

# Машинное обучение

Лекция 3. Решающие деревья и ансамбли



## Напоминание: часто используемые методы

- Линейные модели
- Решающие деревья
- Ансамбли решающих деревьев

# Обсуждаем сегодня: деревья и ансамбли

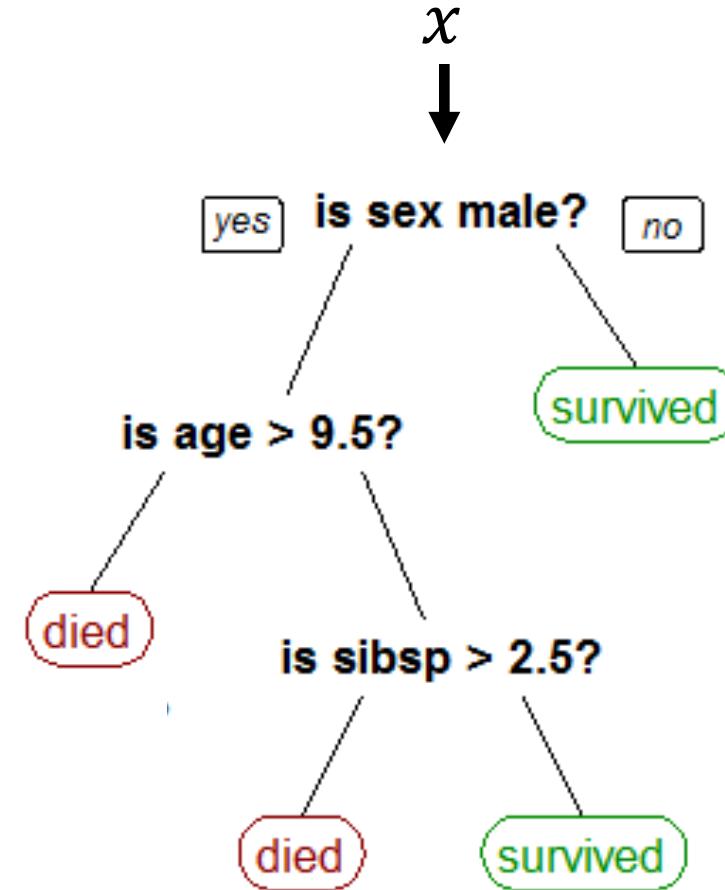
- I. Решающие деревья
- II. Ансамбли деревьев
- III. Общие идеи построения ансамблей

# I. Решающие деревья

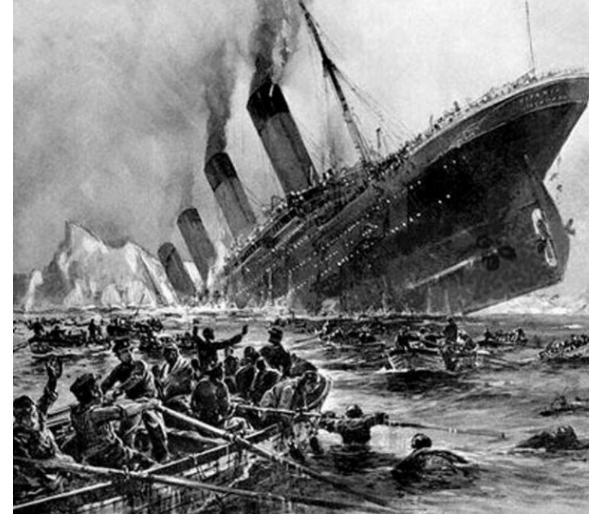
# План

1. Что такое решающие деревья
2. Решающие деревья в классификации и регрессии
3. Как строить решающие деревья
4. Дополнительные темы

# Решающее дерево



# Датасет



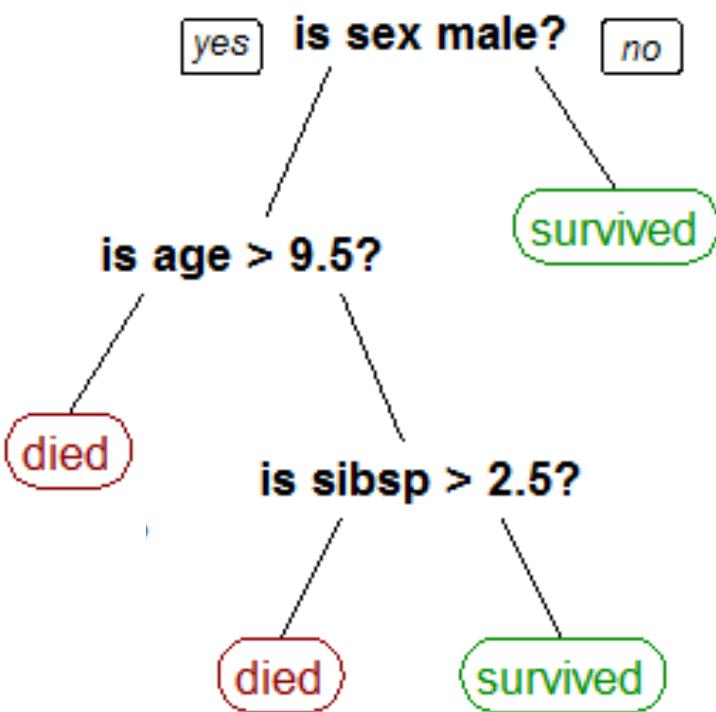
**«Titanic Dataset»** - список пассажиров Титаника, для которых даны возраст, пол, количество членов семьи на борту и другие признаки.

**Целевые значения:** выжил пассажир или нет  
(задача классификации)

# Решающее дерево

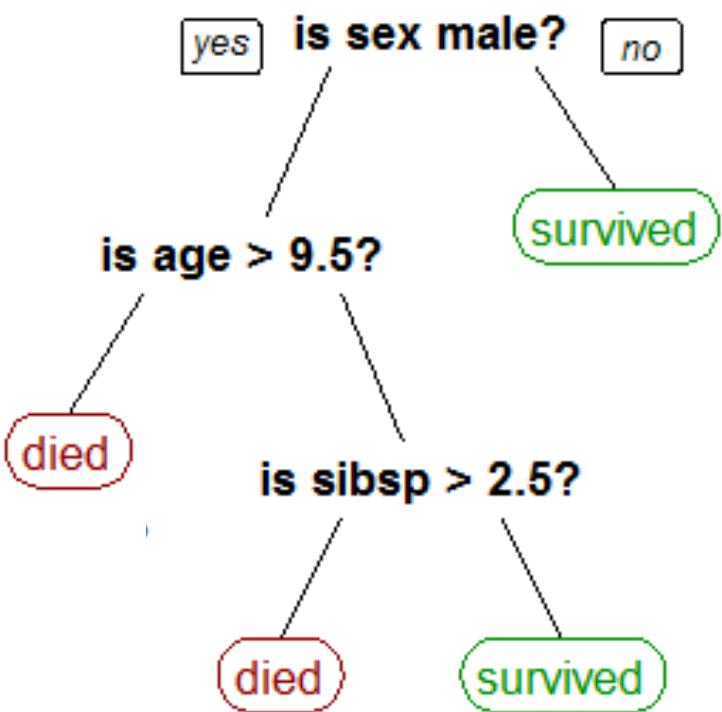
$$x = (9, \text{ male}, 3)$$

↓



# Решающее дерево

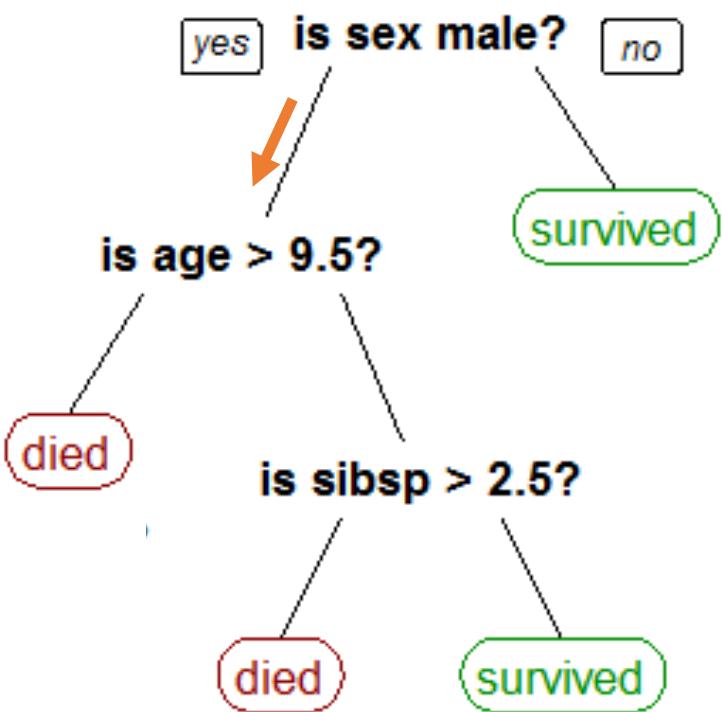
$$x = (9, \underset{\text{age}}{\text{male}}, \underset{\text{sex}}{3}, \underset{\text{sibsp}}{3})$$



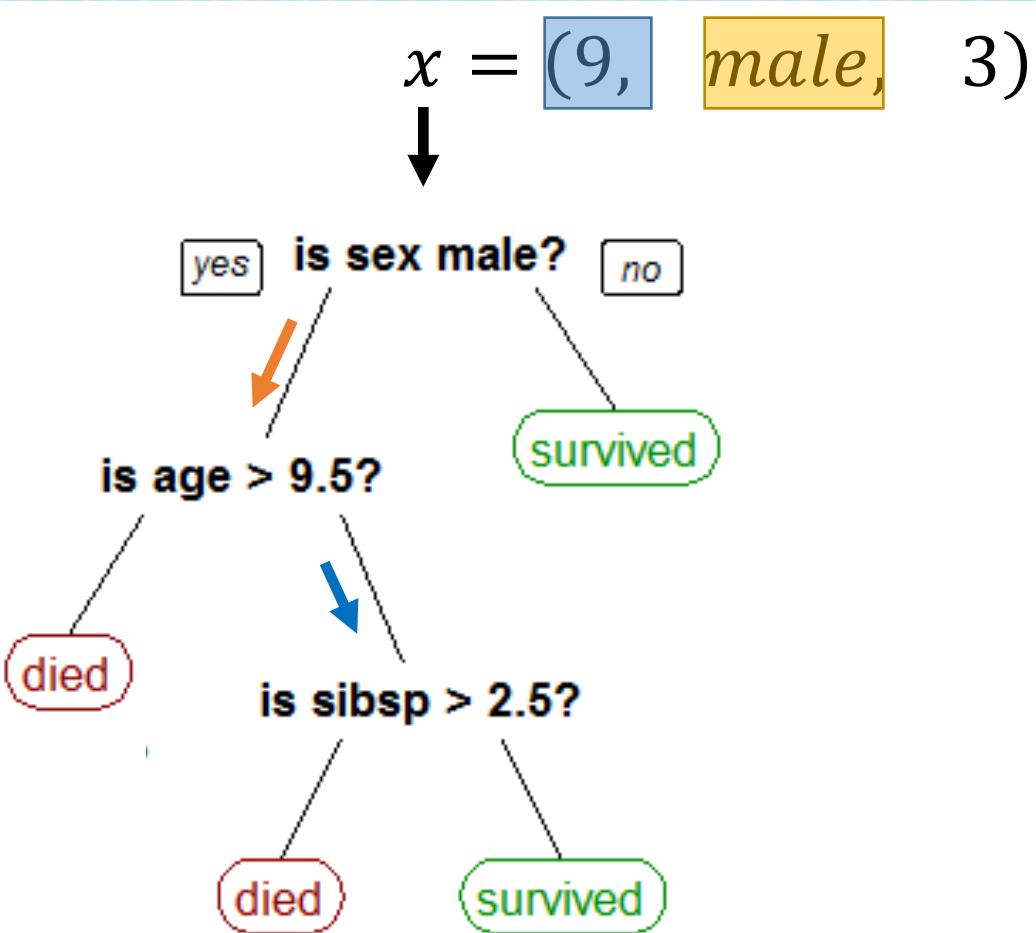
# Решающее дерево

$$x = (9, \text{male}, 3)$$

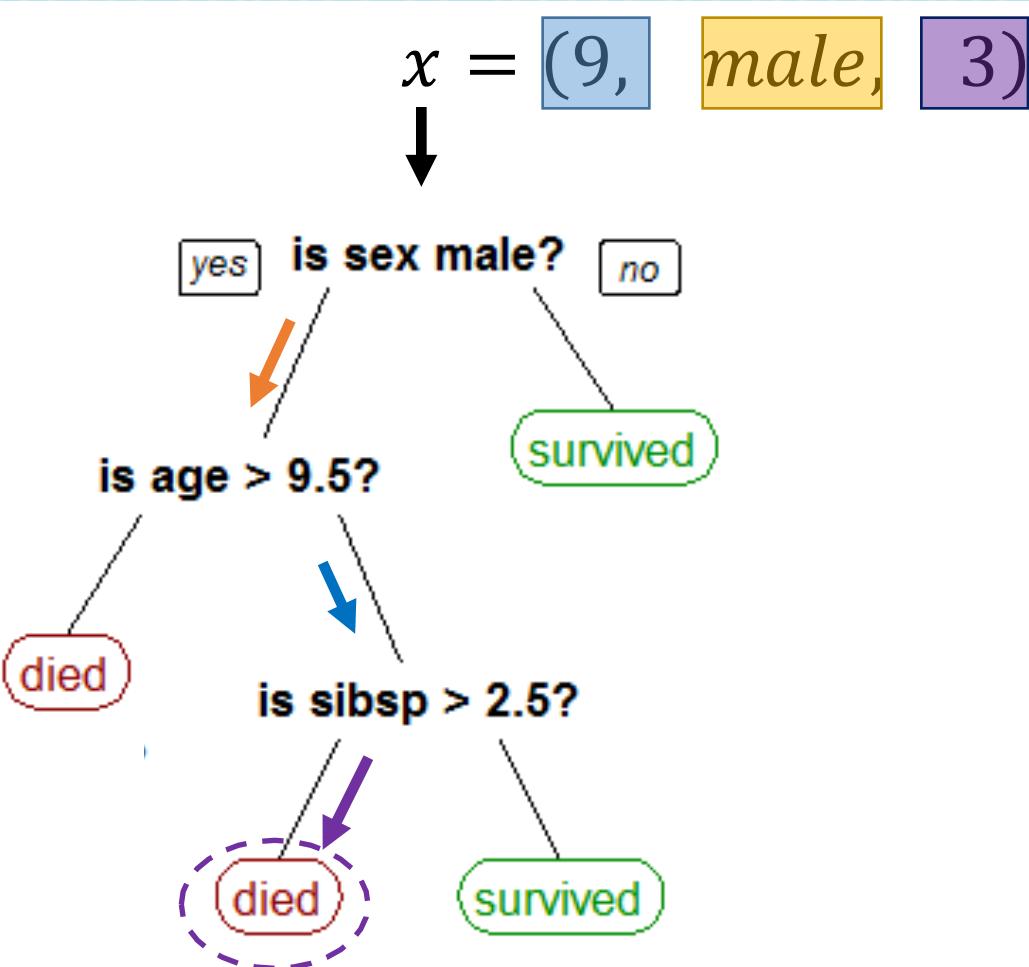
↓



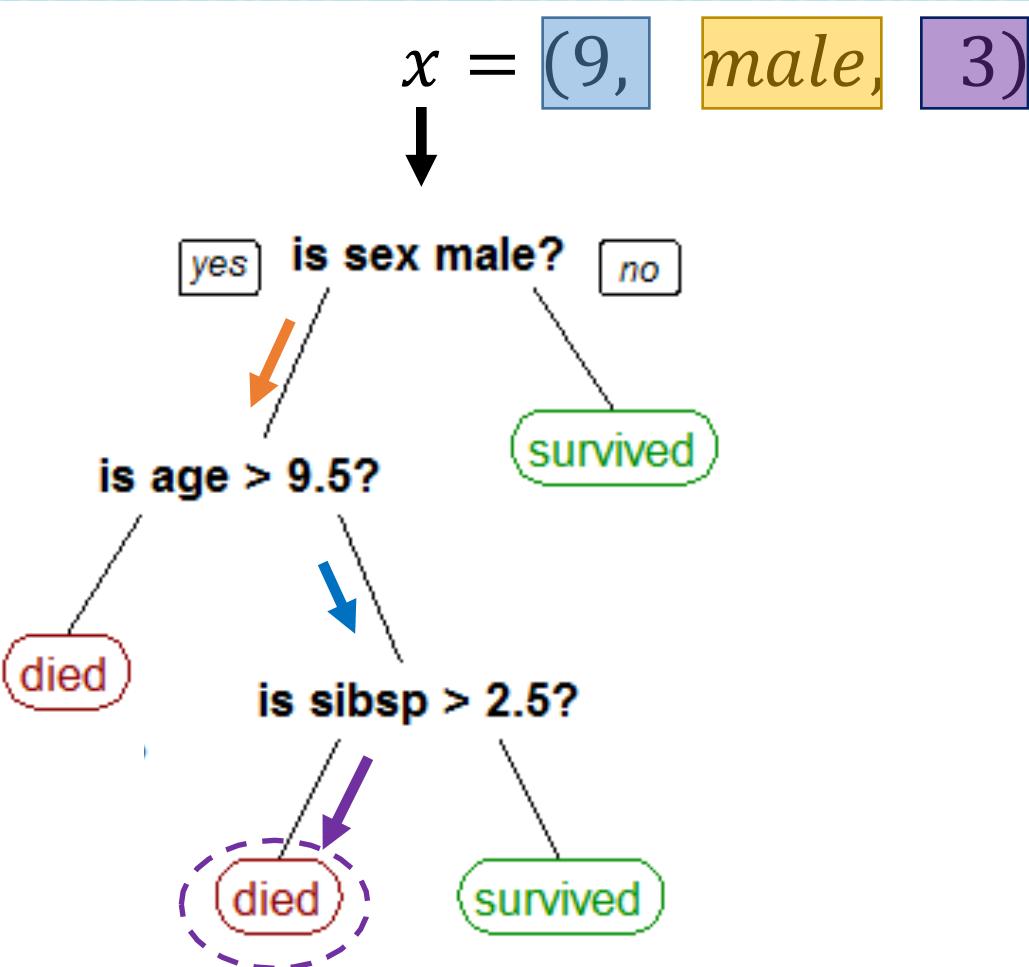
# Решающее дерево



# Решающее дерево

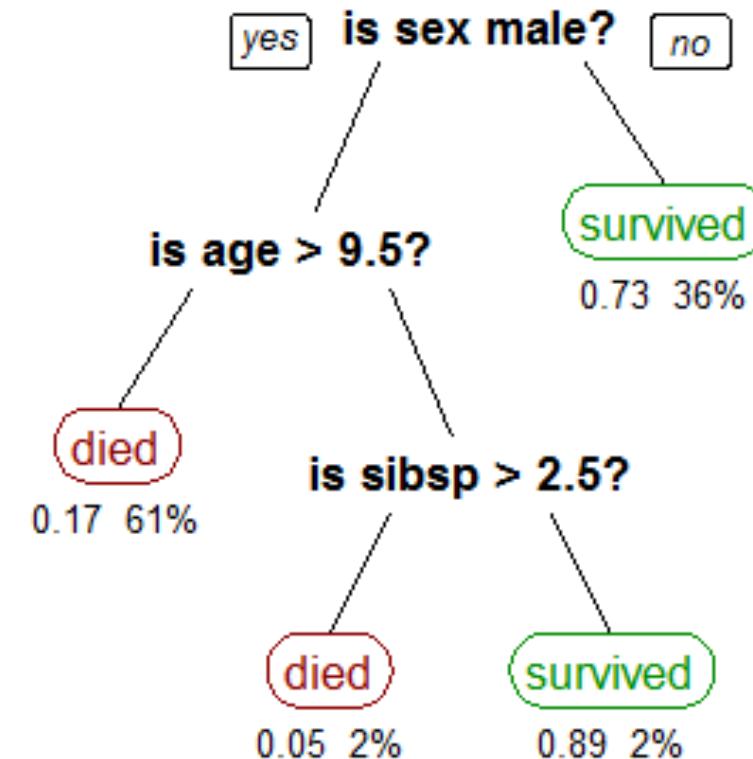


# Решающее дерево

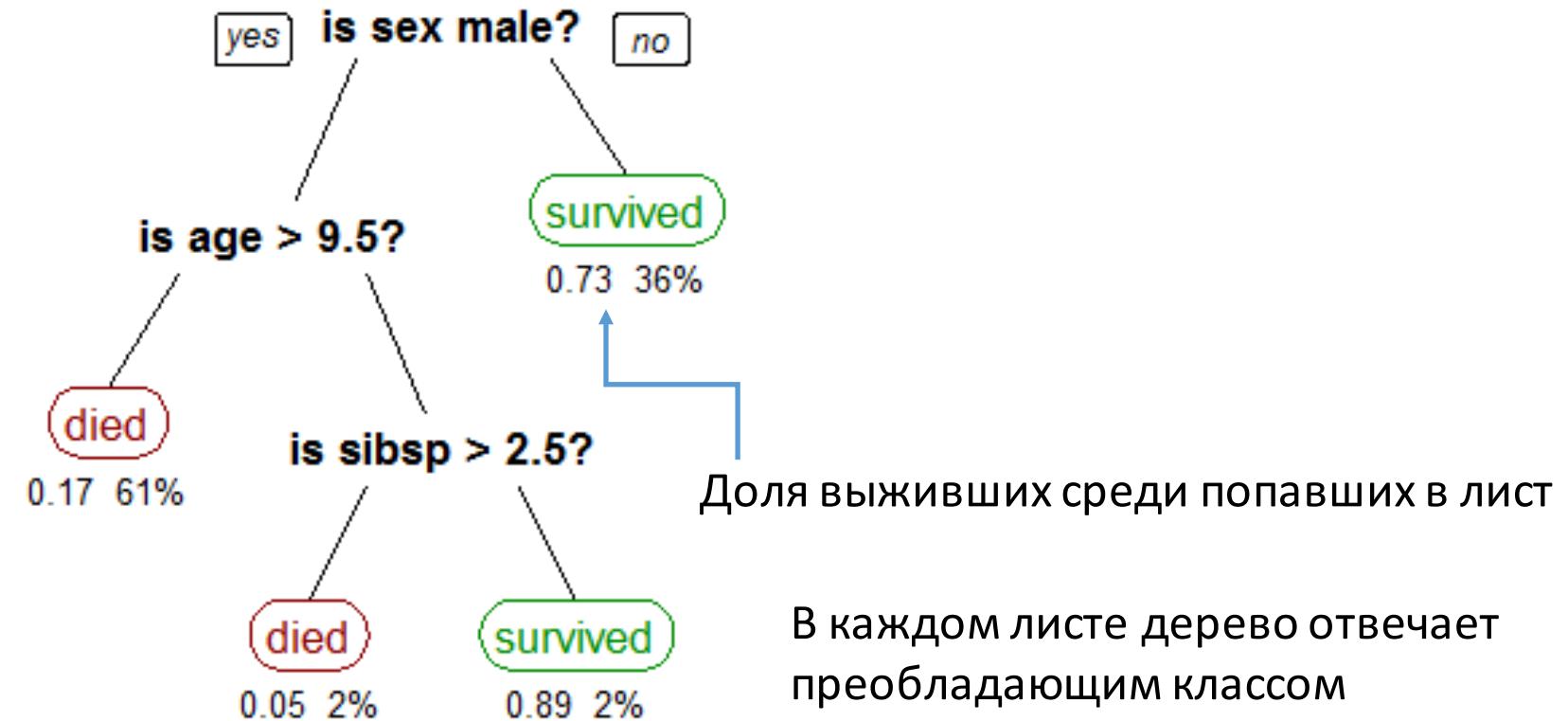


$$y = \text{died}$$

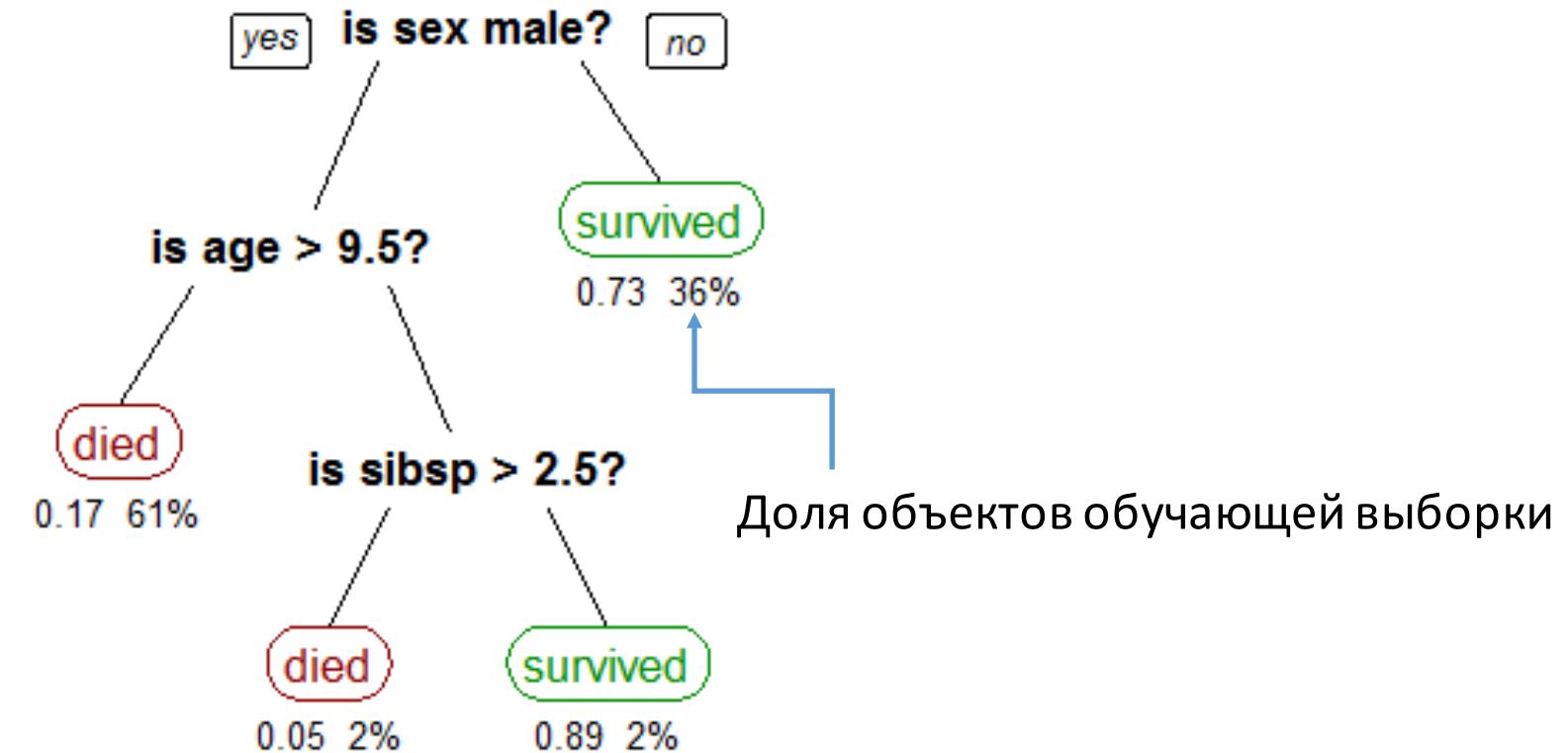
# Решающее дерево: классификация



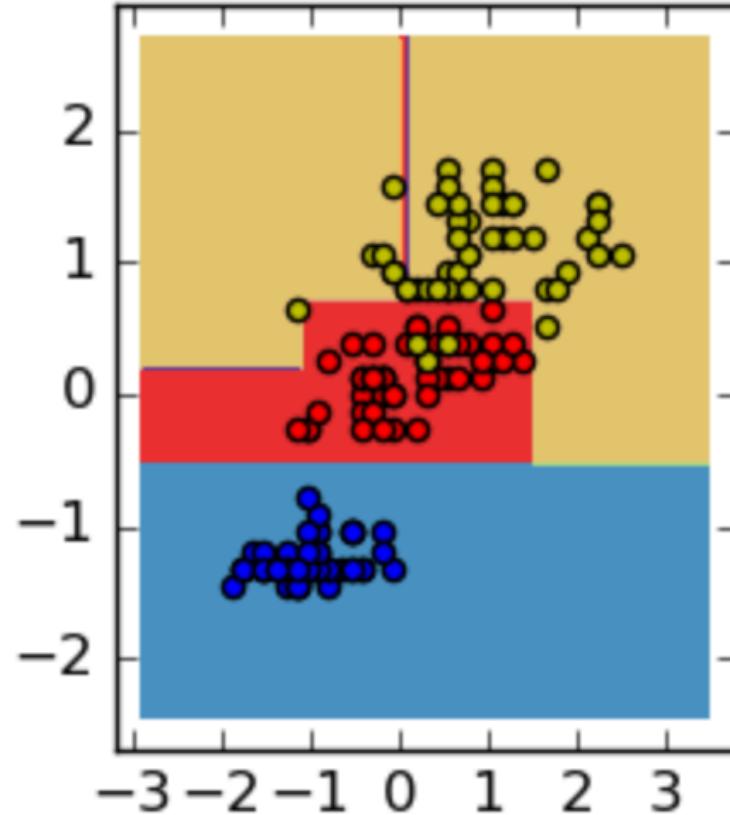
# Решающее дерево: классификация



# Решающее дерево: классификация

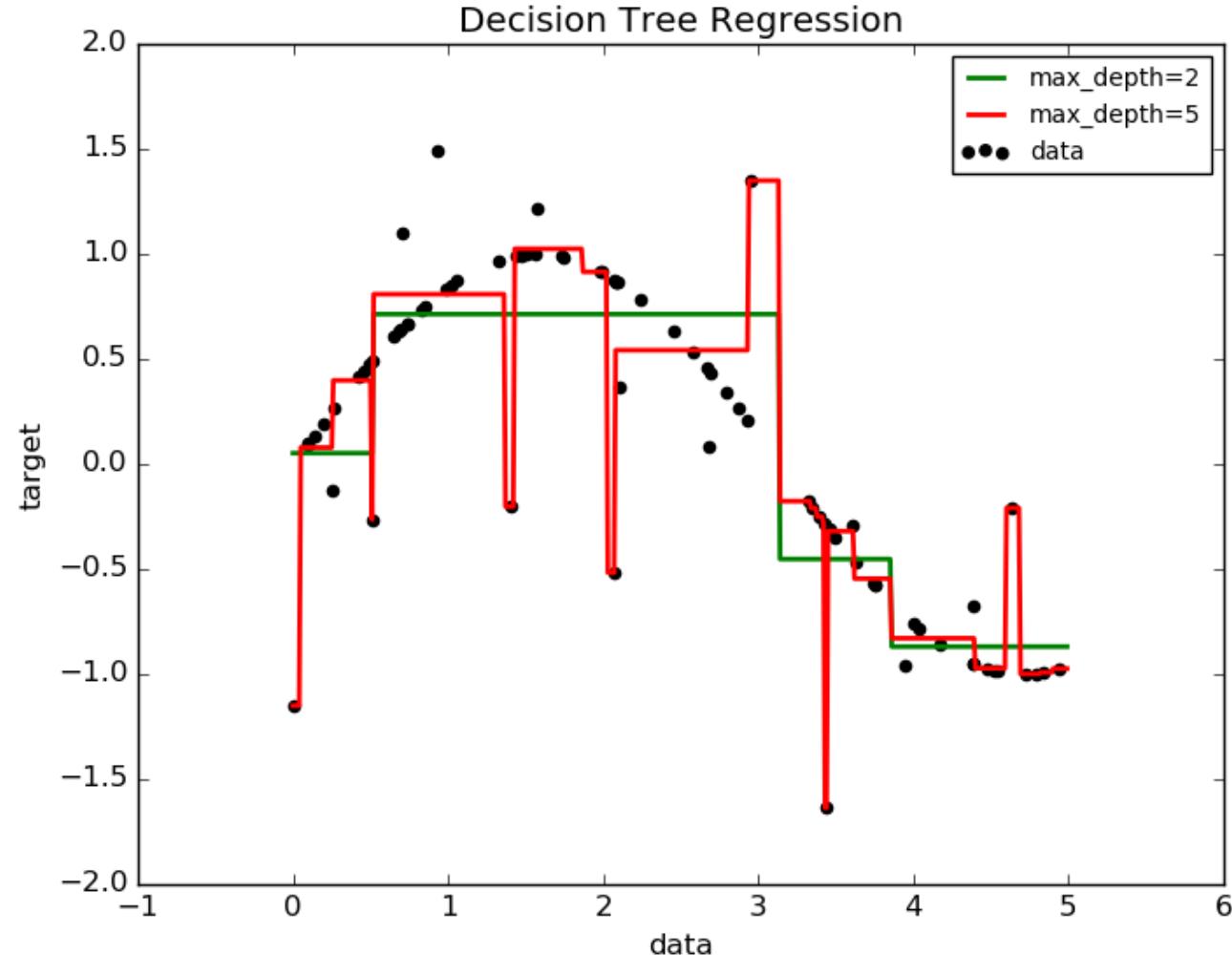


# Решающее дерево: классификация



Пример: 3 класса и 2 признака

# Решающее дерево: регрессия



Пример: восстановление зависимости у от х с помощью решающих деревьев глубины 2 и глубины 5

В каждом листе дерево отвечает некоторой константой

# Рекурсивное построение

Строим разбиение  
выборки по значению  
одного из признаков

$$x^{(j)} < t$$

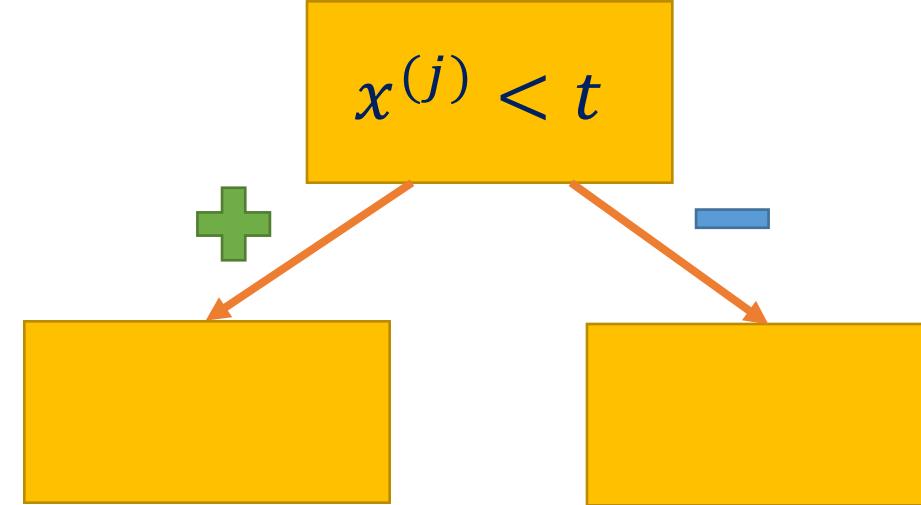
# Рекурсивное построение

Строим разбиение  
выборки по значению  
одного из признаков

$$x^{(j)} < t$$

Фактически нужно  
только выбрать  $j$  и  $t$   
наилучшим образом

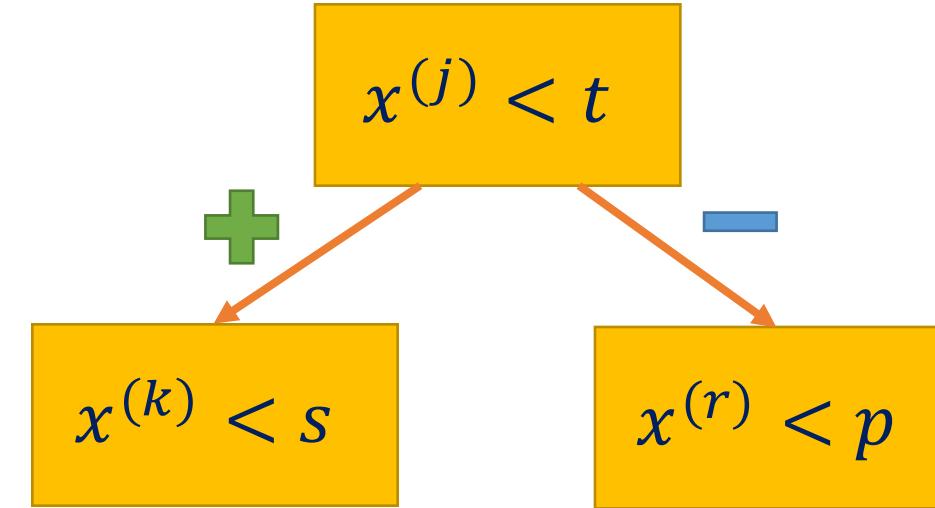
# Рекурсивное построение



Выборка делится  
по этому условию  
на две части

# Рекурсивное построение

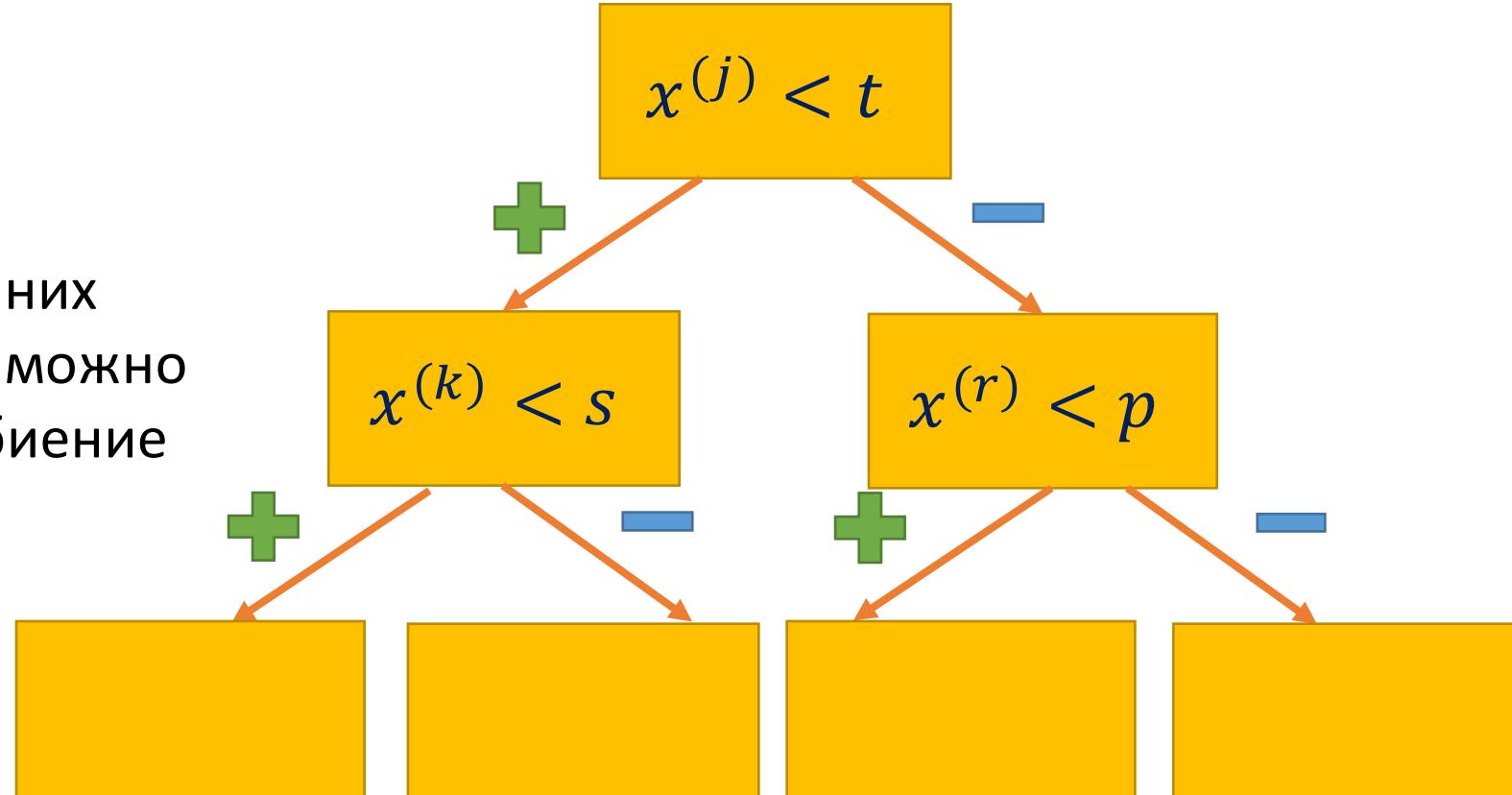
В каждой из них  
теперь тоже можно  
сделать разбиение



Выборка делится  
по этому условию  
на две части

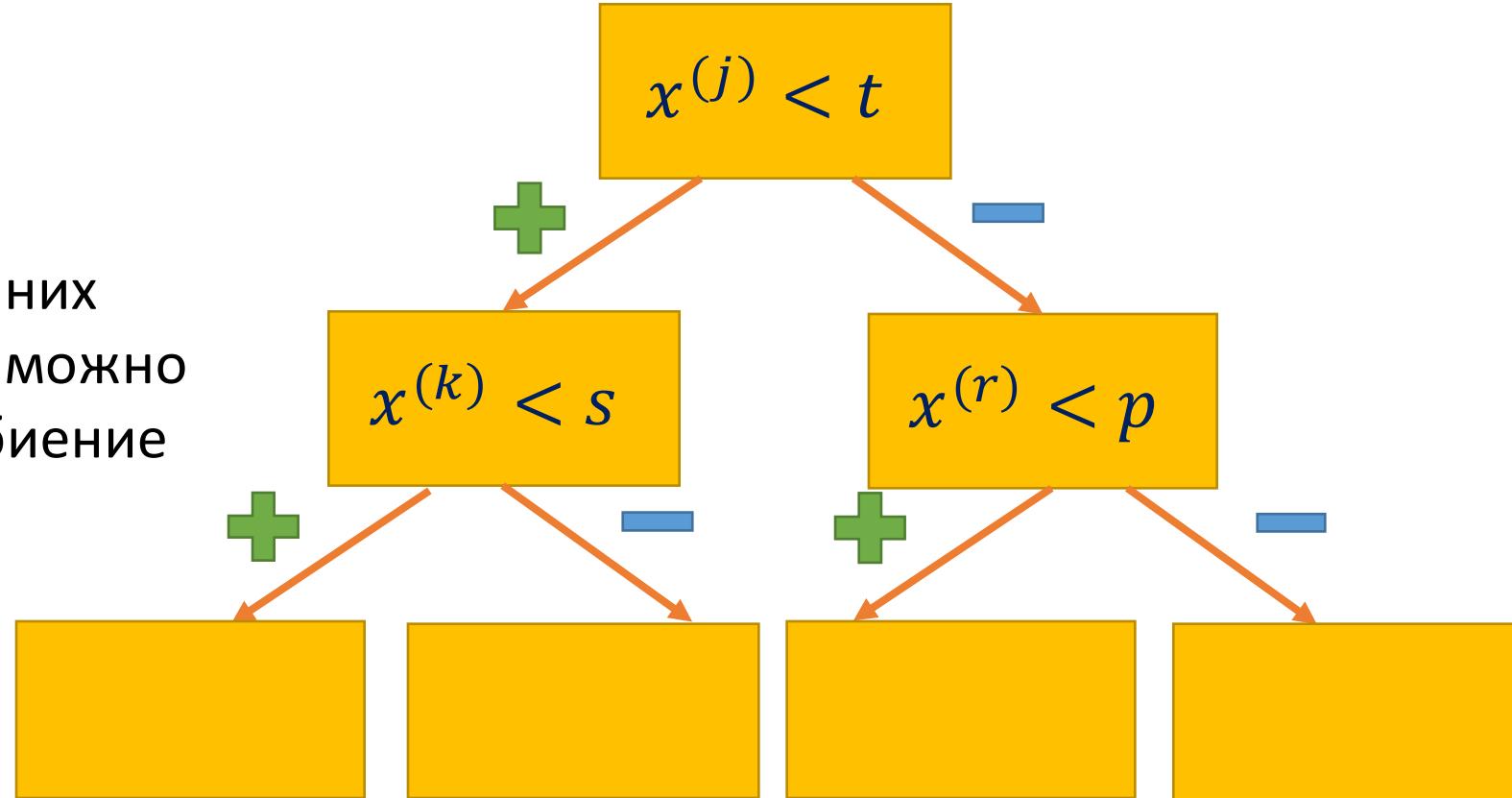
# Рекурсивное построение

В каждой из них  
теперь тоже можно  
сделать разбиение



# Рекурсивное построение

В каждой из них  
теперь тоже можно  
сделать разбиение



Процесс можно продолжать в тех узлах, в  
которые попадает достаточно много объектов

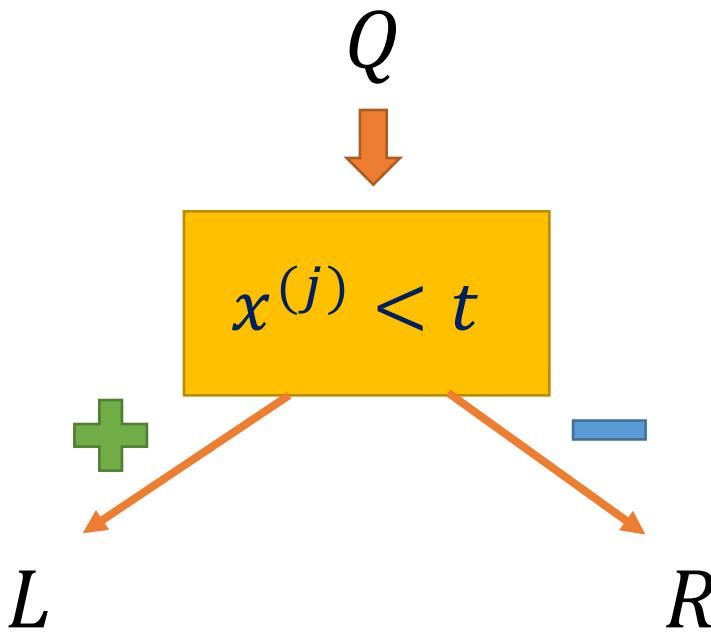
# Выбор разбиения

$Q$

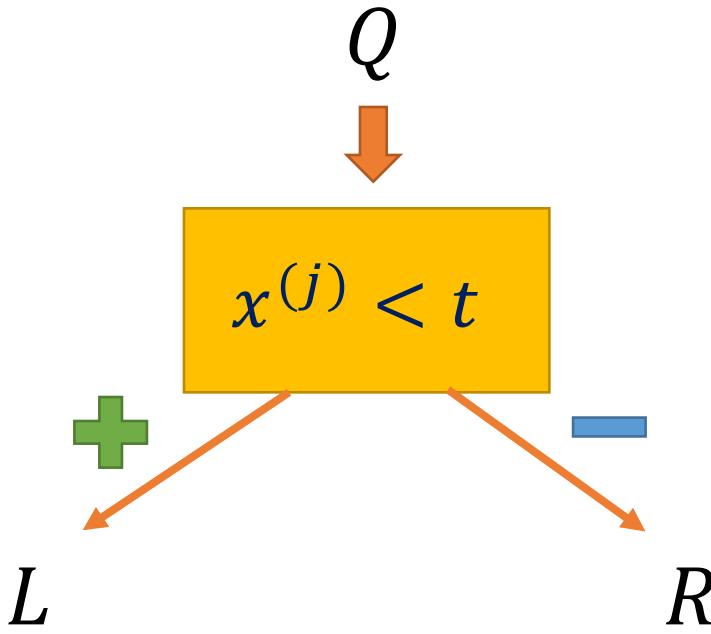


$$x^{(j)} < t$$

# Выбор разбиения

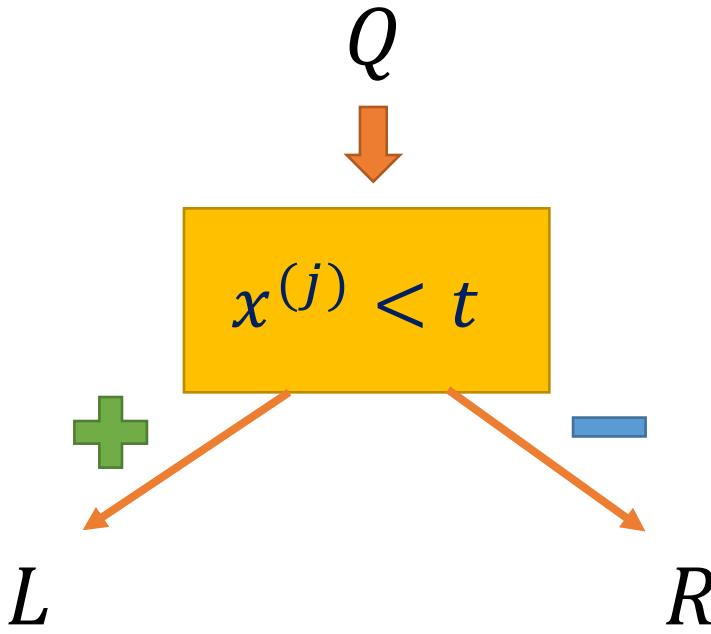


# Выбор разбиения



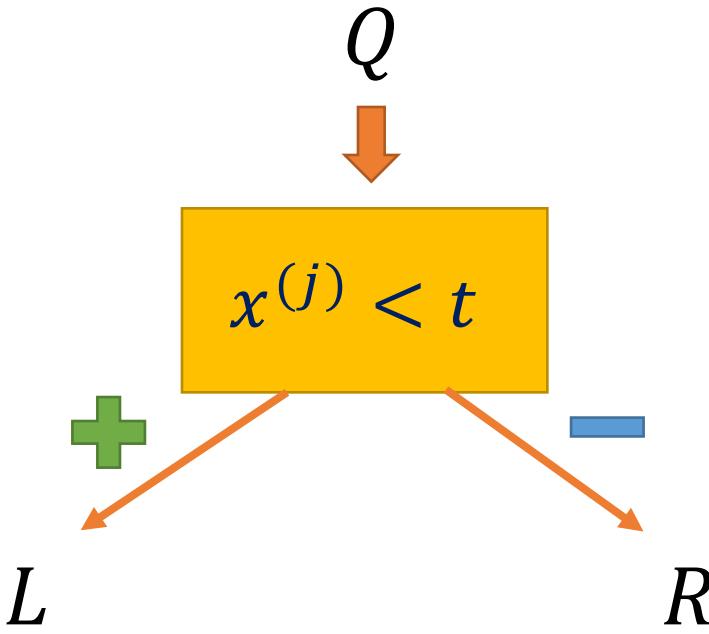
$$G(j, t) = \frac{|L|}{|Q|} H(L) + \frac{|R|}{|Q|} H(R)$$

# Выбор разбиения



$$G(j, t) = \frac{|L|}{|Q|} H(L) + \frac{|R|}{|Q|} H(R) \rightarrow \min_{j,t}$$

# Выбор разбиения



$$G(j, t) = \frac{|L|}{|Q|} H(L) + \frac{|R|}{|Q|} H(R) \rightarrow \min_{j,t}$$

$H(R)$  - мера «неоднородности» множества  $R$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

Пусть мы решаем задачу классификации на 2 класса,  
 $p_0, p_1$  – доли объектов классов 0 и 1 в  $R$

1) Misclassification criteria:  $H(R) = 1 - \max\{p_0, p_1\}$

2) Entropy criteria:  $H(R) = -p_0 \ln p_0 - p_1 \ln p_1$

3) Gini criteria:  $H(R) = 1 - p_0^2 - p_1^2 = 2p_0p_1$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

Пусть мы решаем задачу классификации на  $K$  классов,  
 $p_1, \dots, p_K$  – доли объектов классов 1, ...,  $K$  в  $R$

1) Misclassification criteria:  $H(R) = 1 - p_{max}$

2) Entropy criteria:

$$H(R) = - \sum_{k=1}^K p_k \ln p_k$$

3) Gini criteria:

$$H(R) = \sum_{k=1}^K p_k(1 - p_k)$$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

Чтобы решать задачу регрессии, достаточно взять среднеквадратичную ошибку в качестве  $H(R)$ :

$$H(R) = \frac{1}{|R|} \sum_{x_i \in R} (y_i - \bar{y})^2$$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

Чтобы решать задачу регрессии, достаточно взять среднеквадратичную ошибку в качестве  $H(R)$ :

$$H(R) = \frac{1}{|R|} \sum_{x_i \in R} (y_i - \bar{y})^2$$

$$\bar{y} = \frac{1}{|R|} \sum_{x_i \in R} y_i$$

# Pruning

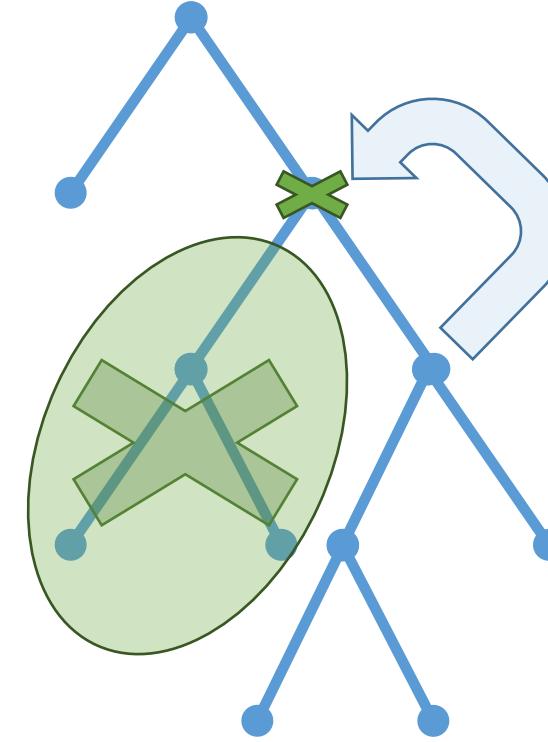
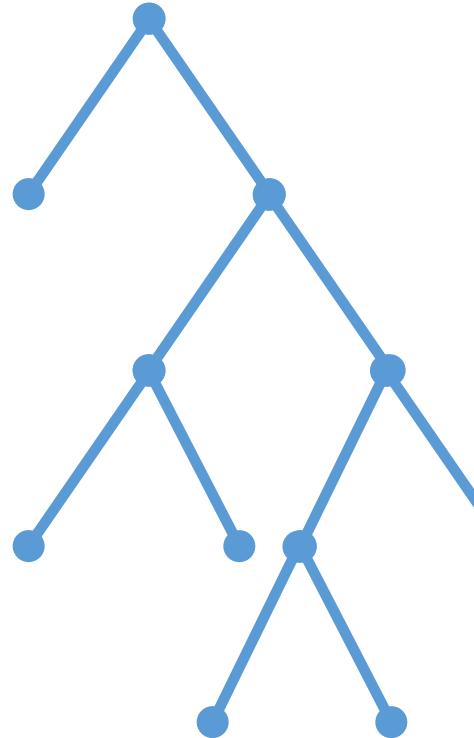
## Pre-prunning:

- Ограничеваем рост дерева до того как оно построено
- Если в какой-то момент информативность признаков в разбиении меньше порога – не разбиваем вершину

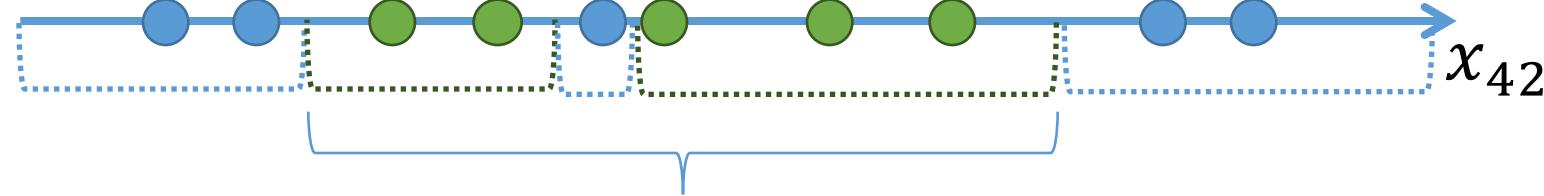
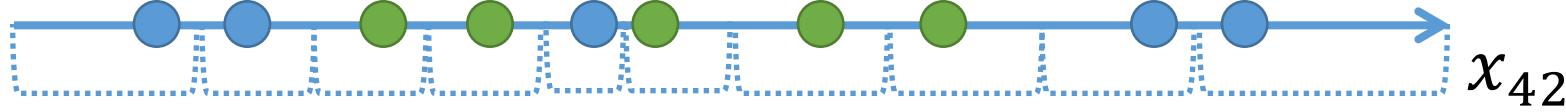
## Post-prunning:

- Упрощаем дерево после того как дерево построено

# Post-pruning



# Бинаризация



# Вариации алгоритма построения

- C4.5
- C5.0
- CART

# Итог

1. Что такое решающие деревья
2. Решающие деревья в классификации и регрессии
3. Как строить решающие деревья
4. Дополнительные темы

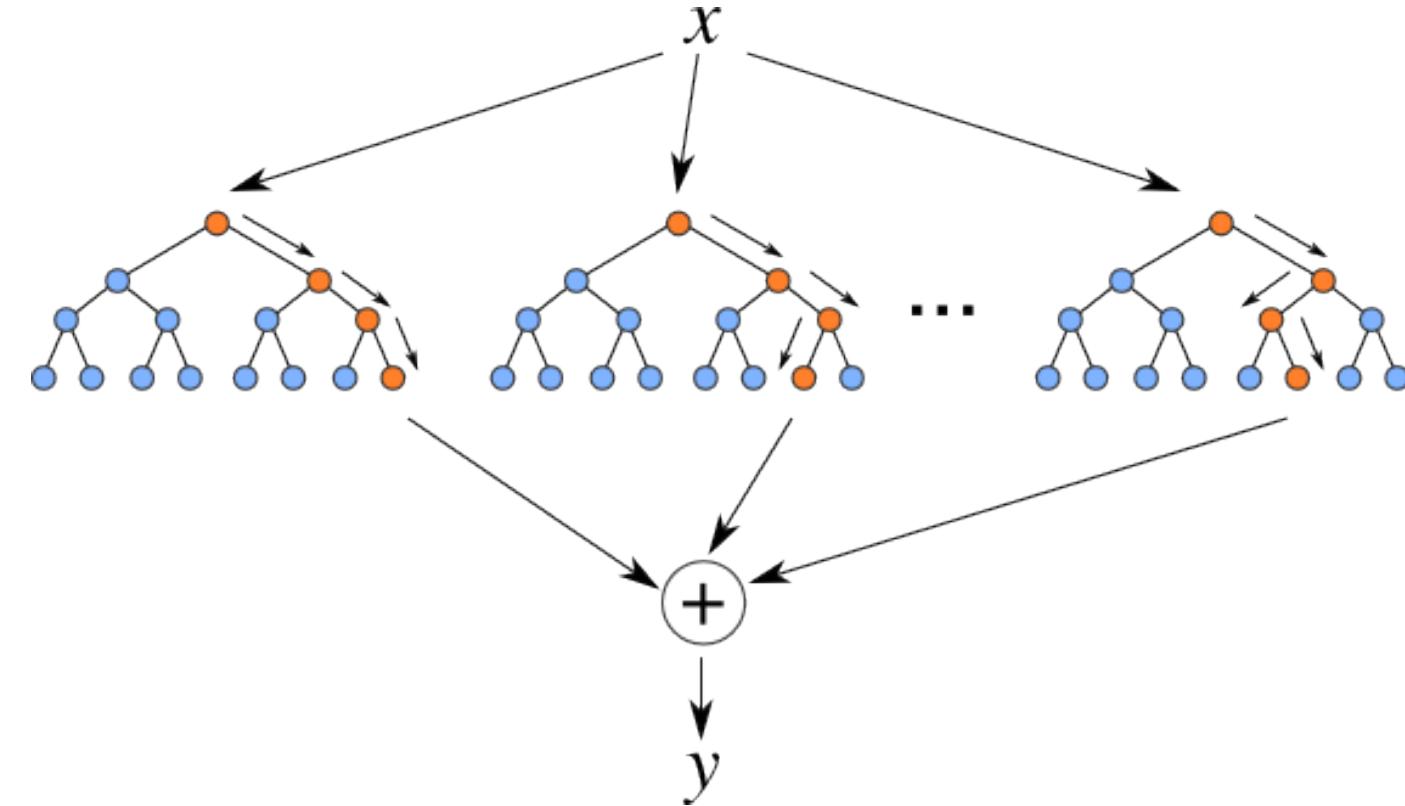
## II. Ансамбли решающих деревьев

# План

1. Random Forest
2. Идея Gradient Boosted Decision Trees (GBDT)
3. Библиотеки
4. Подробное обсуждение GBDT

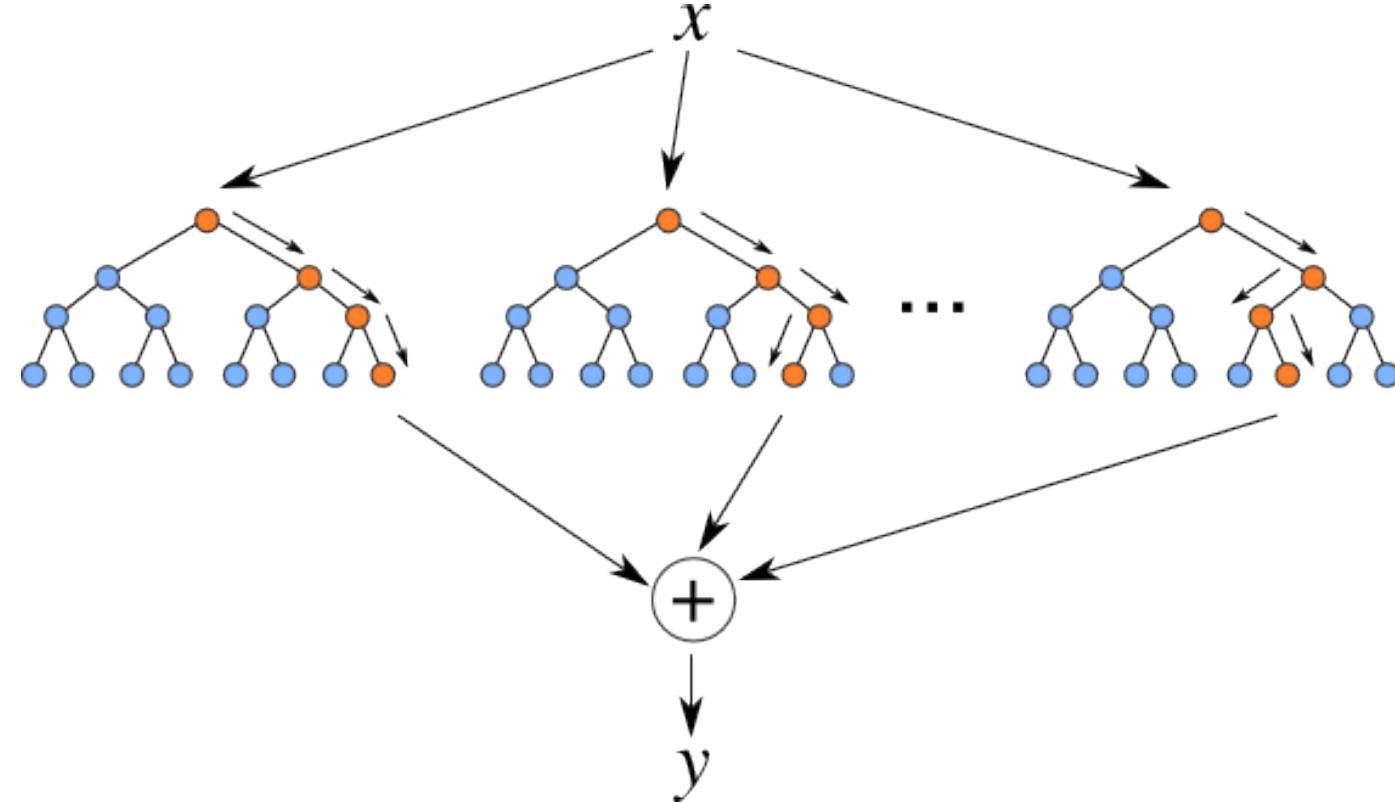
# Random Forest

1. Генерируем  $M$  выборок на основе имеющейся



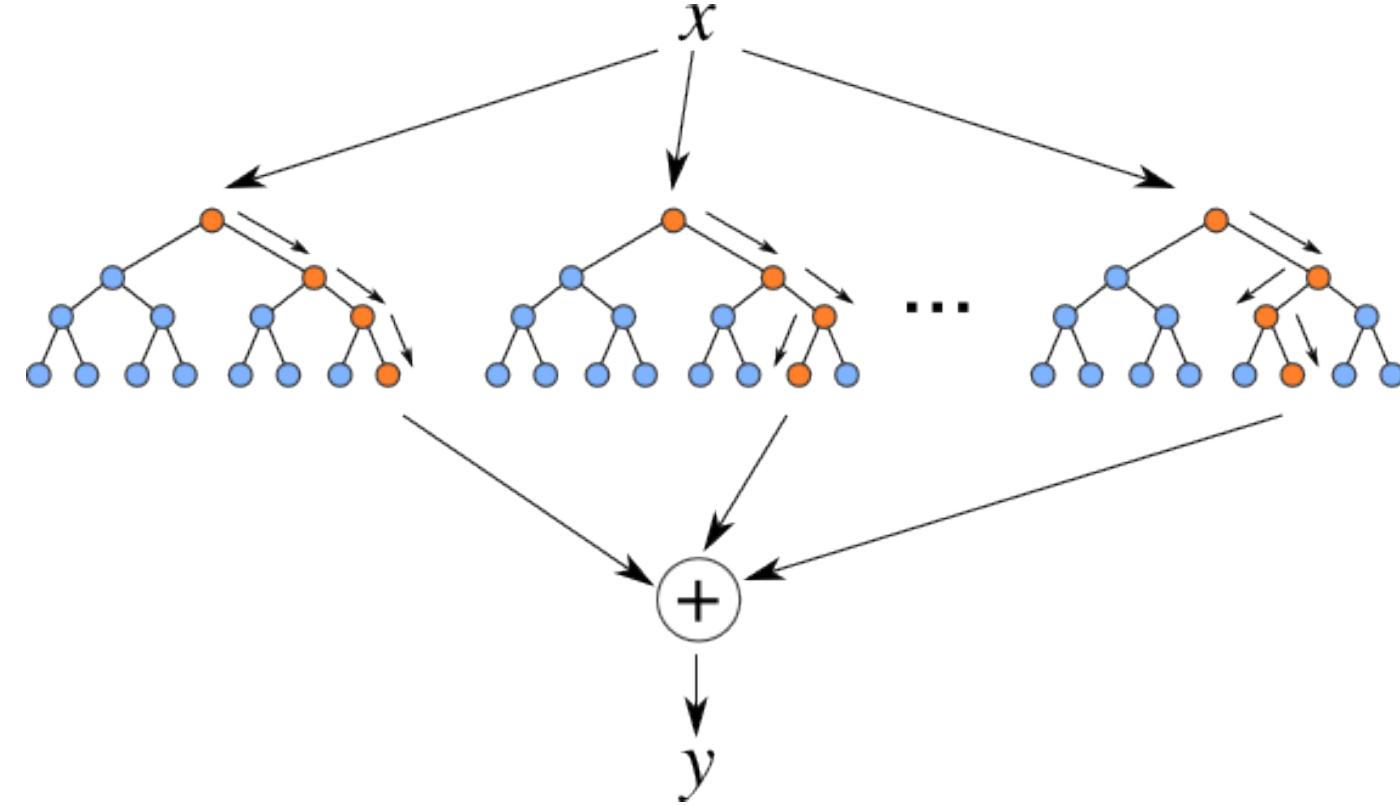
# Random Forest

1. Генерируем  $M$  выборок на основе имеющейся
2. Строим на них деревья с randomизированными разбиениями в узлах: выбираем  $k$  случайных признаков и ищем наиболее информативное разбиение по ним



# Random Forest

1. Генерируем  $M$  выборок на основе имеющейся
2. Строим на них деревья с randomизированными разбиениями в узлах: выбираем  $k$  случайных признаков и ищем наиболее информативное разбиение по ним
3. При прогнозе усредняем ответ всех деревьев

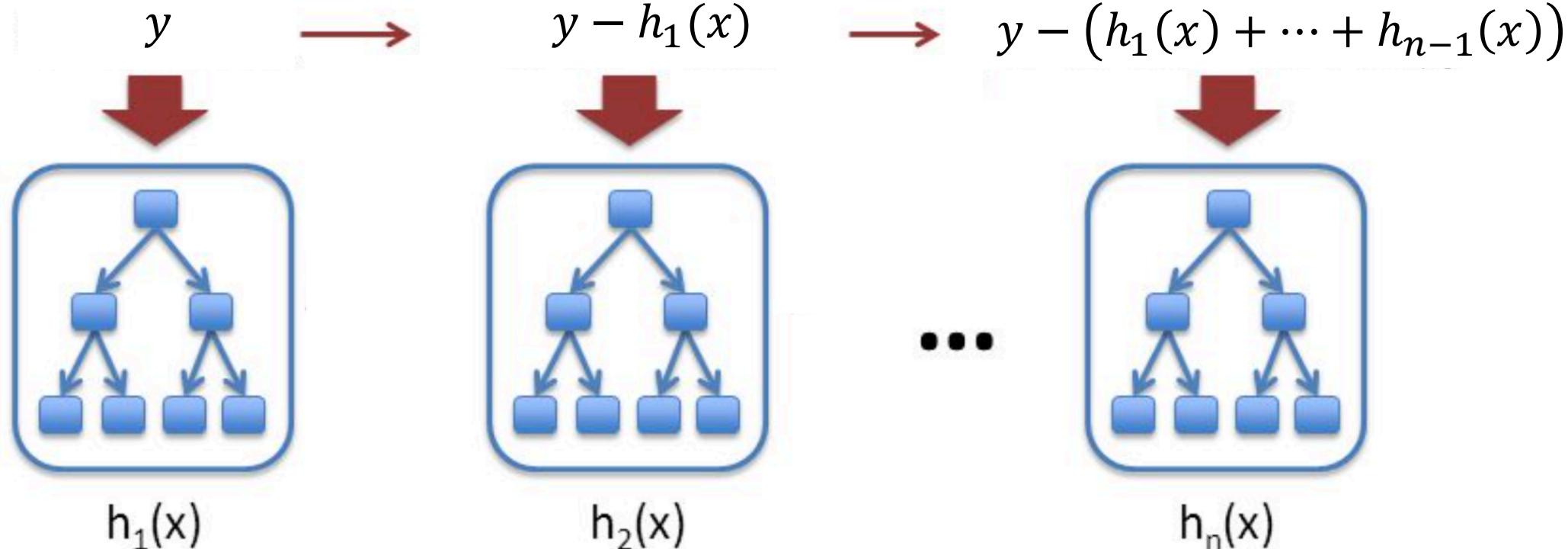


# Идея Gradient Boosted Decision Trees (GBDT)

$$h(x) = h_1(x) + \dots + h_n(x)$$

# Идея Gradient Boosted Decision Trees (GBDT)

$$h(x) = h_1(x) + \dots + h_n(x)$$

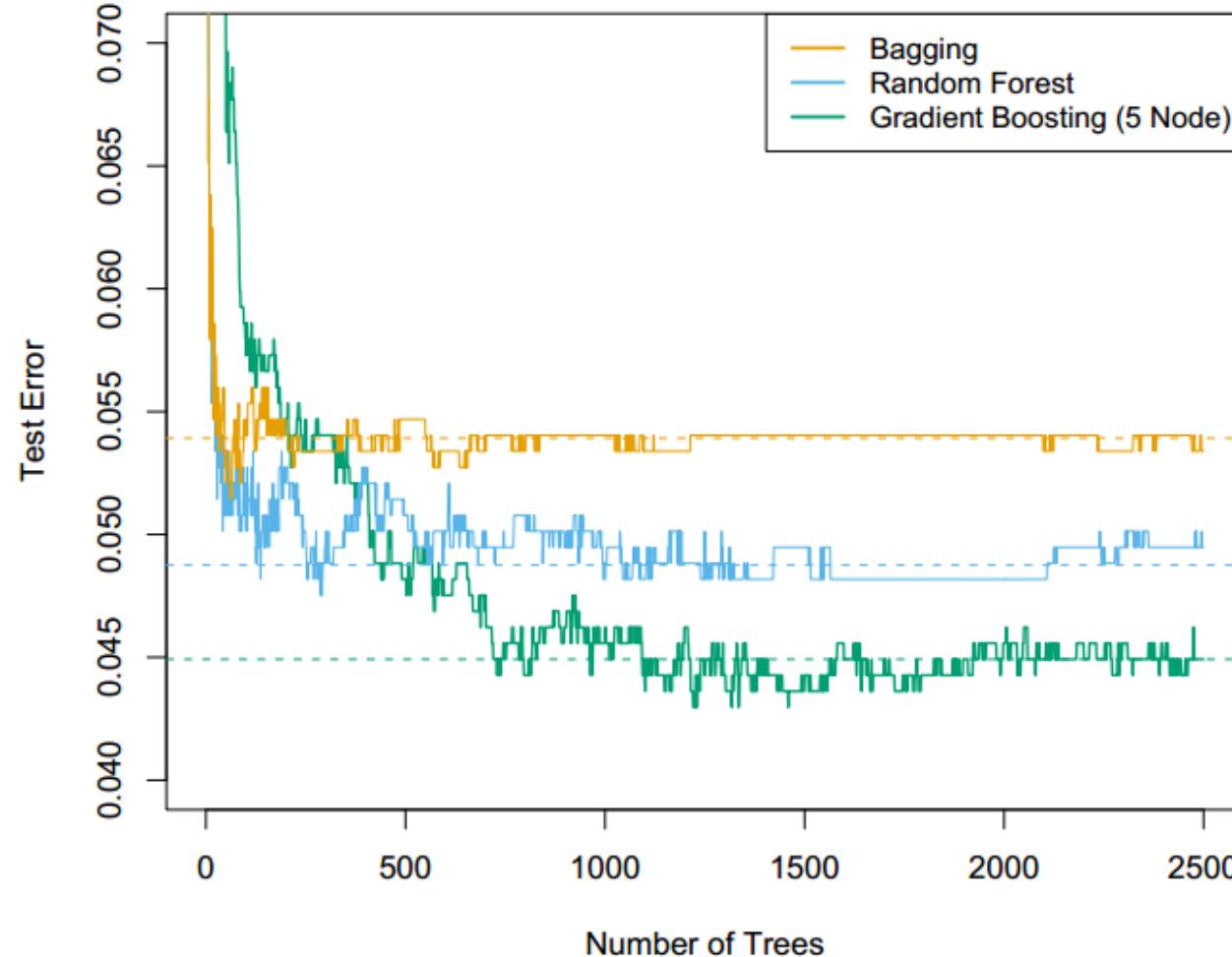


# Gradient Boosted Decision Trees

- Каждое новое дерево  $h_k(x)$  обучаем на ответы  $y_i - h_i$   
 $h_i$  - прогноз всей композиции на  $i$ -том объекте на предыдущей итерации
- Коэффициент  $\alpha_k$  перед новым деревом подбираем с помощью численной оптимизации ошибки

# GBDT и RF

Spam Data

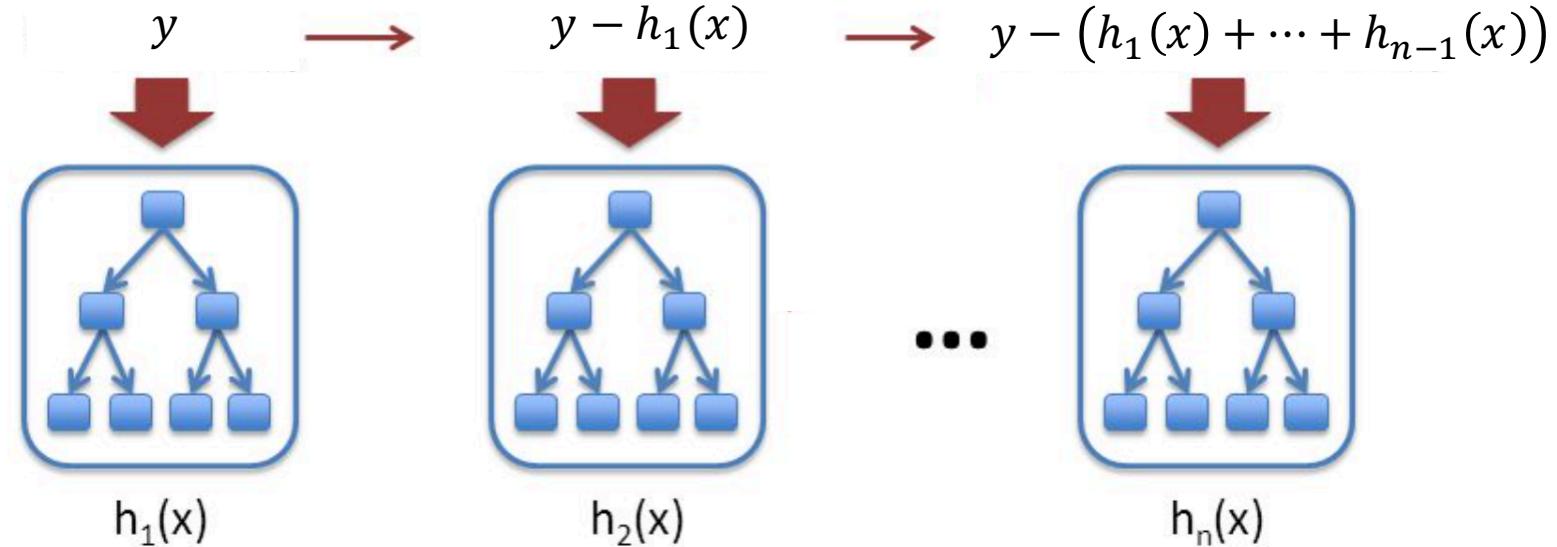


## Библиотеки

- Scikit-learn:
  - `sklearn.ensemble.RandomForestClassifier`
  - `sklearn.ensemble.RandomForestRegressor`
- XGBoost
- LightGBM

# Идея Gradient Boosted Decision Trees

$$a_n(x) = h_1(x) + \cdots + h_n(x)$$



## Аналогия с численной оптимизацией

Нам нужно минимизировать ошибку:

$$Q(\hat{y}, y) = \sum_{i=1}^l (\hat{y}_i - y_i)^2 \rightarrow \min \quad \hat{y}_i = a(x_i)$$

## Аналогия с численной оптимизацией

Нам нужно минимизировать ошибку:

$$Q(\hat{y}, y) = \sum_{i=1}^l (\hat{y}_i - y_i)^2 \rightarrow \min \quad \hat{y}_i = a(x_i)$$

Если бы мы подбирали ответы  $\hat{y}$  итеративно, можно было бы это делать градиентным спуском

## Аналогия с численной оптимизацией

Нам нужно минимизировать ошибку:

$$Q(\hat{y}, y) = \sum_{i=1}^l (\hat{y}_i - y_i)^2 \rightarrow \min \quad \hat{y}_i = a(x_i)$$

Если бы мы подбирали ответы  $\hat{y}$  итеративно, можно было бы это делать градиентным спуском

Но нам нужно подобрать не ответы, а функцию  $a(x)$

# Градиентный бустинг и градиент

В бустинге

$$a(x) = \sum_{t=1}^T \beta_t h_t(x)$$

**Идея:** будем каждый следующий алгоритм выбирать так, чтобы он приближал антиградиент ошибки

$$h_t(x) \approx -\frac{\partial Q(\hat{y}, y)}{\partial \hat{y}}$$

## Градиентный бустинг и градиент

Если  $h_t(x) \approx -\frac{\partial Q(\hat{y}, y)}{\partial \hat{y}}$  и  $Q(\hat{y}, y) = \sum_{i=1}^l (\hat{y}_i - y_i)^2$

$$h_t(x_i) \approx -\frac{\partial Q(\hat{y}_i, y_i)}{\partial \hat{y}_i} = -2(\hat{y}_i - y_i) \propto y_i - \hat{y}_i$$

## GBM с квадратичными потерями

1. Обучаем первый базовый алгоритм  $h_1$ ,  $\beta_1 = 1$
2. Повторяем в цикле по  $t$  от 2 до  $T$ :

обучаем  $h_t$  на ответы  $y_i - a_{t-1}(x_i)$

выбираем  $\beta_t$

## GBM с квадратичными потерями

1. Обучаем первый базовый алгоритм  $h_1$ ,  $\beta_1 = 1$
2. Повторяем в цикле по  $t$  от 2 до  $T$ :

обучаем  $h_t$  на ответы  $y_i - a_{t-1}(x_i)$

выбираем  $\beta_t$

Стратегии выбора  $\beta_t$ :

- всегда равен небольшой константе
- как в методе наискорейшего спуска
- уменьшая с ростом  $t$

## GBM с произвольными потерями

1. Обучаем первый базовый алгоритм  $h_1$ ,  $\beta_1 = 1$
2. Повторяем в цикле по  $t$  от 2 до  $T$ :

обучаем  $h_t$  на  $-\frac{\partial Q(\hat{y}_i, y_i)}{\partial \hat{y}_i} = -\frac{\partial L(\hat{y}_i, y_i)}{\partial \hat{y}_i}$

выбираем  $\beta_t$

$$\text{Здесь } Q(\hat{y}, y) = \sum_{i=1}^l L(\hat{y}_i, y_i) \quad \hat{y}_i = a_{t-1}(x_i)$$

## GBM в наиболее общем виде

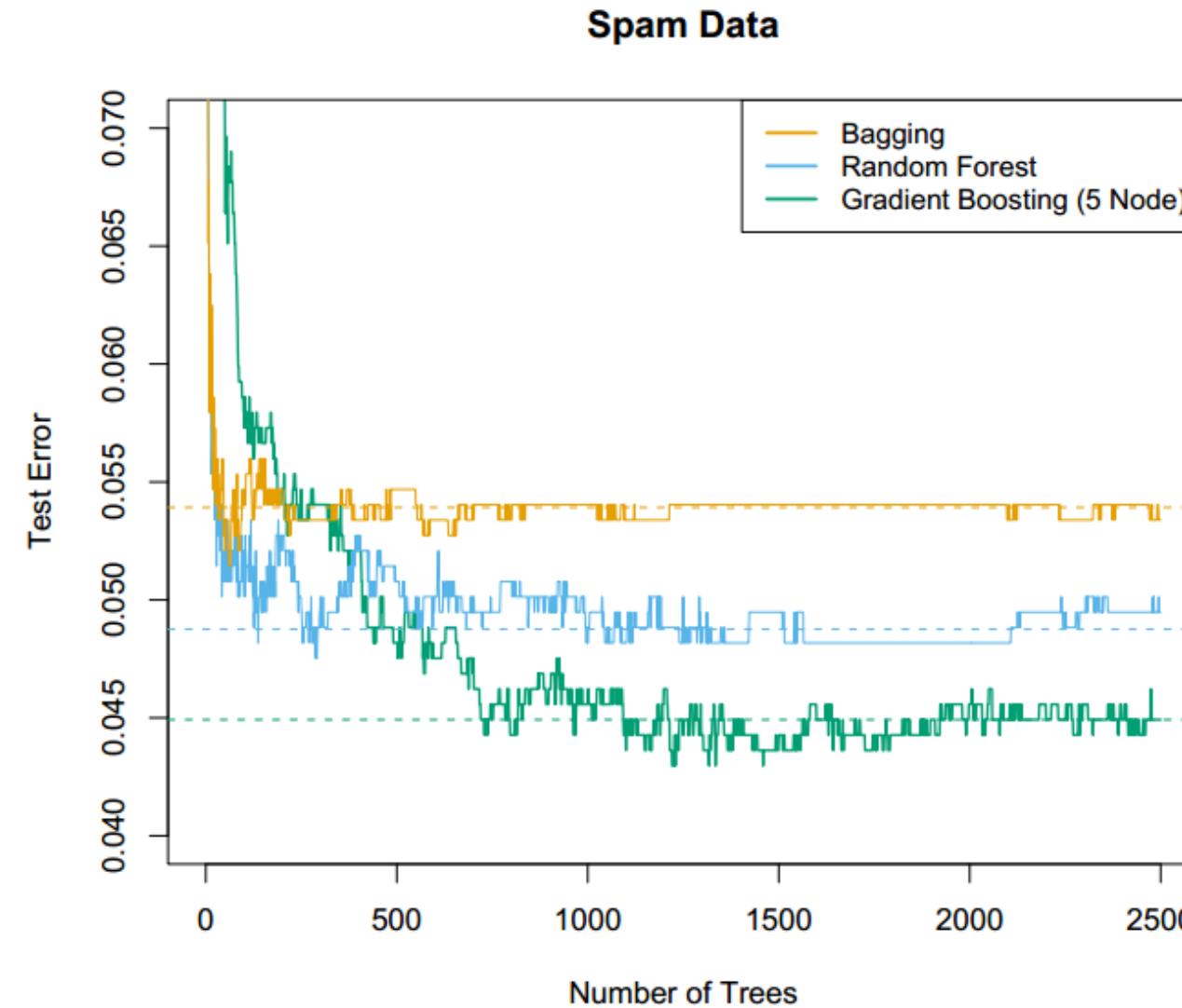
1. Обучаем первый базовый алгоритм  $h_1$ ,  $\beta_1 = 1$
2. Повторяем в цикле по  $t$  от 2 до  $T$ :

$$h_t = \operatorname{argmin}_h \sum_{i=1}^l \tilde{L} \left( h(x_i), -\frac{\partial L(\hat{y}_i, y_i)}{\partial \hat{y}_i} \right)$$

выбираем  $\beta_t$

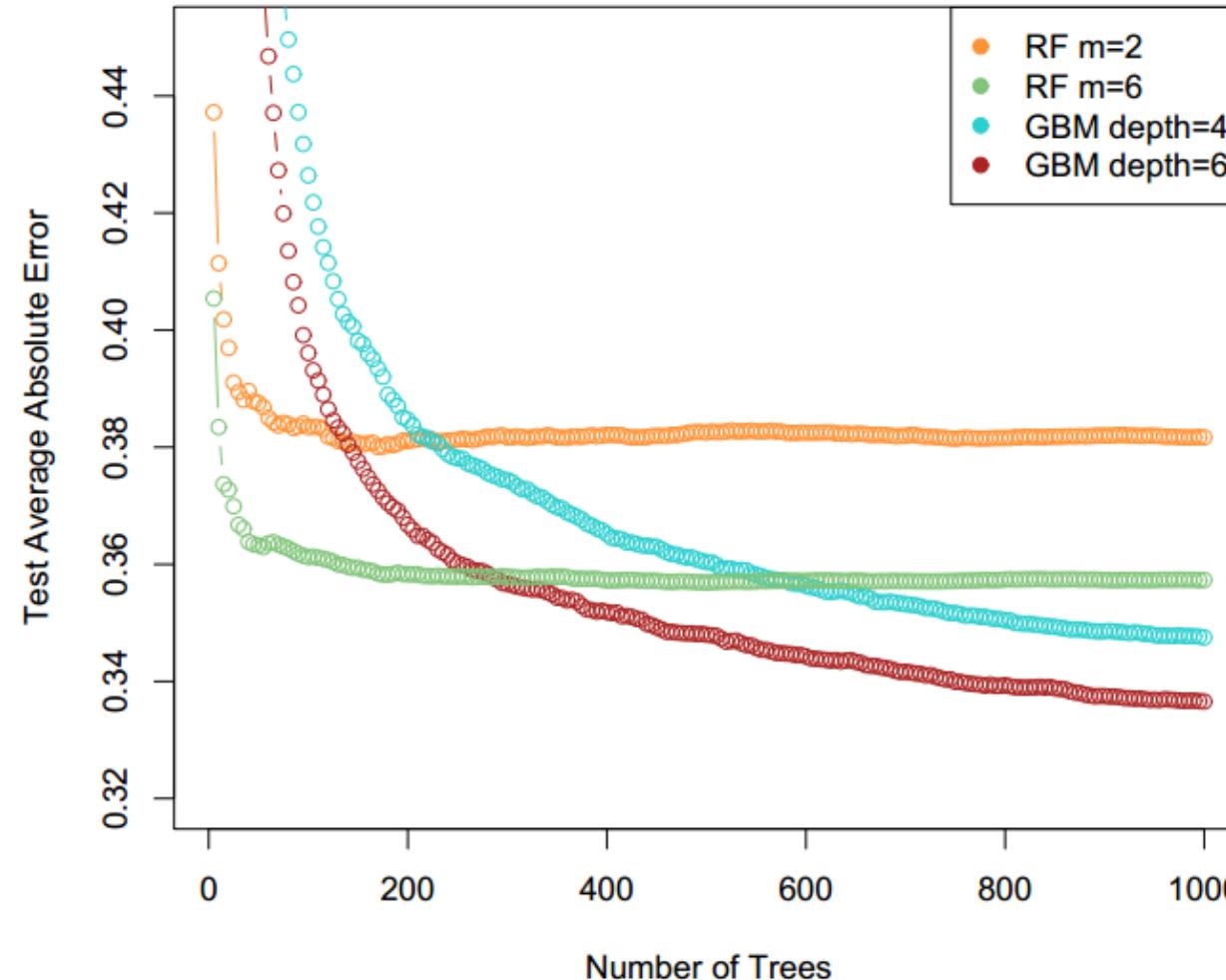
$$\text{Здесь } Q(\hat{y}, y) = \sum_{i=1}^l L(\hat{y}_i, y_i) \quad \hat{y}_i = a_{t-1}(x_i)$$

# Bagging, Random Forest, GBDT



# GTBM и RF

California Housing Data



# Распараллеливание

**Вопрос для обсуждения:**

Какой из ансамблей деревьев больше подходит для распараллеливания? Как это делать в одном и в другом случае?

# Итог

1. Random Forest
2. Gradient Boosted Decision Trees (GBDT)
3. Библиотеки

### **III. Общие идеи построения ансамблей**

# Bagging

Bagging = Bootstrap aggregation

# Бутстрап

Выборка:

№	X
1	3.4
2	2.9
3	3.7
N	3.1

# Бутстреп

Выборка:

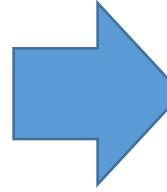
№	X
1	3.4
2	2.9
3	3.7
N	3.1

$$\mathbb{E} X = 3.3 \pm ?$$

# Бутстреп

Выборка:

№	X
1	3.4
2	2.9
3	3.7
N	3.1



№	X
3	3.7
1	3.4
2	2.9
2	3.1

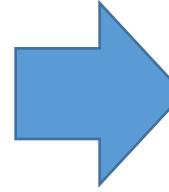
Генерируем новую  
(искусственную) выборку,  
отбрав в нее объекты из  
исходной выборки по  
схеме выбора с  
возвращением

$$\mathbb{E} X = 3.3 \pm ?$$

# Бутстреп

Выборка:

№	X
1	3.4
2	2.9
3	3.7
N	3.1



№	X
3	3.7
1	3.4
2	2.9
2	3.1

№	X
1	3.4
3	3.7
2	2.9
M	3.0

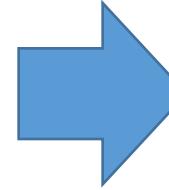
Продолжаем  
генерировать  
такие выборки

$$\mathbb{E} X = 3.3 \pm ?$$

# Бутстреп

Выборка:

№	X
1	3.4
2	2.9
3	3.7
N	3.1



№	X
3	3.7
1	3.4
2	2.9
2	2.9

№	X
1	3.4
3	3.7
2	2.9
M	3.0

...

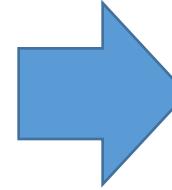
№	X
3	3.7
2	2.9
1	3.4
1	3.4

$$\mathbb{E} X = 3.3 \pm ?$$

# Бутстреп

Выборка:

№	X
1	3.4
2	2.9
3	3.7
N	3.1



№	X
3	3.7
1	3.4
2	2.9
2	2.9

№	X
1	3.4
3	3.7
2	2.9
M	3.0

...

№	X
3	3.7
2	2.9
1	3.4
1	3.4

$$\mathbb{E} X = 3.31 \pm ?$$

$$\mathbb{E} X = 3.25$$

$$\mathbb{E} X = 3.27$$

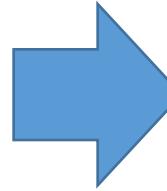
...

$$\mathbb{E} X = 3.39$$

# Бутстреп

Выборка:

№	X
1	3.4
2	2.9
3	3.7
N	3.1



№	X
3	3.7
1	3.4
2	2.9
2	2.9

№	X
1	3.4
3	3.7
2	2.9
M	3.0

...

№	X
3	3.7
2	2.9
1	3.4
1	3.4

$$\mathbb{E} X = 3.31 \pm ?$$

$$\mathbb{E} X = 3.25$$

$$\mathbb{E} X = 3.27$$

...

$$\mathbb{E} X = 3.39$$

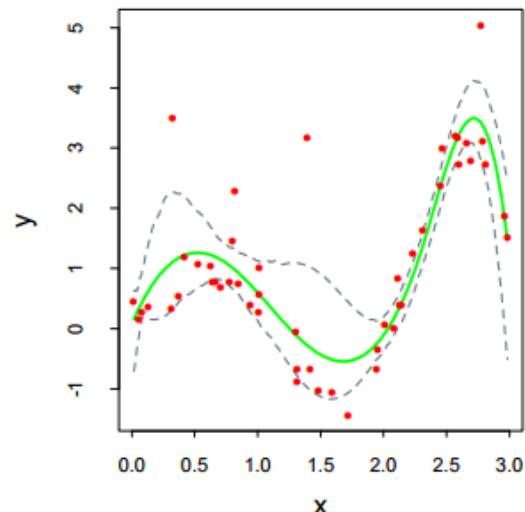
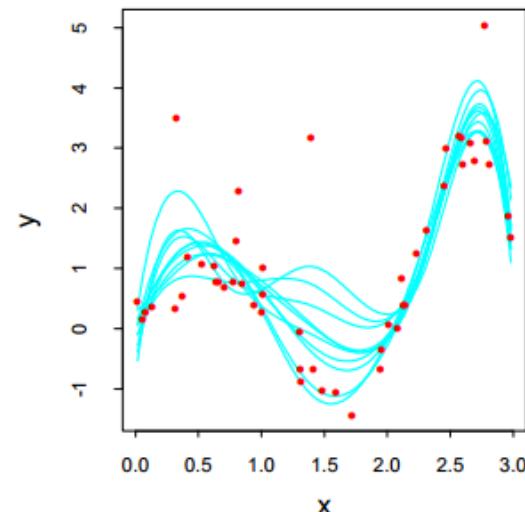
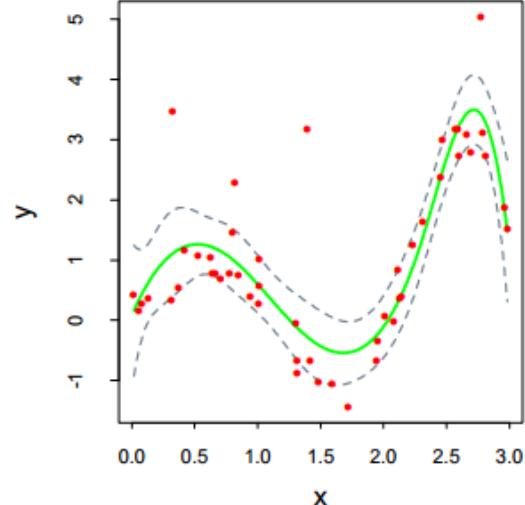
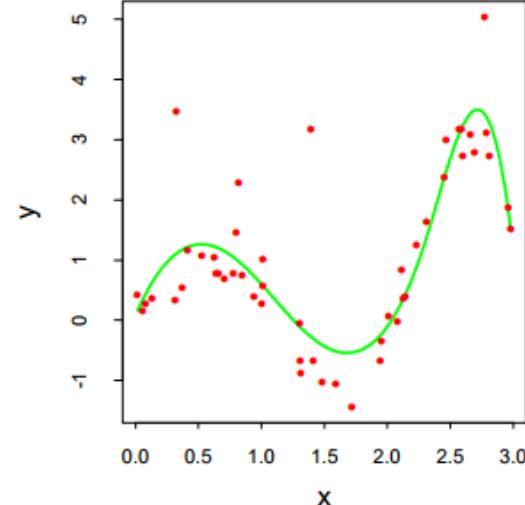
$$\mathbb{E} X = 3.32 \pm 0.06$$

# Bagging

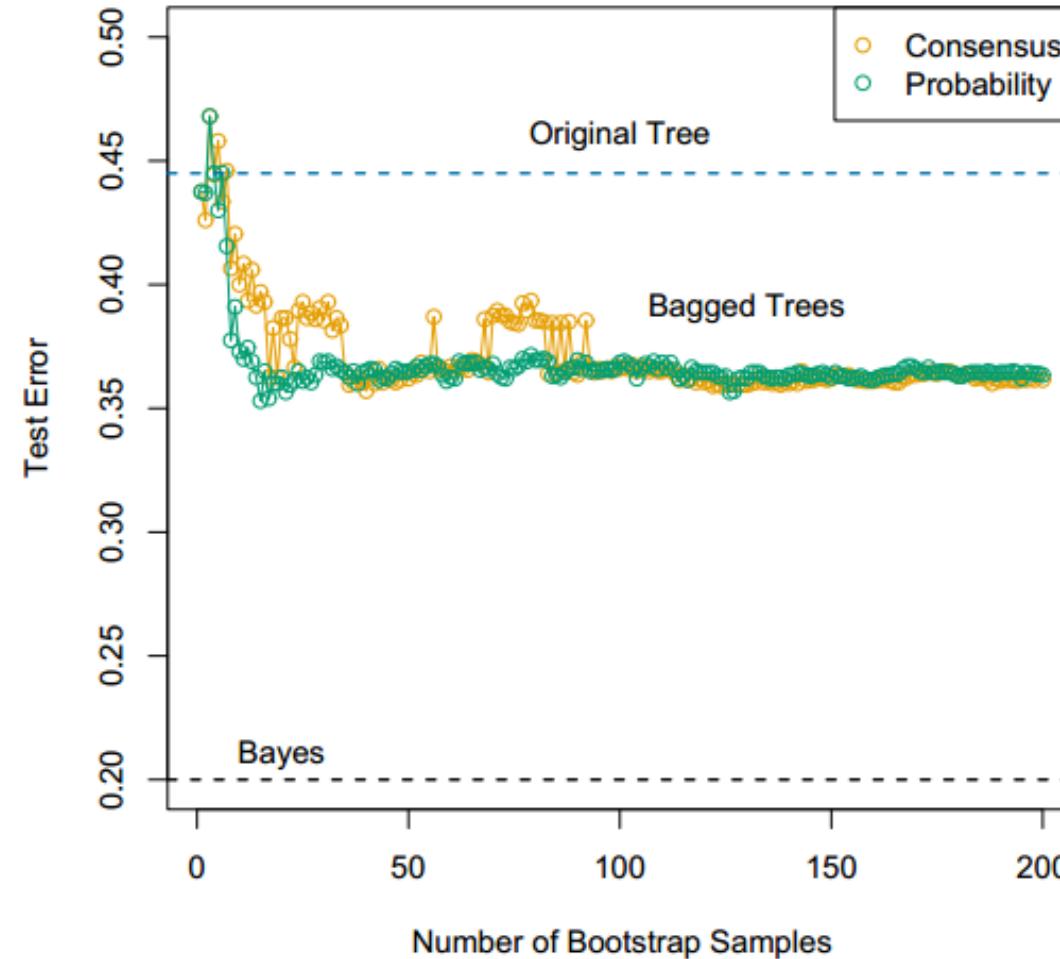
Bagging = Bootstrap aggregation

По схеме выбора с возвращением, генерируем  $M$  обучающих выборок такого же размера, обучаем на них модели и усредняем

# Bagging



# Бэггинг в классификации



## Вариации: Pasting, RSM

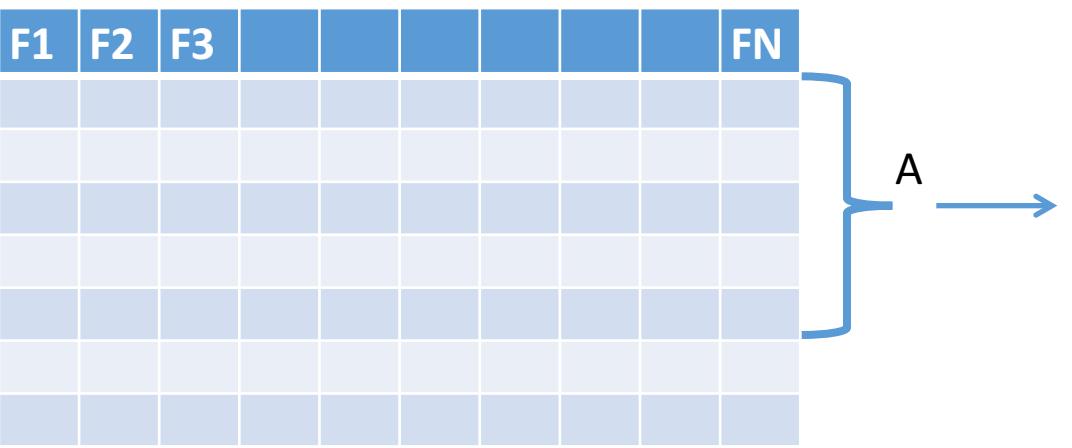
- RSM – Random Subspace Method, выбираем не объекты, а признаки
- Pasting – выбираем объекты без возвращения

# Stacking

## Обучающая выборка:

# Stacking

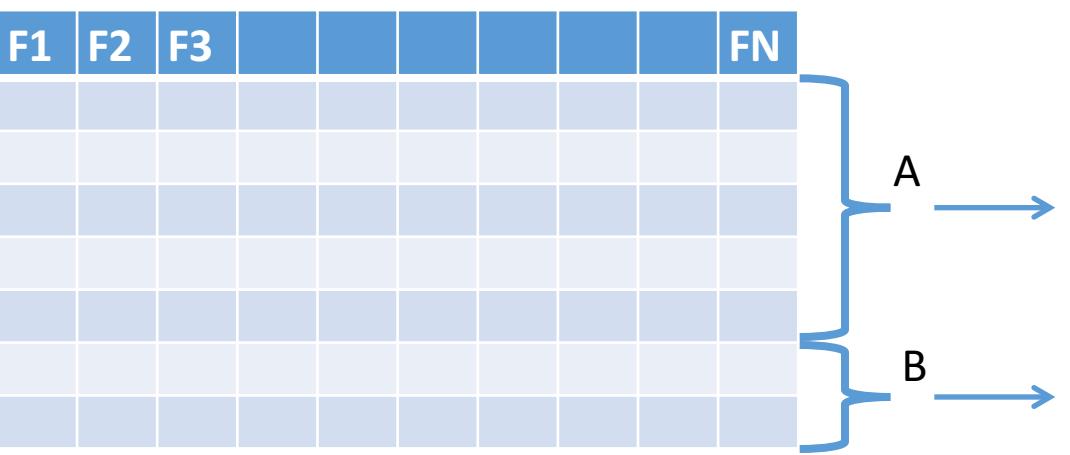
Обучающая выборка:



Обучающая выборка для M  
базовых алгоритмов

# Stacking

Обучающая выборка:

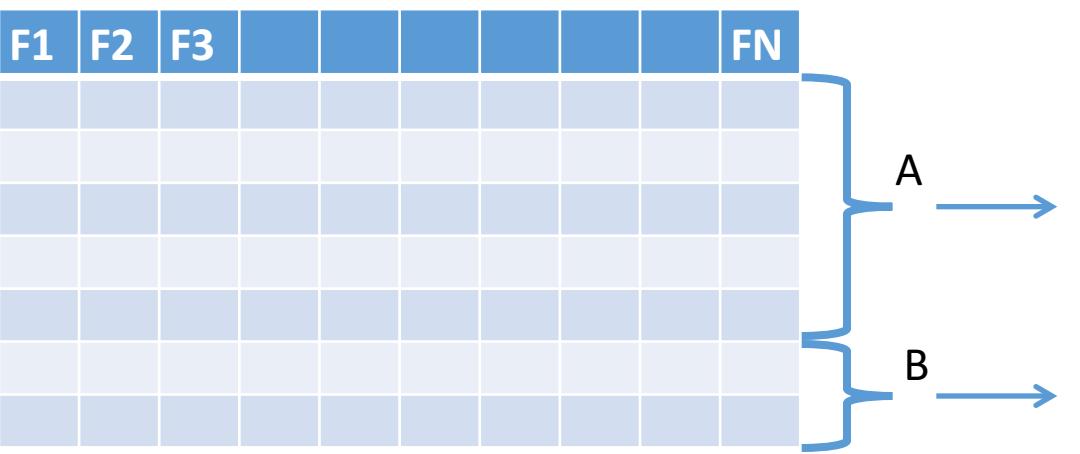


Обучающая выборка для M  
базовых алгоритмов

Обучающая выборка для  
другой модели

# Stacking

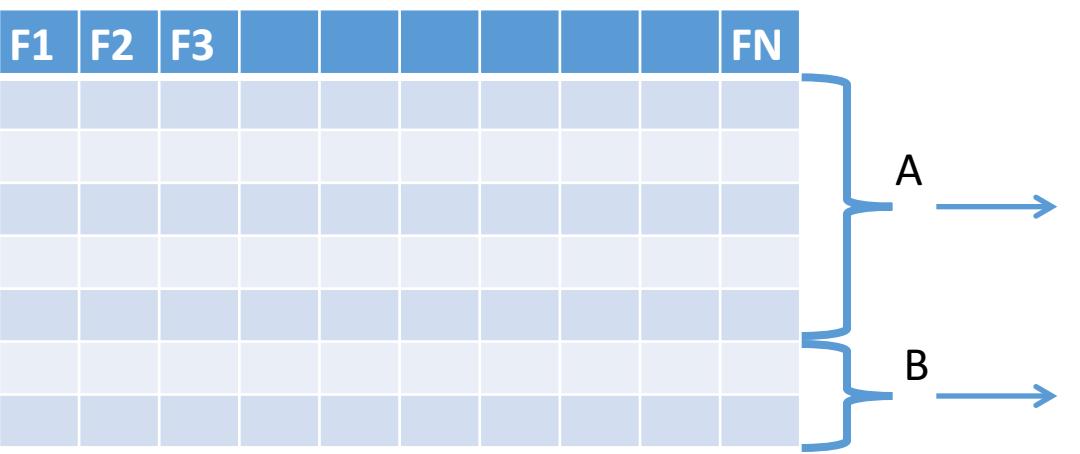
Обучающая выборка:



Обучаем M  
базовых  
алгоритмов на  
выборке А

# Stacking

Обучающая выборка:

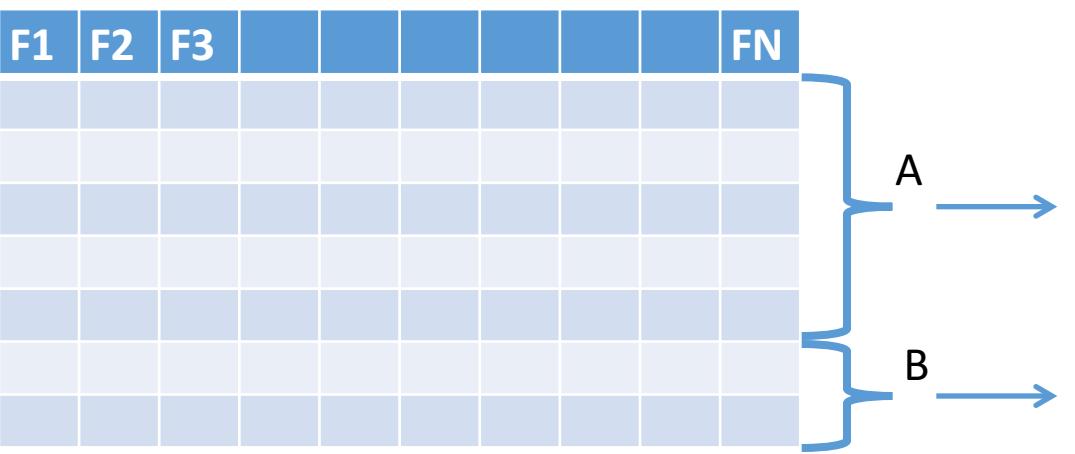


Обучаем M базовых алгоритмов на выборке А

Считаем их прогнозы на выборке В

# Stacking

Обучающая выборка:



Обучающая выборка для M  
базовых алгоритмов

Обучающая выборка для  
другой модели

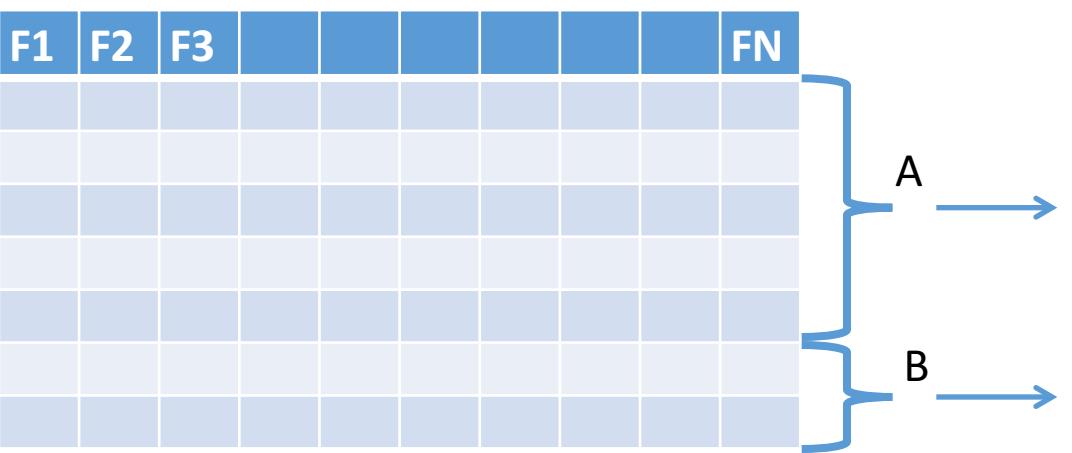
Обучаем M  
базовых  
алгоритмов на  
выборке А

Считаем их  
прогнозы на  
выборке В

B1	B2		BM

# Stacking

Обучающая выборка:



Обучаем M базовых алгоритмов на выборке А

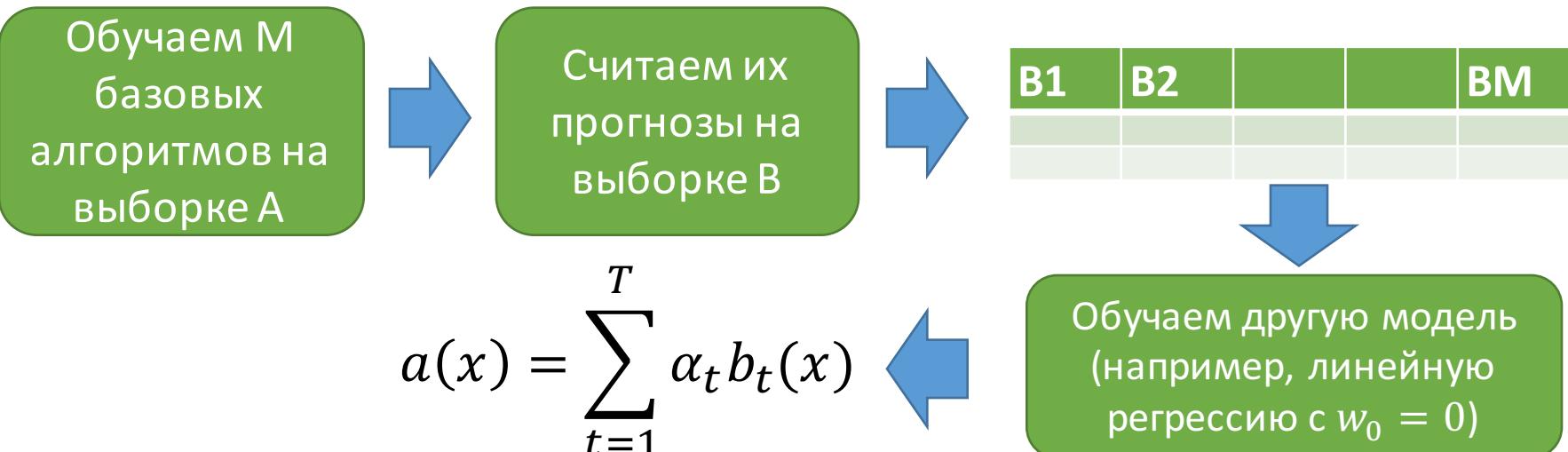
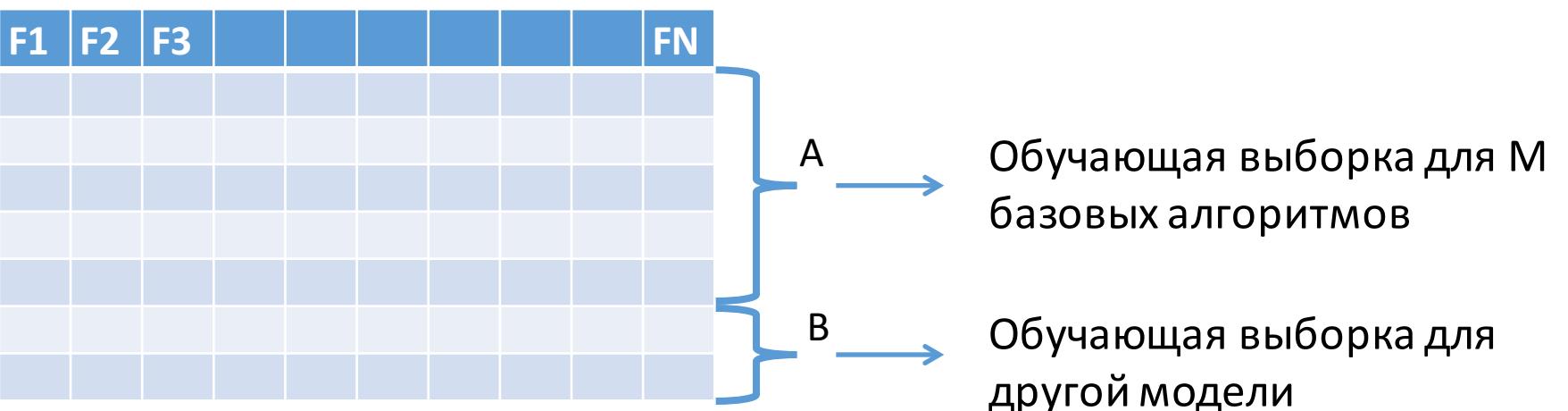
Считаем их прогнозы на выборке В

B1	B2		BM

Обучаем другую модель  
(например, линейную  
регрессию с  $w_0 = 0$ )

# Stacking

Обучающая выборка:



# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

Преимущества и недостатки:

- Очень прост идеально, хорошо работает, логичен

# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

Преимущества и недостатки:

- Очень прост идеально, хорошо работает, логичен
- Иногда надо перебирать веса или использовать дискретную оптимизацию

# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

Преимущества и недостатки:

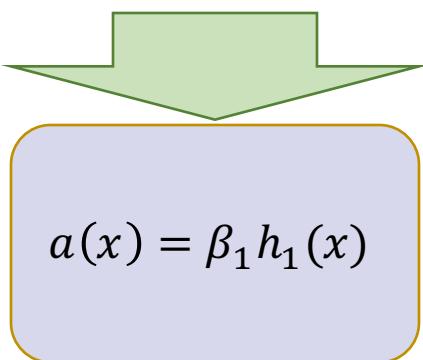
- Очень прост идеально, хорошо работает, логичен
- Иногда надо перебирать веса или использовать дискретную оптимизацию
- Не всегда композиция в виде взвешенной суммы – то, что надо. Иногда нужна более сложная композиция

# Boosting

Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$

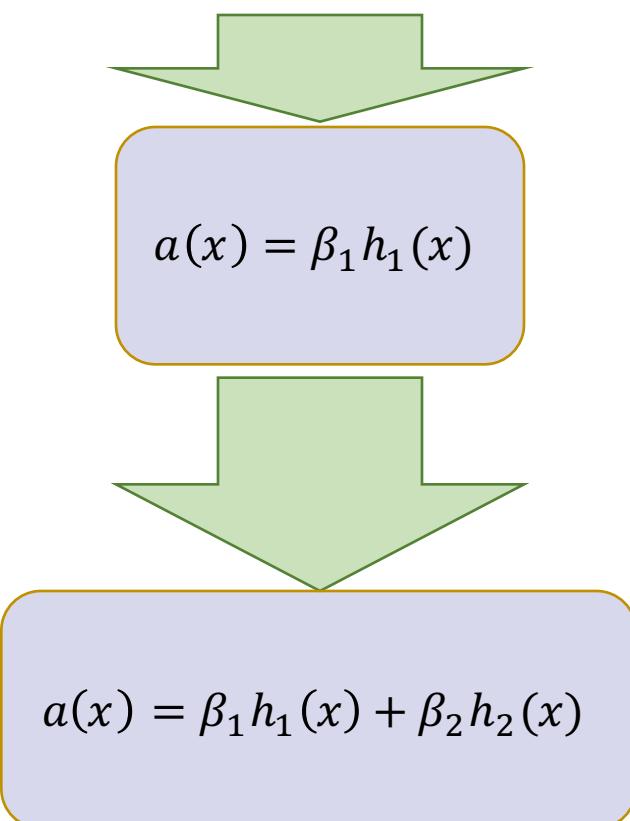
# Boosting

Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$


$$a(x) = \beta_1 h_1(x)$$

# Boosting

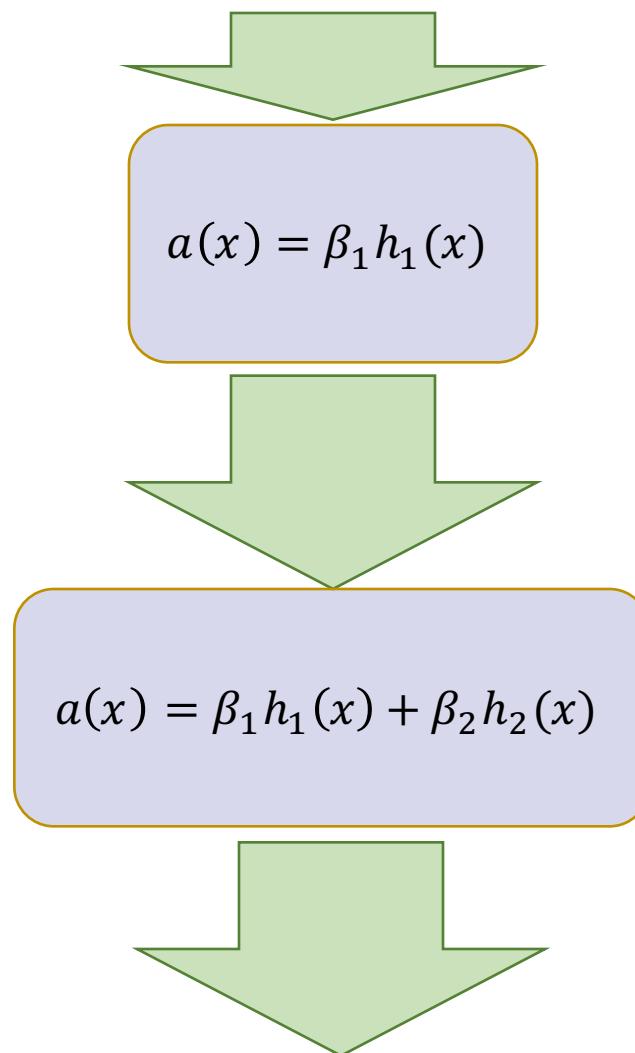
Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$



# Boosting

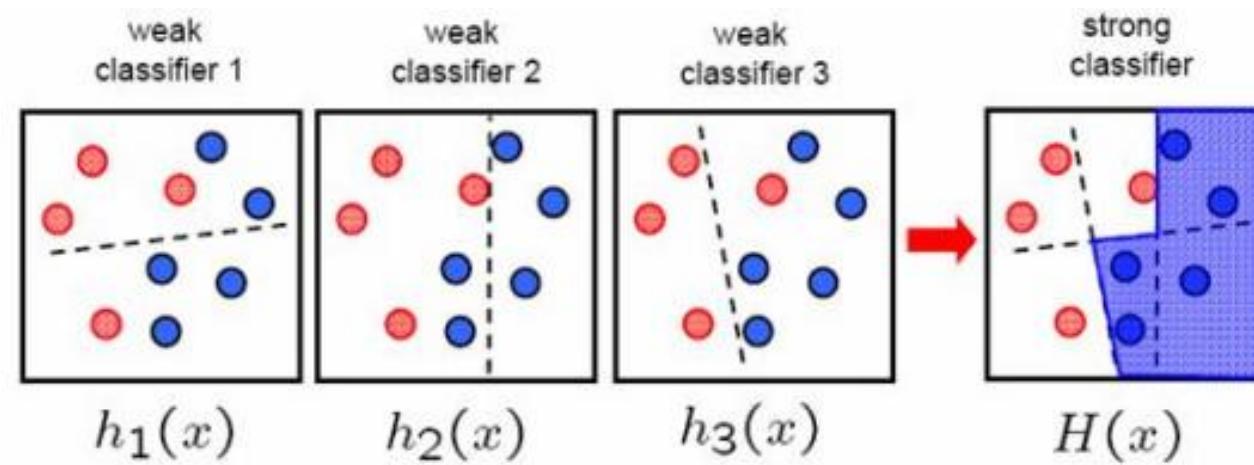
Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$

$$a(x) = \sum_{t=1}^T \beta_t h_t(x)$$

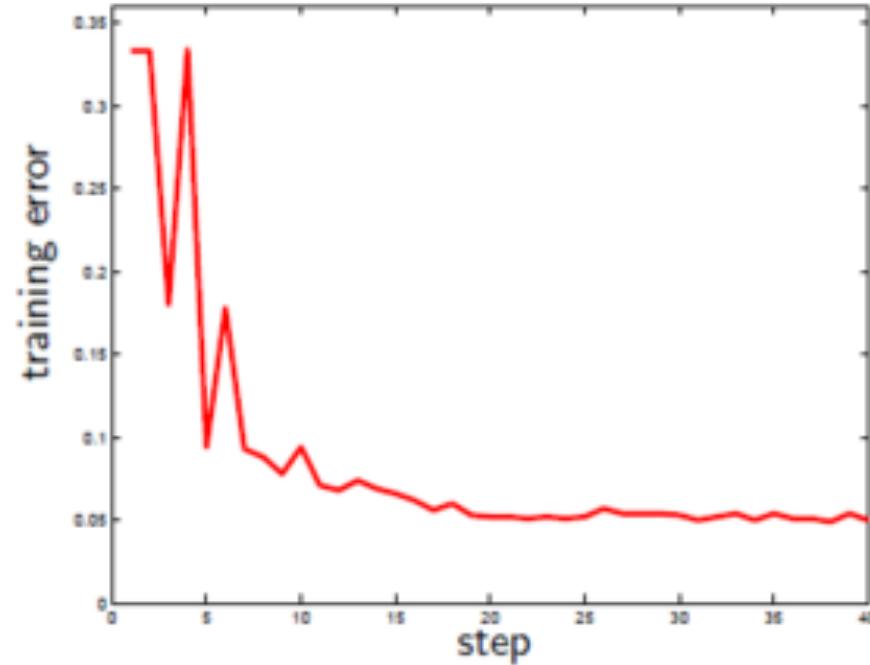
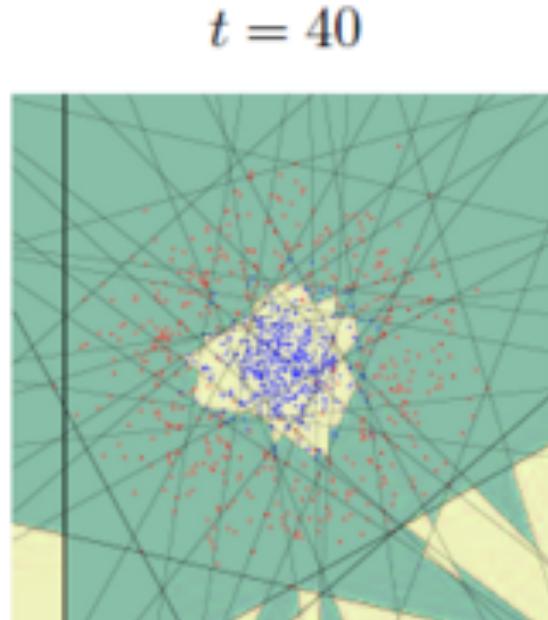


# «Слабые» алгоритмы

$h_k(x)$  – как правило, решающие деревья небольшой глубины или линейные модели



# Пример: бустинг над линейными классификаторами

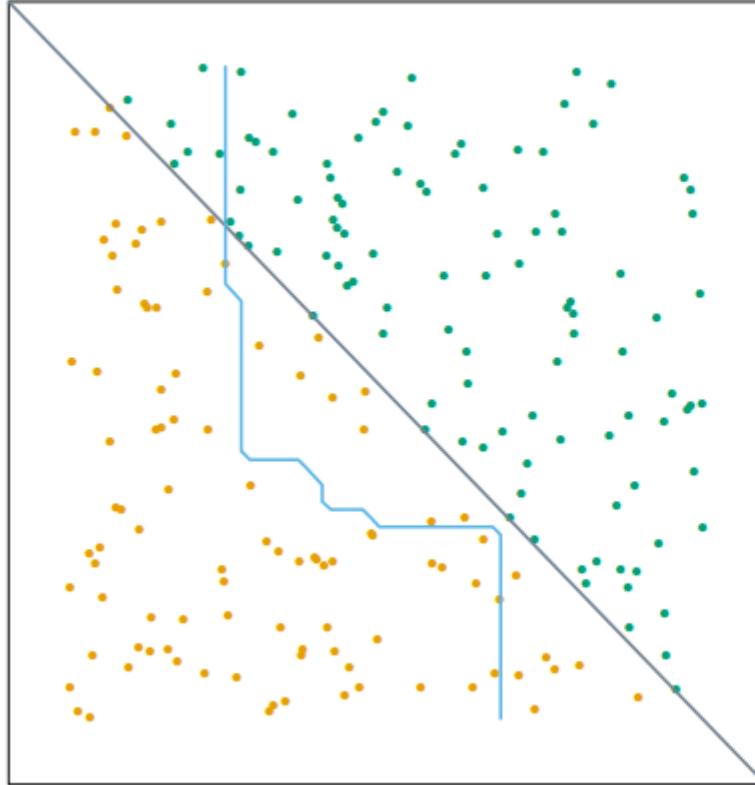


# Алгоритмы бустинга

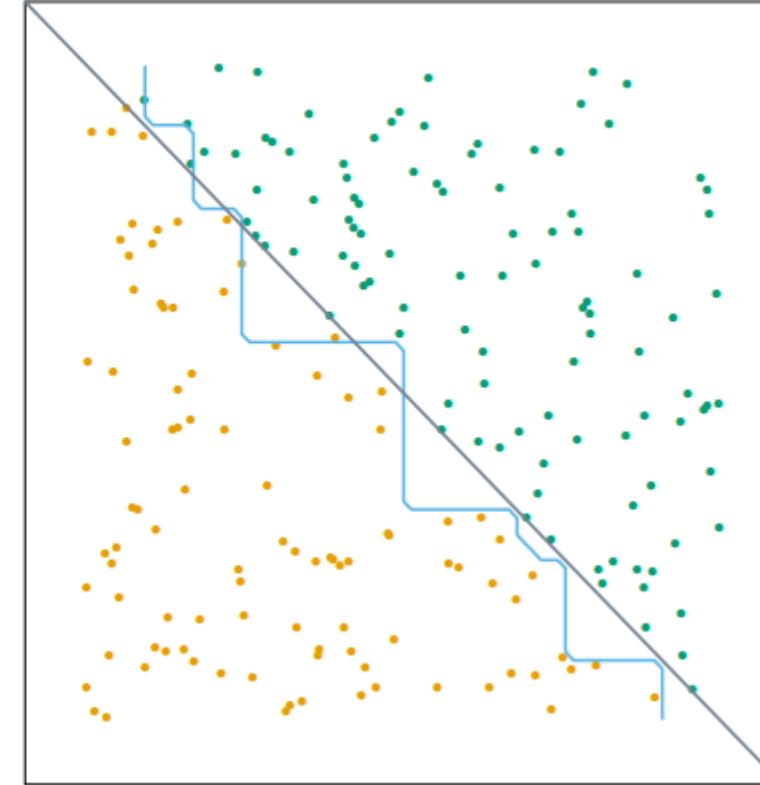
- Основные алгоритмы:
  - Градиентный бустинг
  - Адаптивный бустинг (AdaBoost)
- Вариации AdaBoost:
  - AnyBoost (произвольная функция потерь)
  - BrownBoost
  - GentleBoost
  - LogitBoost
  - ....

# Бэггинг и бустинг

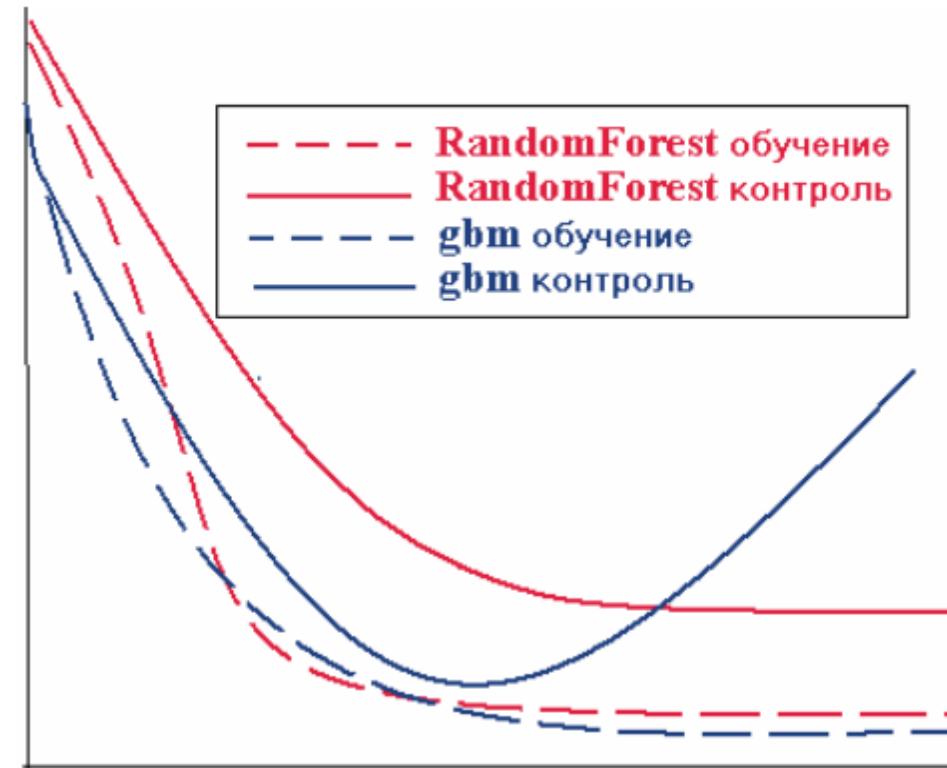
Bagged Decision Rule



Boosted Decision Rule



# Бэггинг и бустинг: переобучение



# Преимущества и недостатки бустинга

- Позволяет очень точно приблизить восстанавливаемую функцию или разделяющую поверхность классов
- Плохо интерпретируем
- Композиции могут содержать десятки тысяч базовых моделей и долго обучаться
- Переобучение на выбросах при избыточном количестве классификаторов

# Резюме

- I. Решающие деревья
- II. Ансамбли деревьев
- III. Общие идеи построения ансамблей

# Резюме

- I. Решающие деревья
- II. Ансамбли деревьев
- III. Общие идеи построения ансамблей
  - 1) Bagging (+Pasting & RSM)
  - 2) Stacking
  - 3) Blending
  - 4) Boosting

# Спасибо за внимание



[dmia@applieddatascience.ru](mailto:dmia@applieddatascience.ru)



[https://t.me/joinchat/B10lThC96v0BQCvs\\_joNew](https://t.me/joinchat/B10lThC96v0BQCvs_joNew)



[https://github.com/vkantor/ml2018jan\\_feb](https://github.com/vkantor/ml2018jan_feb)

# eXtreme Gradient Boosting (XGBoost)

<https://arxiv.org/pdf/1603.02754.pdf>

# eXtreme Gradient Boosting (XGBoost)

$$\sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + b(x_i)) \rightarrow \min_b$$

$$s = \left( - \frac{\partial L}{\partial z} \Big|_{z=a_{N-1}(x_i)} \right)_{i=1}^{\ell} = -\nabla_s \sum_{i=1}^{\ell} L(y_i, a_{N-1}(x_i) + s_i)$$

$$b_N(x) = \arg \min_{b \in \mathcal{A}} \sum_{i=1}^{\ell} (b(x_i) - s_i)^2$$

↓

$$b_N(x) = \arg \min_{b \in \mathcal{A}} \sum_{i=1}^{\ell} \left( b(x_i) - \frac{s_i}{h_i} \right)^2 \quad h_i = \frac{\partial^2 L}{\partial z^2} \Big|_{z=a_{N-1}(x_i)}$$

# eXtreme Gradient Boosting (XGBoost)

$$b_N(x) = \arg \min_{b \in \mathcal{A}} \sum_{i=1}^{\ell} \left( b(x_i) - \frac{s_i}{h_i} \right)^2$$

$$b(x) = \sum_{j=1}^J b_j [x \in R_j]$$

$$\sum_{i=1}^{\ell} \left( -s_i b(x_i) + \frac{1}{2} h_i b^2(x_i) \right) + \lambda J + \frac{\mu}{2} \sum_{j=1}^J b_j^2 \rightarrow \min_b$$

$$\sum_{j=1}^J \left\{ \underbrace{\left( - \sum_{i \in R_j} s_i \right) b_j}_{=-S_j} + \frac{1}{2} \underbrace{\left( \mu + \sum_{i \in R_j} h_i \right) b_j^2}_{=H_j} + \lambda \right\}$$

# eXtreme Gradient Boosting (XGBoost)

$$b_j = \frac{S_i}{H_j + \mu}$$

$$H(b) = \frac{1}{2} \sum_{j=1}^J \frac{S_j^2}{H_j + \mu} + \lambda J$$

$$H(b_l) + H(b_r) - H(b) - \lambda \rightarrow \max$$

# LightGBM

<https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree>