

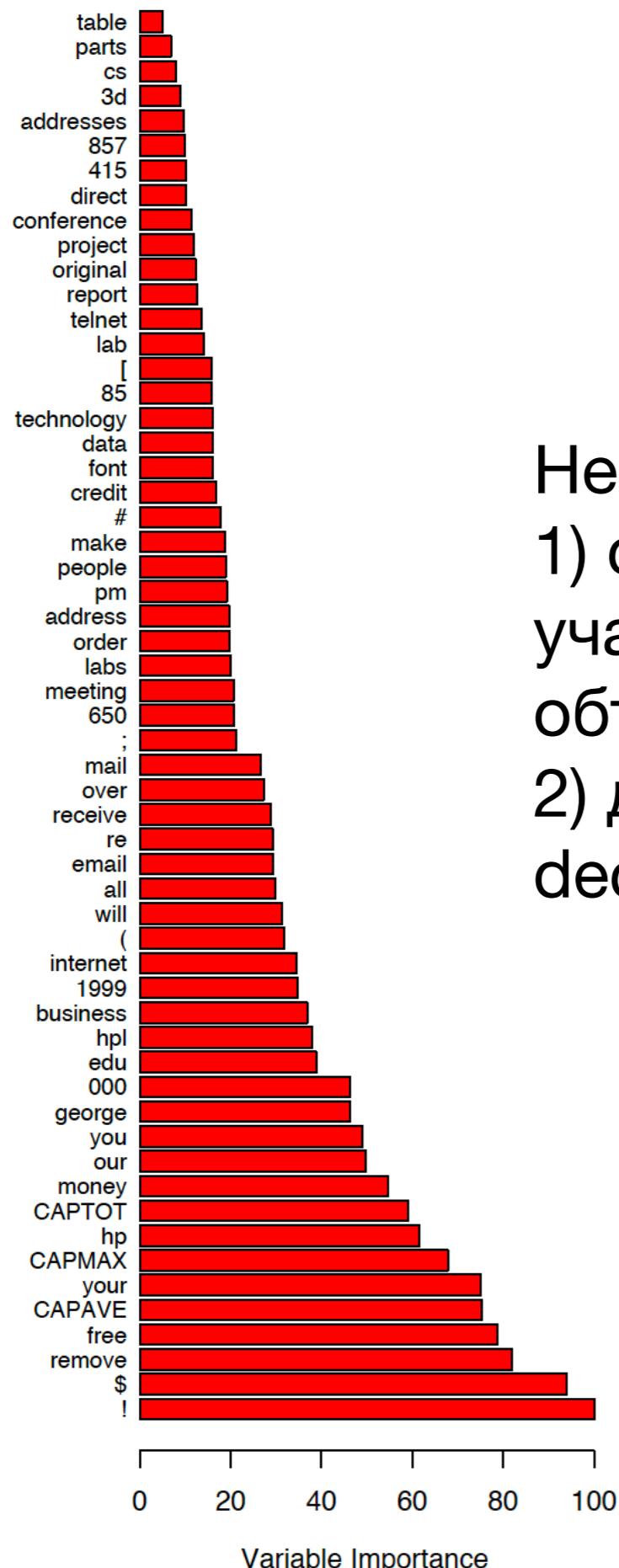
Выбор важных признаков

Gini impurity

Насколько хорошо
переменная помогает
нам разбивать данные

Несколько вариантов реализации:

- 1) считаем число разбиений, в которых участвует признак, умножая на число объектов, участвующих в разбиении
- 2) дополнительно учитываем gini impurity decrease (sklearn)



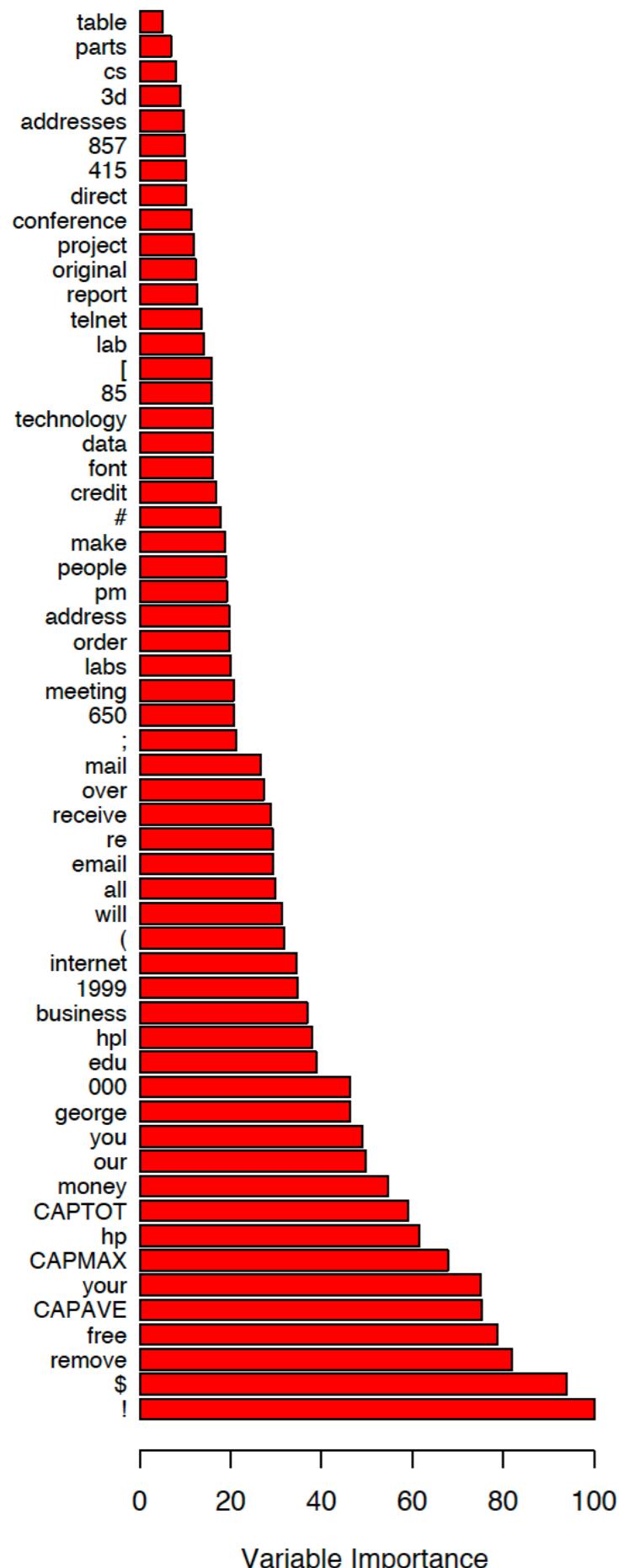
Gini impurity

Насколько хорошо
переменная помогает
нам разбивать данные

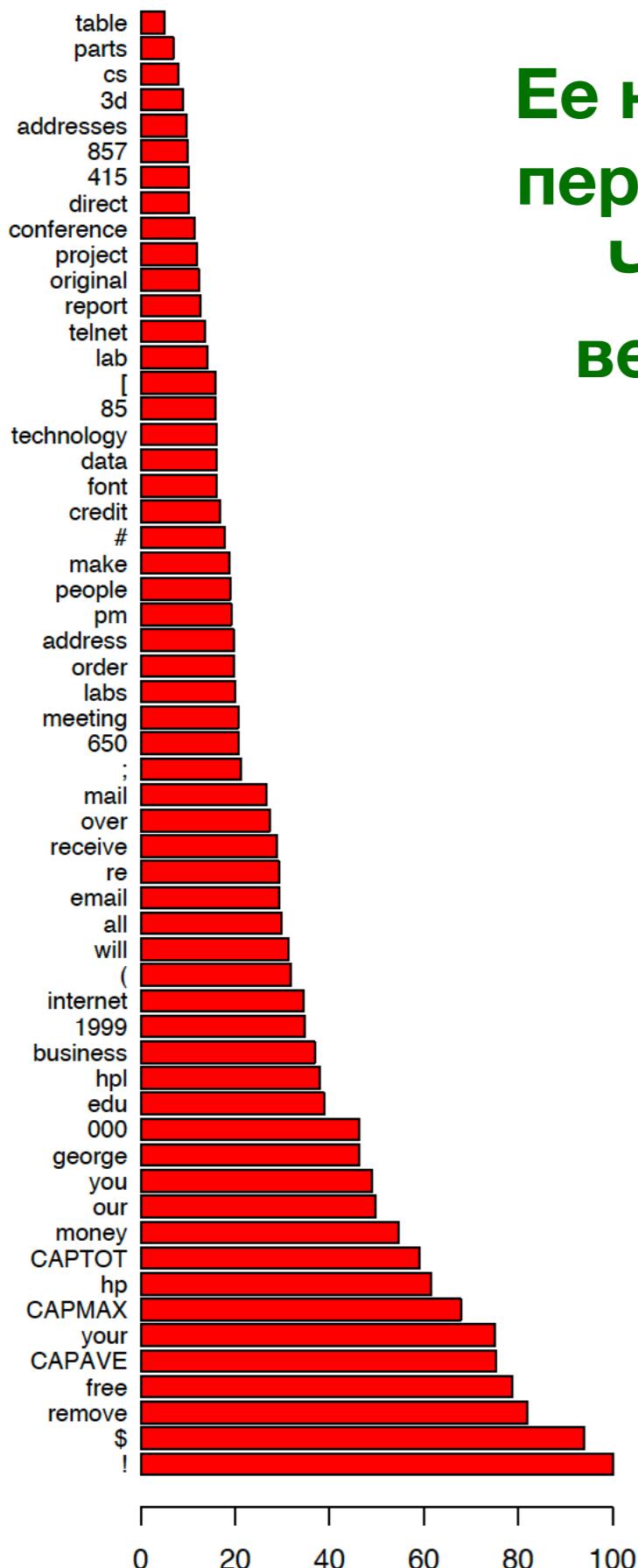
Несколько вариантов реализации:

- 1) считаем число разбиений, в которых участвует признак, умножая на число объектов, участвующих в разбиении
- 2) дополнительно учтываем gini impurity decrease (sklearn)

Какие минусы?



Gini impurity



Ее недостаток - больше любит вещественные
переменные и многомерные категориальные:

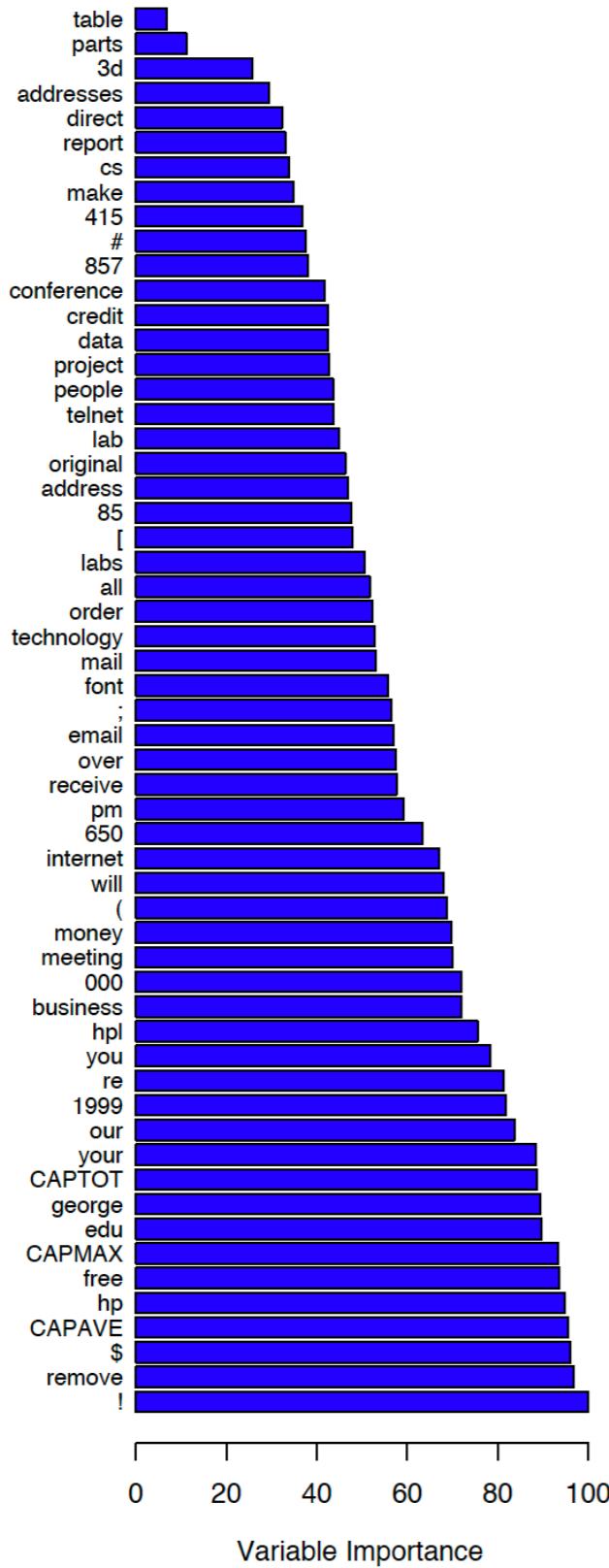
Чем больше вариантов порогов - больше
вероятность получить хорошее разбиение

В случае со случным лесом, как вариант -
можем считать gini impurity на ОOB (out-of-bag
sample), может дать более удобную оценку

[https://cran.r-project.org/web/packages/rfVarImpOOB/vignettes/
rfVarImpOOB-vignette.html](https://cran.r-project.org/web/packages/rfVarImpOOB/vignettes/rfVarImpOOB-vignette.html)

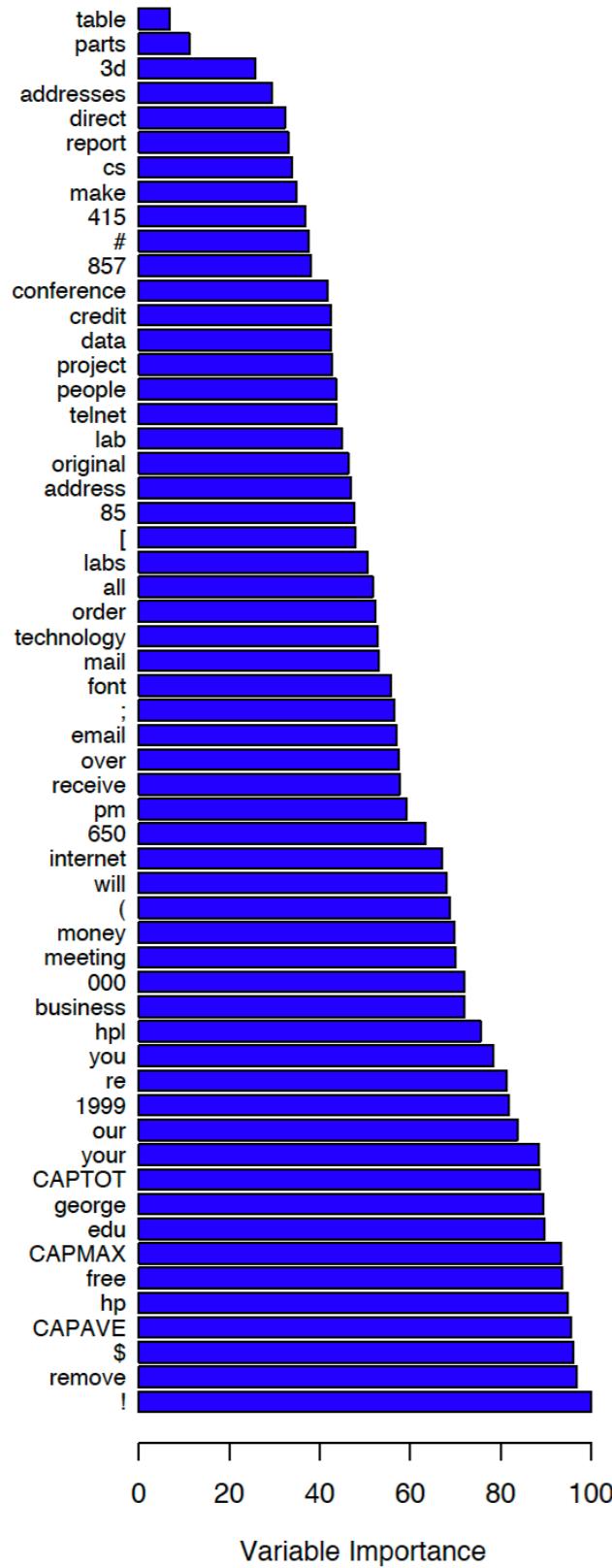
Выбор признаков

**Randomization/Permutation - если перемешаем
значения нашей переменной (она станет
бесмысленной), то какое падение в качестве мы
ожидаем**



Выбор признаков

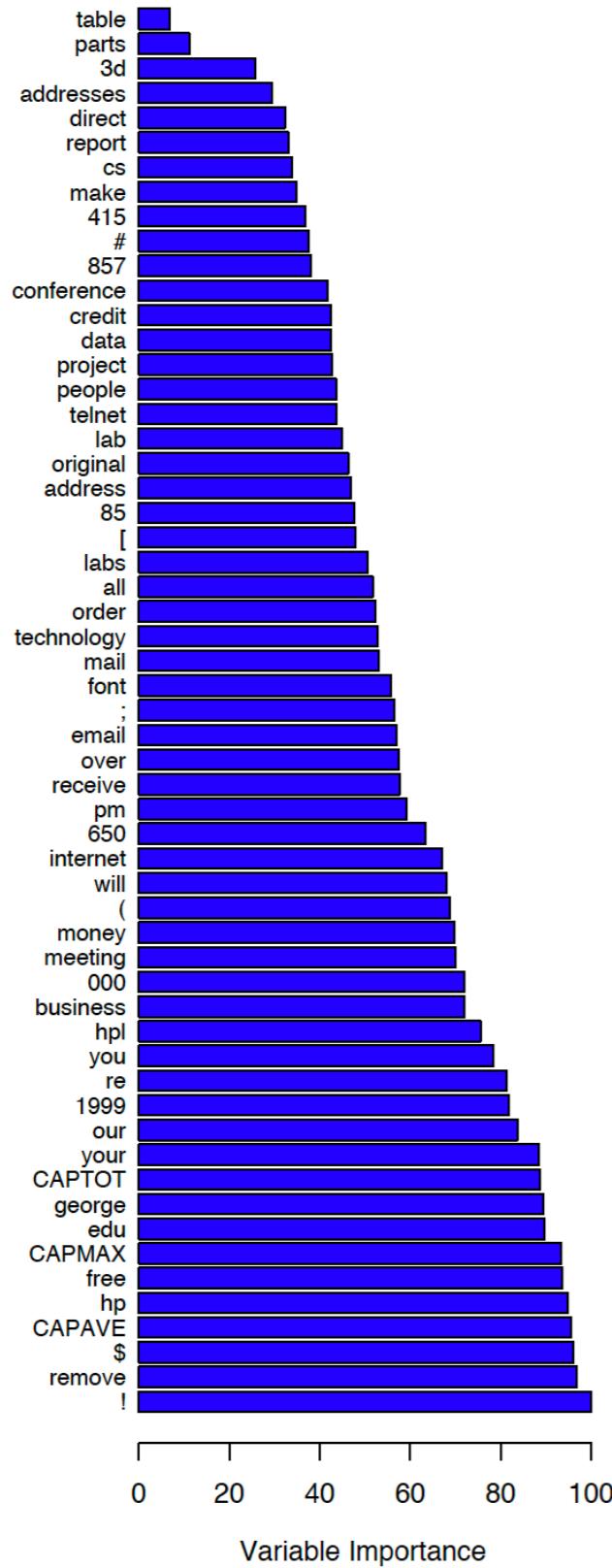
Randomization/Permutation - если перемешаем значения нашей переменной (она станет бесмысленной), то какое падение в качестве мы ожидаем



Первый вариант использования - считаем качество модели на ОOB. Далее по очереди перемешиваем значения каждой переменной в ОOB и оцениваем изменение качества модели

Выбор признаков

Randomization/Permutation - если перемешаем значения нашей переменной (она станет бесмысленной), то какое падение в качестве мы ожидаем

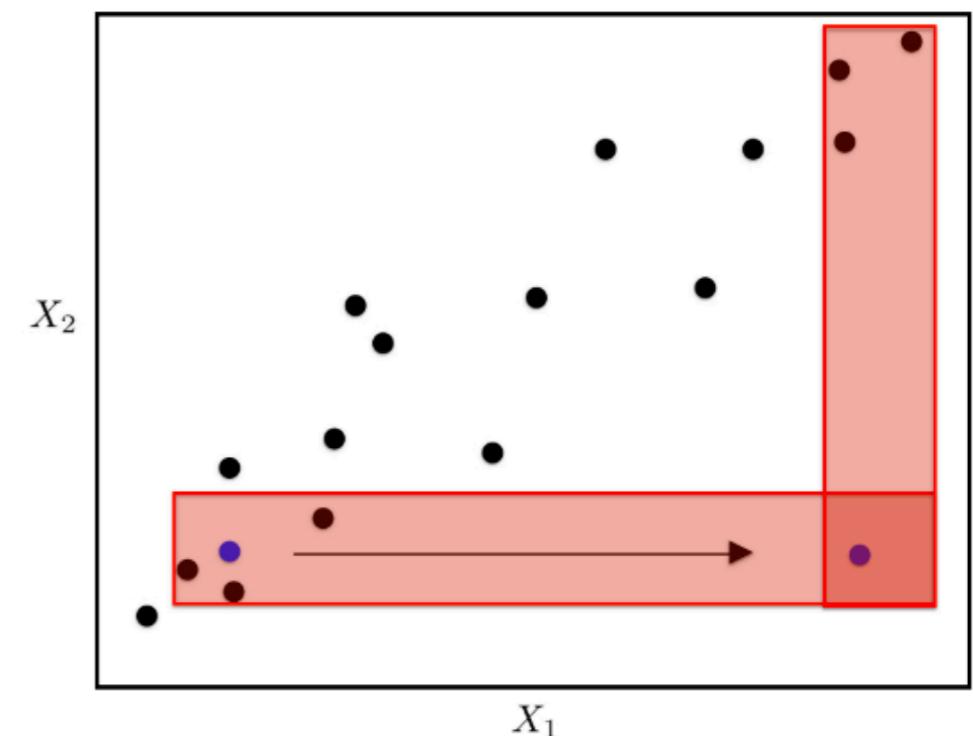
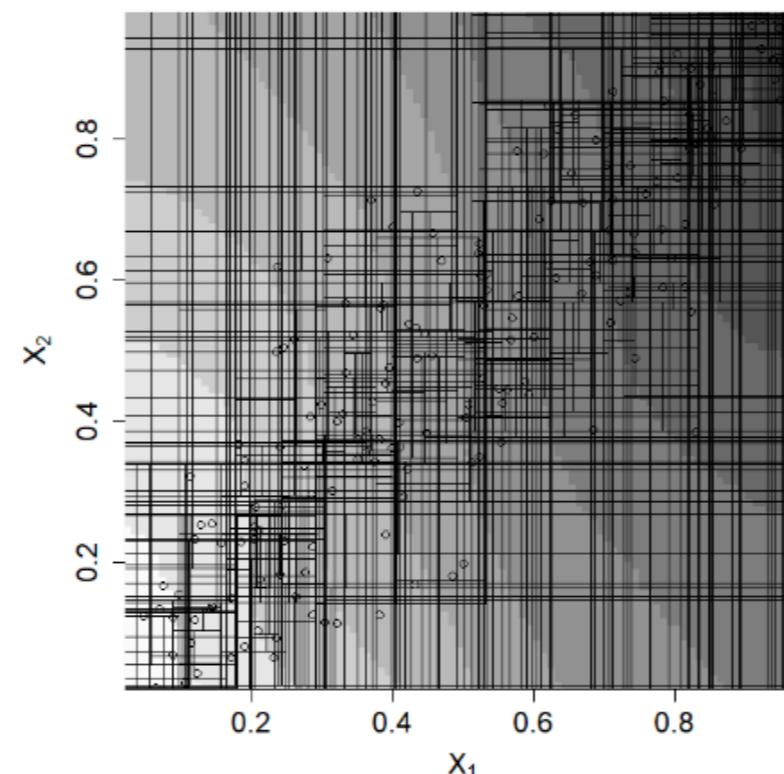
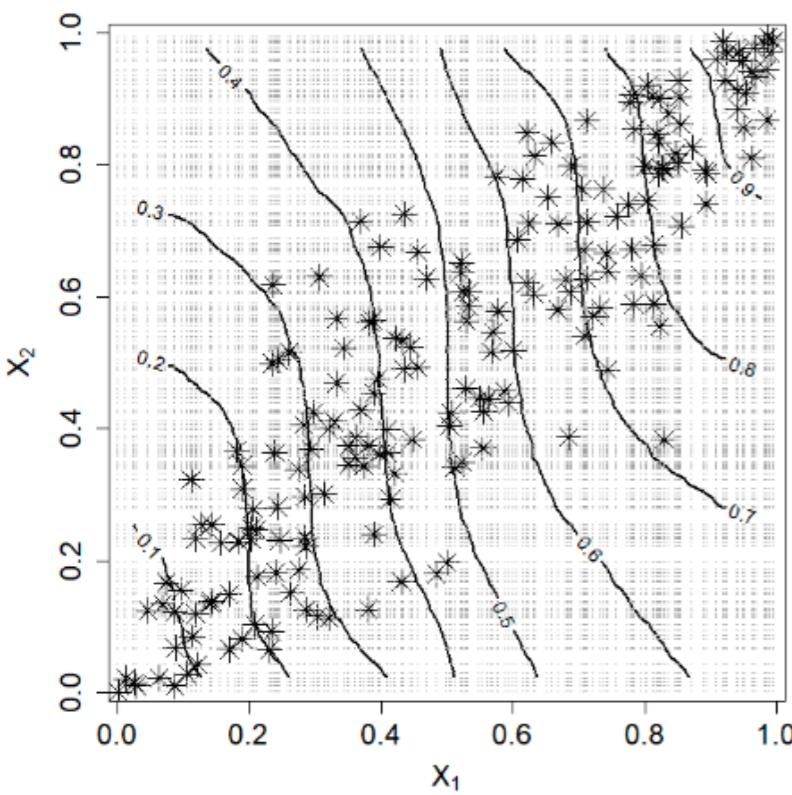


Первый вариант использования - считаем качество модели на ОOB. Далее по очереди перемешиваем значения каждой переменной в ОOB и оцениваем изменение качества модели

Так делать не стоит - <https://blog.ceshine.net/post/please-stop-permuting-features/>

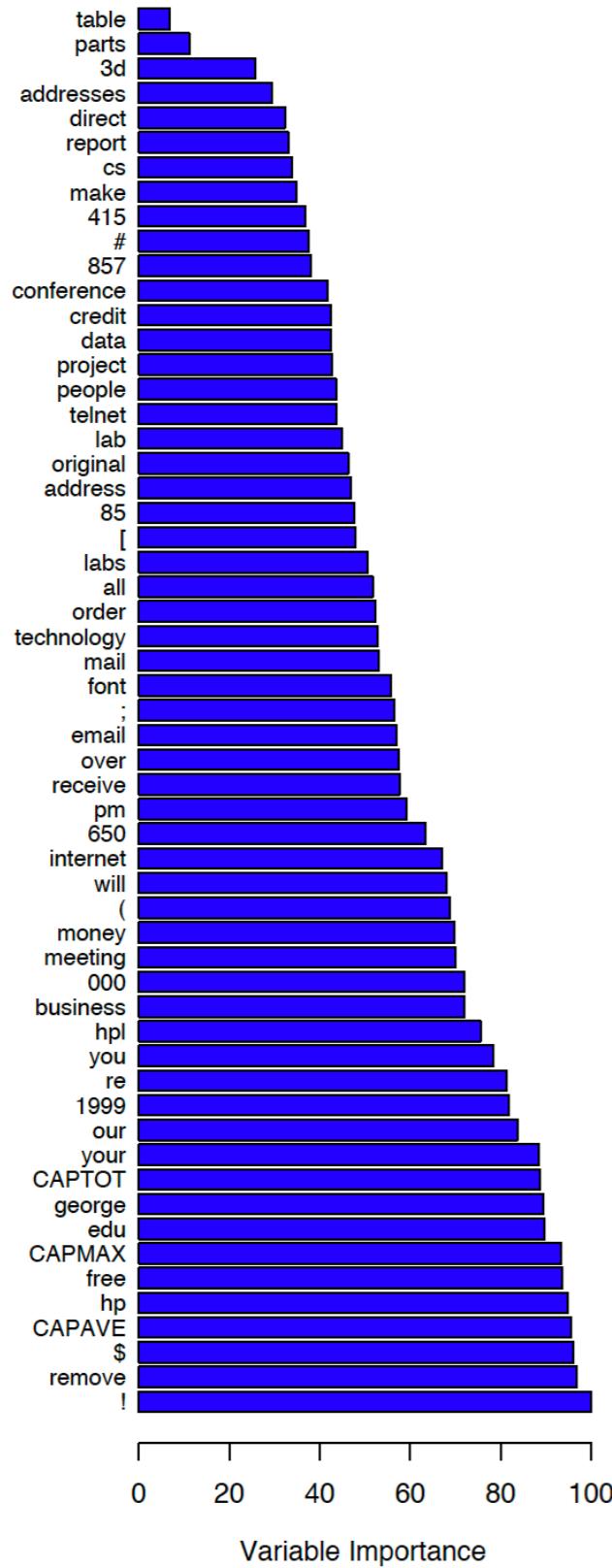
Приводит к завышению качества для коррелирующих признаков

Приводит к завышению качества для коррелирующих признаков.
Почему?



Выбор признаков

Randomization/Permutation - если перемешаем значения нашей переменной (она станет бесмысленной), то какое падение в качестве мы ожидаем

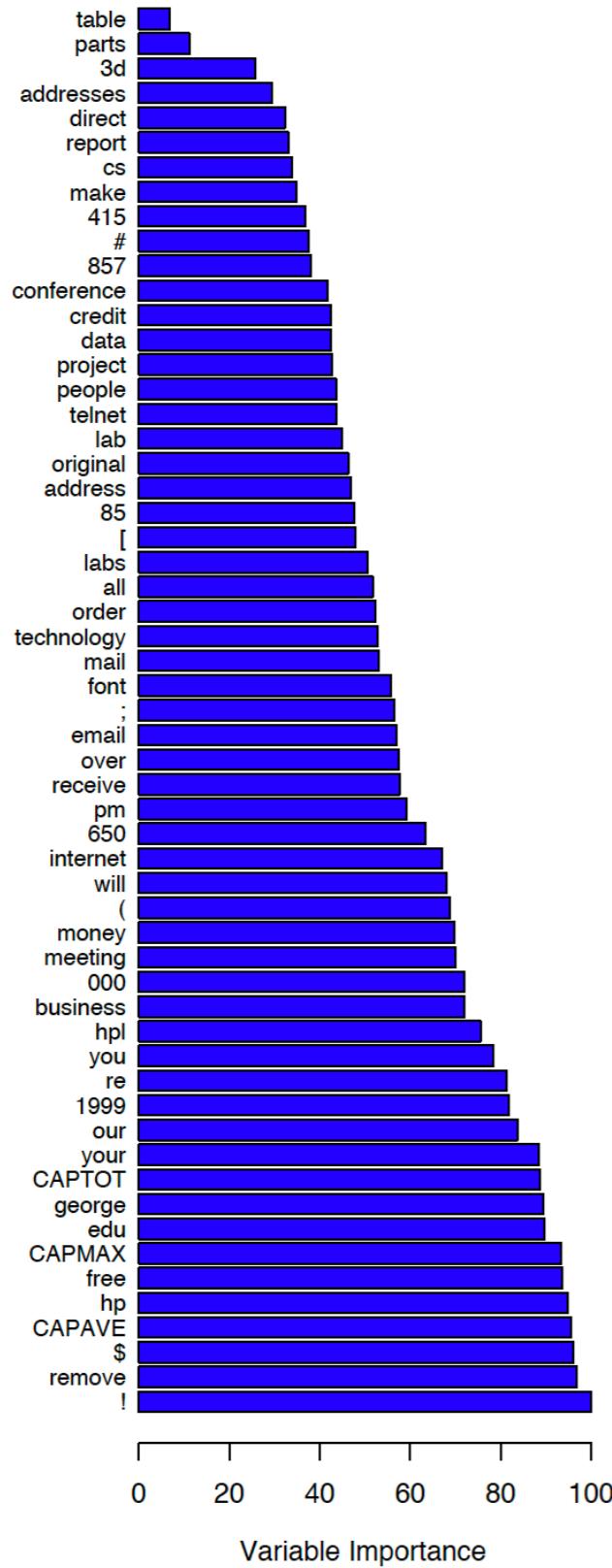


Второй вариант использования - считаем качество модели на ОOB. Но при перемешивании признака мы тренируем модель заново

Работает адекватно. Но: долго(

Выбор признаков

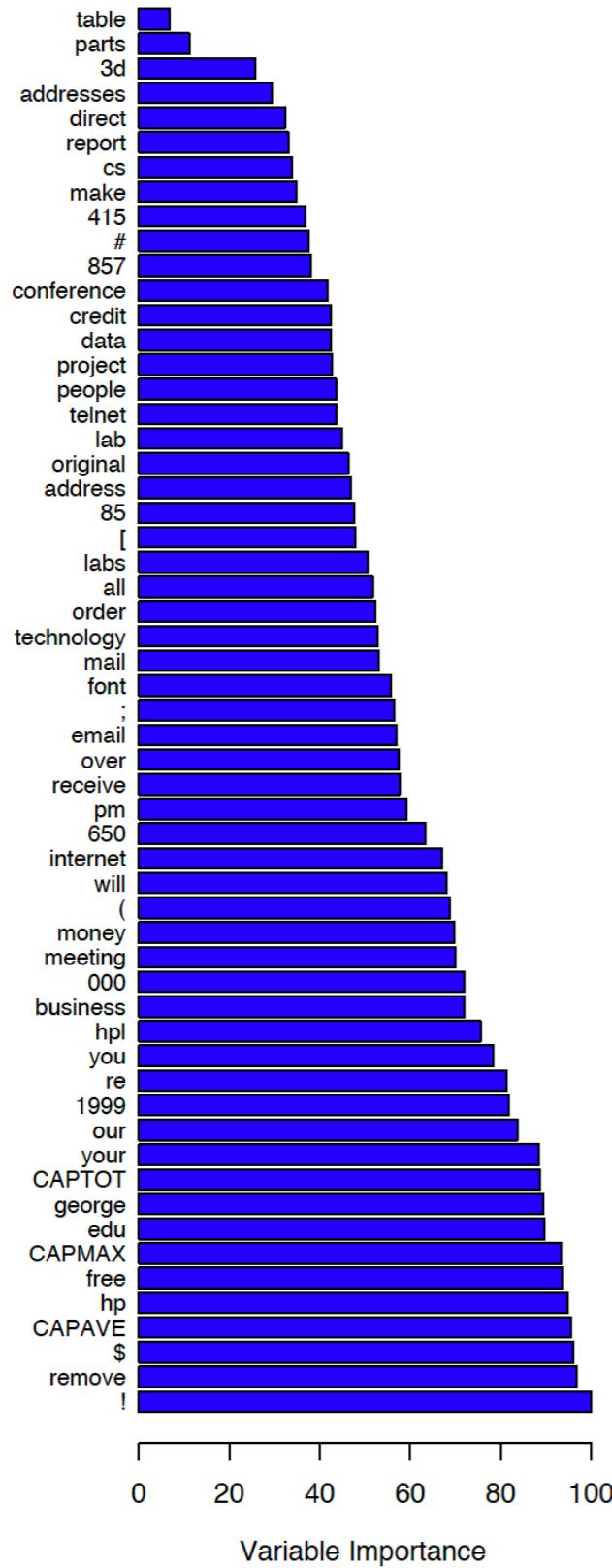
Randomization/Permutation - если перемешаем значения нашей переменной (она станет бесмысленной), то какое падение в качестве мы ожидаем



Третий вариант использования - считаем качество модели на ОOB. Но при перемешивании признака перемешиваем его, обуславливаясь на значения других признаков

Выбор признаков

Randomization/Permutation - если перемешаем значения нашей переменной (она станет бесмысленной), то какое падение в качестве мы ожидаем



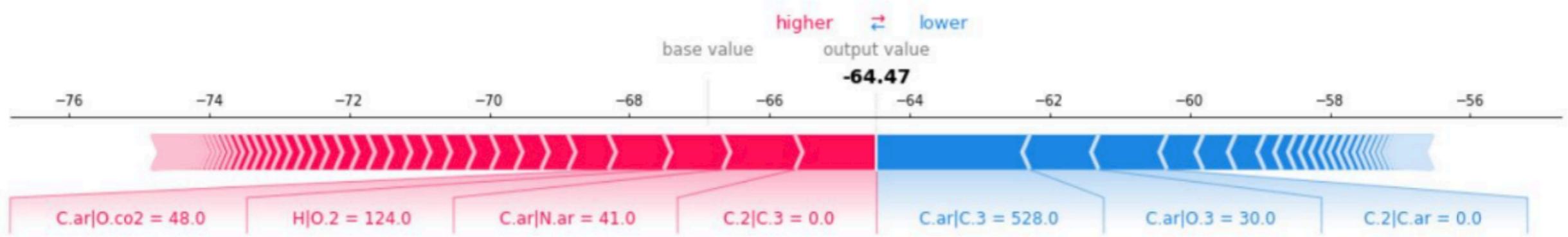
Четвертый вариант использования - считаем качество модели на ОOB. Но при перемешивании признака перемешиваем его, обуславливаясь на значения других признаков и переучиваем модель.

Выбор признаков

Dropped variable importance - если выкинуть переменную, как изменится качество модели

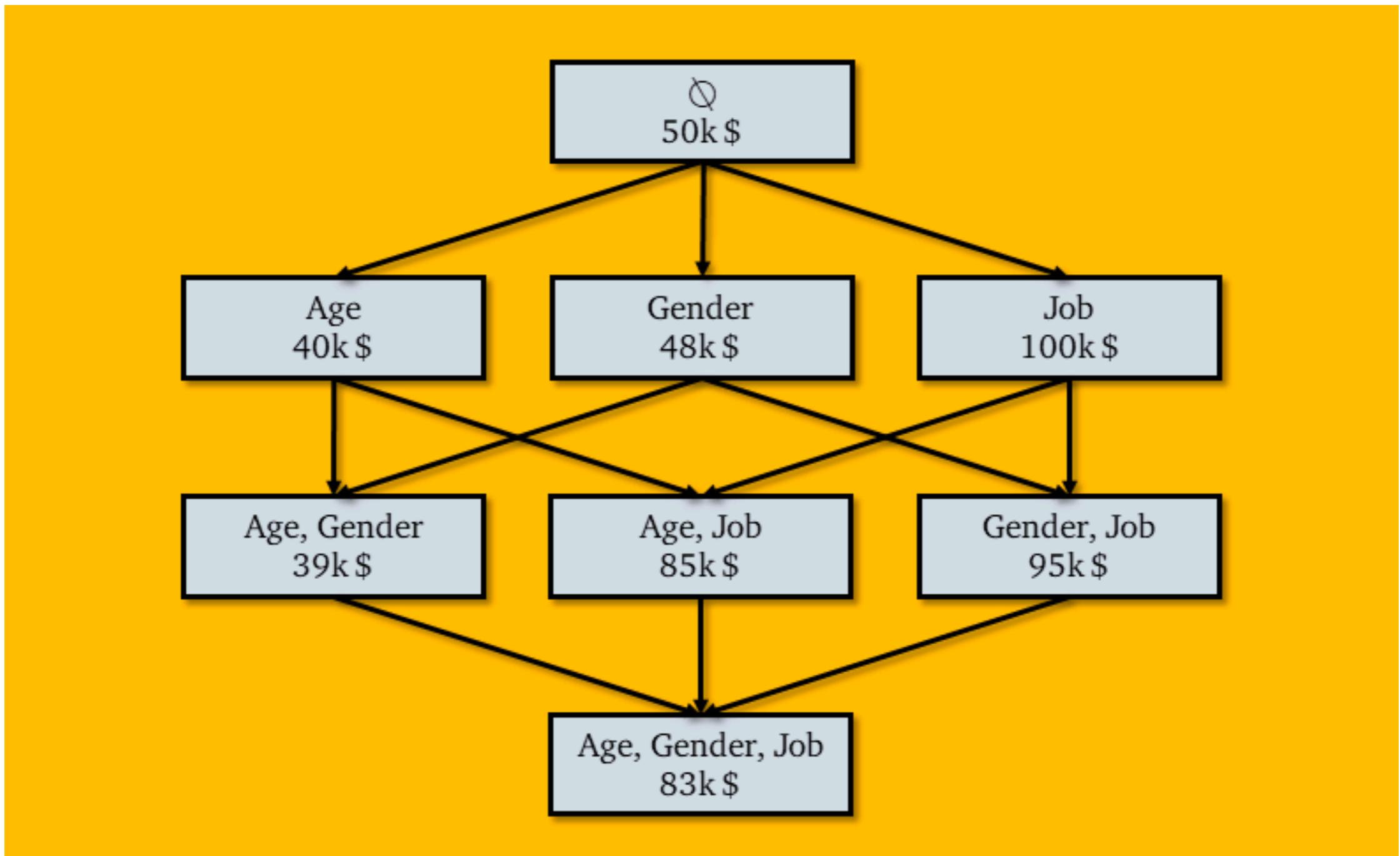
Тоже приводит к артефактам, и учить модели долго

SHAP



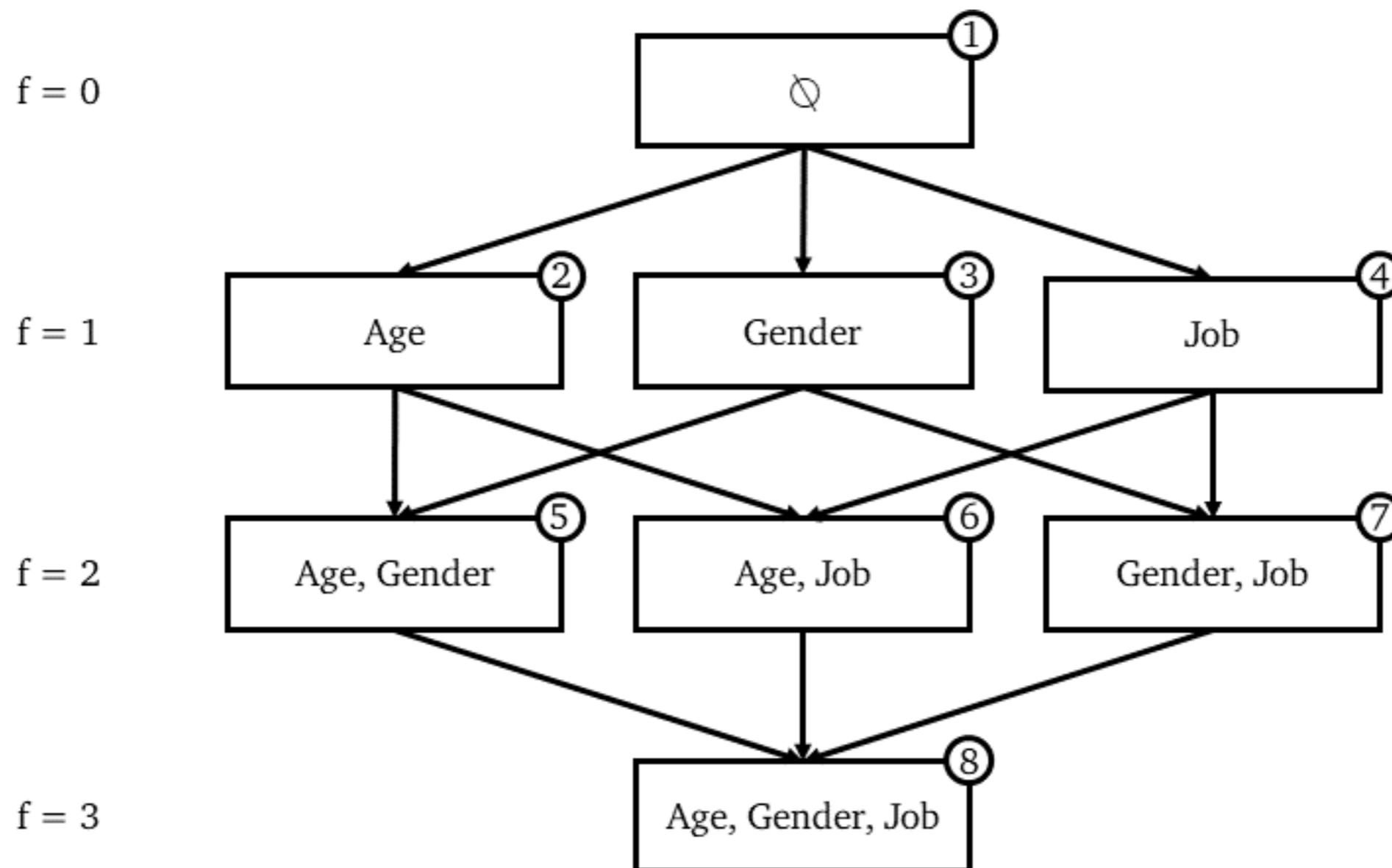
<https://github.com/slundberg/shap>

SHAP

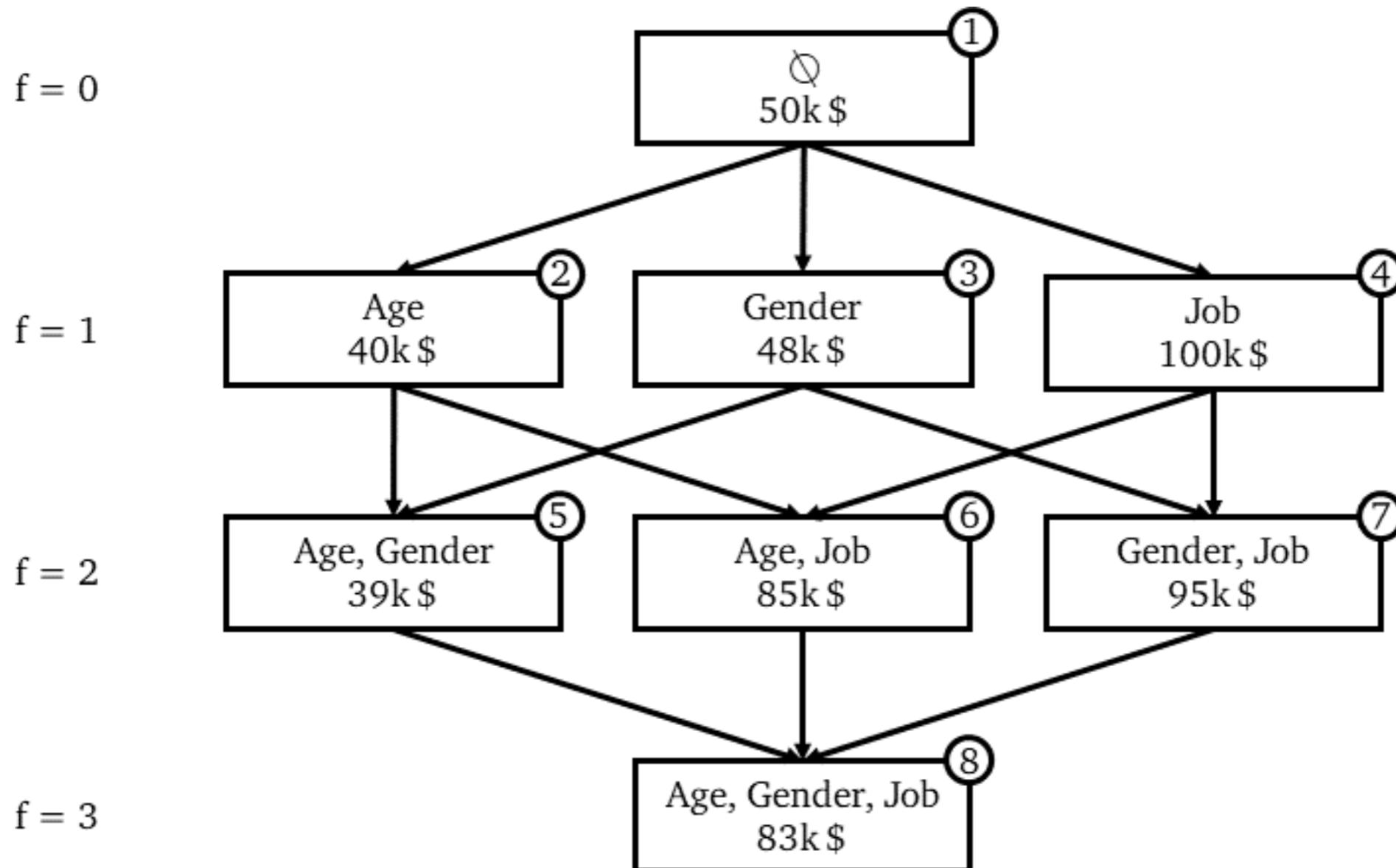


<https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30>

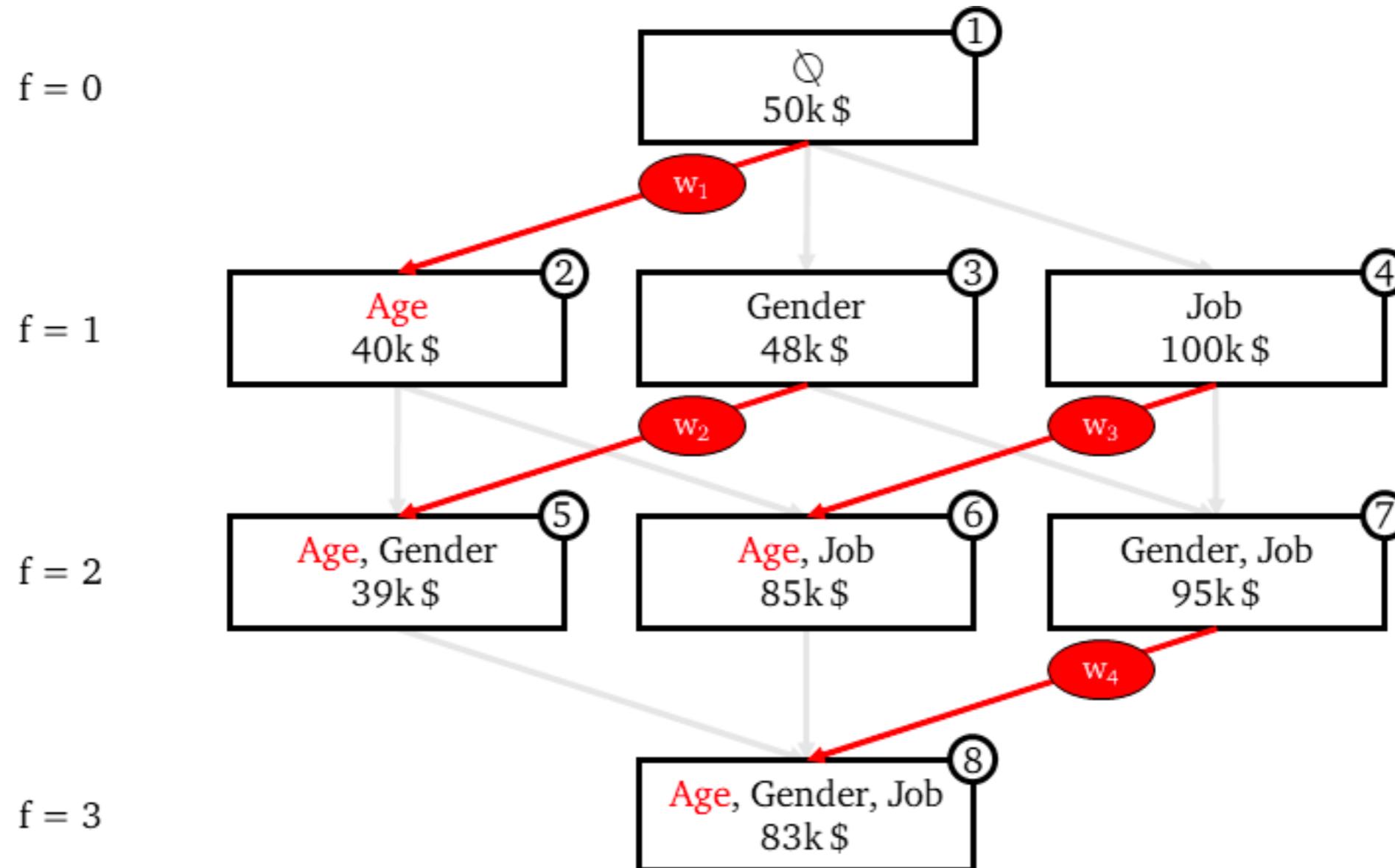
SHAP



SHAP

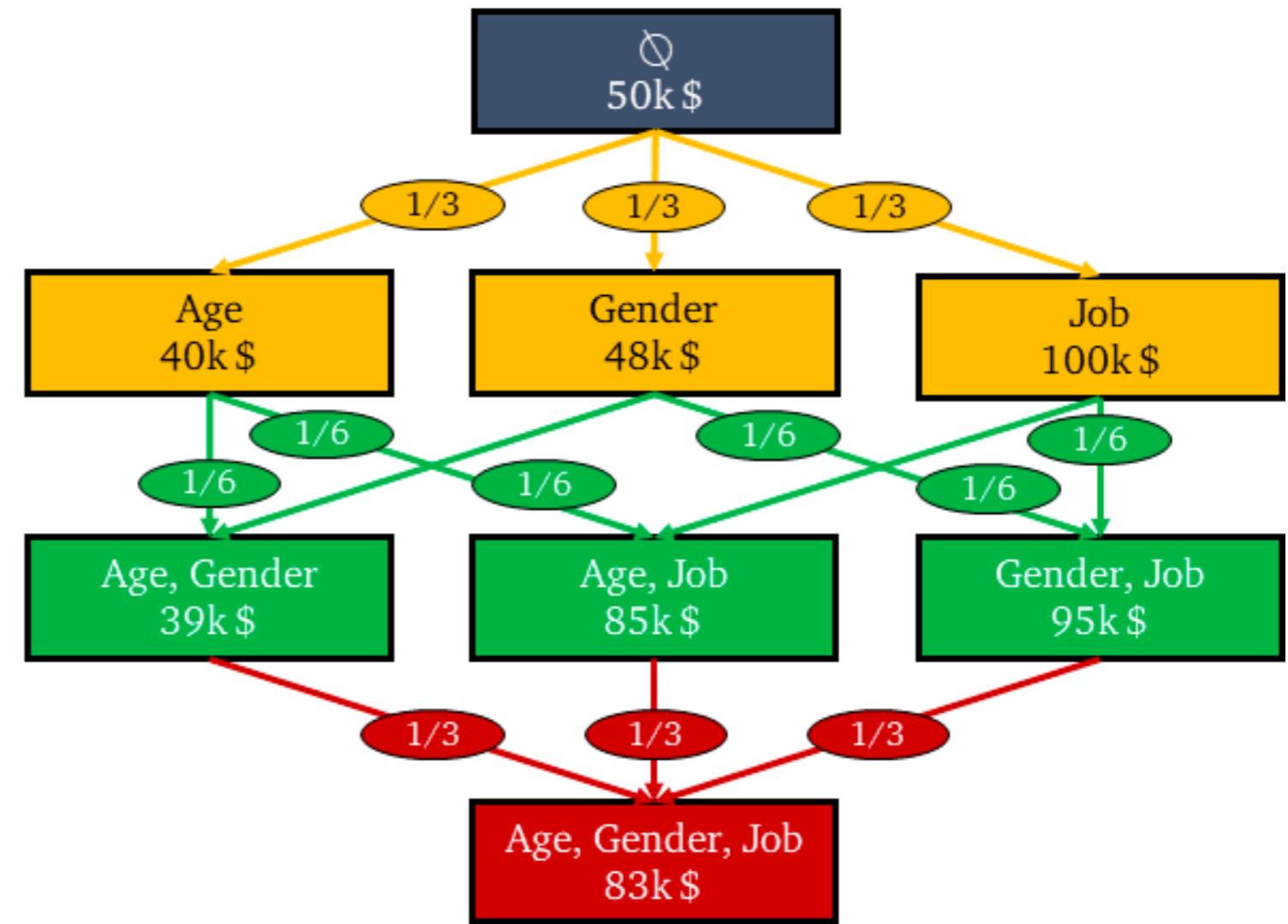


SHAP



SHAP

	N. of Nodes $\binom{F}{f}$	N. of Edges $f \times \binom{F}{f}$
$f = 0$	1	
$f = 1$	3	3
$f = 2$	3	6
$f = 3$	1	3
Sum	$2^F = 8$	$F \times 2^{F-1} = 12$



SHAP

$$\begin{aligned} SHAP_{Age}(x_0) &= [(1 \times \binom{3}{1})^{-1} \times MC_{Age, \{Age\}}(x_0) + \\ &\quad [(2 \times \binom{3}{2})^{-1} \times MC_{Age, \{Age, Gender\}}(x_0) + \\ &\quad [(2 \times \binom{3}{2})^{-1} \times MC_{Age, \{Age, Job\}}(x_0) + \\ &\quad [(3 \times \binom{3}{3})^{-1} \times MC_{Age, \{Age, Gender, Job\}}(x_0) + \\ &= \frac{1}{3} \times (-10k\$) + \frac{1}{6} \times (-9k\$) + \frac{1}{6} \times (-15k\$) + \frac{1}{3} \times (-12k\$) \\ &= -11.33k\$ \end{aligned}$$

Как подсчитать значимость (чтобы, собственно, выбрать признаки)?

Bootstrap

Изначальные объекты выборки

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----

Bootstrap-выборки

13	9	10	14	15	15	13	6	8	10	14	6	7	7	15
----	---	----	----	----	----	----	---	---	----	----	---	---	---	----

11	4	7	1	7	9	6	7	11	7	2	3	5	12	11
----	---	---	---	---	---	---	---	----	---	---	---	---	----	----

...

14	4	13	1	13	5	8	9	5	14	11	5	3	7	6
----	---	----	---	----	---	---	---	---	----	----	---	---	---	---

Bootstrap

Для каждой bootstrap-выборки оцениваем значения коэффициентов, строим распределение для каждого коэффициента.

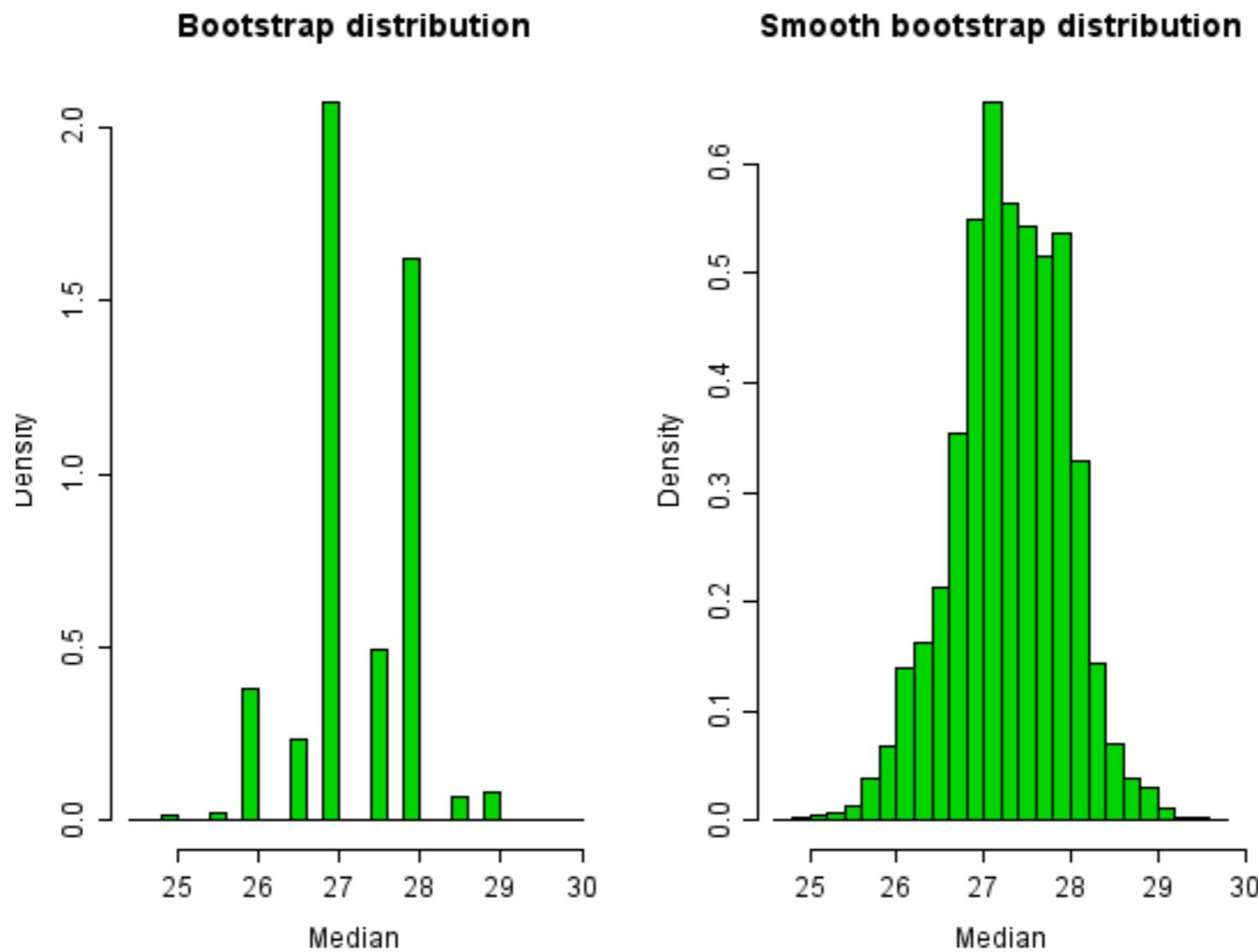
Bootstrap-выборки

13	9	10	14	15	15	13	6	8	10	14	6	7	7	15
----	---	----	----	----	----	----	---	---	----	----	---	---	---	----



Значения коэффициентов при признаках для данной bootstrap-выборки

Bootstrap



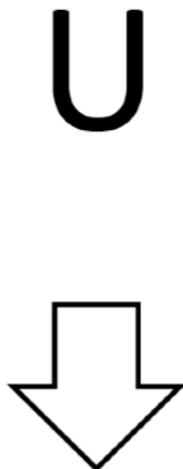
Boruta

Датасет с исходными признаками

	Gene 1	Gene 2	Gene 3
Sample1	1	12	4
Sample2	2	15	3
Sample3	1	13	17
Sample4	3	18	89

Датасет и теневыми признаками

	Gene_s 1	Gene_s 2	Gene_s 3
Sample1	3	18	17
Sample2	1	12	4
Sample3	2	15	3
Sample4	1	13	89



	Gene 1	Gene_s 1	Gene 2	Gene_s 2	Gene 3	Gene_s 3
Sample1	1	3	12	18	4	17
Sample2	2	1	15	12	3	4
Sample3	1	2	13	15	17	3
Sample4	3	1	18	13	89	89

Boruta

Данные с замешанными теневыми признаками



Обучаем модель

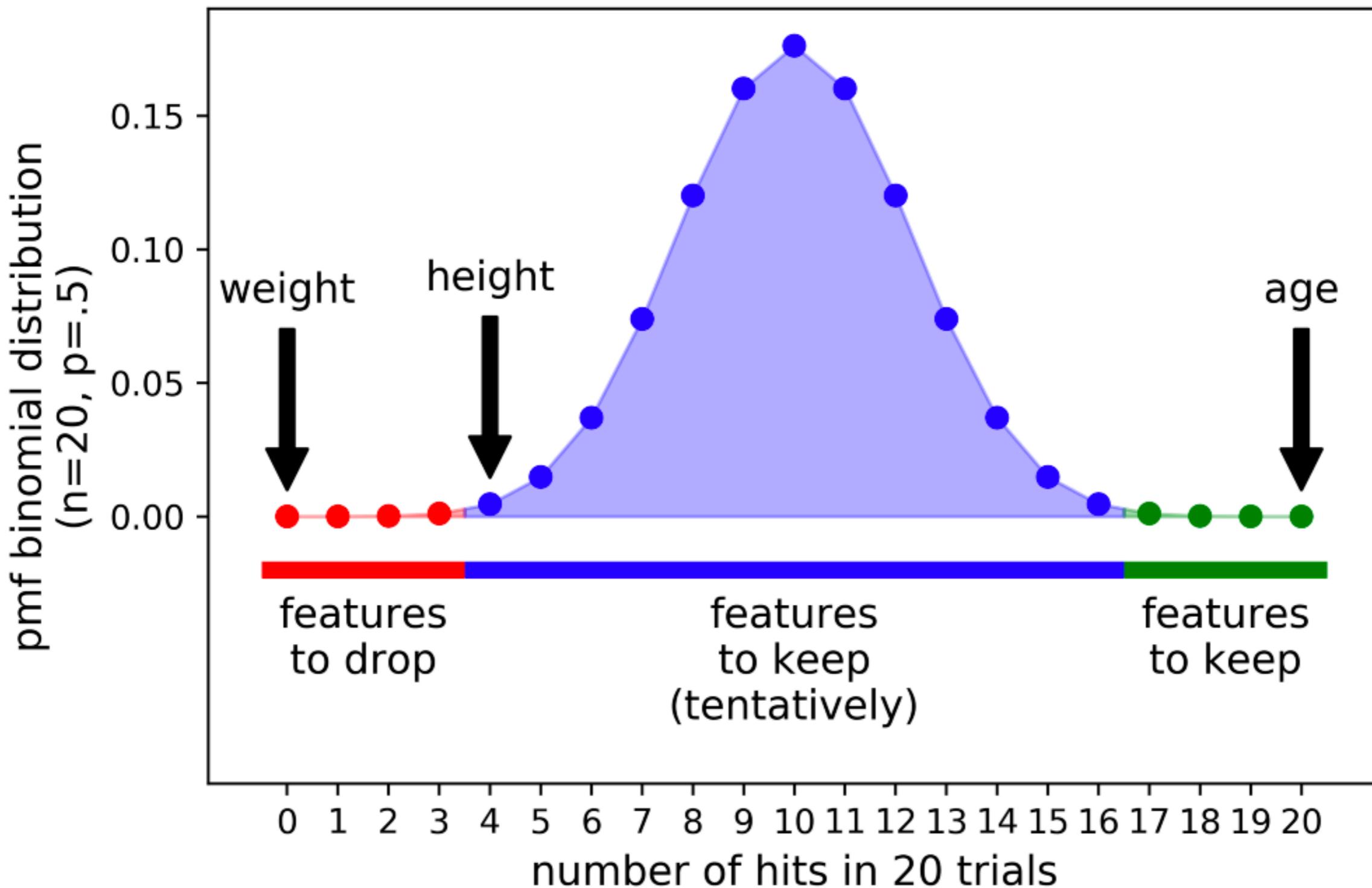


Отбираем только те признаки, которые по качеству лучше всех теневых

Повторить это несколько раз

Boruta

Самая большая неопределенность относительно брать/не брать признак - это
 $p=0.5$



Взвешенное голосование

$$h(x) = \sum_i b_i a_i(x)$$



Bagging vs boosting

Стрельба в тире



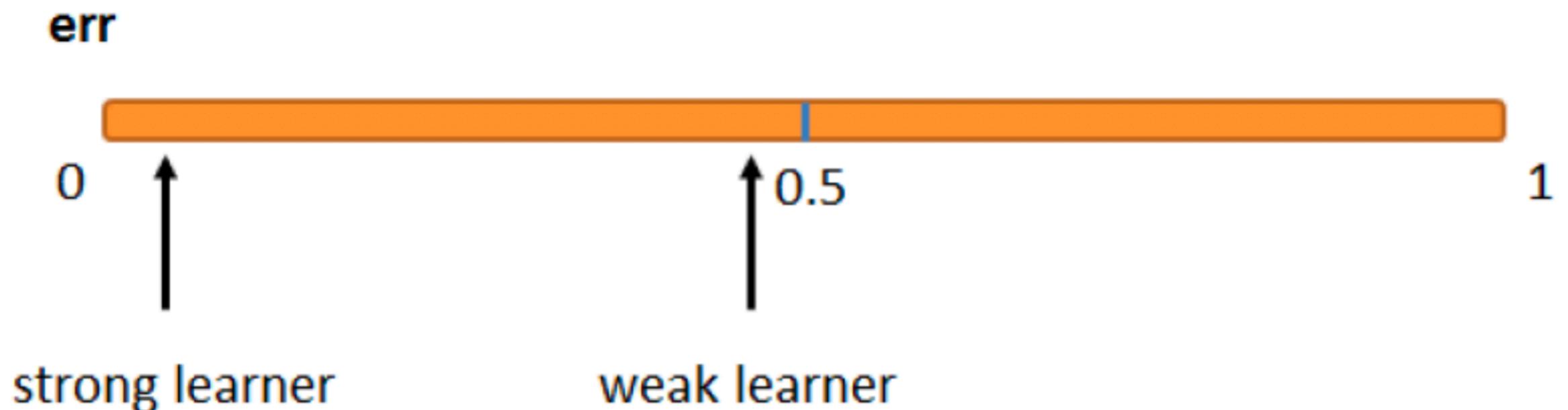
Игра в гольф



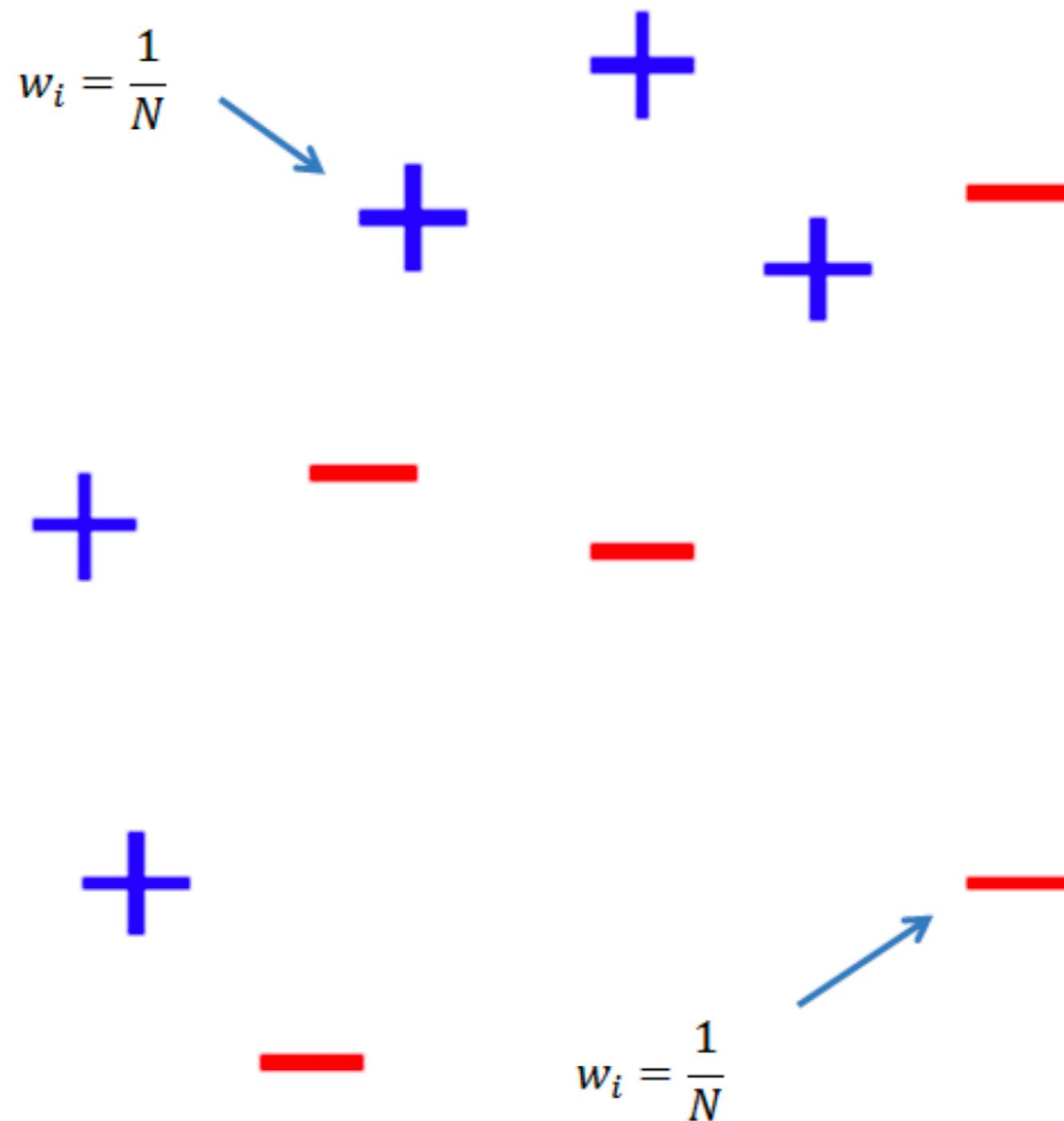
Взято из презентации В. Гулина, Техносфера

Boosting

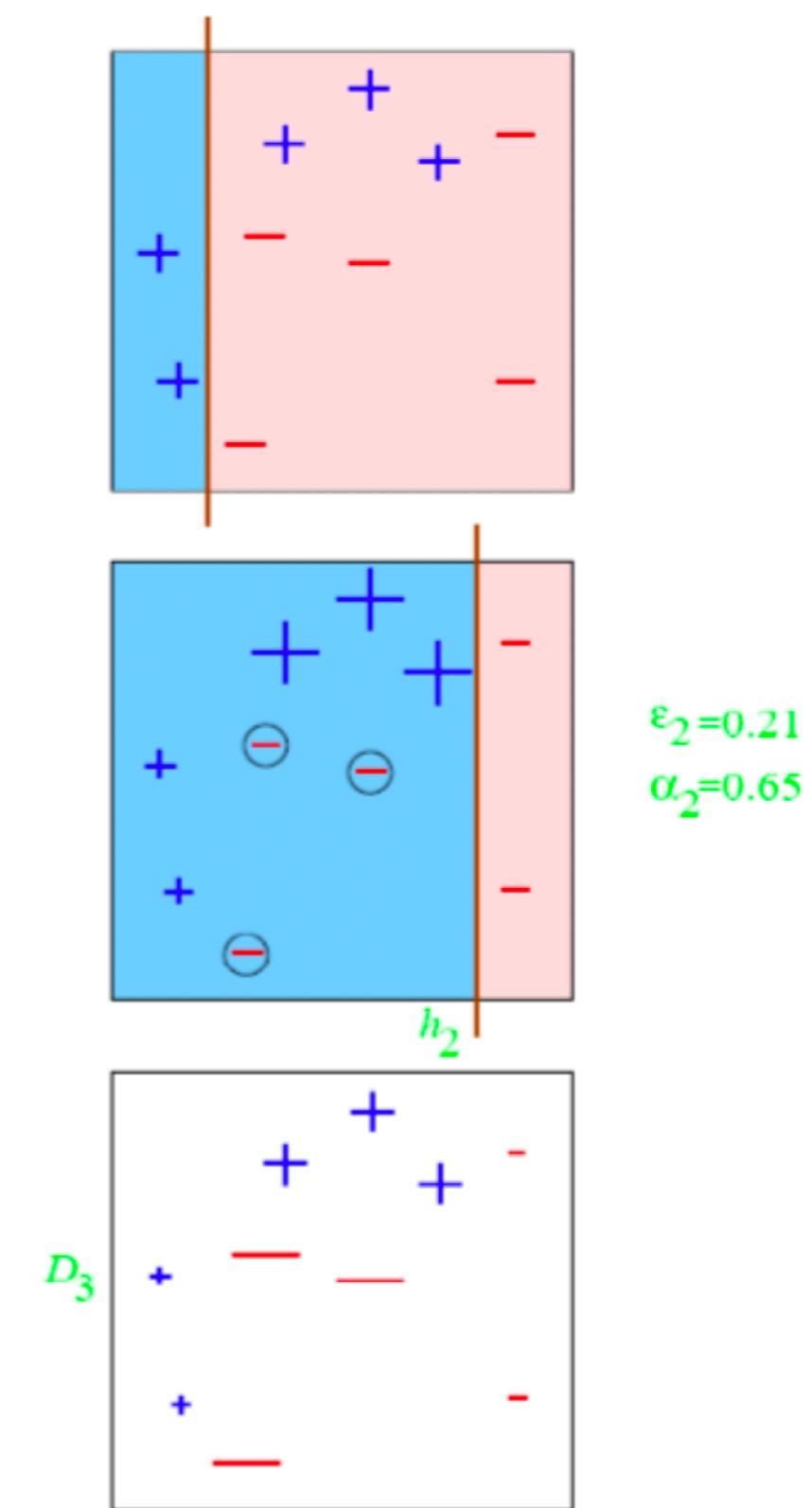
Возможно ли построение сильного предсказателя из набора “слабых” моделей - моделей, качество которых недалеко от случайного?



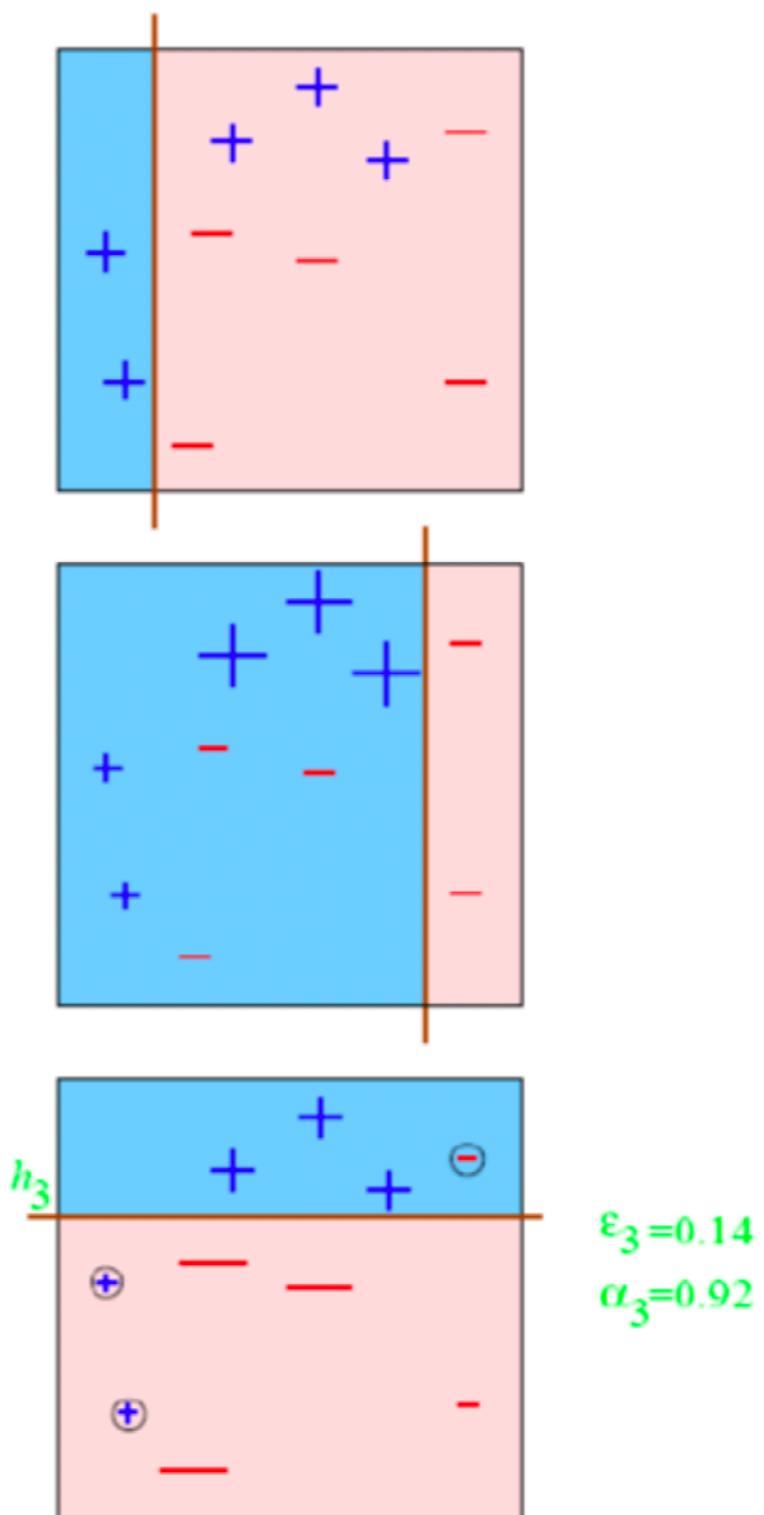
AdaBoost (Freund & Schapire, 1996)



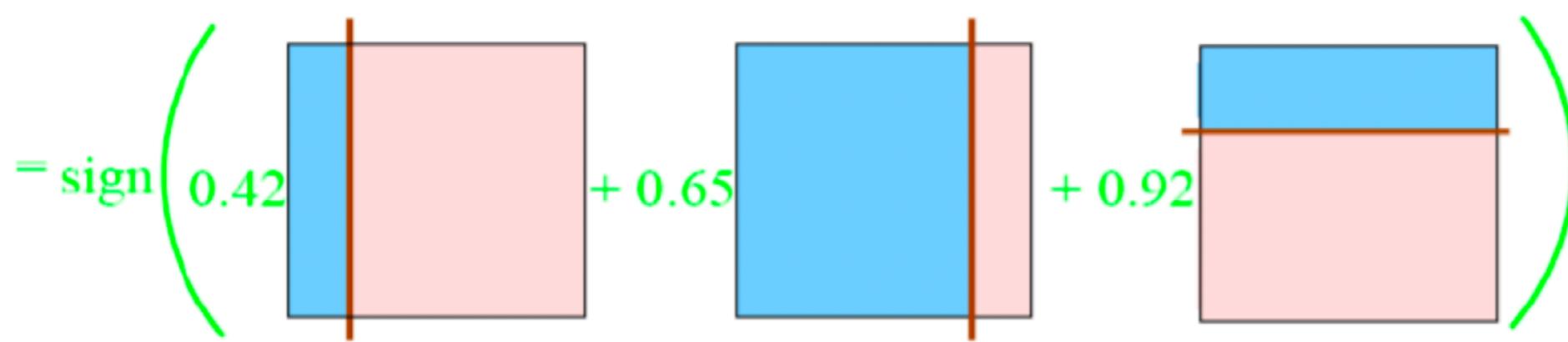
AdaBoost

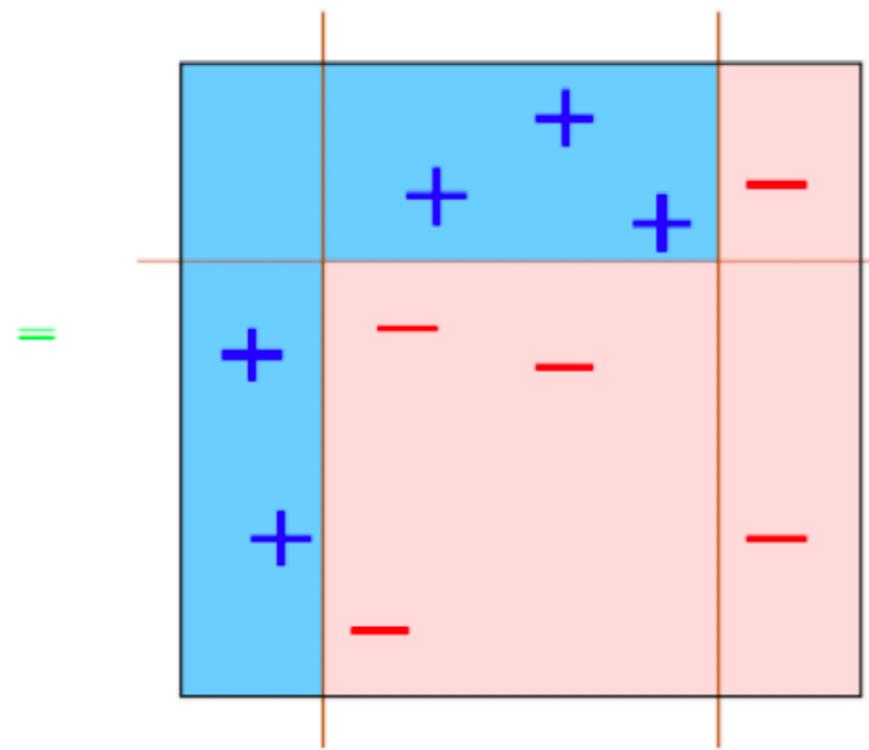


AdaBoost

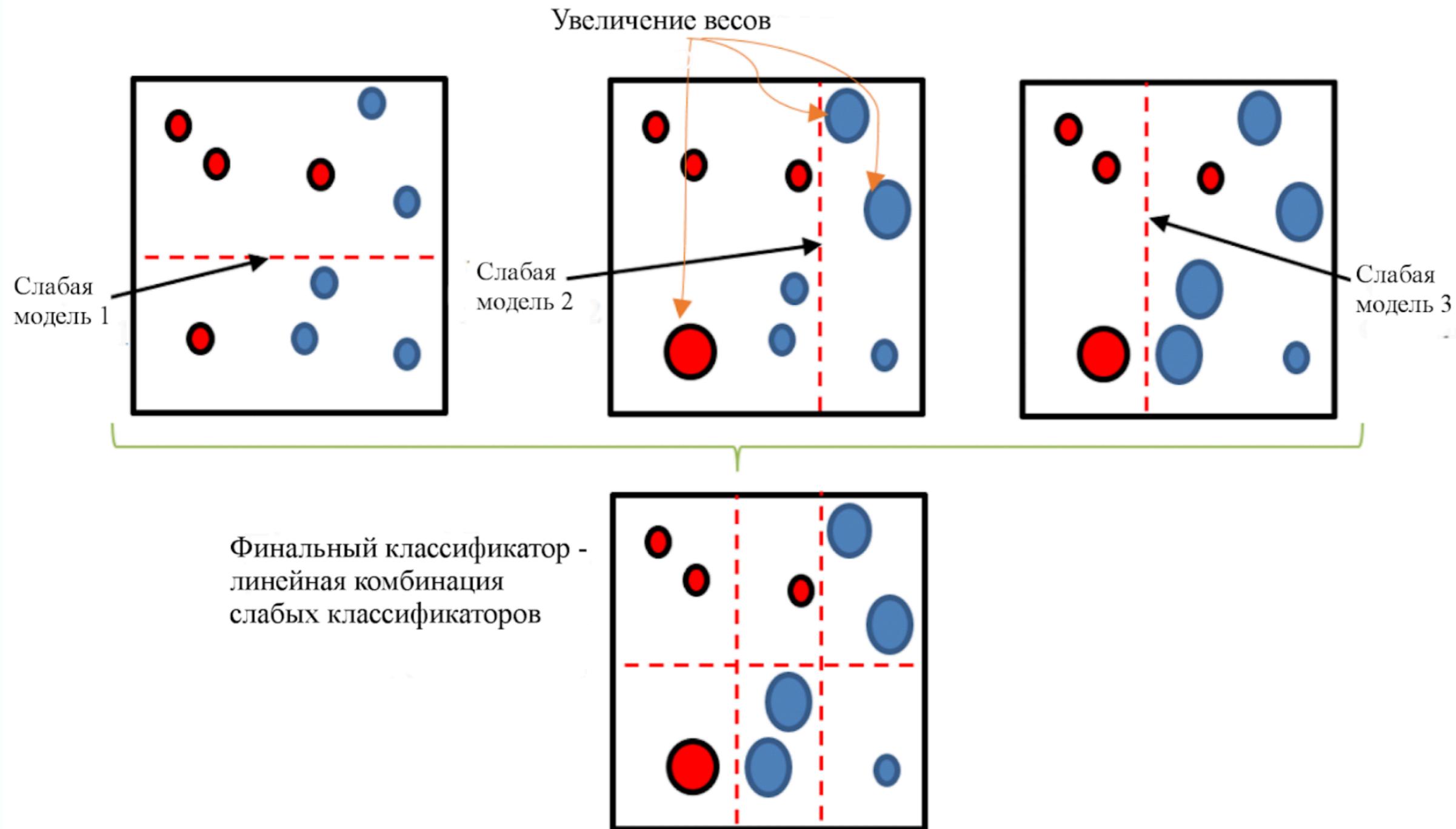


AdaBoost

$$= \text{sign} \left(0.42 + 0.65 + 0.92 \right)$$




AdaBoost



Zerrouki et al, 2018

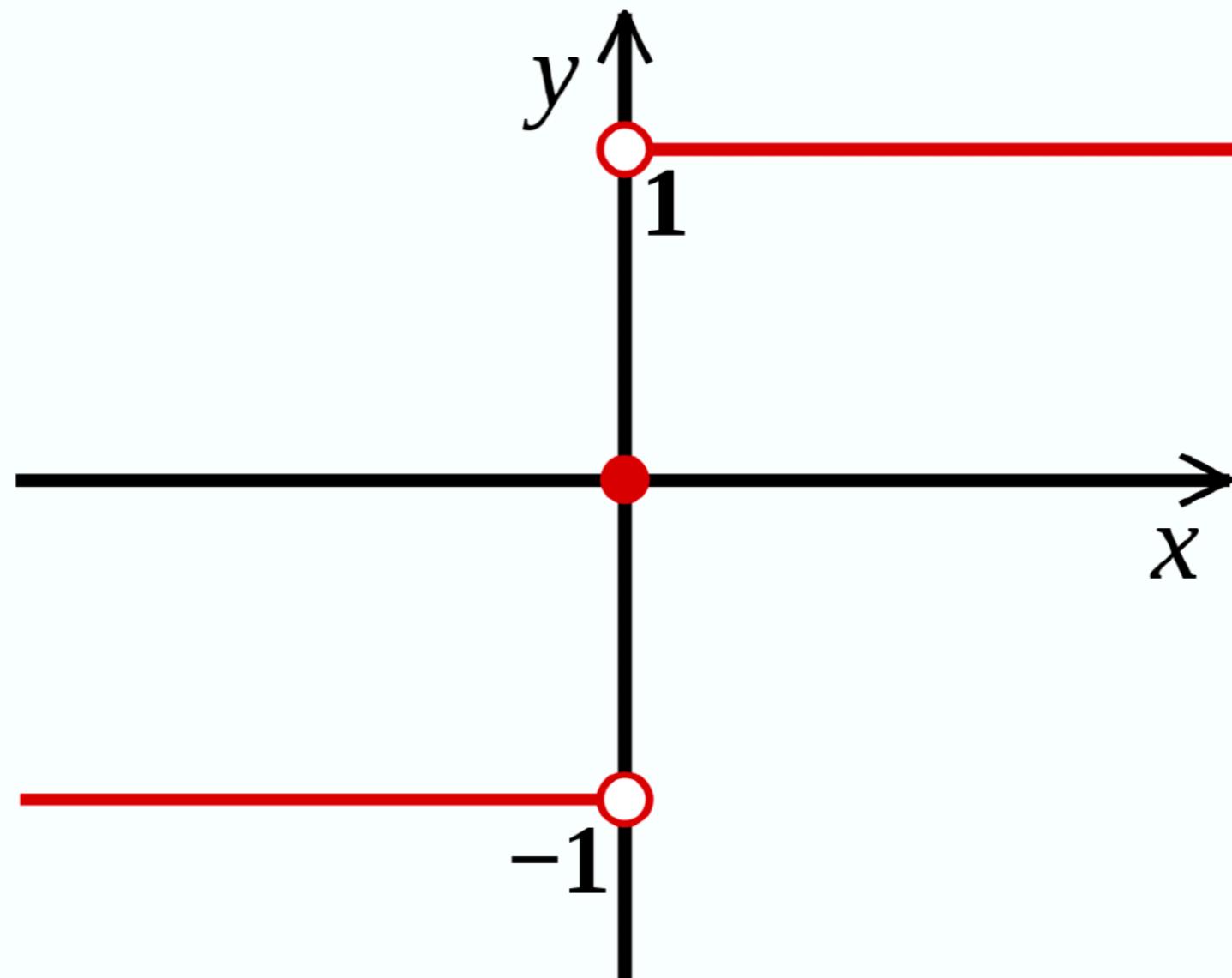
Функция для классификации

Если нас волнует только классификация, то какая функция будет самой правильной?

Функция для классификации

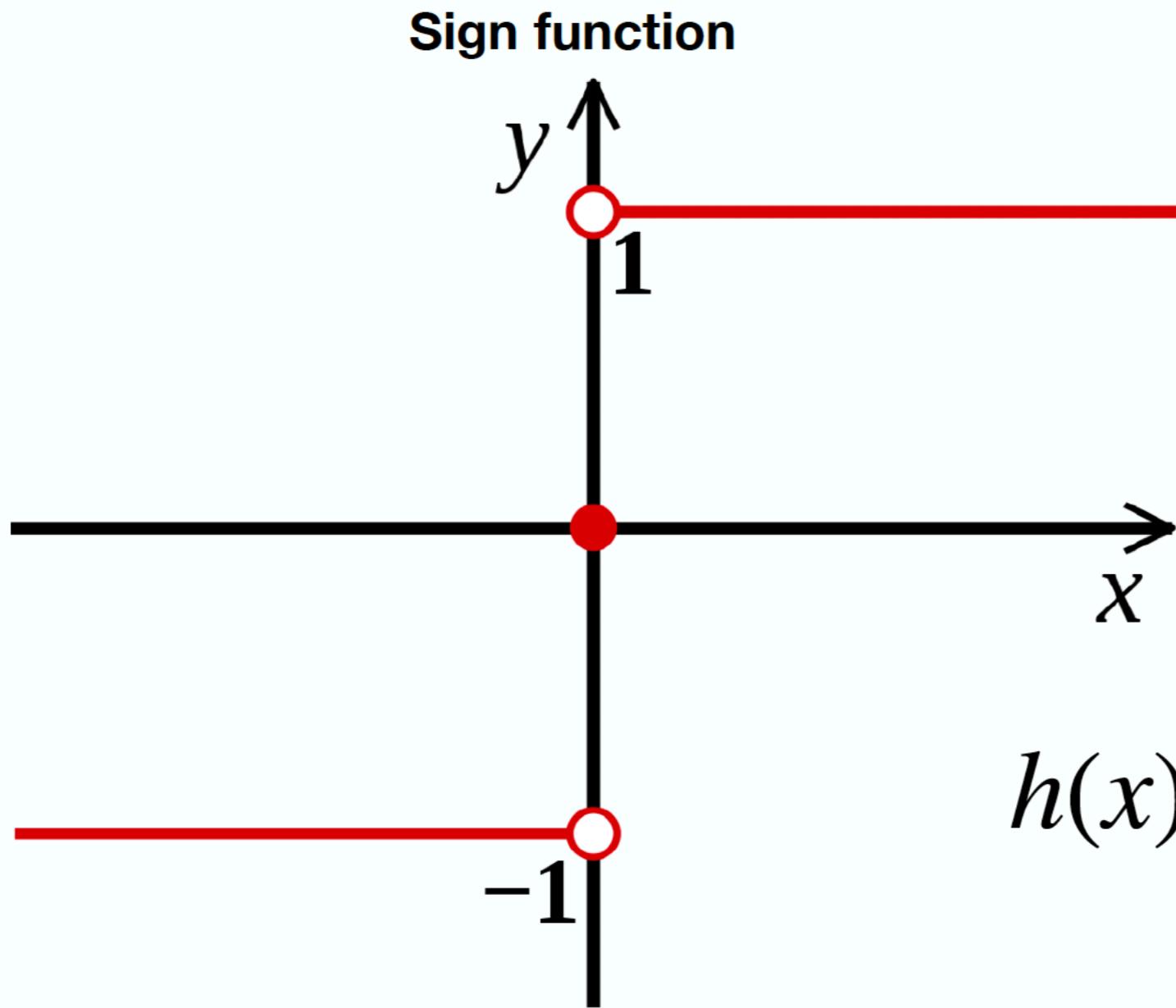
Если нас волнует только классификация, то какая функция будет самой правильной?

Sign function



Функция для классификации

Если нас волнует только классификация, то какая функция будет самой правильной?



$$h(x) = \text{sign}\left(\sum_i b_i a_i(x)\right)$$

AdaBoost

Algorithm 14.1 AdaBoost.M1.

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \dots, N$.
 2. For $m = 1$ to M :
 - (a) Fit a classifier $G_m(x)$ to the training data using weights w_i .
 - (b) Compute
$$\text{err}_m = \frac{\sum_{i=1}^N w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^N w_i}.$$
 - (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
 - (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \dots, N$.
 3. Output $G(x) = \text{sign} \left[\sum_{m=1}^M \alpha_m G_m(x) \right]$.
-

**После каждого шага можно нормировать веса,
чтобы они в сумме давали 1**

На каждом шаге j:

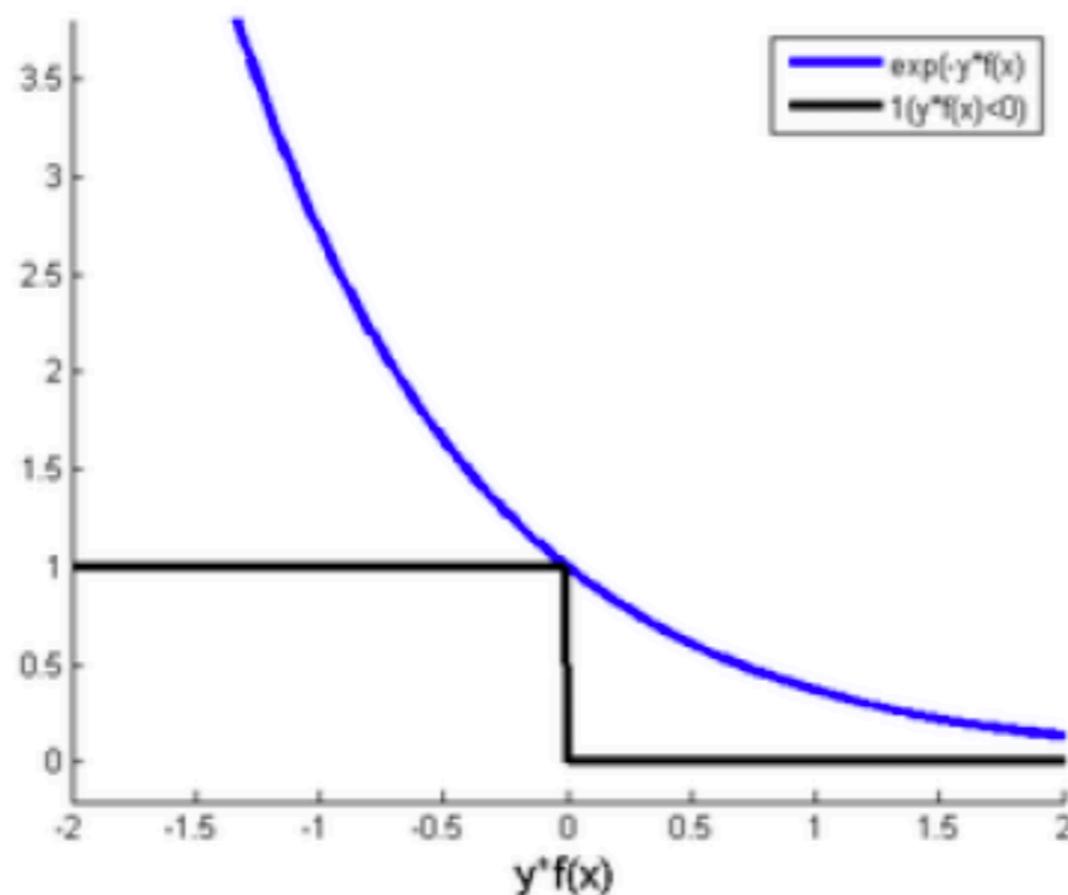
- 1) Если объект правильно классифицирован, его вес не меняется
- 2) Если объект классифицирован неверно, то возможны два случая:
 - a) наш классификатор плохой - предсказывает не лучше монетки ($L_j = 0.5$). Тогда вес объекта не меняется
 - b) Иначе вес объекта увеличивается

$$w_i = \frac{w_i}{\sum w_j}$$

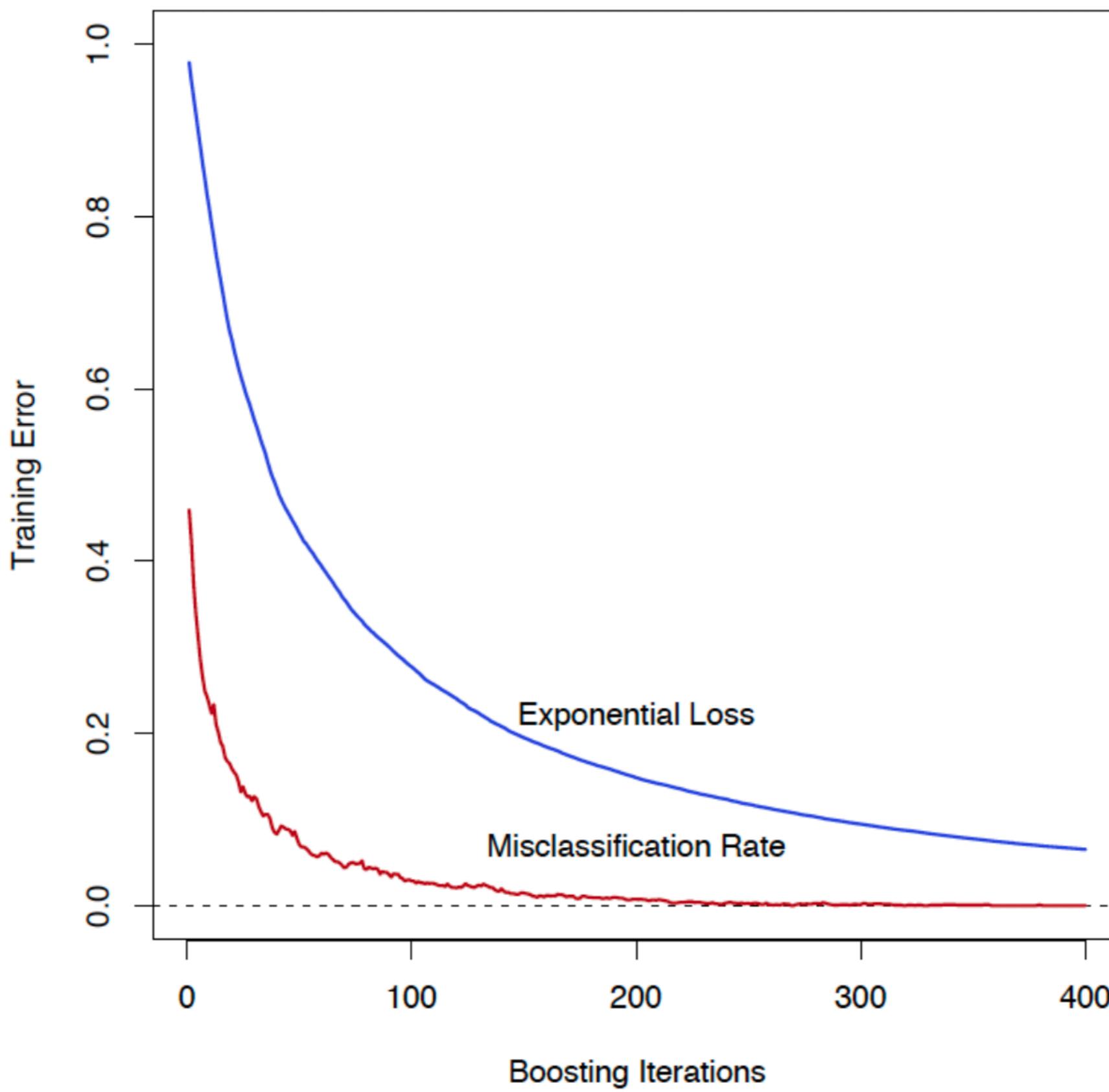
AdaBoost

- Почему экспонента?

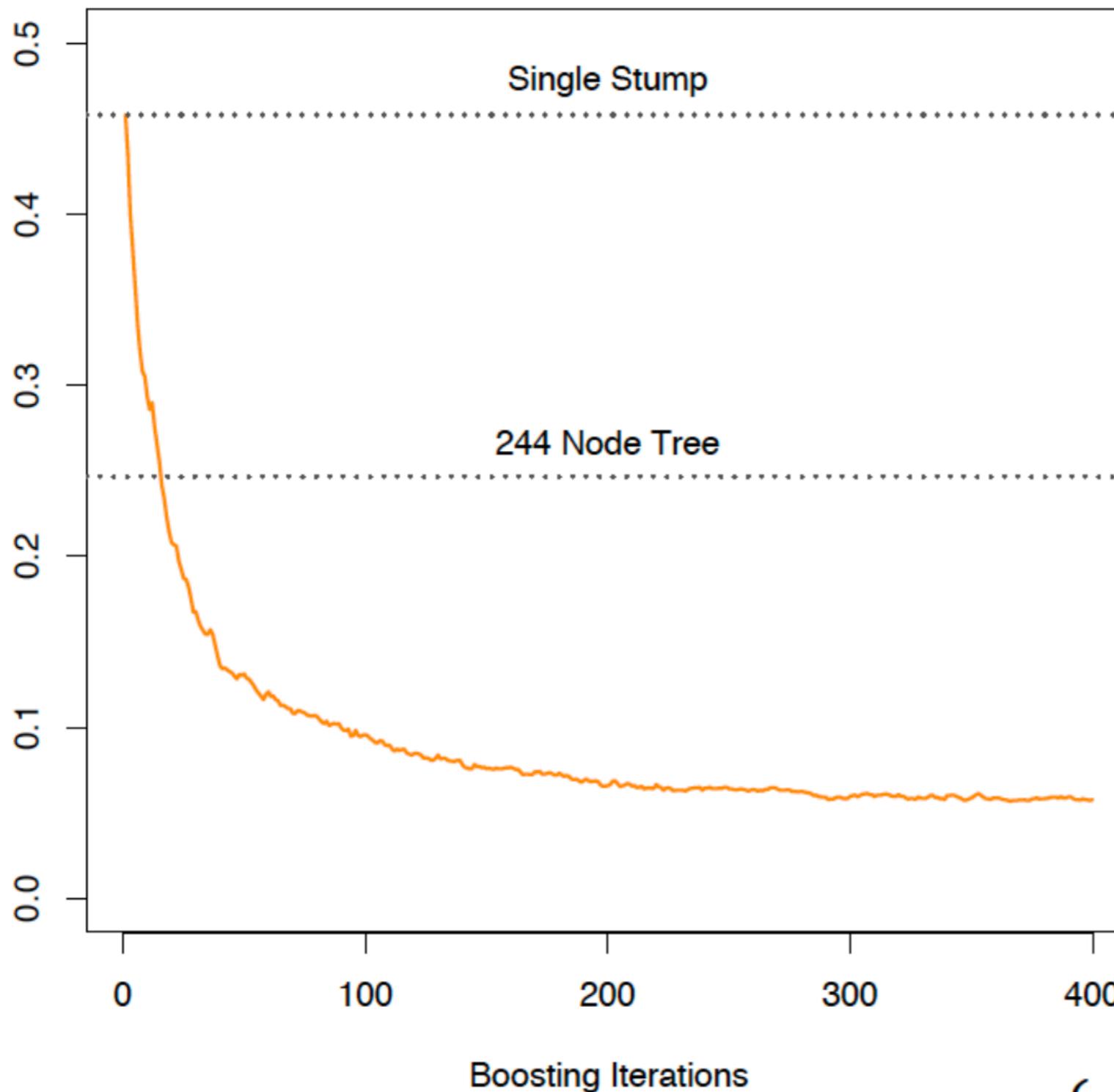
$$I(y \neq a(x)) = I(y \cdot a(x) \leq 0) \leq e^{-y \cdot a(x)}$$



AdaBoost



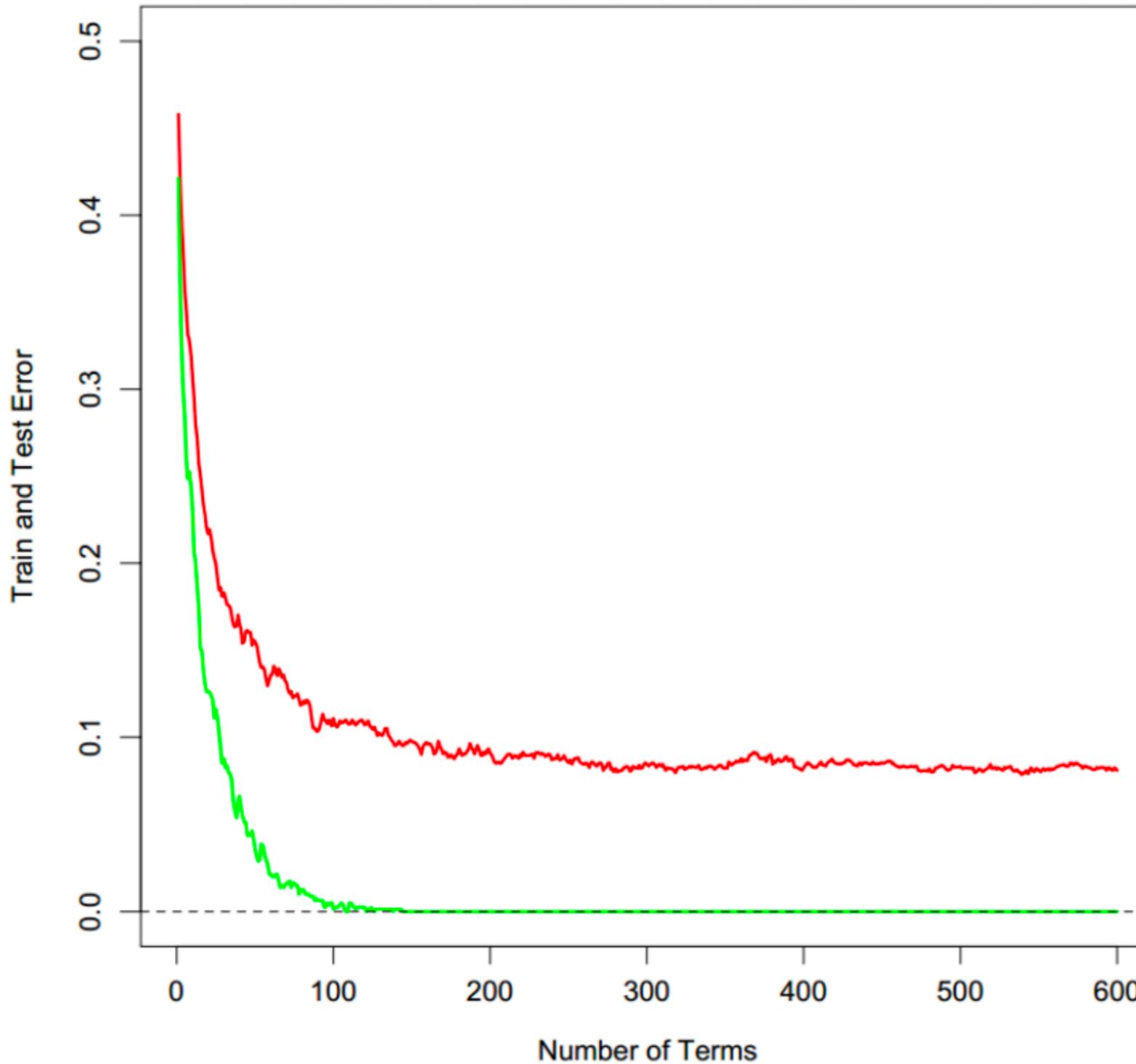
AdaBoost



Boosting Iterations

$$Y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > \chi^2_{10}(0.5), \\ -1 & \text{otherwise.} \end{cases}$$

AdaBoost



**“Не переобучается” -
переобучается,
деревьев да фичей
побольше**

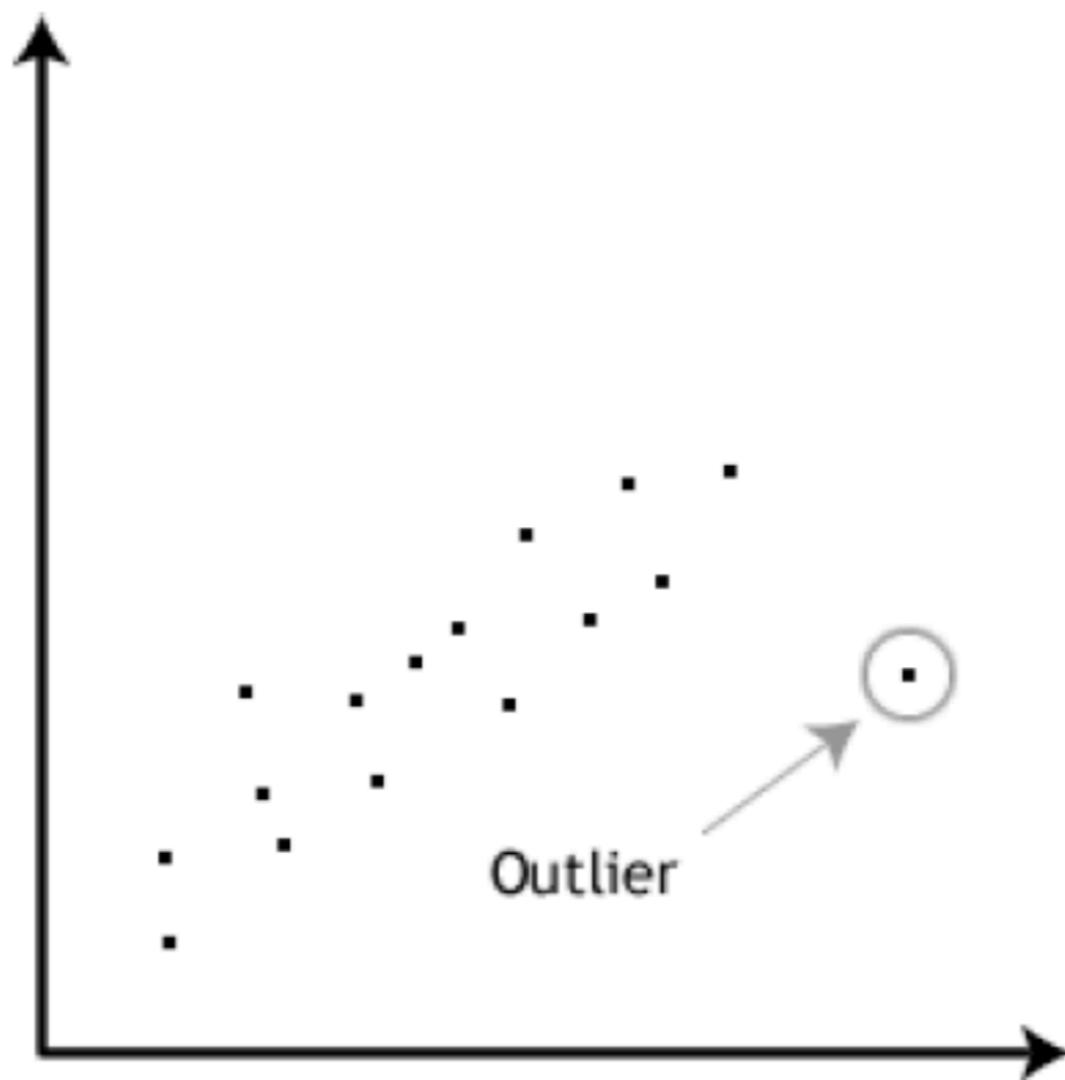
**Ошибка на обучении
может достигнуть
минимума, но при этом
ошибка на teste
продолжает
уменьшаться**

AdaBoost

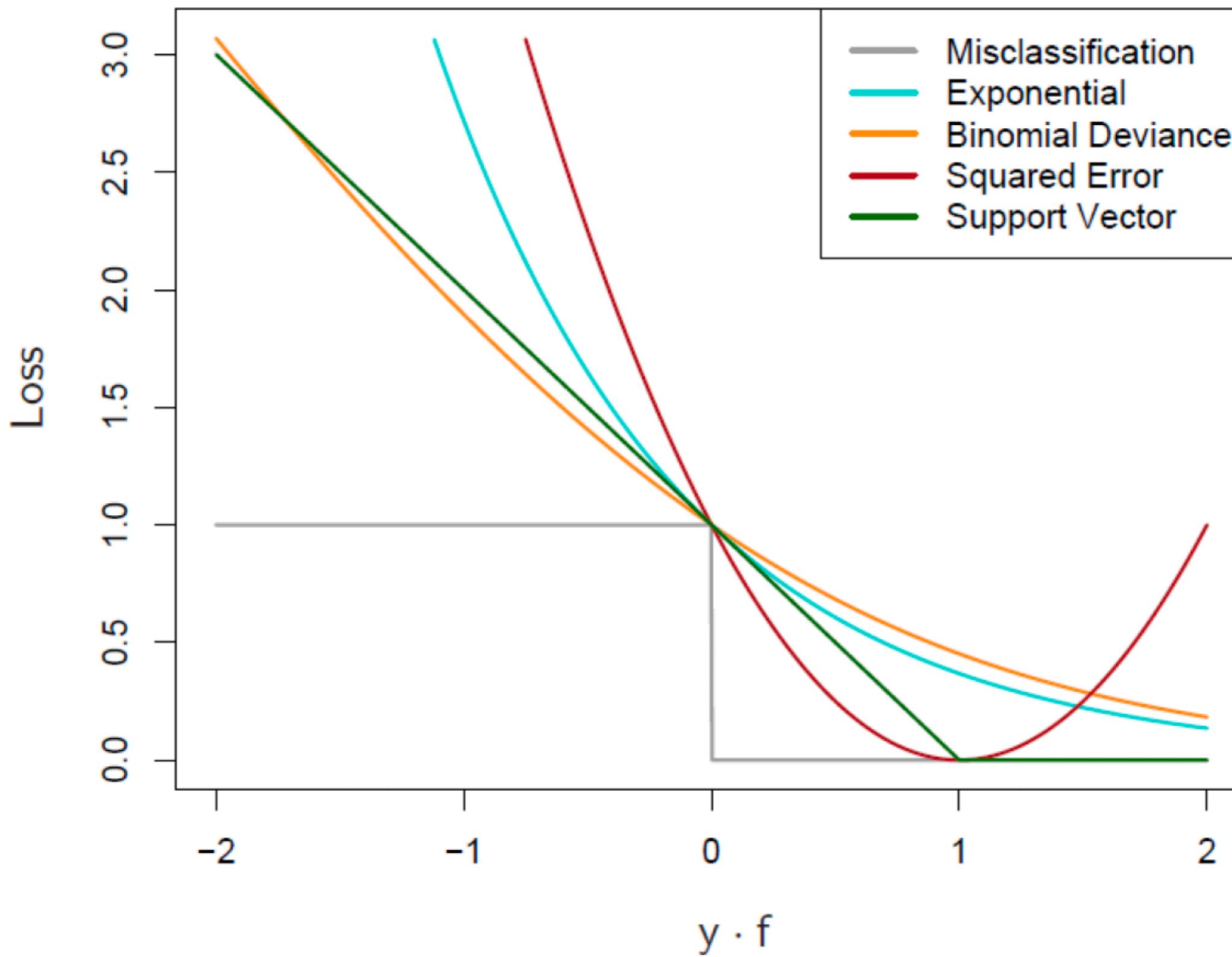
Какие проблемы у экспоненты?

AdaBoost

Экспонента обращает внимание на выбросы!



Можно использовать другие аппроксимирующие функции



Leo Breiman

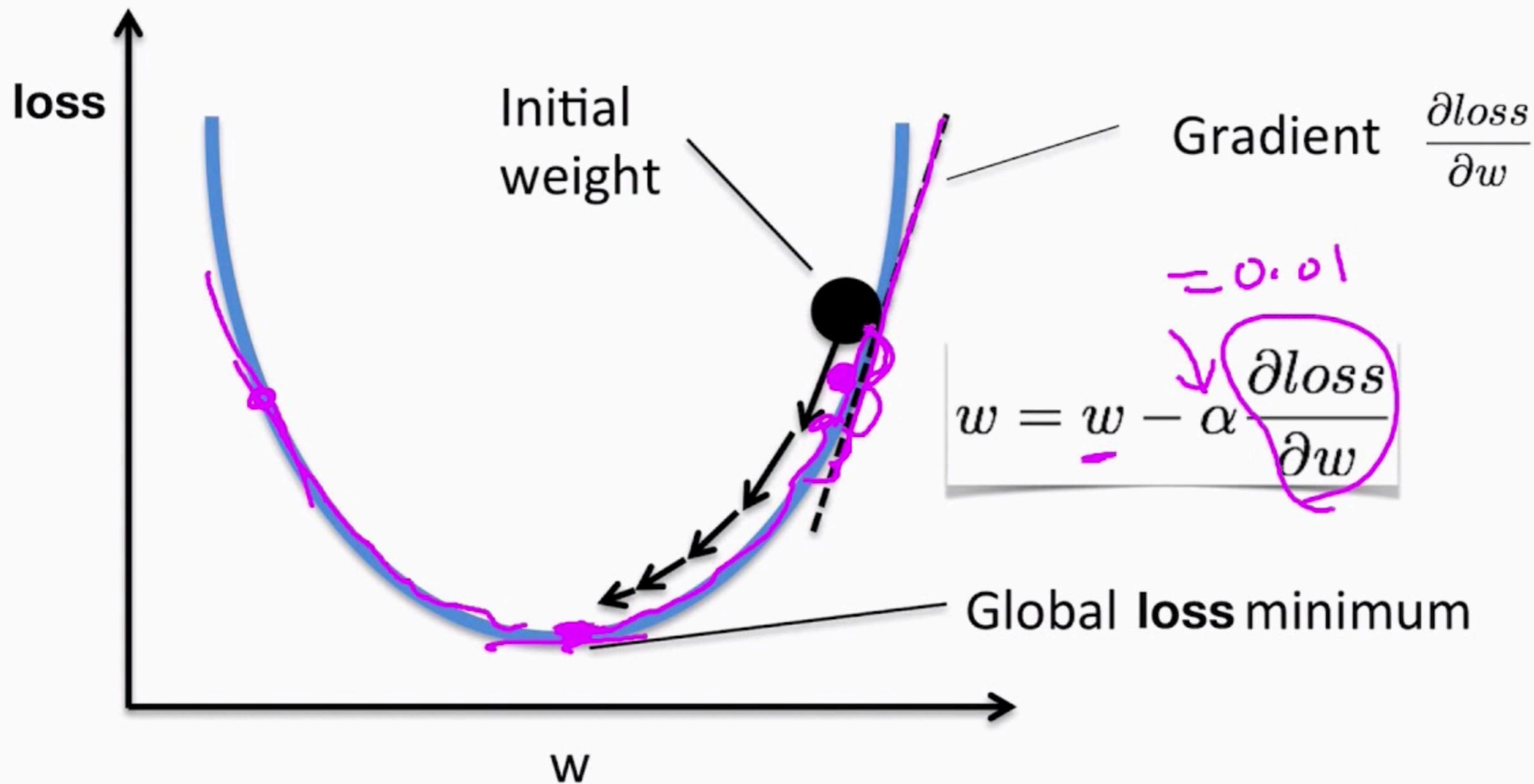


**Бустинг можно
рассматривать
как минимизацию некой
целевой функции ошибки**

Как минимизировать целевую функцию?

Как оптимизировать целевую функцию?

Gradient descent algorithm



Gradient boosting (Friedman)

1. Инициализировать $h_0(\mathbf{x}) = \operatorname{argmin}_\gamma \sum_{j=1}^N L(y_j, \gamma)$

2. Для всех i от 1 до T :

(a) Для всех $j = 1, 2, \dots, N$ вычислить

$$g_{i,j} = - \left[\frac{\partial L(y_j, h(\mathbf{x}_j))}{\partial h(\mathbf{x}_j)} \right]_{h=h_{i-1}}$$

(b) Построить базовую модель a_i на ответах $g_{i,j}$

$$a_i = \arg \min_a \sum_{j=1}^N (g_{i,j} - a(\mathbf{x}_j))^2$$

(c) Определить вес b_i

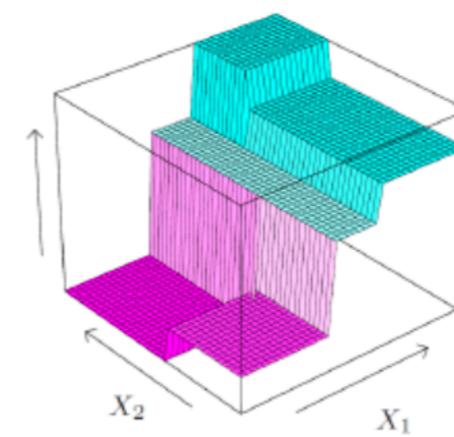
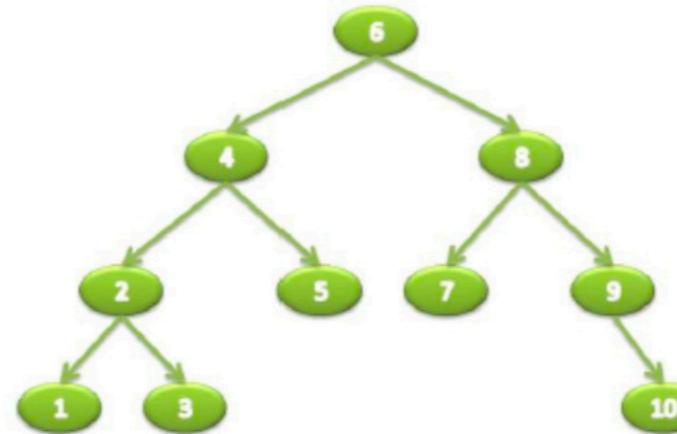
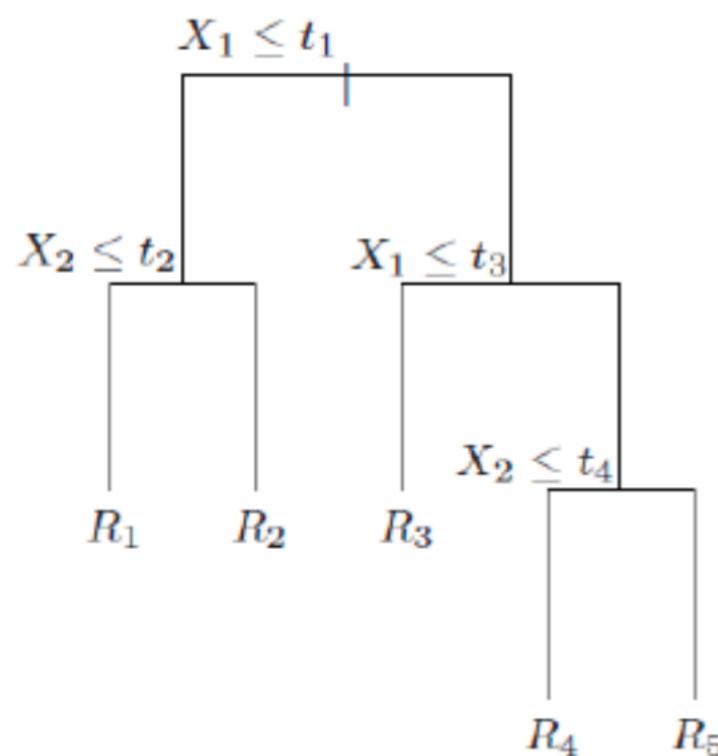
$$b_i = \arg \min_b \sum_{j=1}^N L(y_j, h_{i-1}(\mathbf{x}) + b \cdot a_i(\mathbf{x}_j))$$

(d) Присвоить $h_i(\mathbf{x}) = h_{i-1}(\mathbf{x}) + b_i \cdot a_i(\mathbf{x})$

3. Вернуть $h(\mathbf{x}) = h_T(\mathbf{x})$

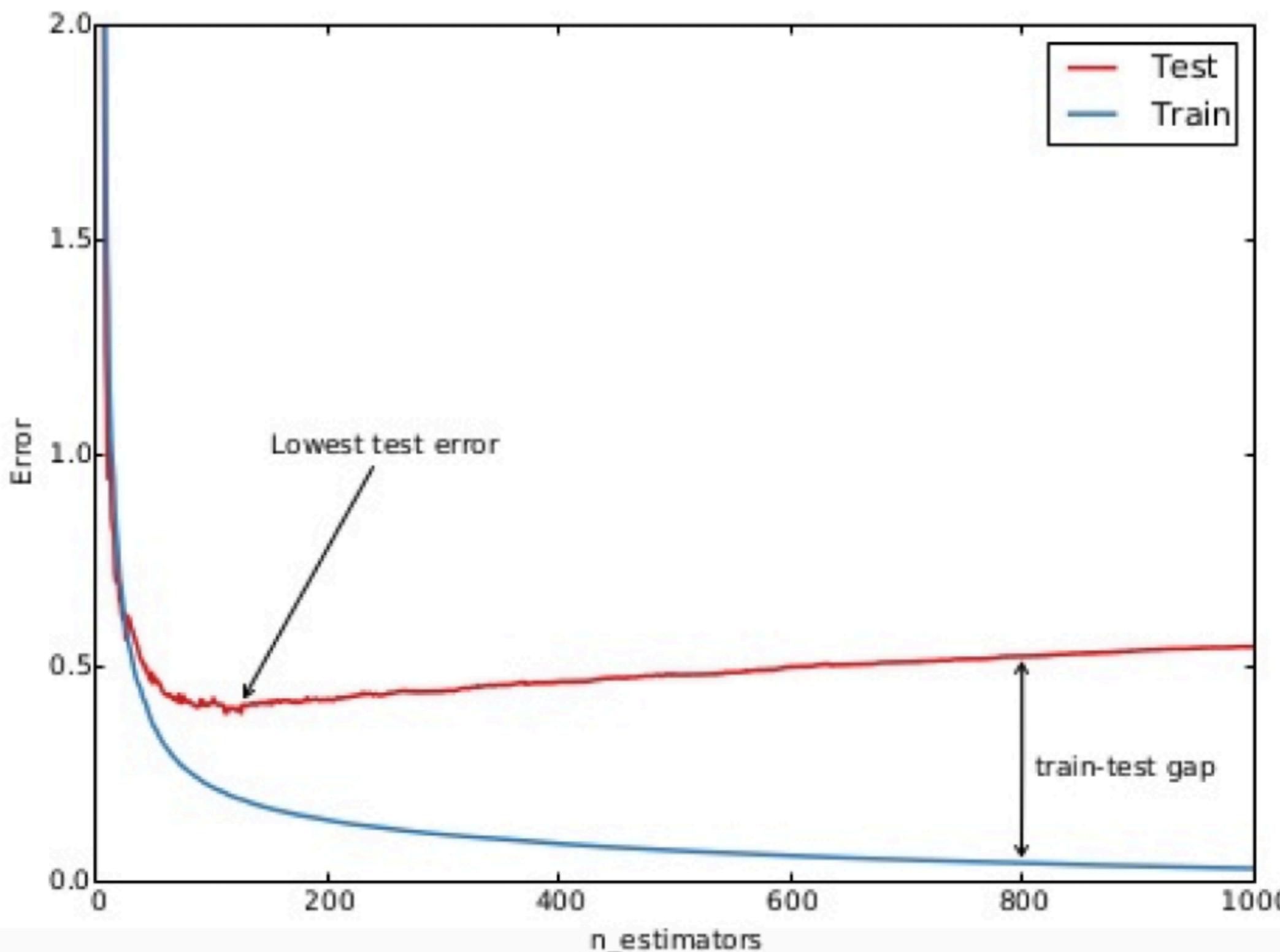
Дерево решений - фактически, сумма моделей

$$T(x) = \sum_d c_d I(x \in R_d)$$

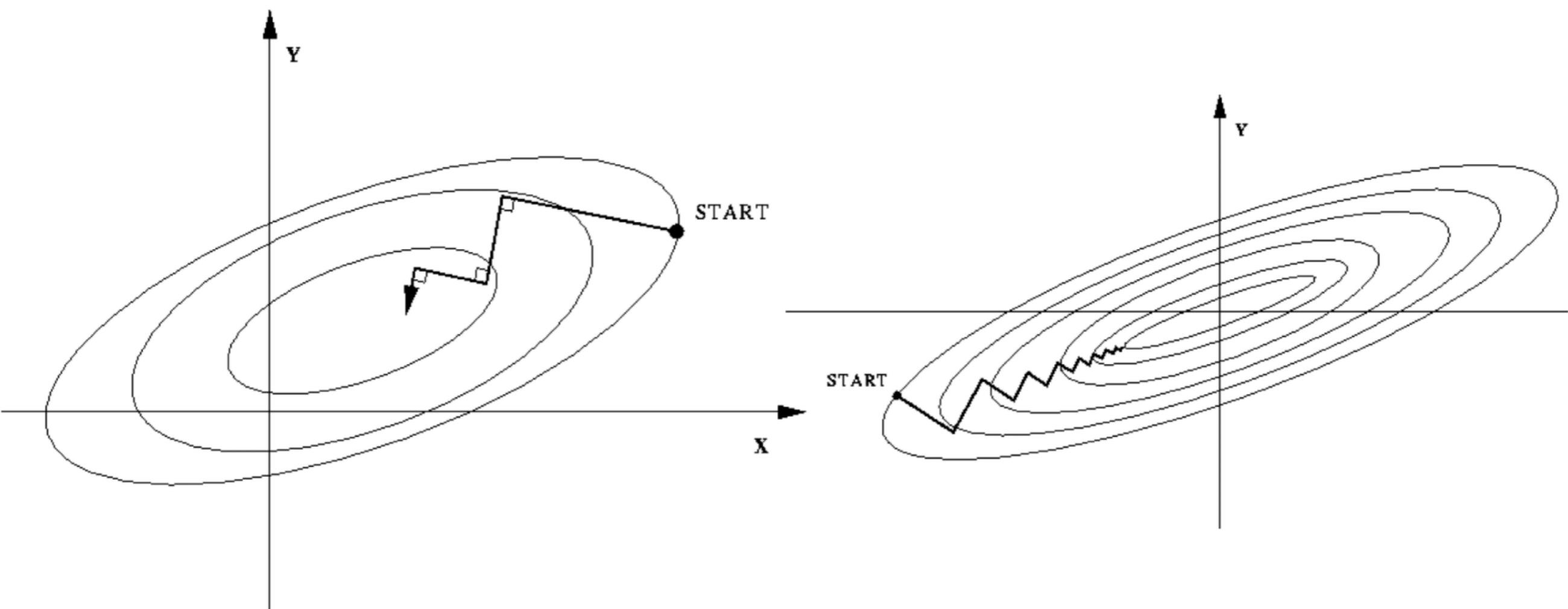


Gradient boosting

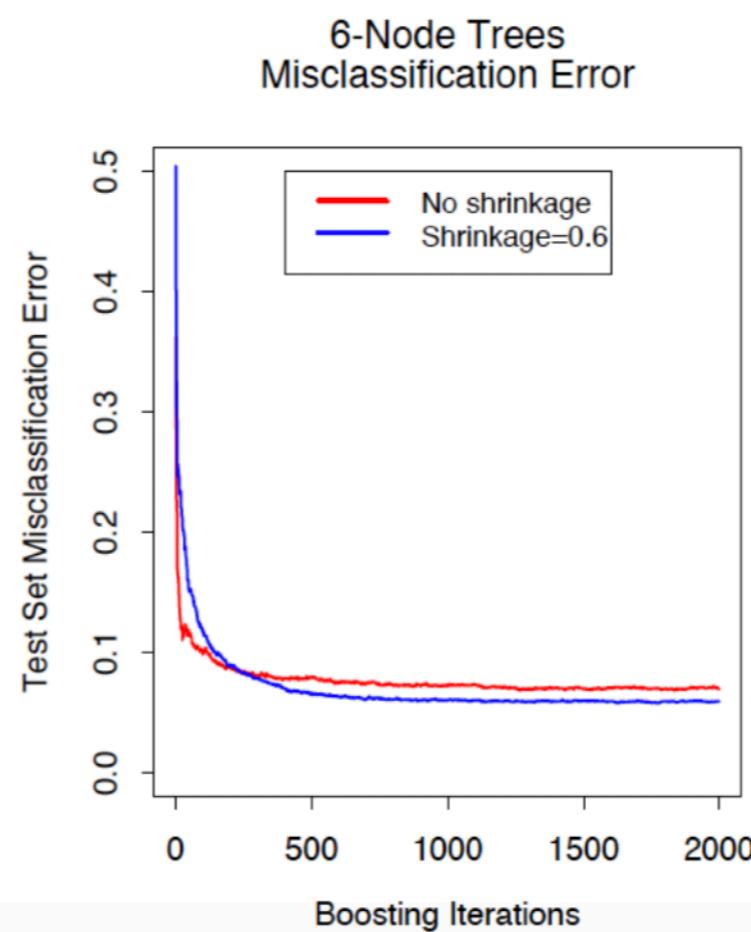
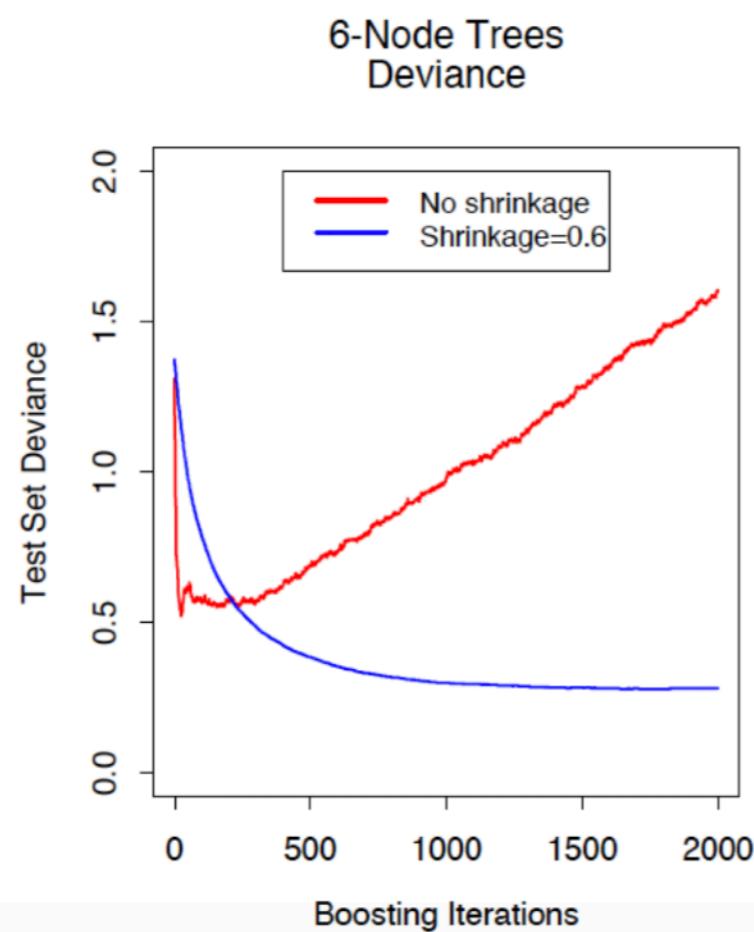
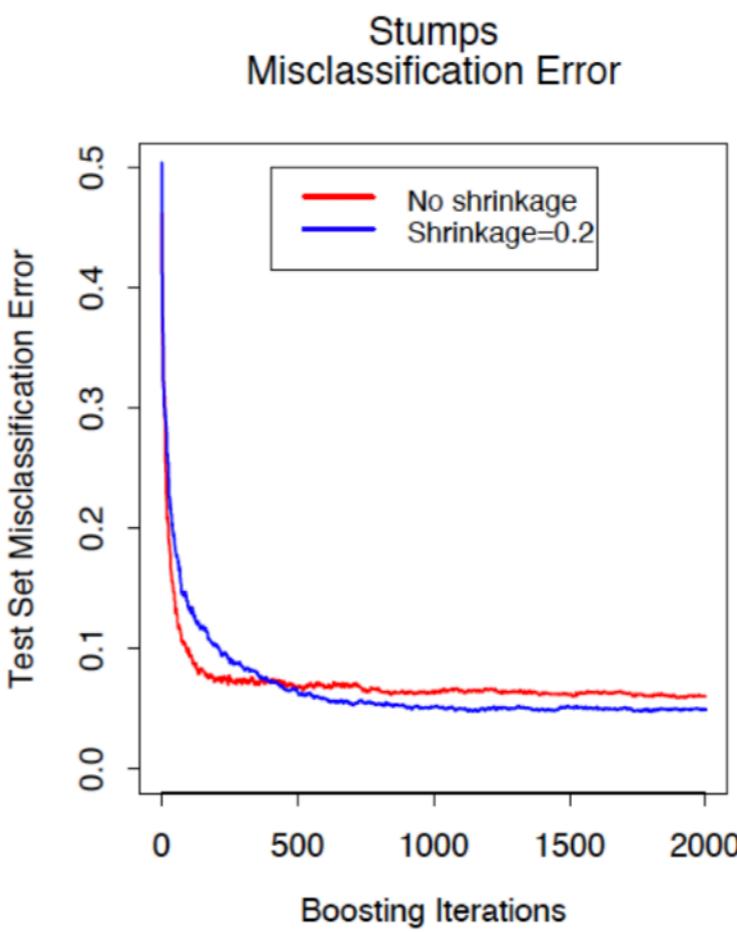
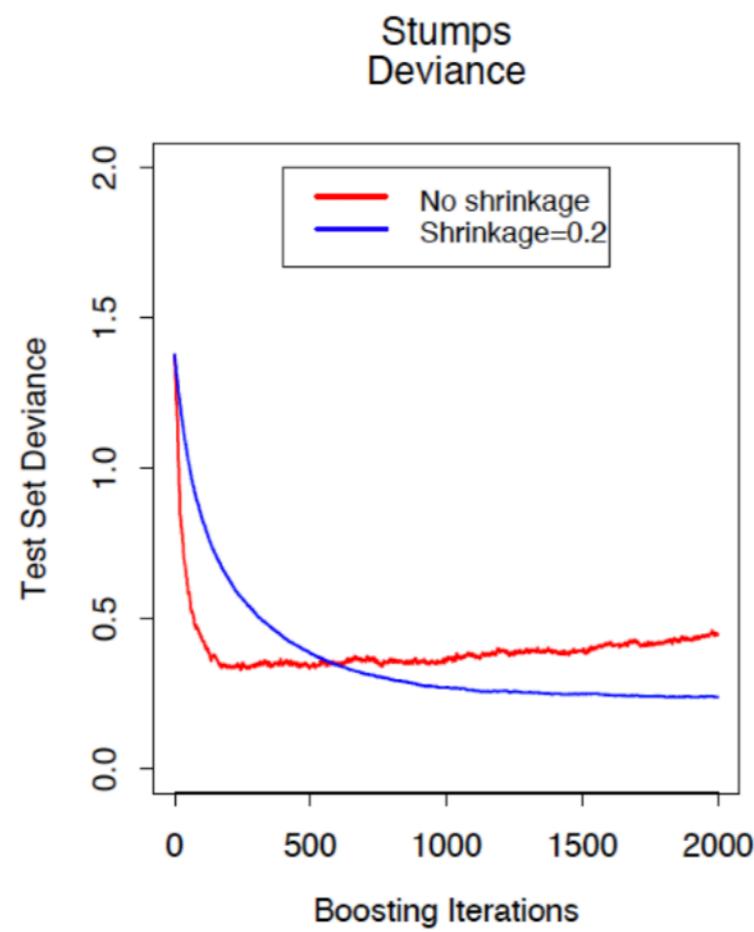
Переобучается((



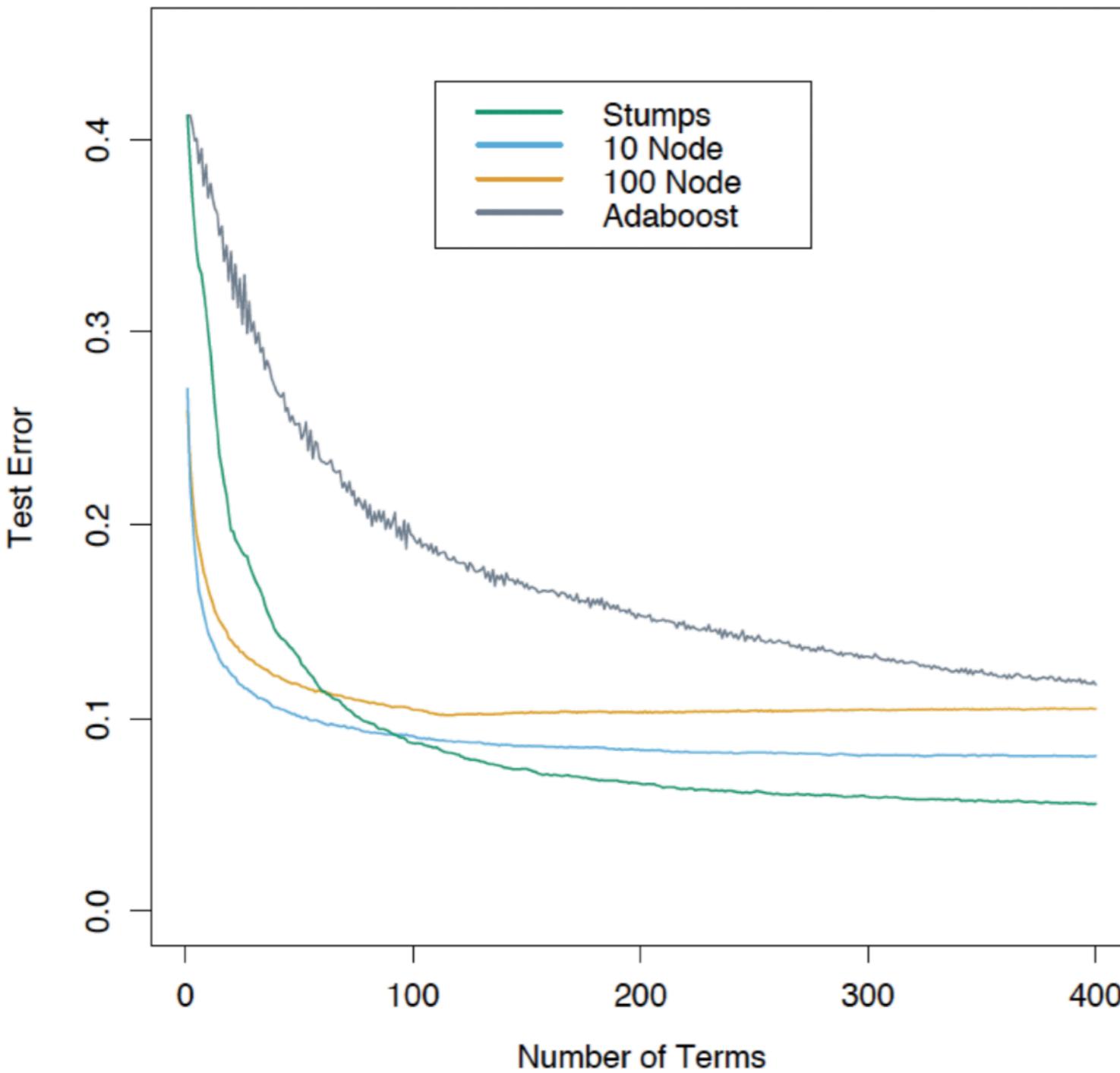
Shrinkage (learning rate)



Домножаем каждую добавляемую модель на еще один коэффициент, < 1 . Вклад каждой модели становится менее значим



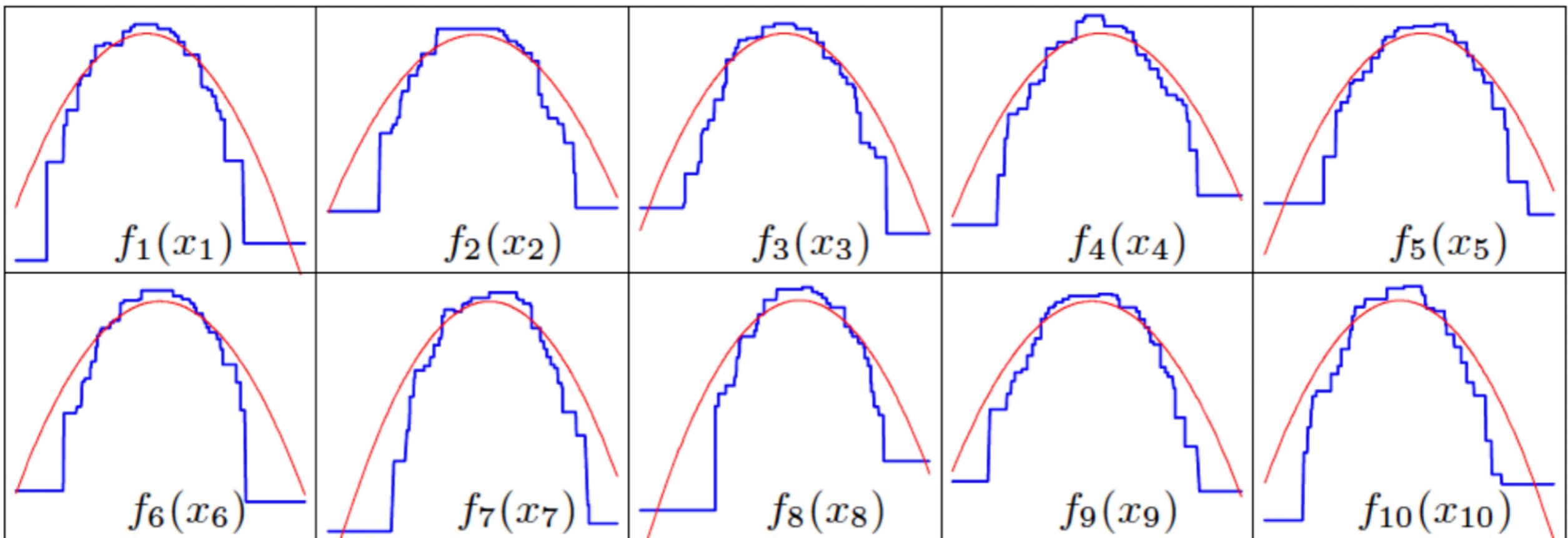
Gradient boosting



$$Y = \begin{cases} 1 & \text{if } \sum_{j=1}^{10} X_j^2 > \chi^2_{10}(0.5) \\ -1 & \text{otherwise.} \end{cases}$$

Gradient boosting

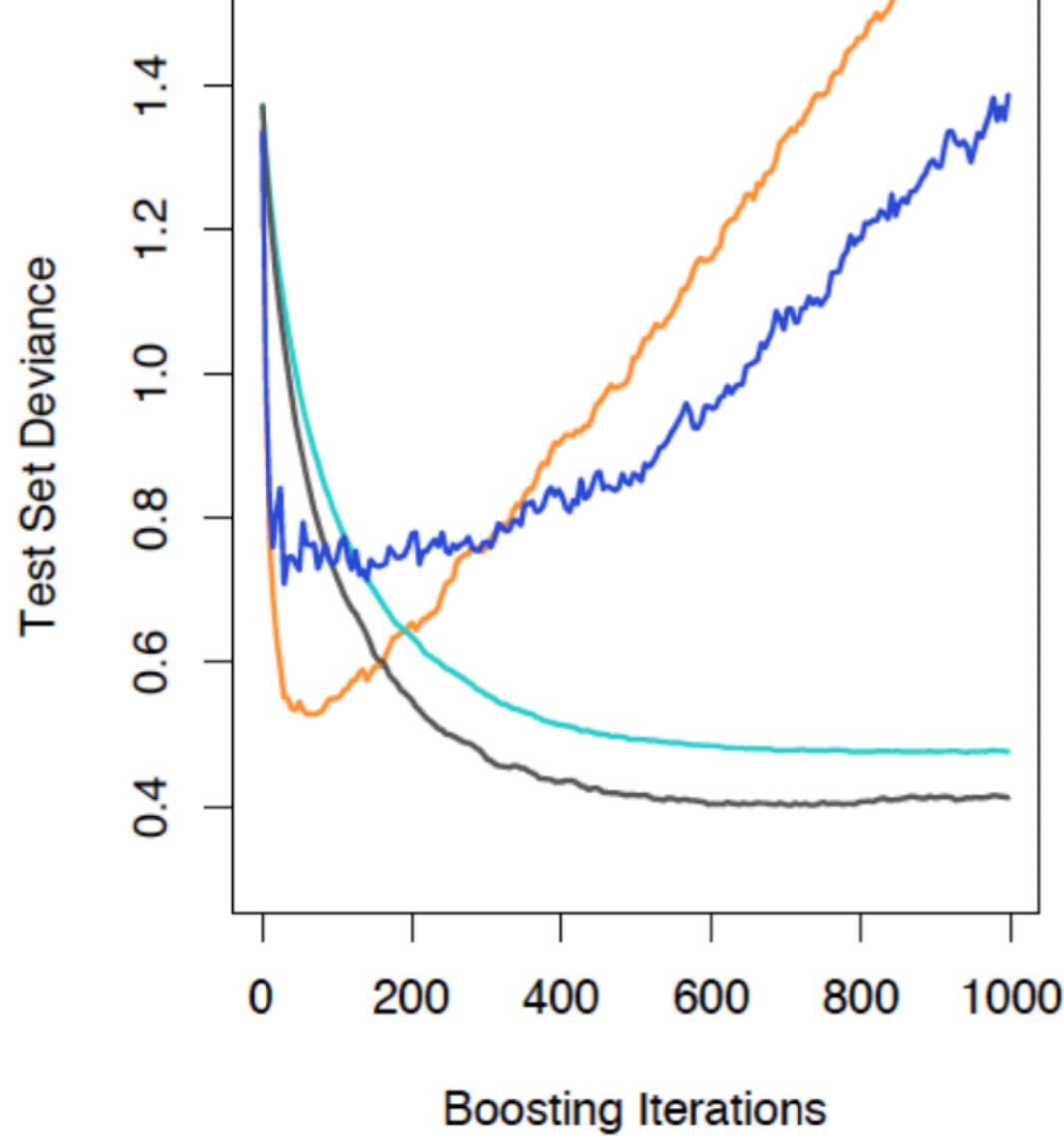
Coordinate Functions for Additive Logistic Trees



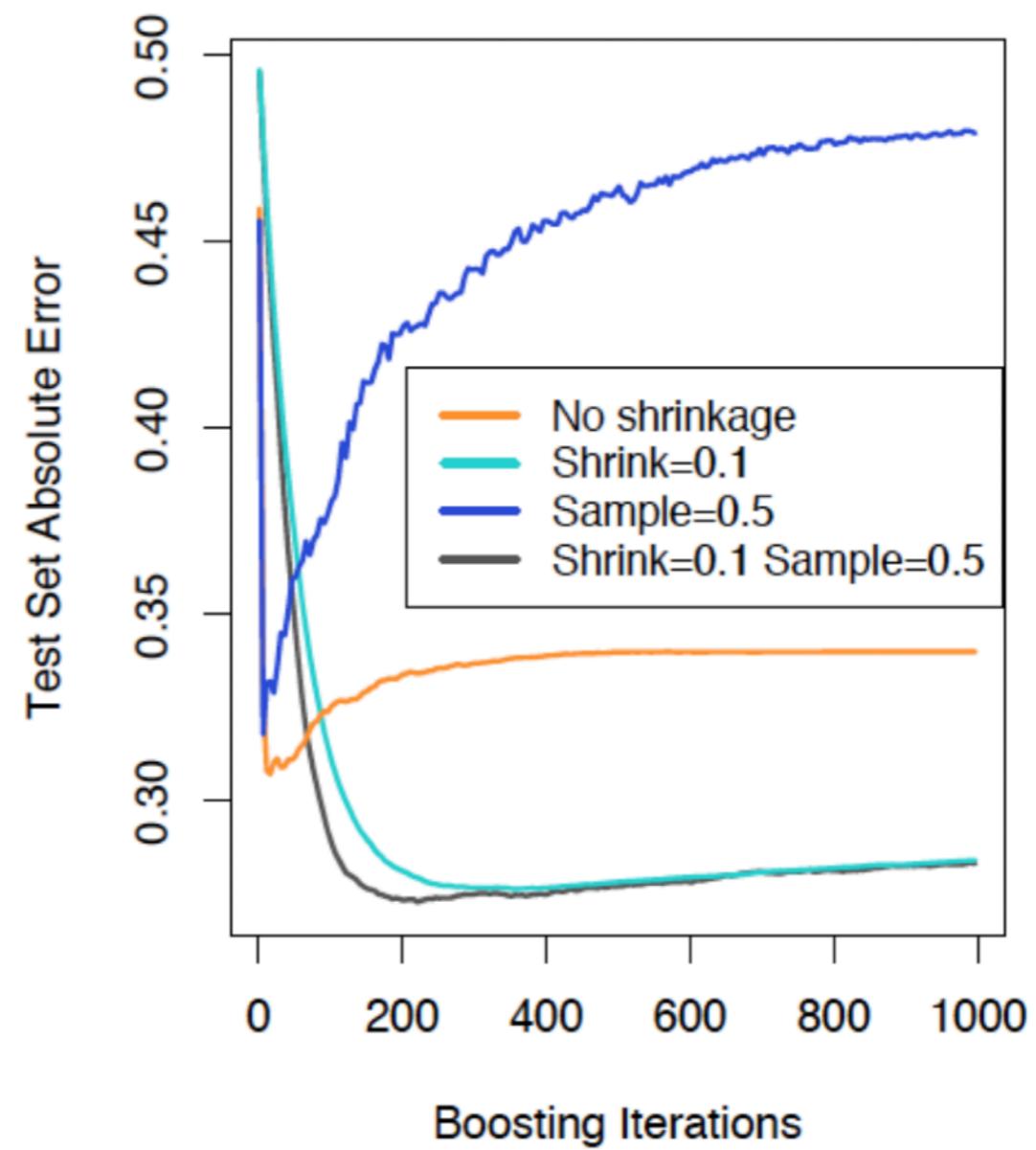
Subsample

4 Node Trees

Deviance

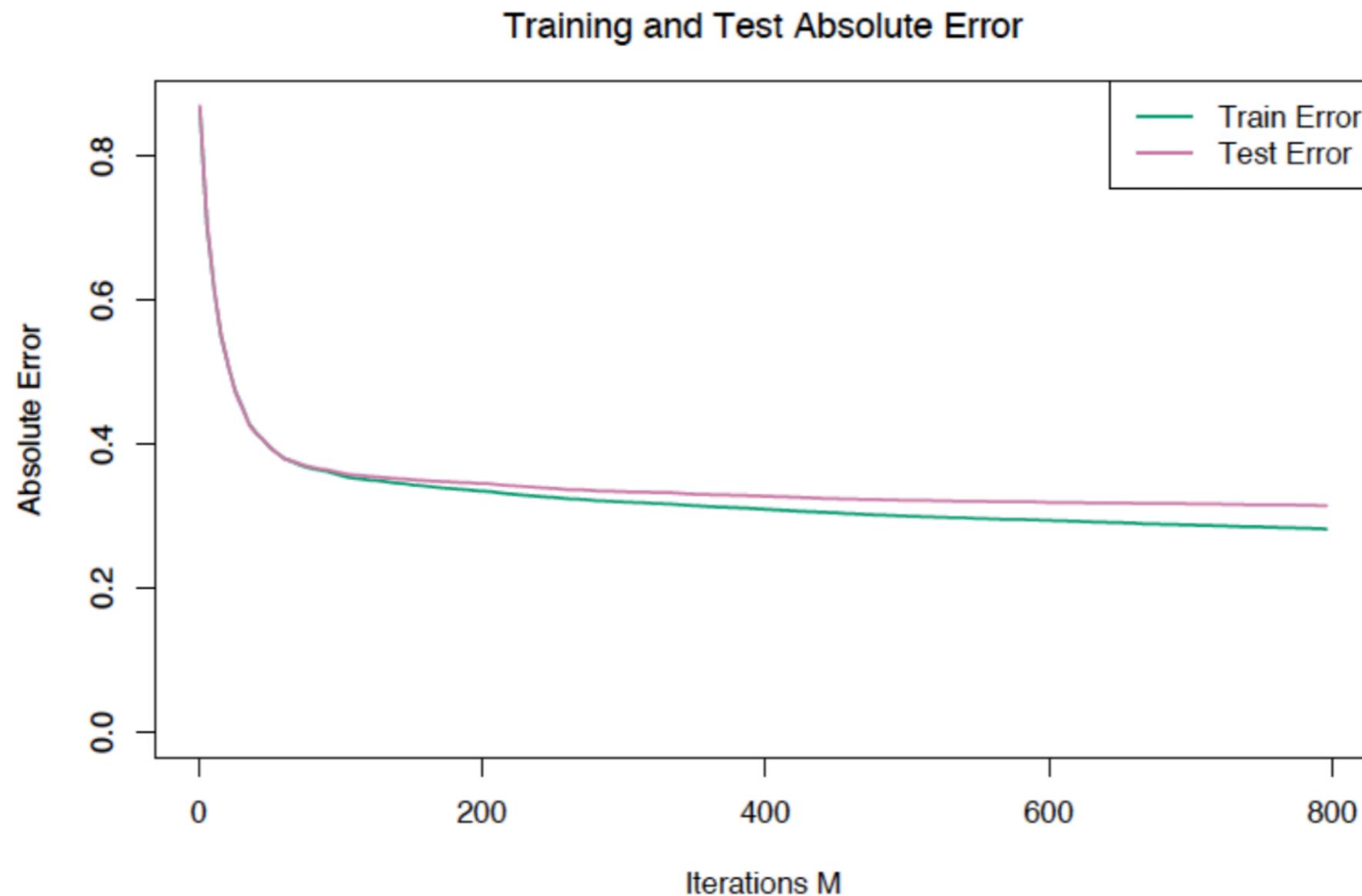


Absolute Error



Можно сэмплировать признаки, можно сэмплировать объекты

“Не переобучается”



Если подобрать хорошие параметры, в частности, хороший shrinkage, можно получить отсутствие переобучения. Но как такое детектировать - вопрос.