



# Boosting и Bagging



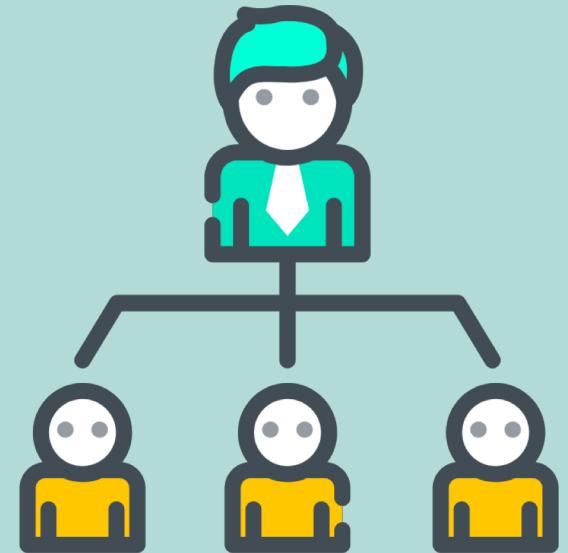
# **План**

1. Напоминание

2. Что стоит помнить

3. ML Puzzles

# 1. Напоминание



# Коротко о главном (bagging)

- Композиция сильных алгоритмов
- Можно оптимизировать произвольную функцию
- Очень хороши для структурированных данных

# Немного математики (bagging)

Пусть  $\xi_1, \xi_2, \dots, \xi_n$  — случайные одинаково распределённые случайные величины с дисперсией  $\sigma^2$  и коэффициентом корреляции  $\rho$ . Тогда

$$\begin{aligned} \text{Var} \frac{1}{n} \sum_{i=1}^n \xi_i &= \frac{1}{n^2} \text{Var} \sum_{i=1}^n \xi_i = \frac{1}{n^2} \text{cov} \left( \sum_{i=1}^n \xi_i, \sum_{j=1}^n \xi_j \right) = \frac{1}{n^2} \left( \sum_{i=1, j=1}^n \text{cov}(\xi_i, \xi_j) \right) = \\ &= \frac{1}{n^2} \left( \sum_{i=1}^n \text{cov}(\xi_i, \xi_i) + \sum_{i \neq j} \text{cov}(\xi_i, \xi_j) \right) = \frac{1}{n^2} (n\sigma^2 + n(n-1)\rho\sigma^2) = \frac{1 + \rho(n-1)}{n} \sigma^2 \end{aligned}$$

# Аналогия для bagging



# Коротко о главном (GB)

- Композиция слабых алгоритмов (обычно decision trees)
- Можно оптимизировать произвольную дифференцируемую функцию
- Очень хороши для структурированных данных (наверное даже лучше)

# Немного математики (GB)

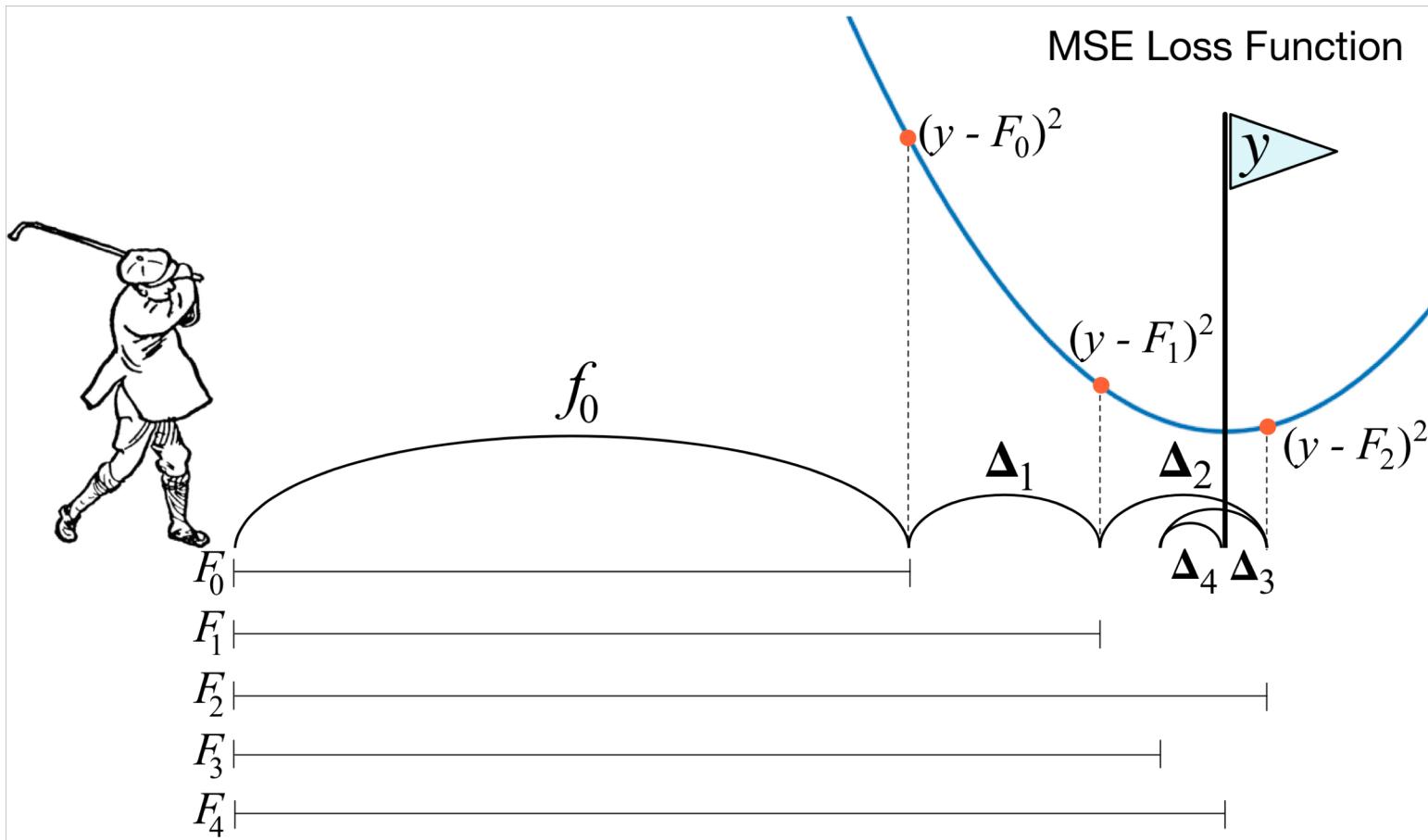
$$\sum_{i=1}^l L(y_i, a_{N-1}(x_i) + \gamma_N b_N(x_i)) \rightarrow \min_{b_N, \gamma_N}$$

- 1.**  $s_i = -\left. \frac{\partial L}{\partial z} \right|_{z=a_{N-1}(x_i)}$
- 2.**  $b_N = \arg \min_b \sum_{i=1}^l (b(x_i) - s_i)^2$
- 3.**  $\gamma_N = \arg \min_\gamma \sum_{i=1}^l L(y_i, a_{N-1}(x_i) + \gamma b_N(x_i))$

$$\begin{aligned} \sum_{i=1}^l L(y_i, a_{N-1}(x_i) + b_N(x_i)) &\approx \\ &\approx \sum_{i=1}^l L(y_i, a_{N-1}(x_i)) + g_i b_N(x_i) + \frac{1}{2} h_i b_N(x_i)^2 + \Omega(b_N) \end{aligned}$$

Regularization:  $\Omega(b_N) = \gamma T + \lambda \|w\|^2$ ,  $T$  – leaf count,  $w$  – leaf values.

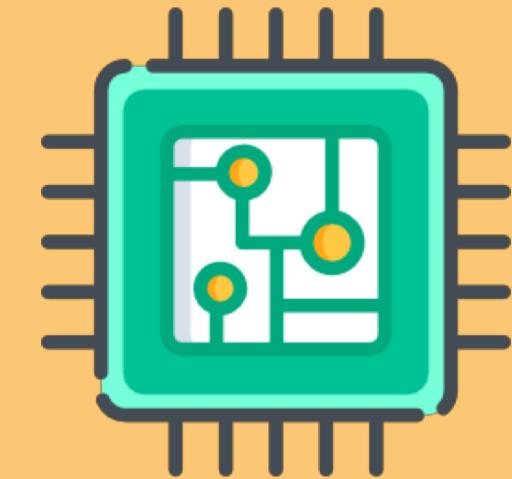
# Аналогия для GB



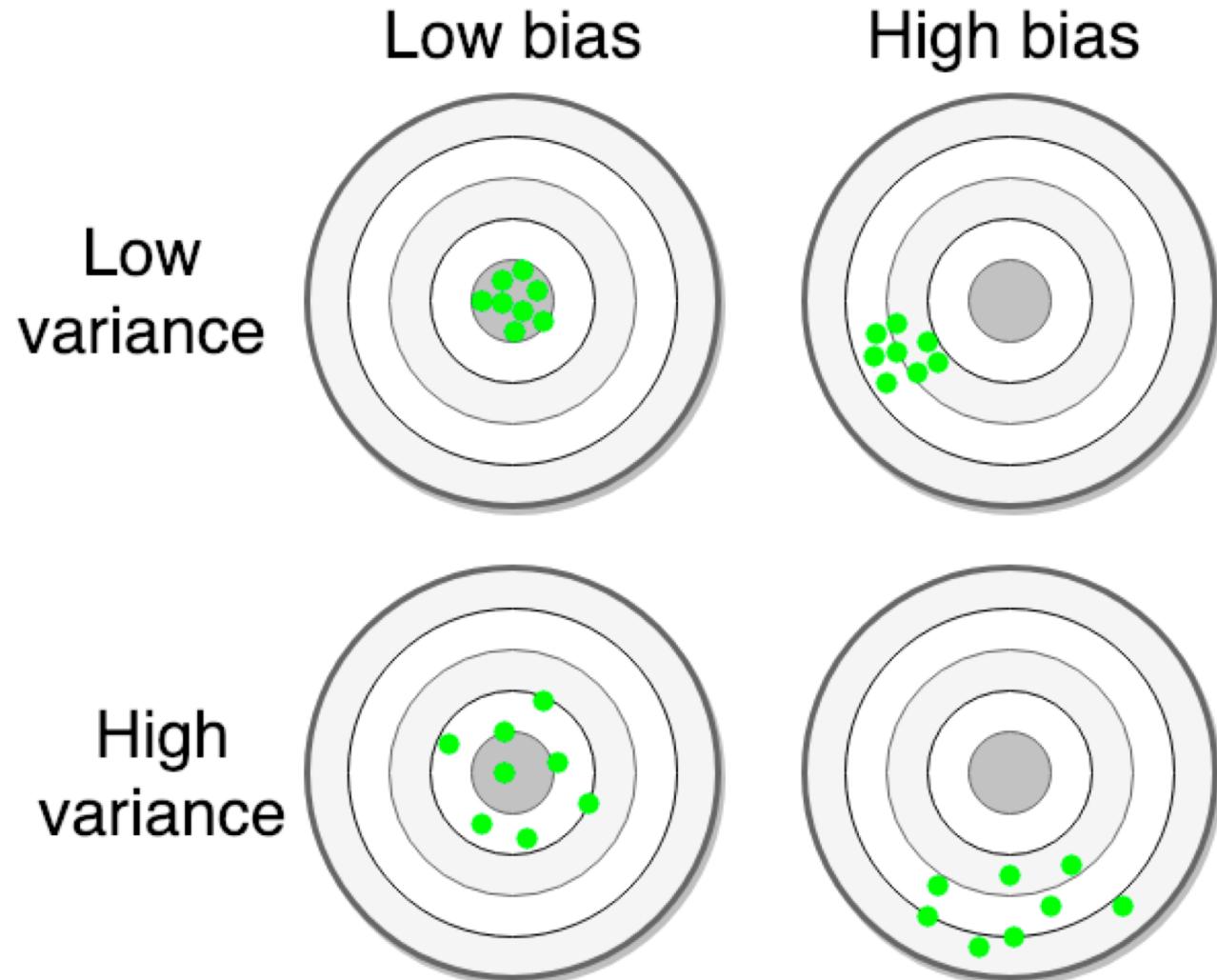
## Полезные ссылки

- [Greedy function approximation: A gradient boosting machine](#)
- [XGBoost: A Scalable Tree Boosting System](#)
- [LightGBM: A Highly Efficient Gradient Boosting Decision Tree](#)
- [Fighting biases with dynamic boosting](#)
- [А. Дьяконов «Градиентный бустинг»](#)
- [Видео про градиентный бустинг и фишki](#)

## **2. Что стоит помнить**



# Bias-Variance tradeoff



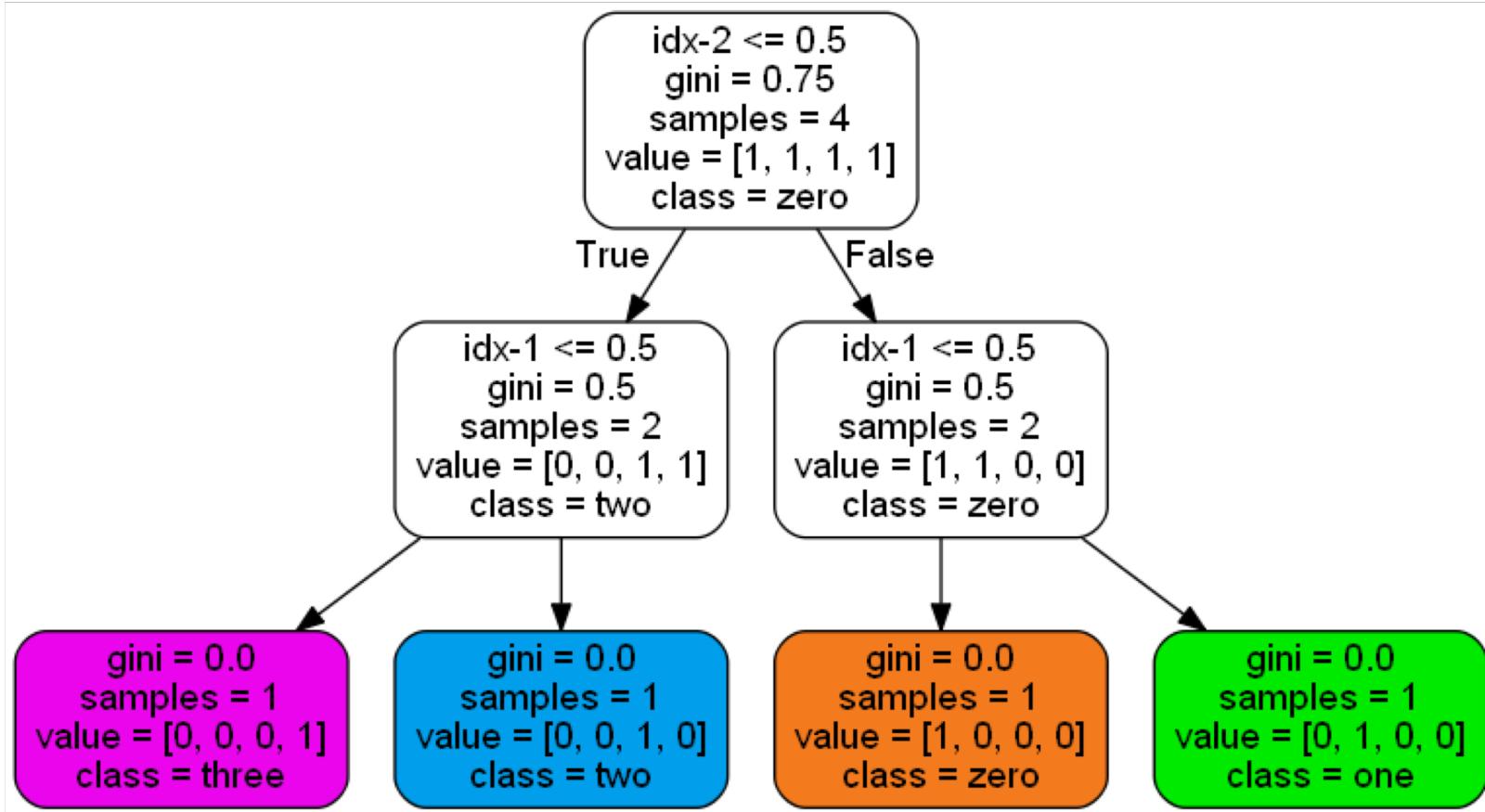
# Bias-Variance tradeoff

- Bagging уменьшает ?
- Boosting уменьшает ?

# Bias-Variance tradeoff

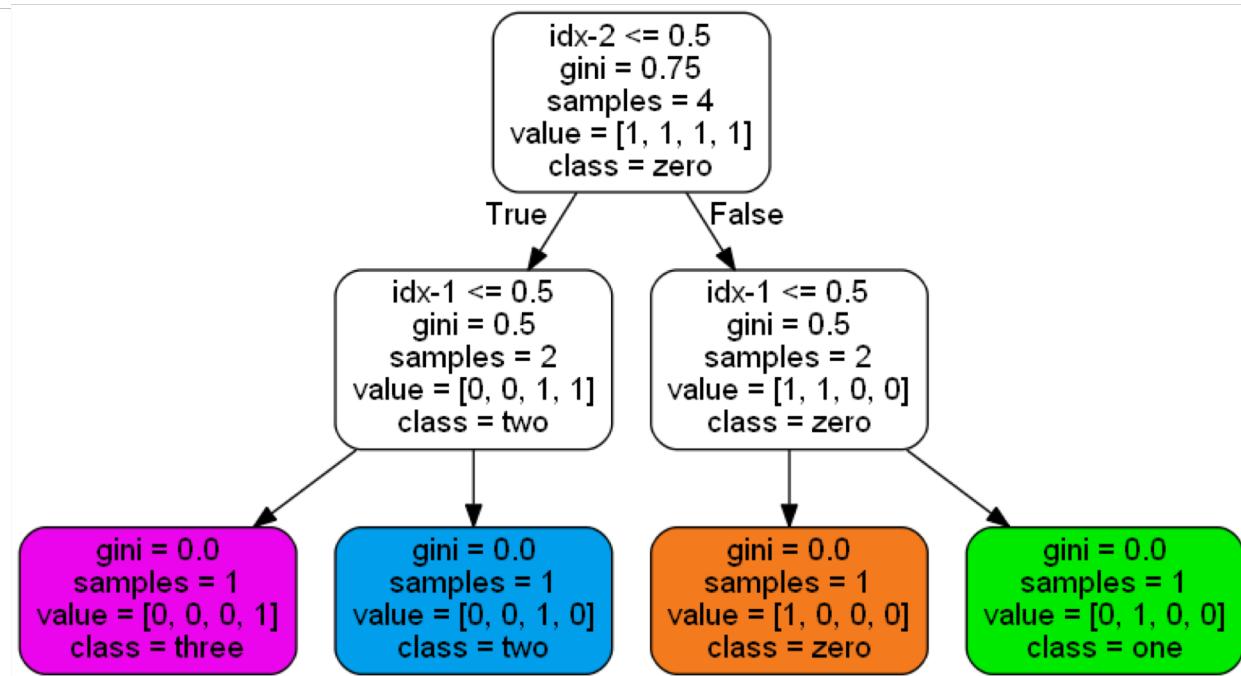
- Bagging уменьшает variance
- Boosting уменьшает bias

# Альтернативное использование деревьев 1



# Альтернативное использование деревьев 2

`sklearn.ensemble.RandomTreesEmbedding`



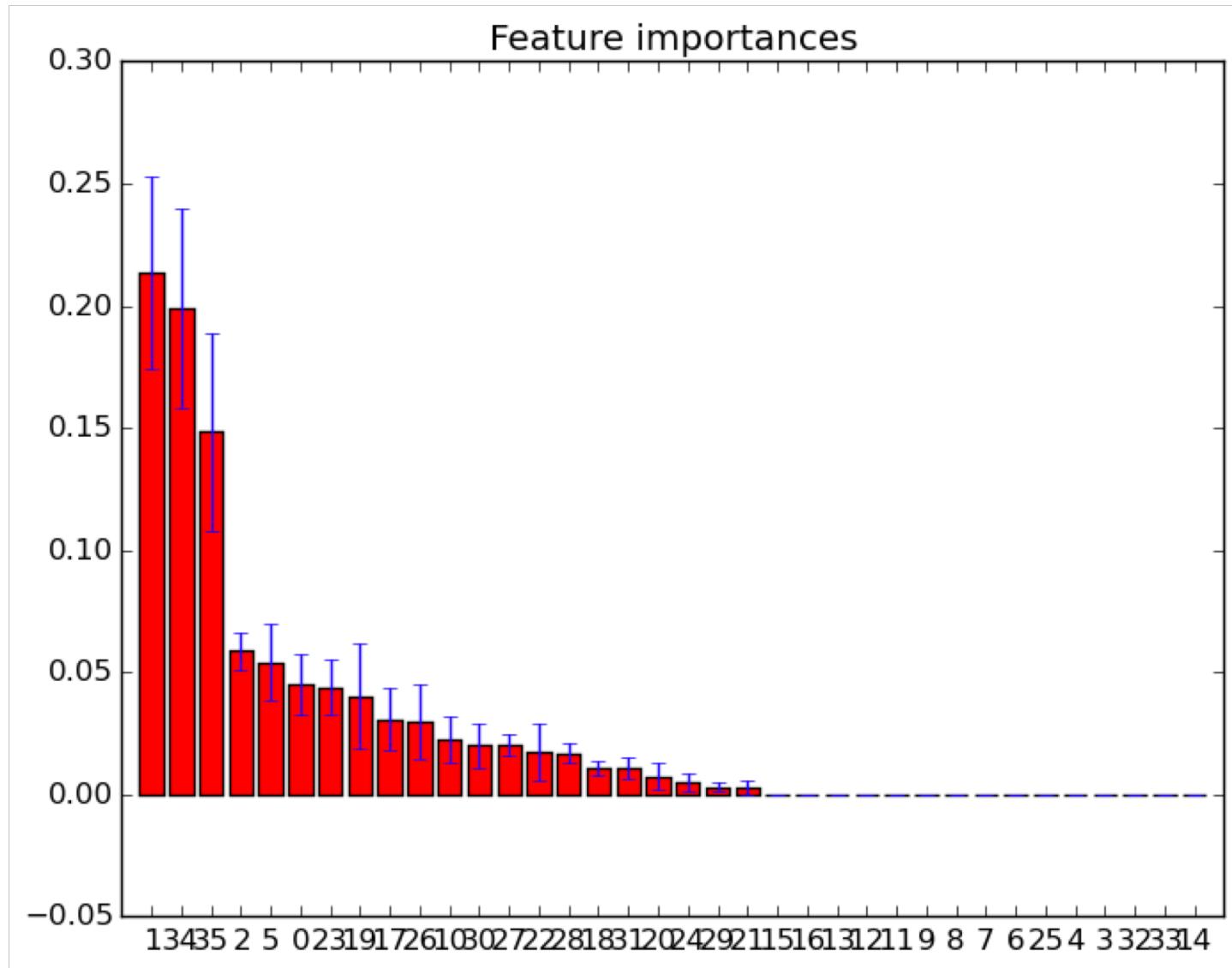
0

0

1

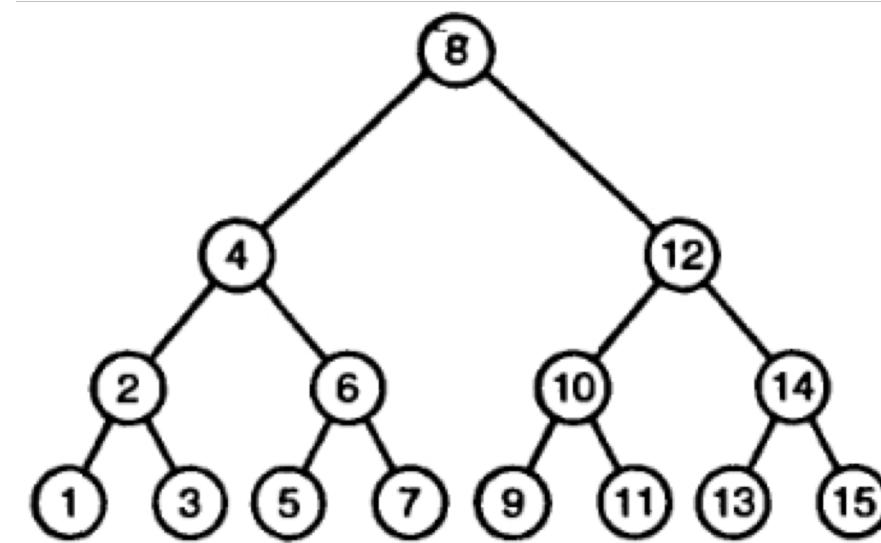
0

# Альтернативное использование деревьев 3



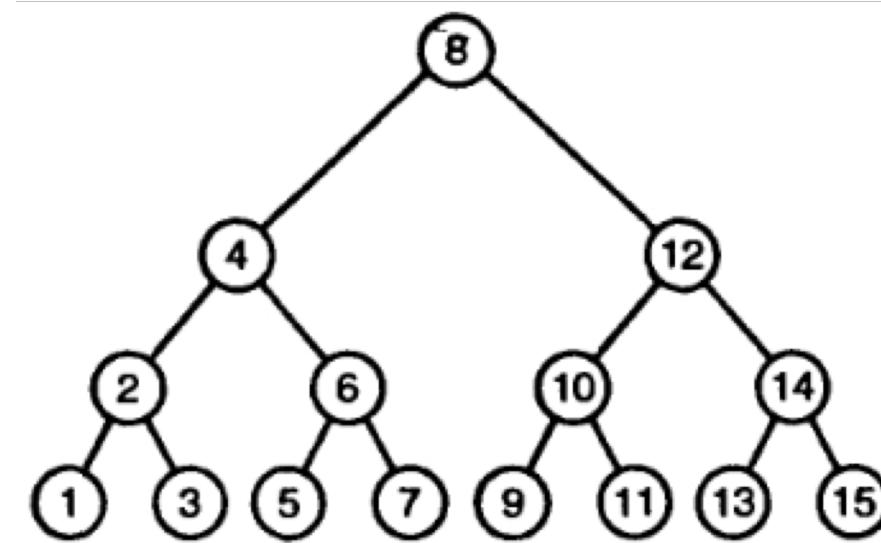
# Накладные расходы

- есть полное бинарное дерево глубины  $h$
- сколько в нём вершин?



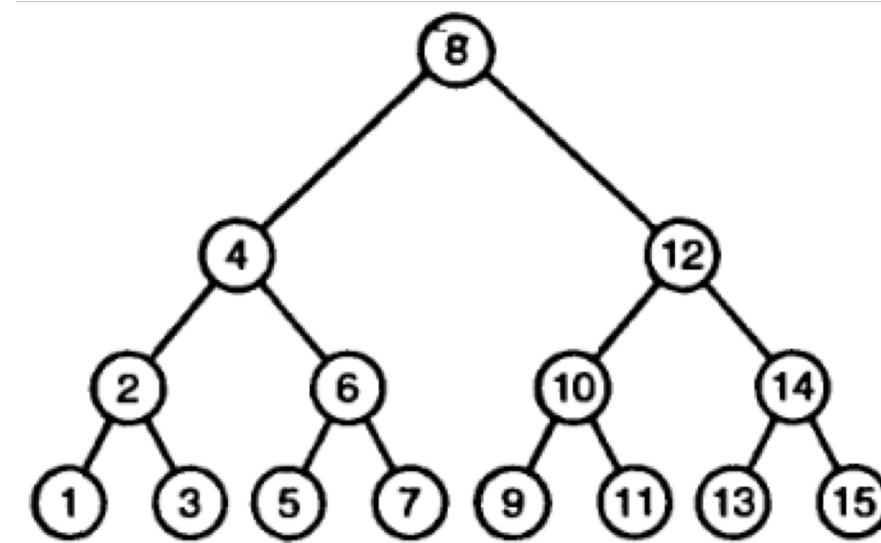
# Накладные расходы

- есть полное бинарное дерево глубины  $h$
- сколько в нём вершин?  $2^h$



# Накладные расходы

- есть полное бинарное дерево глубины  $h$
- сколько в нём вершин?  $2^h$
- сколько памяти займет лес на 1000 деревьев глубины 20?



## 3. ML Puzzles



# Сумма

Может ли GB или RF выучить сумму двух чисел?

**слагаемое**

$$3 + 2 = 5$$

**сумма**

**сумма**

# Отрицательные числа

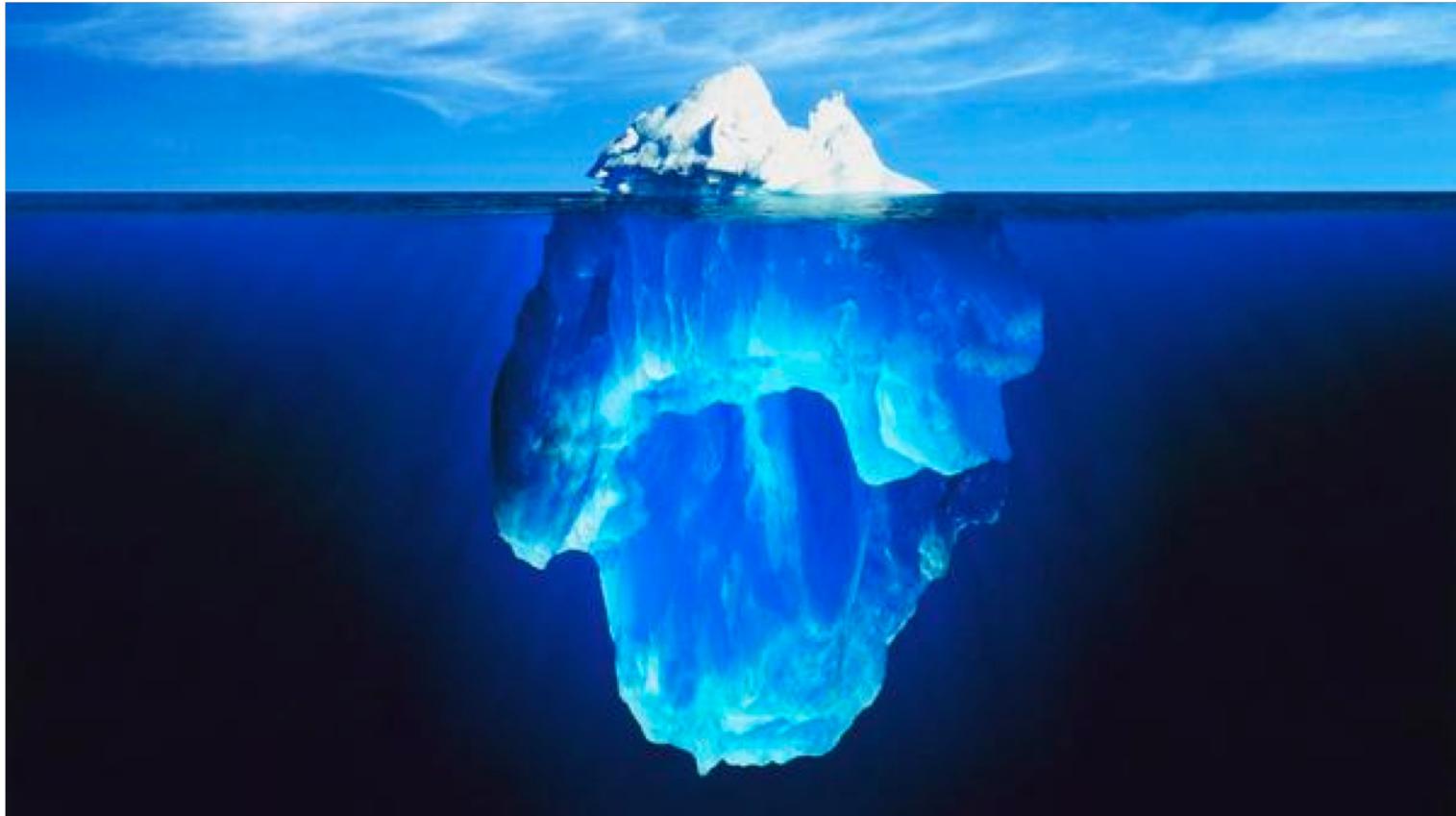
В обучении есть положительные таргеты, а на teste  
есть отрицательные предсказания:

- GB
- RF
- Linear models
- kNN



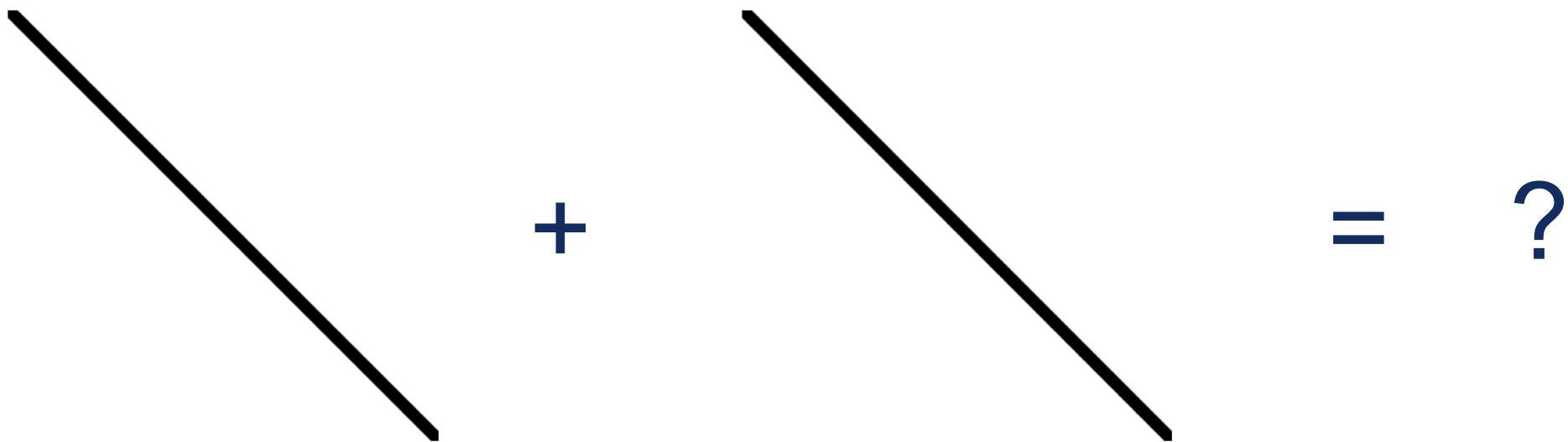
# Глубина деревьев

Какая глубина дерева характерна для бустинга, а какая для бэггинга? И почему?



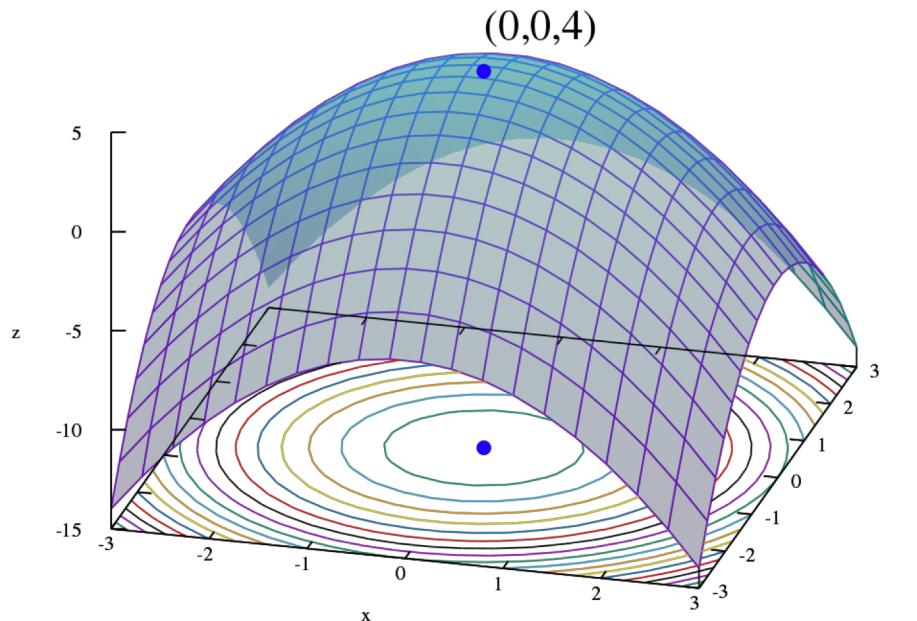
# Немного о линейных моделях

Имеет ли смысл делать бустинг и бэггинг над линейными моделями?



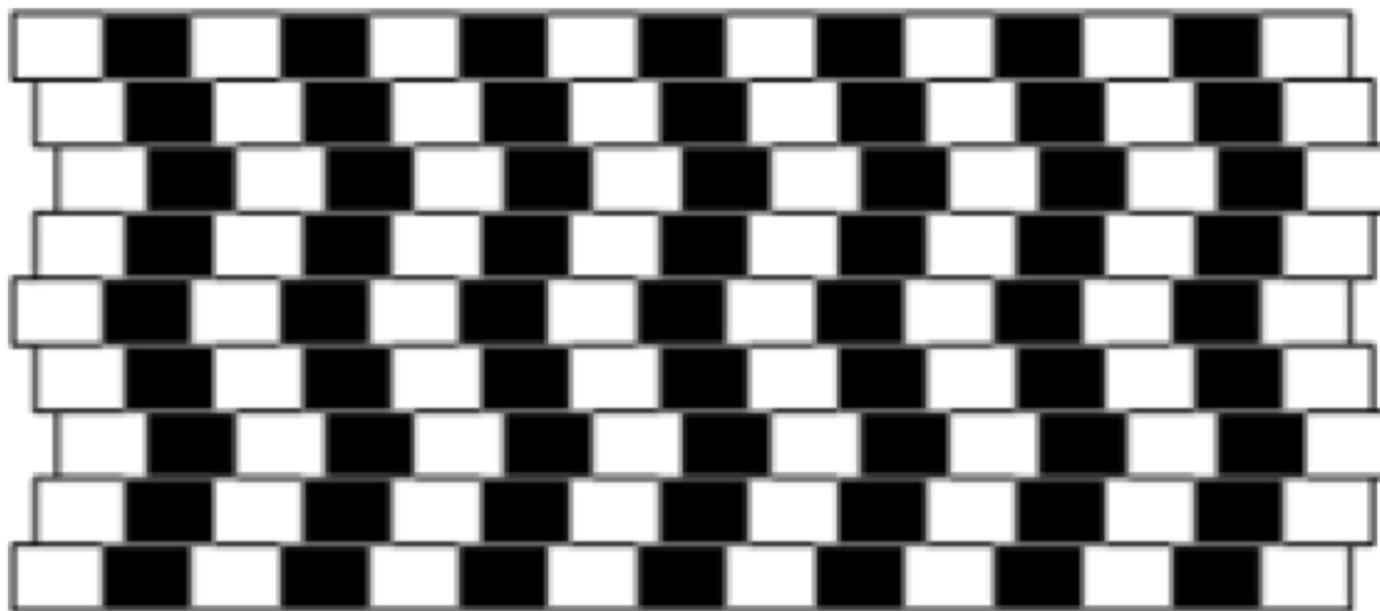
# О разном

Можно ли использовать GB модель для оптимизации  
фичей с целью максимизации таргета?



# О параллелизме

Как распараллелить бэггинг? А бустинг?



**GNUparallel**