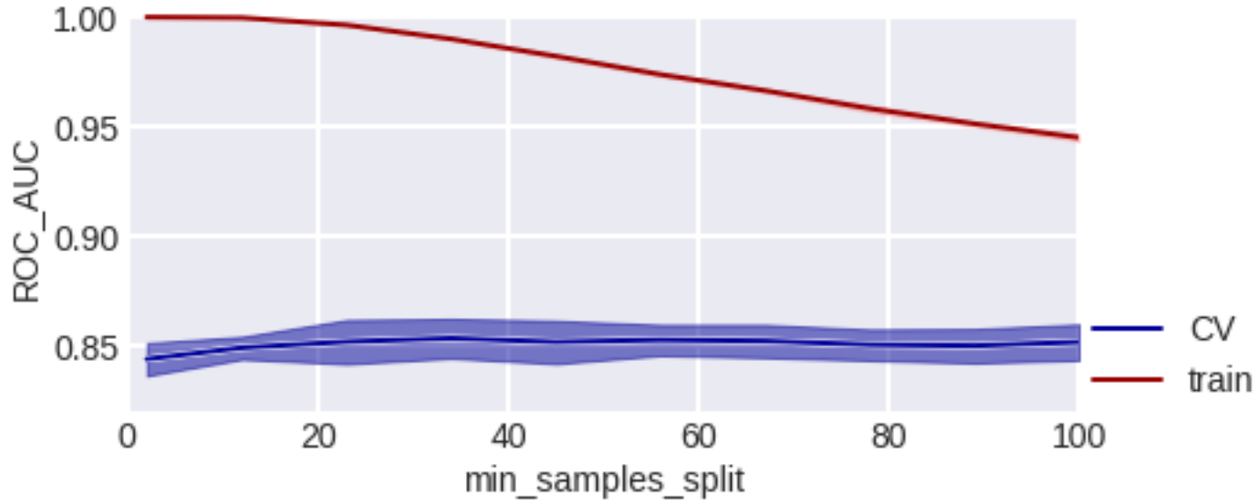
умолчание: 1 – классификация, 5 – регрессия

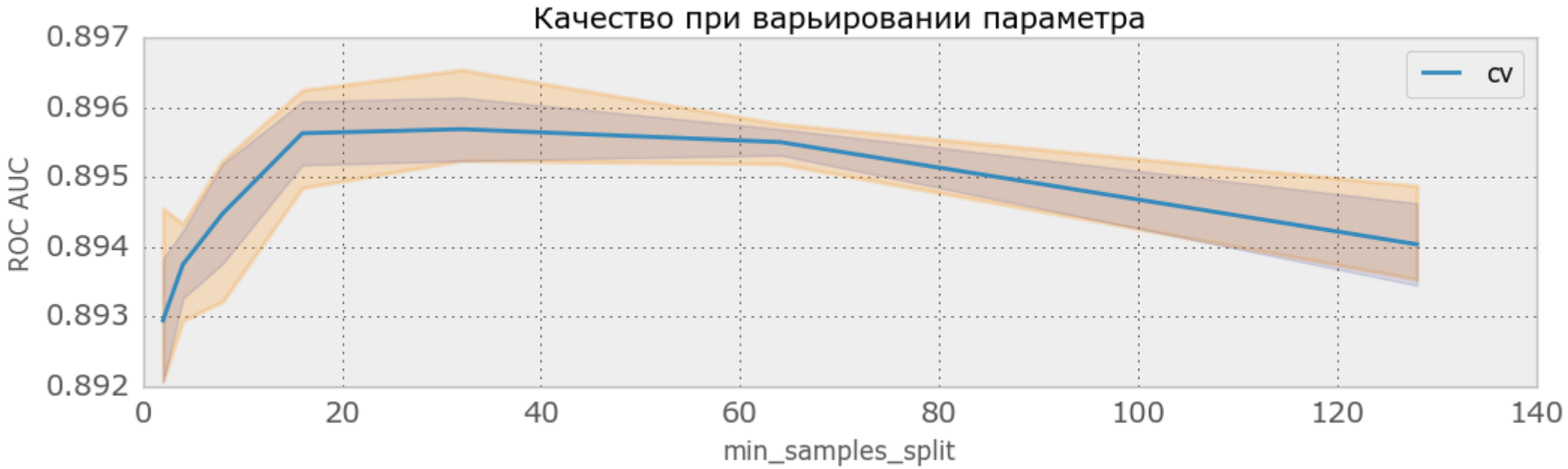


RandomForestClassifier: min_samples_leaf

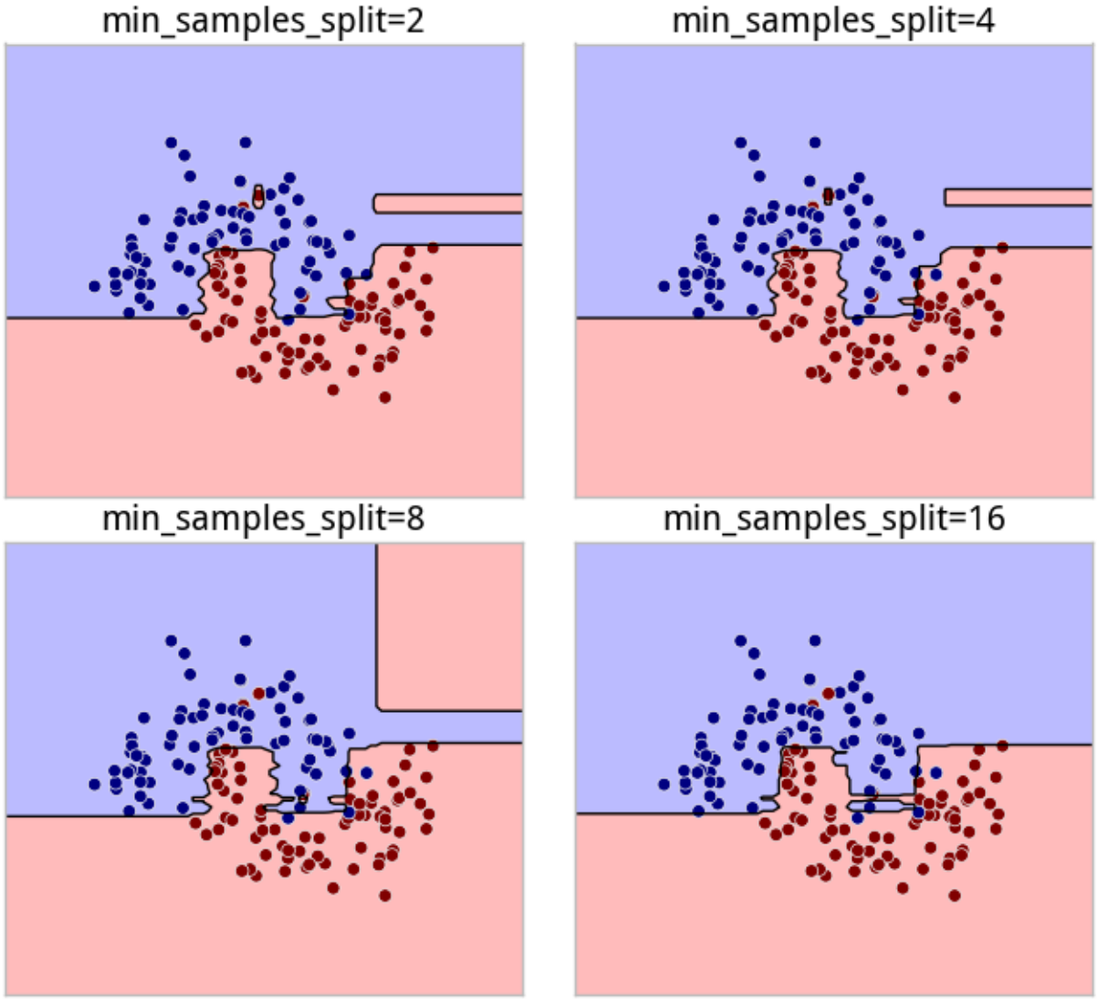


RandomForestClassifier: min_samples_split





RandomForestClassifier: min_samples_split



Проблемы RF

Может долго считаться…

Вместо CV – разбиение на обучение и контроль (hold out)

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.33, random_state=42)
```

sklearn: не забывать `n_jobs`

Proximity

при построении деревьев можно много чего считать…

**Чем чаще 2 объекта попадают в один лист,
тем они ближе…**

Какую метрику можно придумать?

Extreme Random Trees (ExtraTrees)

- нет бутстрепа (используем всю выборку)
- генерируем несколько пар (признак, порог)
  – выбираем оптимальную для разбиения пару
- также есть параметр «число признаков для просмотра»

- ET быстрее RF
- ET чуть хуже RF, когда много шумных признаков

```
from sklearn.ensemble import ExtraTreesClassifier
clf = ExtraTreesClassifier(n_estimators=10, max_depth=None,
                           min_samples_split=2, random_state=0)
```

Ансамбли на базе RF

Синтетический случайный лес (Synthetic RF) – стекинг лесов с разным nodesize / min_samples_leaf

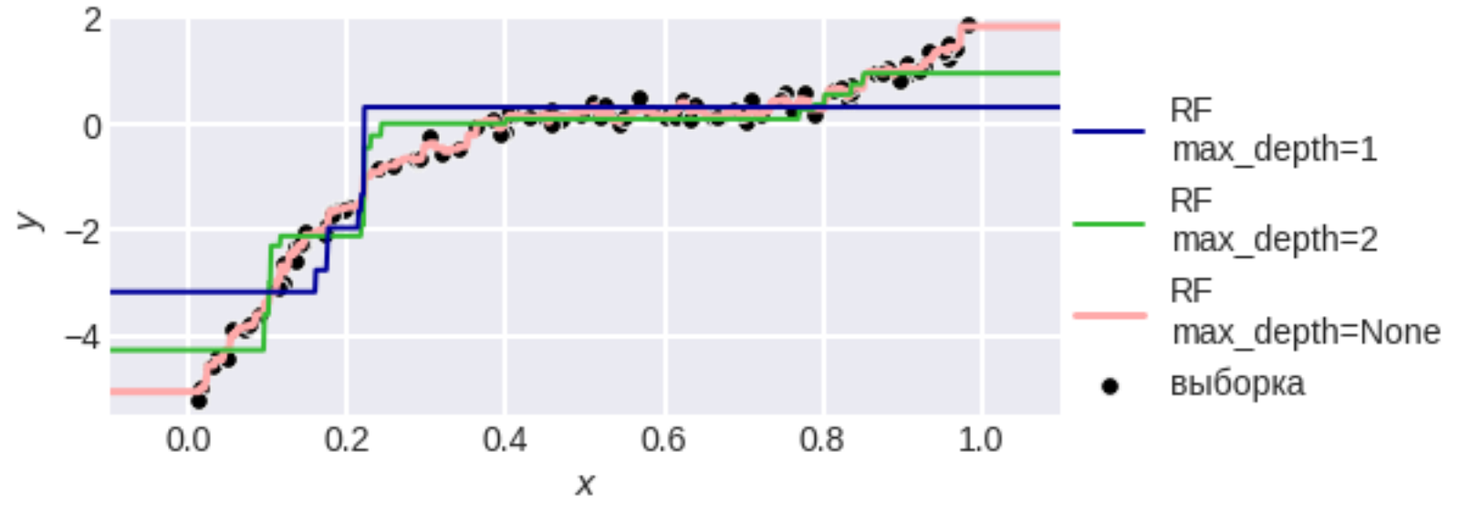
Algorithm 1 *Synthetic Random Forests (SRF)*

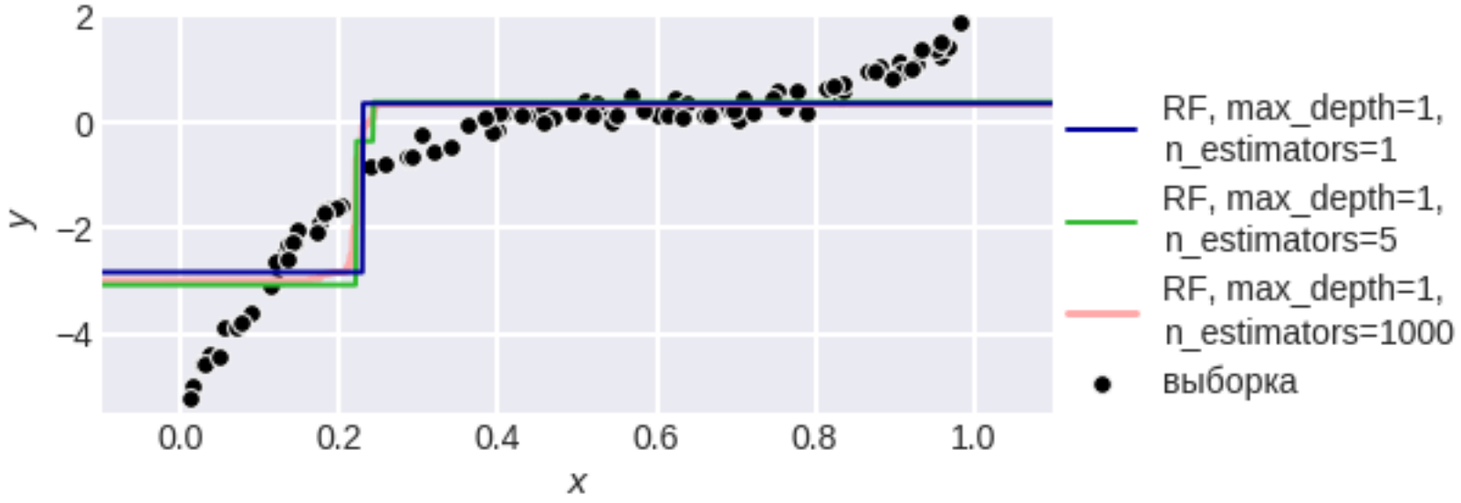
1: Choose a set of candidate nodesize values $\mathcal{N} = \{n_1, n_2, \ldots, n_D\}$.
2: Fit a RF with *nodesize* $= n_j$ for $j = 1, \ldots, D$. Use the same *ntree* and *mtry* value for each forest. Denote the resulting forests by $\mathrm{RF}_1, \ldots, \mathrm{RF}_D$.
3: Calculate the predicted value for each random forest RF_j, $j = 1, \ldots, D$. We call the predicted value the synthetic feature.
4: Fit a RF using for features both the newly created synthetic features and the original p features (using the same *ntree* and *mtry* value as before). We call this the synthetic RF.

чтобы не переобучаться используется OOB-прогноз

Ishwaran H, Malley JD. «Synthetic learning machines». BioData Min. 2014;7(1):28 // https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4279689/pdf/13040_2014_Article_28.pdf

Когда плохи методы, основанные на деревьях…





Итог

Число признаков (max_features) – самый важный параметр (унимодальность)

Число базовых алгоритмов (n_estimators) – чем больше, тем лучше

Глубина (max_depth) – скорее всего, максимальная

Параметры сложности (min_samples_leaf, min_samples_split) – чуть подкорректировать (не очень важно)

Подвыборка (samsize) – брать всё (часто и выбора нет)

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…

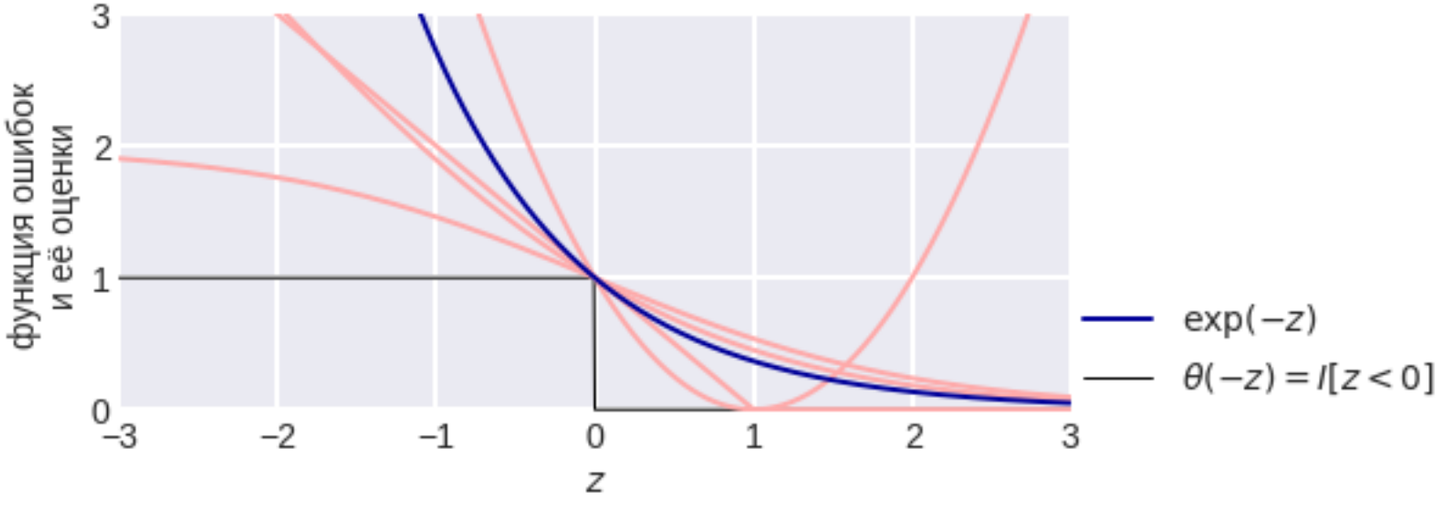
AdaBoost: постановка задачи

– FSAM для бинарной задачи классификации $Y = \{+1, -1\}$

базовые классификаторы генерируют классы $b(x) \in \{+1, -1\}$

Ансамбль

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



exponential loss

$$L(y,a) = \text{exploss}(y,a) \equiv$$

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

заметим, что

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

AdaBoost: весовая схема

У каждого объекта – вес (распределение!)

$$W = \left(w_1, \ldots, w_m\right) \geq 0$$

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

Взвешенное число ошибок:

$$e_W(a) = \sum_{t\,:\,a(x_t)\neq y(x_t)} w_t = \sum_{t=1}^{m} w_t I[a(x_t) \neq y(x_t)]$$

«ошибка, порождённая распределением»

(формально не имеет общего с экспоненциальной ошибкой, но мы используем в оценке)

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, который допускает ошибку $e_W(b_j)$

(вычисляется по распределению W)

предполагаем, что $0 < e_W(b_j) < 0.5$

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

«перестроить» распределение $W = (w_1, \ldots, w_m)$**:**

$$w_t \leftarrow \frac{w_t \exp(-\alpha_j y(x_t) b_j(x_t))}{\sum_{i=1}^{m} w_i \exp(-\alpha_j y(x_i) b_j(x_i))}$$

+ нормировка

вариант: перенастраивать веса только объектов, на которых ошибки…

AdaBoost: вывод формул для экспоненциальной ошибки

Напомним, что экспоненциальная ошибка:

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

Число ошибок оценивается так:

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

$$= \sum_{t=1}^{m} \underbrace{\exp\left(-y_t \sum_{j=1}^{s-1} \alpha_j b_j(x_t) \right)}_{\sim w_t} \exp\left(-y_t \alpha_s b_s(x_t) \right) \sim \sum_{t=1}^{m} w_t \exp\left(-y_t \alpha_s b_s(x_t) \right)$$

первый множитель пропорционален весу объекта (с точностью до знаменателя)

AdaBoost: вывод формул для экспоненциальной ошибки

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

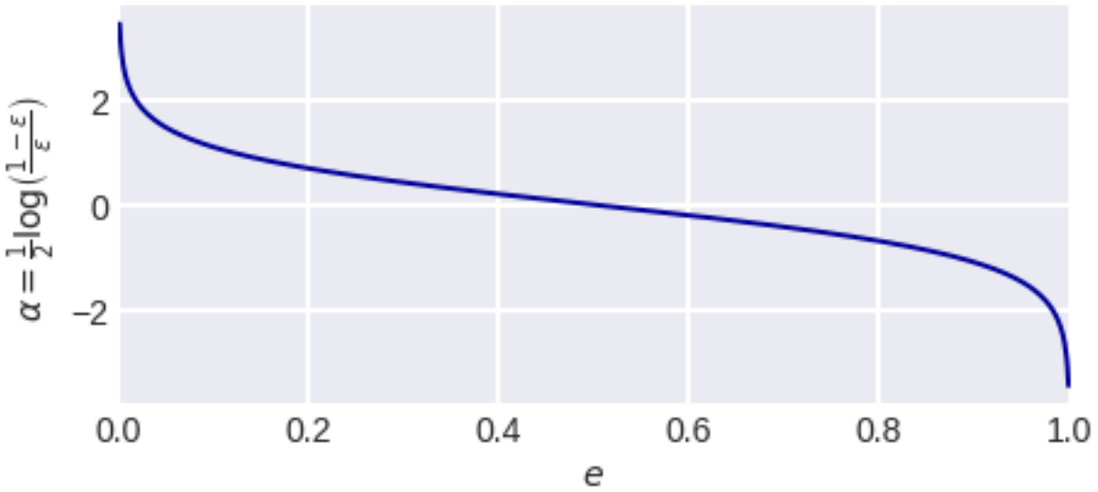
$$= \sum_{t:y_t=a_s(x_t)} w_t \exp\left(-\alpha_s\right) + \sum_{t:y_t\neq a_s(x_t)} w_t \exp\left(\alpha_s\right) =$$

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

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 пересчёта весов…

$$w_t \leftarrow \frac{w_t \exp(-\alpha_j y(x_t) b_j(x_t))}{\sum_{i=1}^{m} w_i \exp(-\alpha_j y(x_i) b_j(x_i))}$$

если рекурсивно пересчитать…

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

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

$$= \sum_{t=1}^{m} \text{exploss}(y_t, a_s(x_t))$$

AdaBoost: вывод формул для экспоненциальной ошибки

Получили – после j-й итерации

$$w_t \sim \text{exploss}(y(x_t), a_j(x_t))$$

вес объекта пропорционален ошибке на этом объекте

теперь смотрим на формулу ошибки

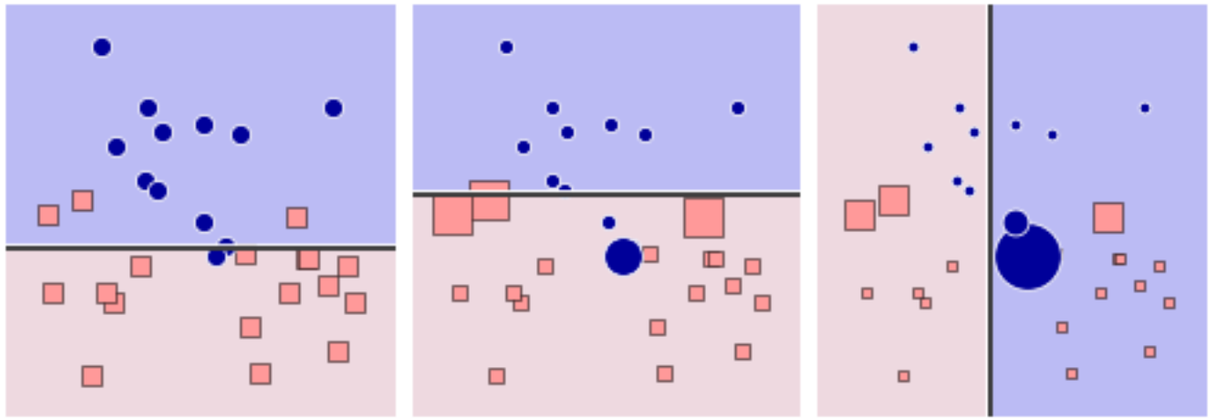
$$\sim \sum_{t=1}^{m} \underbrace{w_t \quad \exp\left(-y_t \alpha_s b_s(x_t)\right)}_{\text{exploss на } s}$$

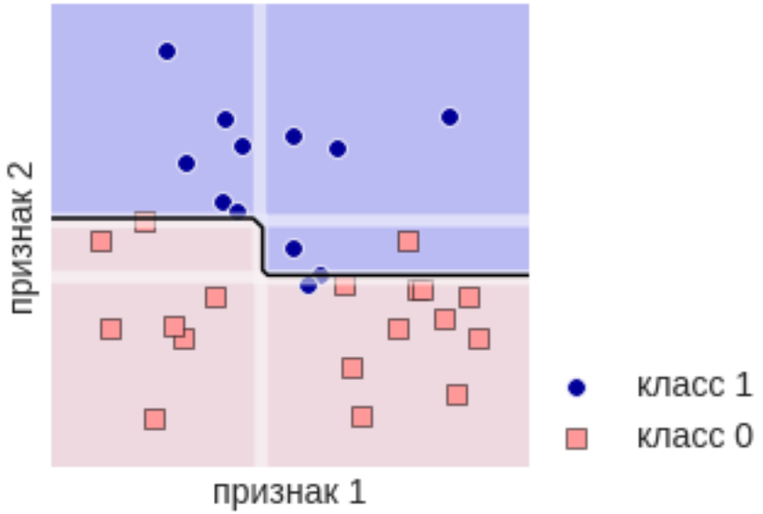
exploss на s-1

это просто учёт в весах новых ответов

это обосновывает предложенный способ пересчёта

AdaBoost: пример





площадь объектов пропорциональна весу…

в итоге – комбинация классификаторов

Как реализуется минимизация $e_w(b)$

- **встроенная весовая минимизация**
- **пересэмплирование**

AdaBoost: теория

Если на каждом шаге мы можем построить слабый (weak) классификатор:

$e_W(b_j) \le 0.5 - \varepsilon,\ \varepsilon > 0$, **то ошибка ансамбля**

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

на обучении оценивается как

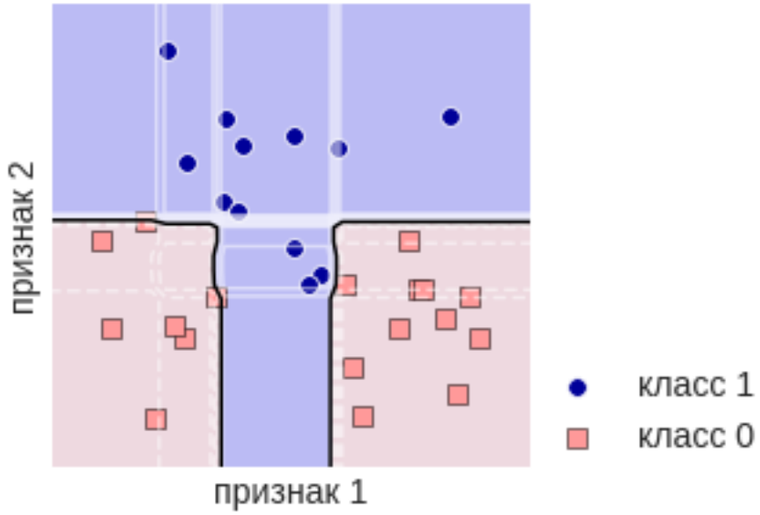
$$e_W(a) = \frac{1}{m} \sum_{t=1}^{m} I[a(x_t) \ne y(x_t)] \le \exp(-2\varepsilon^2 s)$$

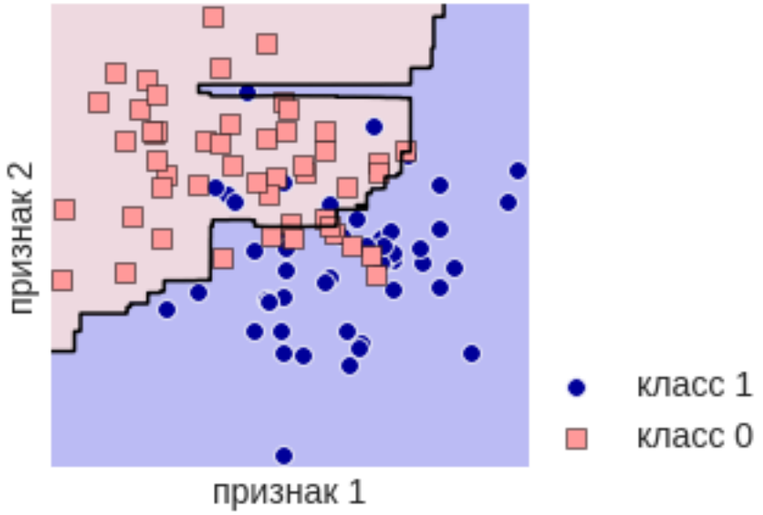
т.е. всего лишь из предположения, что слабый классификатор на ε лучше случайного

В чём небольшая некорректность в этой фразе?

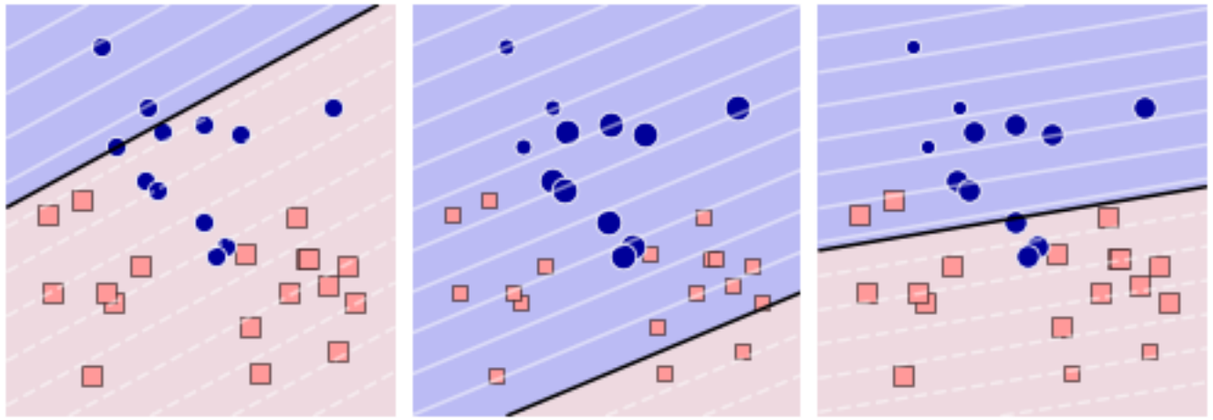
AdaBoost: минутка кода

```
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)
```





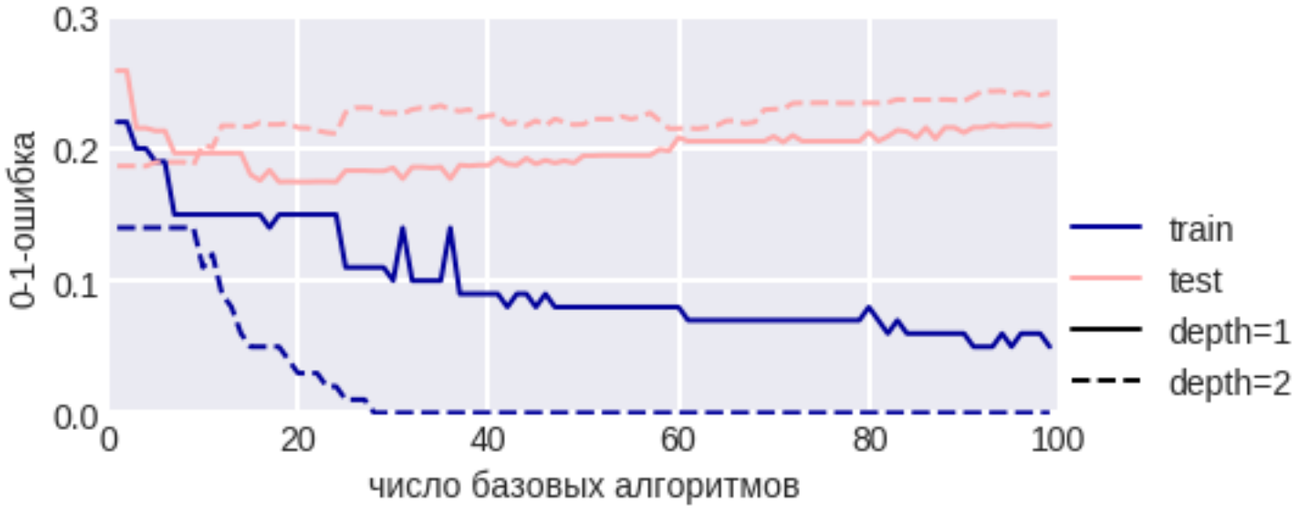
AdaBoost: недостатки

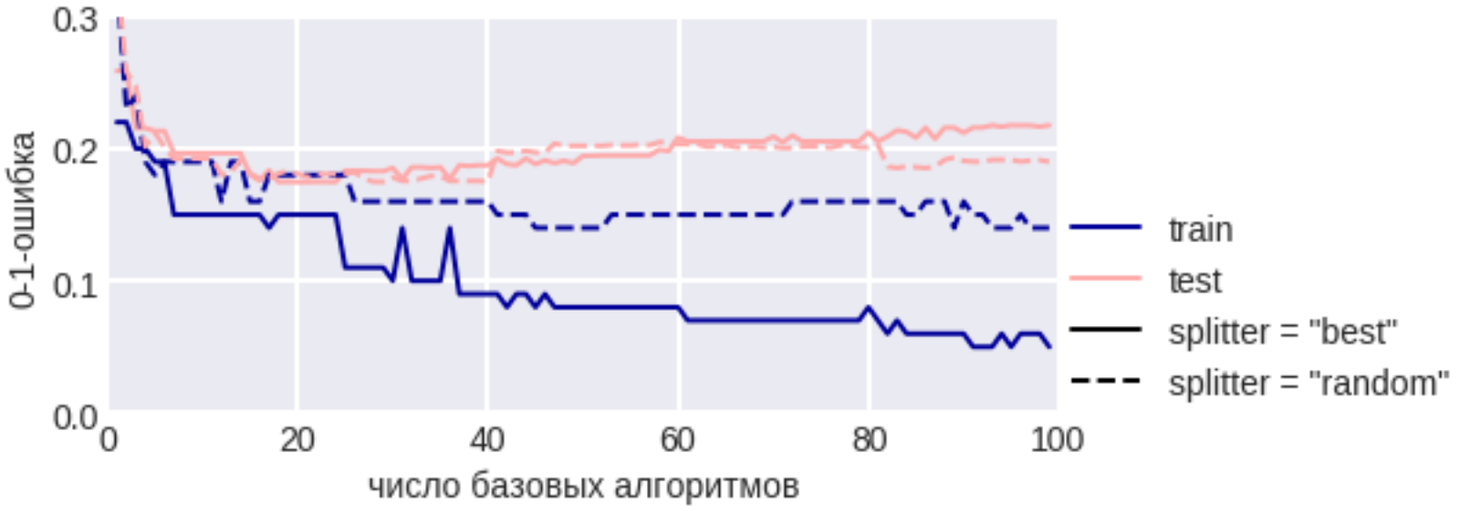


Бустинг плох, когда есть выбросы.

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

$$f(a_1, a_2)$$

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

Amazon Employee Access Challenge

Литература

A. Liaw, M. Wiener Classification and Regression by randomForest // R News (2002) Vol. 2/3 p. 18.

http://www.bios.unc.edu/~dzeng/BIOS740/randomforest.pdf

И. Генрихов О критериях ветвления, используемых при синтезе решающих деревьев // Машинное обучение и анализ данных, 2014, Т.1, №8, С.988-1017

http://jmlda.org/papers/doc/2014/no8/Genrikhov2014Criteria.pdf