



# Введение в АД

Лекция 7



## Список основных обозначений

$X = (X_1, \dots, X_n)$  — случайная выборка

$x = (x_1, \dots, x_n)$  — неслучайная выборка

$i \in \{1, \dots, n\}$  — индексация элементов выборки

$j \in \{1, \dots, d\}$  — индексация размерностей  $x \in \mathbb{R}^d$  (признаков)

$k \in \{1, \dots, K\}$  — обозначения классов

$x_i$  — объект выборки

$x_j$  — значение  $j$ -го признака объекта  $x$

$x_{ij}$  — значение  $j$ -го признака объекта  $x_i$

$\mathcal{I}$  — множество индексов элементов выборки

$X_{\mathcal{I}}$  — подвыборка с индексами  $\mathcal{I}$



# Решающие деревья



# Свойства линейной регрессии

- ▶ Легко обучается с помощью градиентного спуска
- ▶ Легко интерпретируема
- ▶ Восстанавливает только простые зависимости  
мало степеней свободы:  
обычно число параметров  $\approx$  количество признаков
- ▶ Не всегда отражает то, как люди принимают решения.  
Иногда не логично расставлять коэффициенты перед признаками.



# Как люди принимают решения?





# Решающие деревья



# Регрессионное дерево

Идея построения бинарного дерева по выборке  $(x_1, Y_1), \dots, (x_n, Y_n)$ .

- ▶ **Начало:** один лист с меткой  $\bar{Y}$ , к нему относятся все объекты.
- ▶ **Деление листа**

Пусть  $\mathcal{I}_{leaf}$  — индексы объектов в текущем листе.

Делим лист на два с подмножествами индексов  $\mathcal{I}_\ell \sqcup \mathcal{I}_r = \mathcal{I}_{leaf}$ .

- ▶ **Правило деления**

$x_j < t \Rightarrow$  объект  $x$  попадает в  $\ell$ , иначе в  $r$ .

- ▶ **Принцип деления:** наилучшее приближение двумя константами

Правило 
$$\sum_{i \in \mathcal{I}_\ell} (Y_i - y_\ell)^2 + \sum_{i \in \mathcal{I}_r} (Y_i - y_r)^2 \rightarrow \min_{\mathcal{I}_\ell, \mathcal{I}_r}$$

Метки в новых листах 
$$y_\ell = \frac{1}{|\mathcal{I}_\ell|} \sum_{i \in \mathcal{I}_\ell} Y_i, \quad y_r = \frac{1}{|\mathcal{I}_r|} \sum_{i \in \mathcal{I}_r} Y_i$$

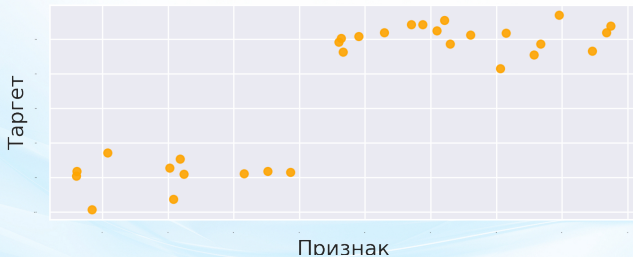


# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.

Т.е. строится кусочно-постоянная функция.

Пример 1:





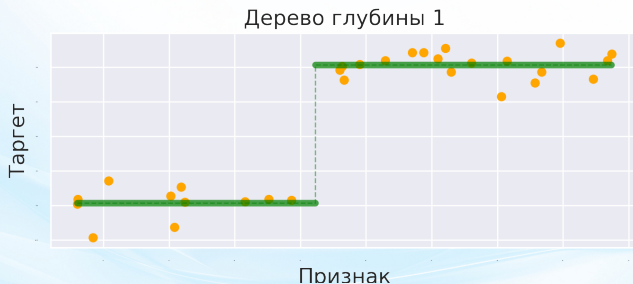


# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.

Т.е. строится кусочно-постоянная функция.

Пример 1:





# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.

Т.е. строится кусочно-постоянная функция.

Пример 2:



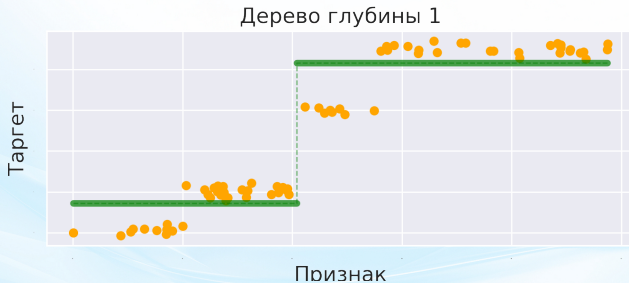


# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.

Т.е. строится кусочно-постоянная функция.

Пример 2:





# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.

Т.е. строится кусочно-постоянная функция.

Пример 2:





# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.

Т.е. строится кусочно-постоянная функция.

Пример 3:

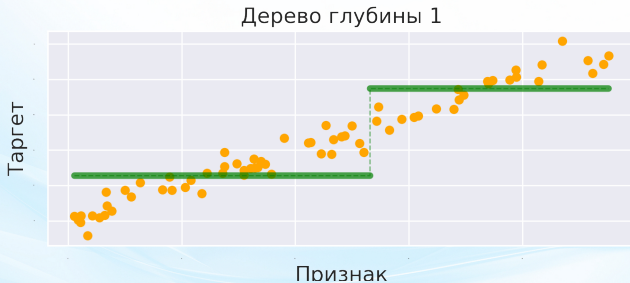




# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.  
Т.е. строится кусочно-постоянная функция.

Пример 3:

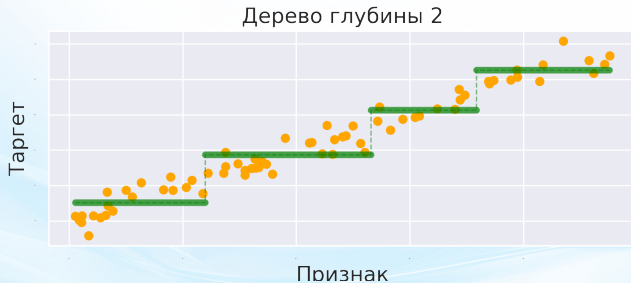




# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.  
Т.е. строится кусочно-постоянная функция.

Пример 3:

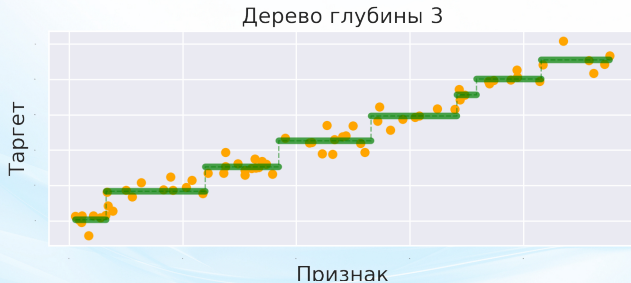




# Регрессионное дерево

Оценка отклика — метка листа, в который попадет объект.  
Т.е. строится кусочно-постоянная функция.

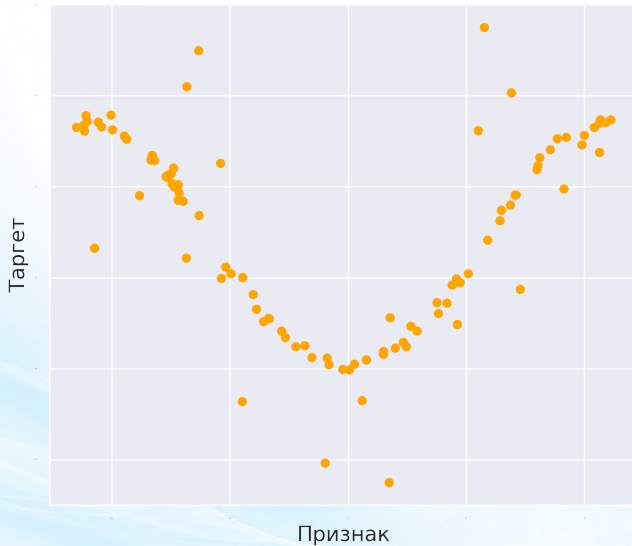
Пример 3:





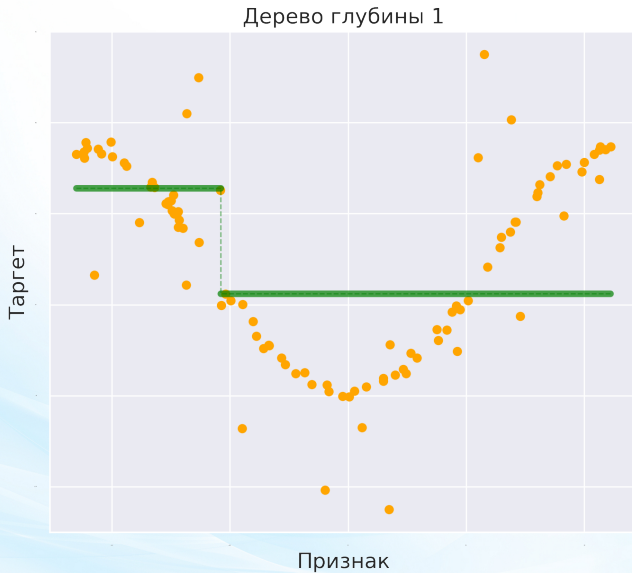


# Регрессионное дерево и выбросы



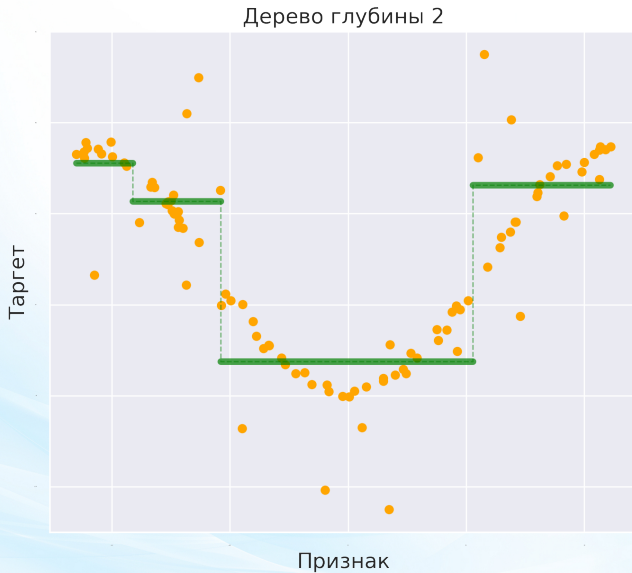


# Регрессионное дерево и выбросы



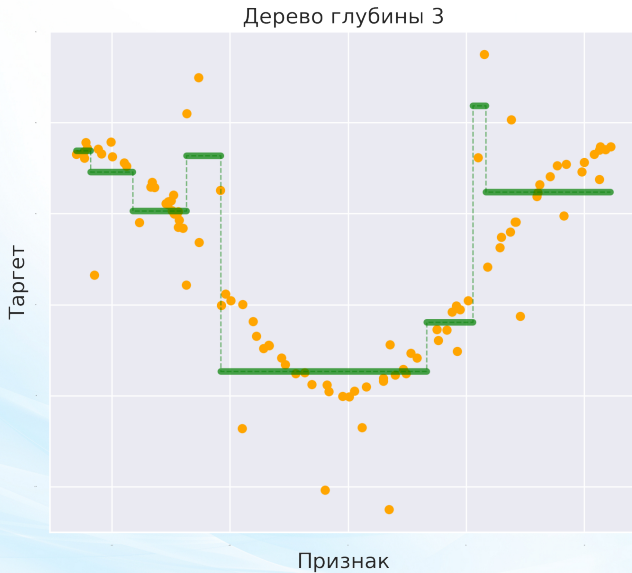


# Регрессионное дерево и выбросы



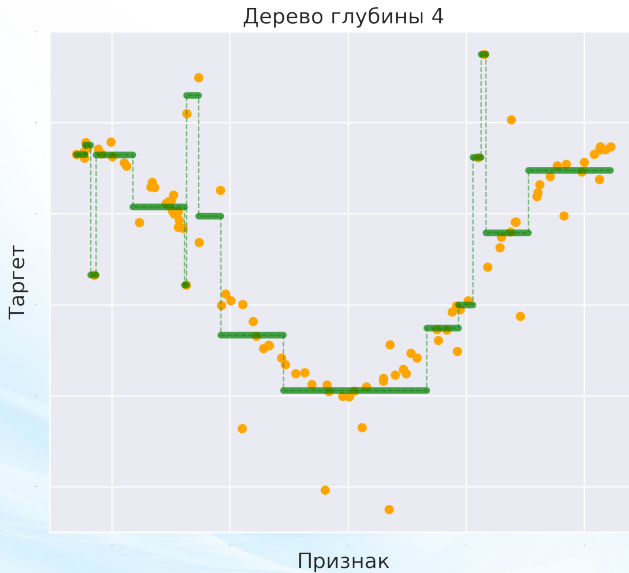


# Регрессионное дерево и выбросы



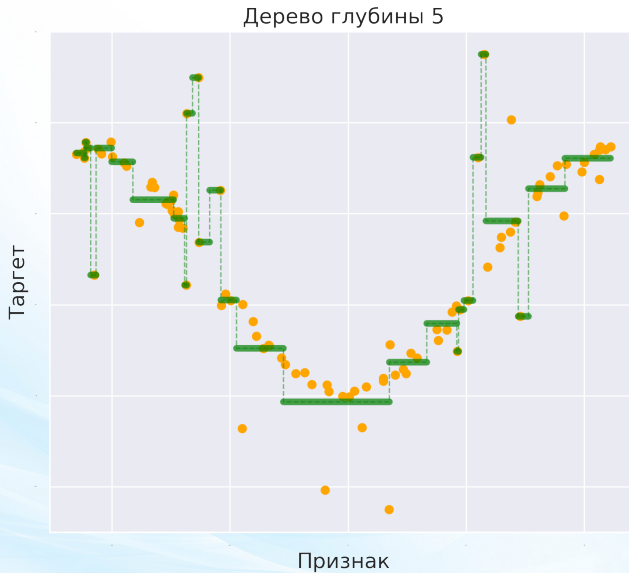


# Регрессионное дерево и выбросы



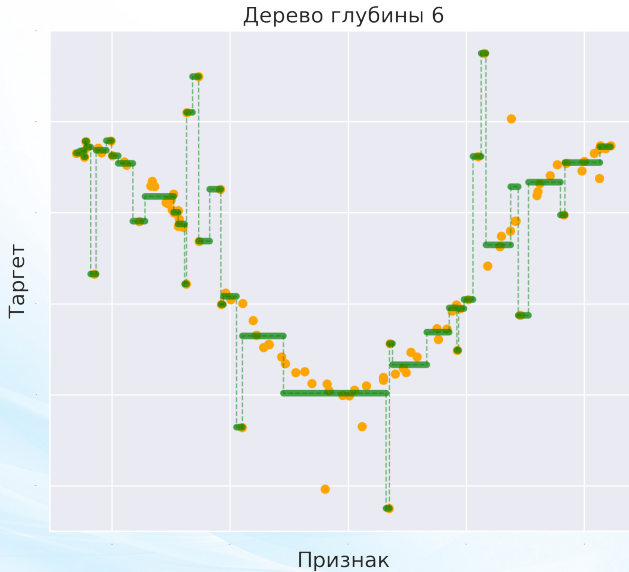


# Регрессионное дерево и выбросы





# Регрессионное дерево и выбросы





# Общий случай

## Решающее дерево:

- ▶ Бинарное дерево.
- ▶ В каждой вершине записано некоторое условие.
- ▶ В зависимости от условия идем в правую или левую вершину.
- ▶ В листьях дерева — предсказания.

*Замечание:* Существуют и не бинарные решающие деревья, однако в основном используются именно бинарные

## Условия в вершинах

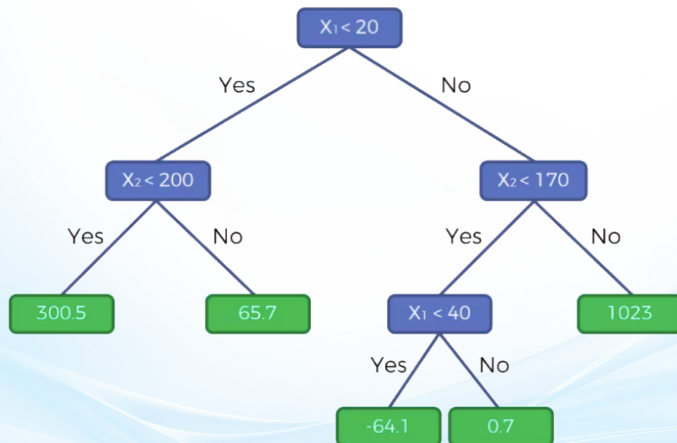
Самый популярный подход — правила вида  $I\{x_j < t\}$ .

Оптимальные значения порога  $t$  и признака  $x_j$  подбираются по некоторому критерию.





# Пример






## Пример

# Классификация котиков



котик	порода	рост	шерсть
	Саванна	50 см	да



# Переобучение

## *Утверждение*

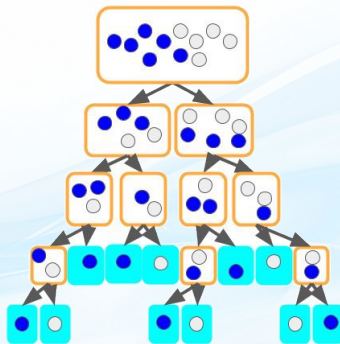
Для любой обучающей выборки можно построить решающее дерево с нулевой ошибкой на обучении.

## *Доказательство:*

Построим решающее дерево, в котором каждый лист содержит только один объект.

Метка в листе определяется только этим одним объектом.

⇒ предсказание для обучающей выборки не содержит ошибок.





# Построение дерева



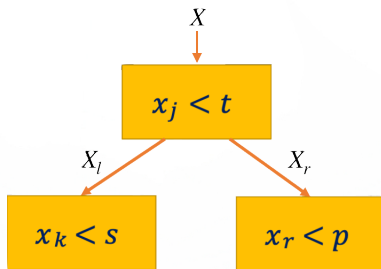
# Как строится дерево?

Пусть  $X$  — обучающая выборка.

## Деление

Найдем правило  $I\{x_j < t\}$ ,  
оптимальное по некоторому критерию.

Данное правило разобьет  $X$   
на две части:  $X_\ell$  и  $X_r$ .



## Итерации

Процедура выполняется рекурсивно для двух дочерних вершин  
с обучающими выборками  $X_\ell$  и  $X_r$  соответственно.

Если выполнен некий критерий остановки,  
то не делим текущую вершину.



# Выбор разбиения

Пусть в вершине  $m$  оказалась выборка  $X_m$ .

Пусть  $Q(X_m, j, t)$  — критерий ошибки условия  $I\{x_j < t\}$ .

Оптимизируем его **перебором** по всем возможным признакам  $j$ ,  
для каждого признака по всем возможным его порогам  $t$ :

$$Q(X_m, j, t) \longrightarrow \min_{j, t}$$

## Критерий информативности (impurity)

$$Q(X_m, j, t) = \frac{|X_\ell|}{|X_m|} \cdot H(X_\ell) + \frac{|X_r|}{|X_m|} \cdot H(X_r)$$

Оценивает величину разброса таргета в вершине.

Хорошее разбиение: после него больше уверены в ответе в вершине.  
Т.е. хотим разбить вершину на две так, чтобы полученные две вершины  
были более однородны по таргетам.



## Выбор разбиения

$$Q(X_m, j, t) = \frac{|X_\ell|}{|X_m|} \cdot H(X_\ell) + \frac{|X_r|}{|X_m|} \cdot H(X_r)$$

$H(X_\ell)$  и  $H(X_r)$  нормируются на доли объектов, которые идут вправо и влево.

*Зачем это нужно?*

Пусть  $|X_m| = 1000$ ,  $|X_\ell| = 990$ ,  $|X_r| = 10$

В  $X_\ell$  все объекты имеют один класс  $\Rightarrow H(X_\ell)$  маленький.

$X_r$  содержит объекты всех возможных классов  $\Rightarrow H(X_r)$  большой.

Не так страшно, что  $X_r$  получилось плохим, главное, что 990 попали в правильную вершину.



# Критерий информативности

## Неформальное определение

- ▶  $H(X)$  зависит от меток в выборке  $X$
- ▶ Показывает разброс ответов в  $X$
- ▶ Чем меньше разброс ответов в  $X$ , тем меньше  $H(X)$

## Для регрессии

*Как показать разброс ответов для задачи регрессии?*

Возьмем выборочную дисперсию ответов в качестве  $H(X)$ :

$$H(X) = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} (Y_i - \bar{Y})^2,$$

где  $\mathcal{I}$  — мн-во индексов, соотв. подвыборке  $X$ , а  $\bar{Y} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} Y_i$ .





# Критерий информативности: задача классификации

Пусть решаем задачу классификации на  $K$  классов.

$p_k = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} I\{Y_i = k\}$  — доля объектов в классе  $k$  в  $X_{\mathcal{I}}$ .

## Энтропийный критерий

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

Мера отличия распределения классов от вырожденного.

Считаем, что  $0 \ln 0 = 0$

Свойства:

- ▶  $H(X) \geq 0$
- ▶ При  $p_1 = 1, p_2 = 0, \dots, p_K = 0$  выполнено  $H(X) = 0$

## Критерий Джини

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

Вероятность ошибки случайного классификатора, который выдает ответы пропорционально  $p_k$ .



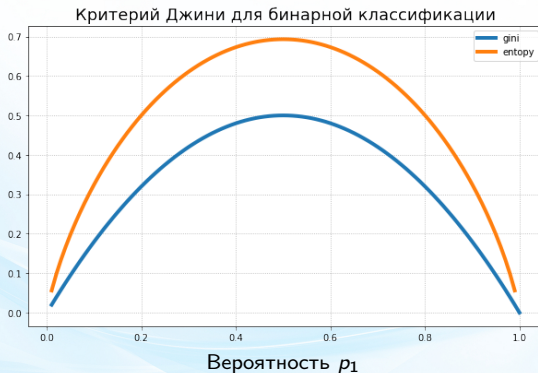
# Критерий информативности: задача классификации

## Энтропийный критерий

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

## Критерий Джини

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





# Критерий остановки



# Критерий останова

*Как понять, разбивать ли вершину далее или сделать ее листовой?*

- ▶ **Все объекты в вершине одного класса**

Если выборка сложная, то сработает только когда в вершине осталось 1-2 объекта.

- ▶ **В вершину попало  $\leq k$  объектов.**

$k$  — гиперпараметр. Нужно выбирать таким, чтобы по  $k$  объектам в листе можно было построить надежный прогноз.

- ▶ **Глубина дерева превысила порог.**

Грубый критерий, не зависит ни от распределения классов, ни от числа объектов.

Хорошо работает в композициях, где много моделей объединяются в одну сложную модель.



## Критерий останова

*Как понять, разбивать ли вершину далее или сделать ее листовой?*

- ▶ **Число листьев в дереве превысило порог.**
- ▶ **Функционал ошибки при делении не уменьшился.**  
Если лучшее из разбиений приводит к росту функционала ошибки, не разбиваем эту вершину.
- ▶ **Функционал ошибки при делении уменьшился на  $< s\%$ .**  
Если лучшее из разбиений не уменьшает функционал на  $s\%$ , не разбиваем эту вершину.



# Метки в листьях



## Ответ в листе

*Какое предсказание выдавать для объекта, попавшего в лист?*

► Классификация:

1. Самый популярный класс в листе:

$$\hat{y} = \arg \max_k \sum_{i \in \mathcal{I}_{leaf}} I\{Y_i = k\}$$

2. Оценки вероятности классов  $\hat{p} = (\hat{p}_1, \dots, \hat{p}_K)$ , где

$$\hat{p}_k = \frac{1}{|\mathcal{I}_{leaf}|} \sum_{i \in \mathcal{I}_{leaf}} I\{Y_i = k\}$$

► Регрессия с критерием MSE:

$$\hat{y} = \frac{1}{|\mathcal{I}_{leaf}|} \sum_{i \in \mathcal{I}_{leaf}} Y_i$$

*Почему это оптимально? Узнаем в статистике!*



# Случайные леса





# Решающее дерево



Один в поле не воин...



Лес





# Нужны разные деревья



"Танцующий лес", нац. парк Куршская коса, Калининградская обл.



# Метод Random Forest

Композиция вида  $F = \frac{1}{B} \sum_{b=1}^B f_b$  из  $B$  деревьев, где  $f_b$  — дерево, обученное на случайной выборке, причем при обучении в каждом узле дерева использовался RSM — случайное подпространство признаков.

Деревья строятся максимальной глубины, сильно переобученные.

**Случайная выборка:** упорядоченный выбор с возвращением.

**Рекомендуемые размерности подпр-ств в RSM:**

- ▶ для задач регрессии размерность  $\sim d/3$ ;
- ▶ для задач классификации размерность  $\sim \sqrt{d}$ .



# Метод Random Forest

## Преимущества:

- ▶ не требует сложных экспериментов по настройке параметров;
- ▶ в среднем не переобучается с увеличением количества деревьев;
- ▶ работает с признаками разной природы;
- ▶ возможность распараллеливания.

## Недостатки:

- ▶ плохо обрабатывает линейные зависимости;
- ▶ долго работает на больших данных.



**ВСЁ!**