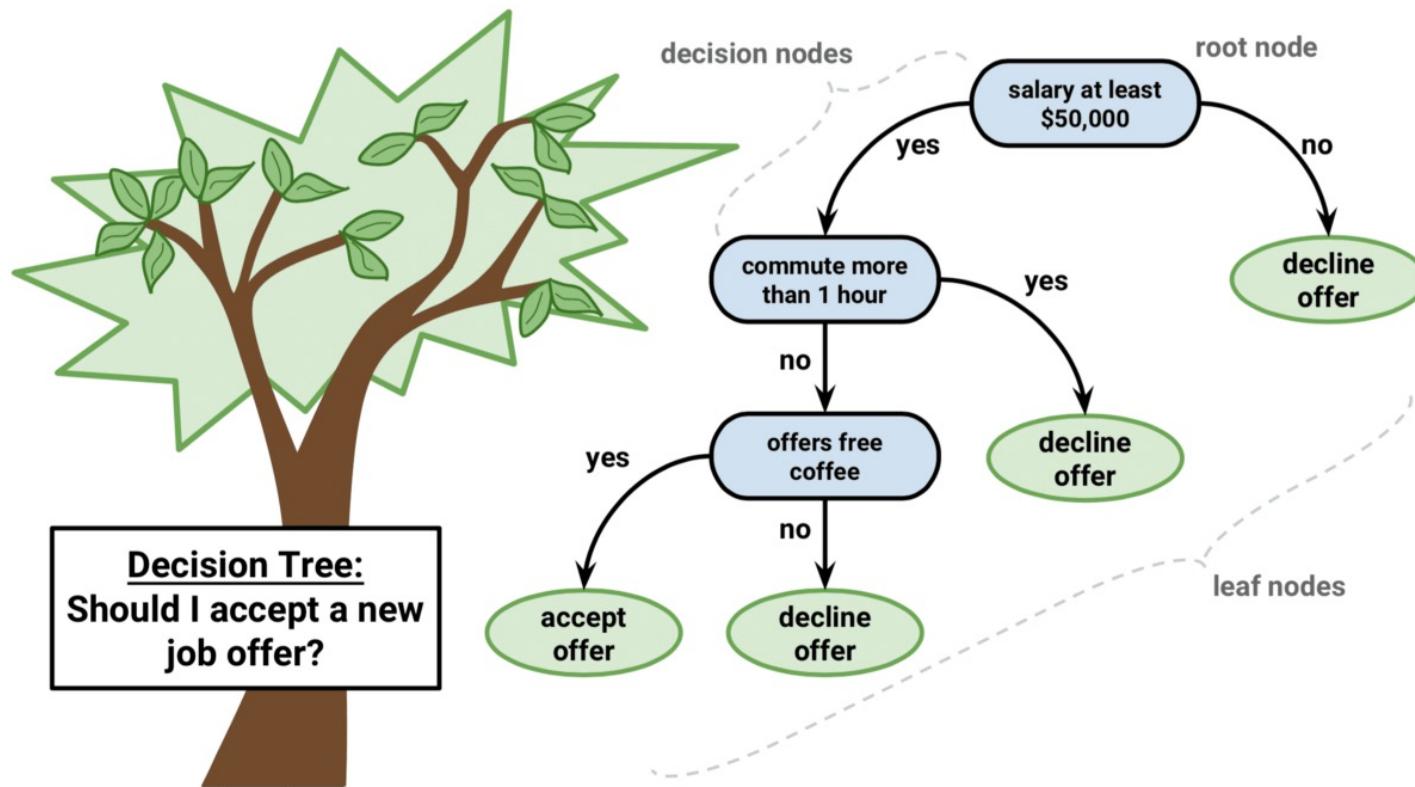


Древья решений и  
ансамбли

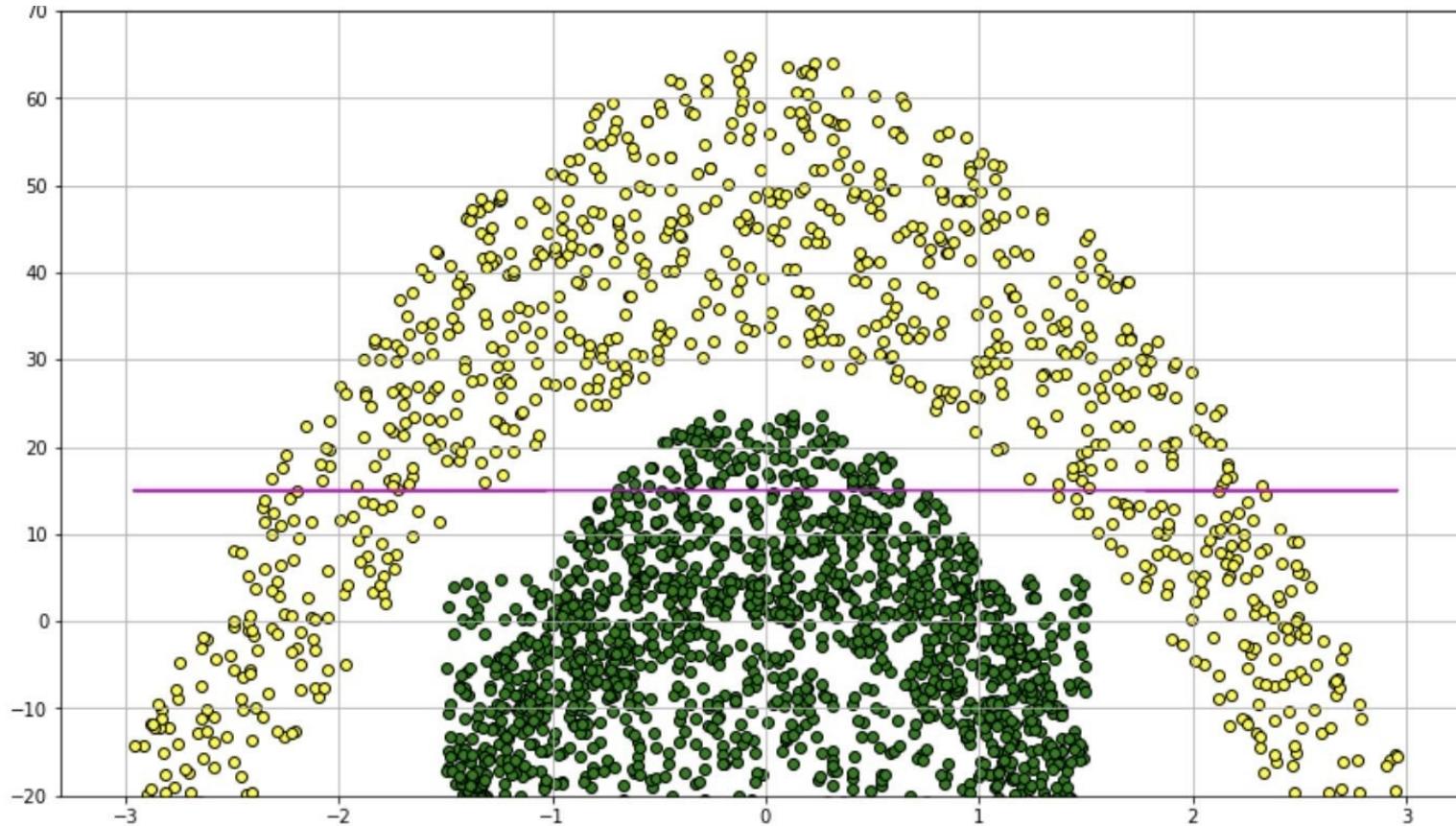
# Деревья решений. Что это вообще такое?

Дерево решений как алгоритм машинного обучения: объединение логических правил вида "Значение признака а меньше Х И Значение признака б меньше Y => Класс 1" в структуру данных "Дерево".



# Интуиция применения деревьев

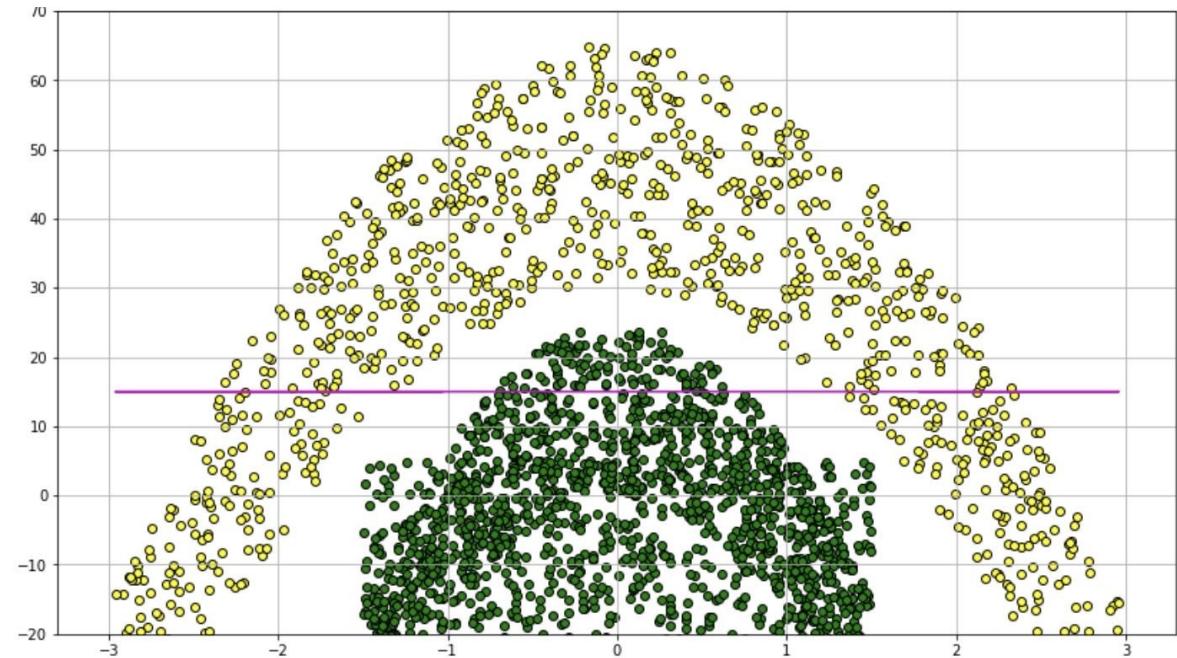
Logistic Regression, f-measure ~0.755



Целевая переменная **нелинейно** зависит от признаков

# Интуиция применения деревьев

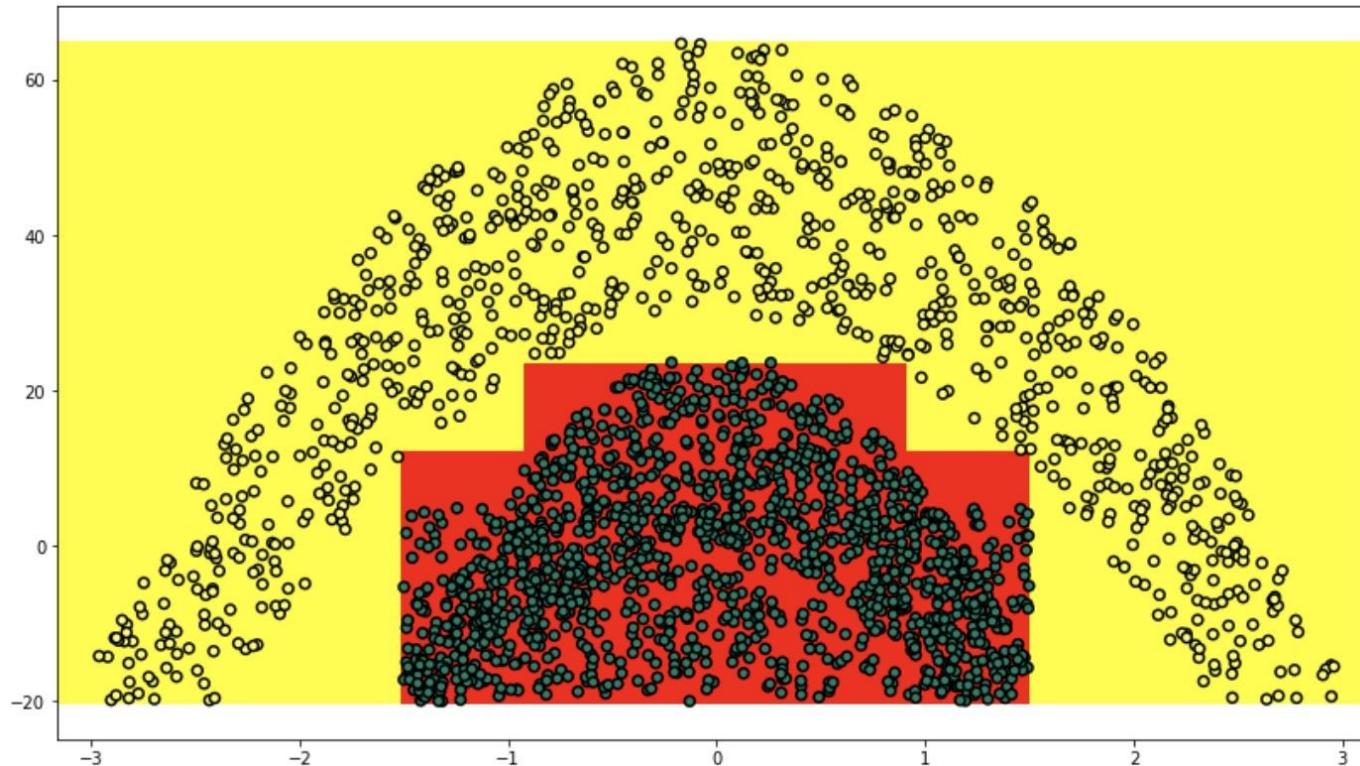
Logistic Regression, f-measure ~0.755



- Лог. рег. хорошо работает при линейной зависимости признаков и целевой переменной
- Экспериментировать с преобразованием признаков и добиться более хорошего качества, но такой подход является эвристиком и вы можете потратить на много времени и при этом не получить желаемый результат

# Интуиция применения деревьев

Decision Tree, f-measure = 1



- разделяет пространство на многомерные прямоугольники (подпространства)
- в подпространстве формируется ответ на основе обучающей выборки

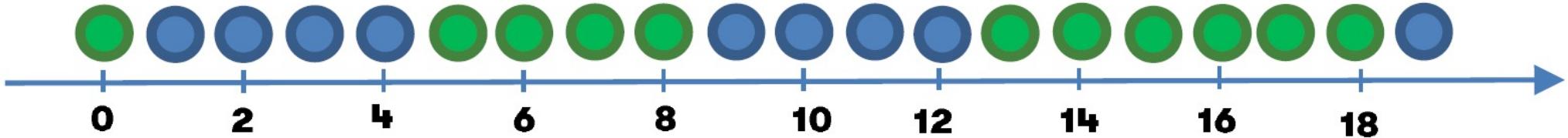
# Как строится дерево решений

- Логические правила – это по своей сути уточняющие вопросы.
- Вопрос тем лучше, чем меньше неопределённости в ответе
- Как нам можно замерить эту «неопределенность»?

$$S = - \sum_{i=1}^N p_i \log_2 p_i,$$

Энтропия Шеннона для системы с N состояниями, где  $P_i$  - вероятность нахождения системы в i-ом состоянии.

# Пример построения



Какая у нас сейчас энтропия?

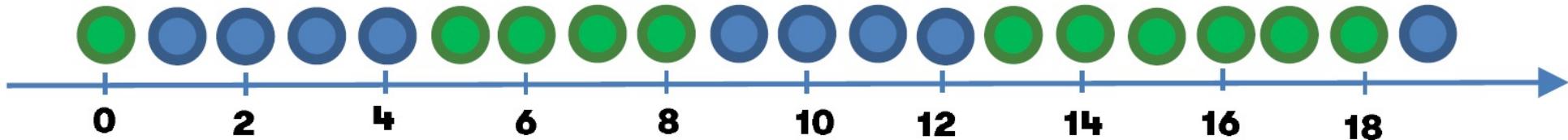
Вероятность синего шара –  $p_1 = \frac{9}{20}$

Вероятность зеленого шара –  $p_2 = \frac{11}{20}$

Значит Финальная энтропия считается, как

$$S_0 = -\frac{9}{20} \log_2 \frac{9}{20} - \frac{11}{20} \log_2 \frac{11}{20} \approx 1$$

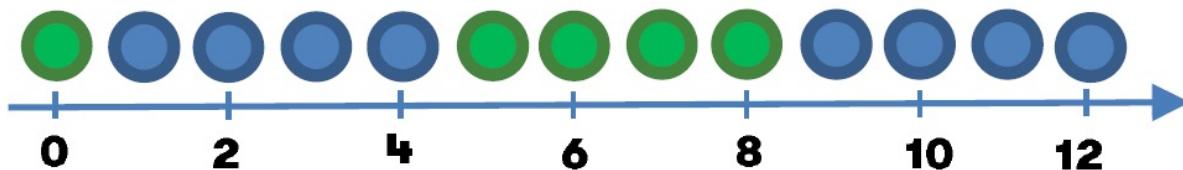
# Пример построения



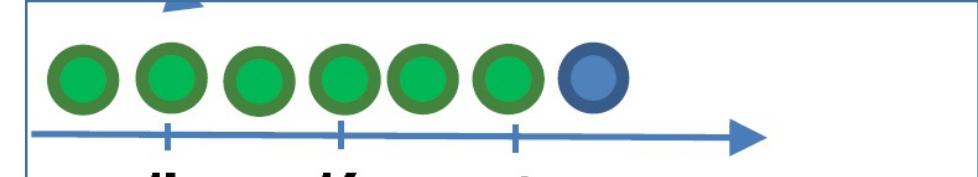
$$X \leq 12$$

Да

Нет



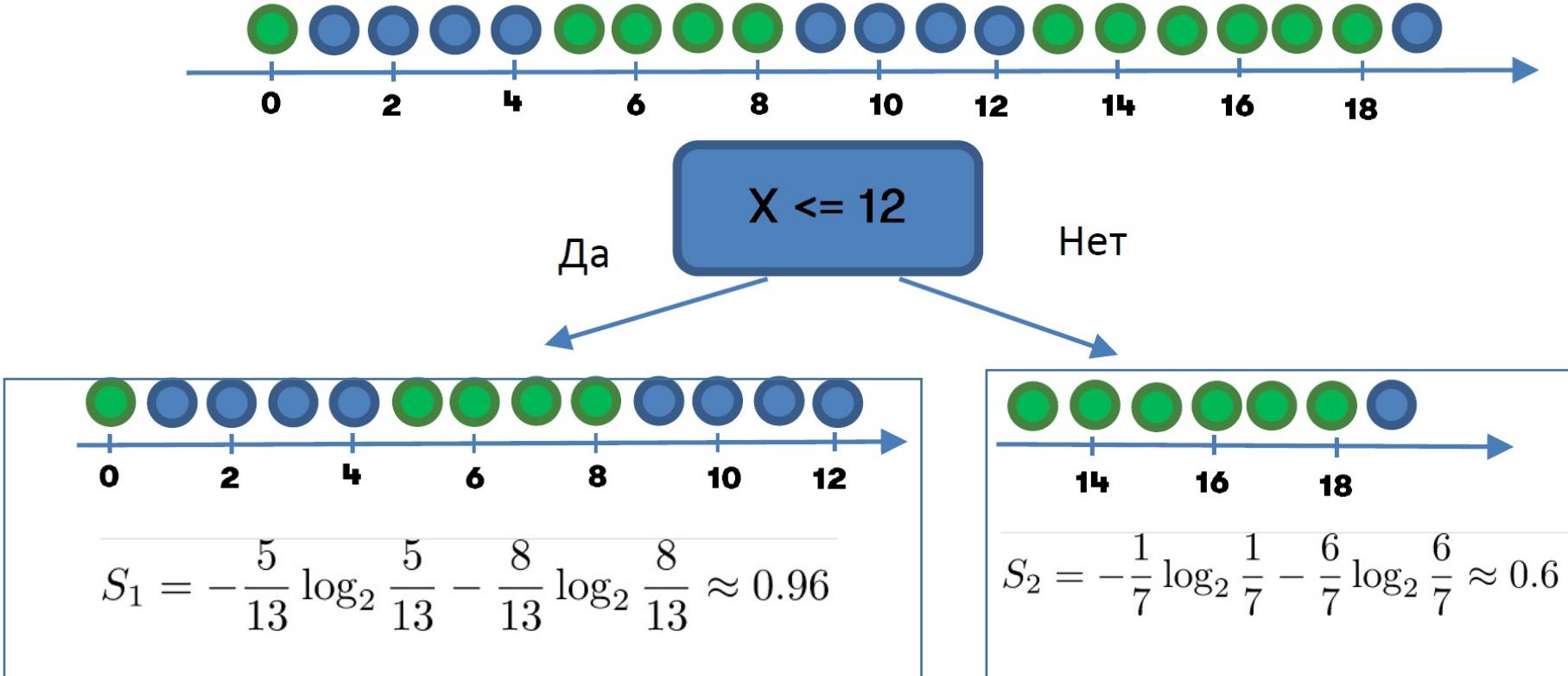
$$S_1 = -\frac{5}{13} \log_2 \frac{5}{13} - \frac{8}{13} \log_2 \frac{8}{13} \approx 0.96$$



$$S_2 = -\frac{1}{7} \log_2 \frac{1}{7} - \frac{6}{7} \log_2 \frac{6}{7} \approx 0.6$$

# Пример построения

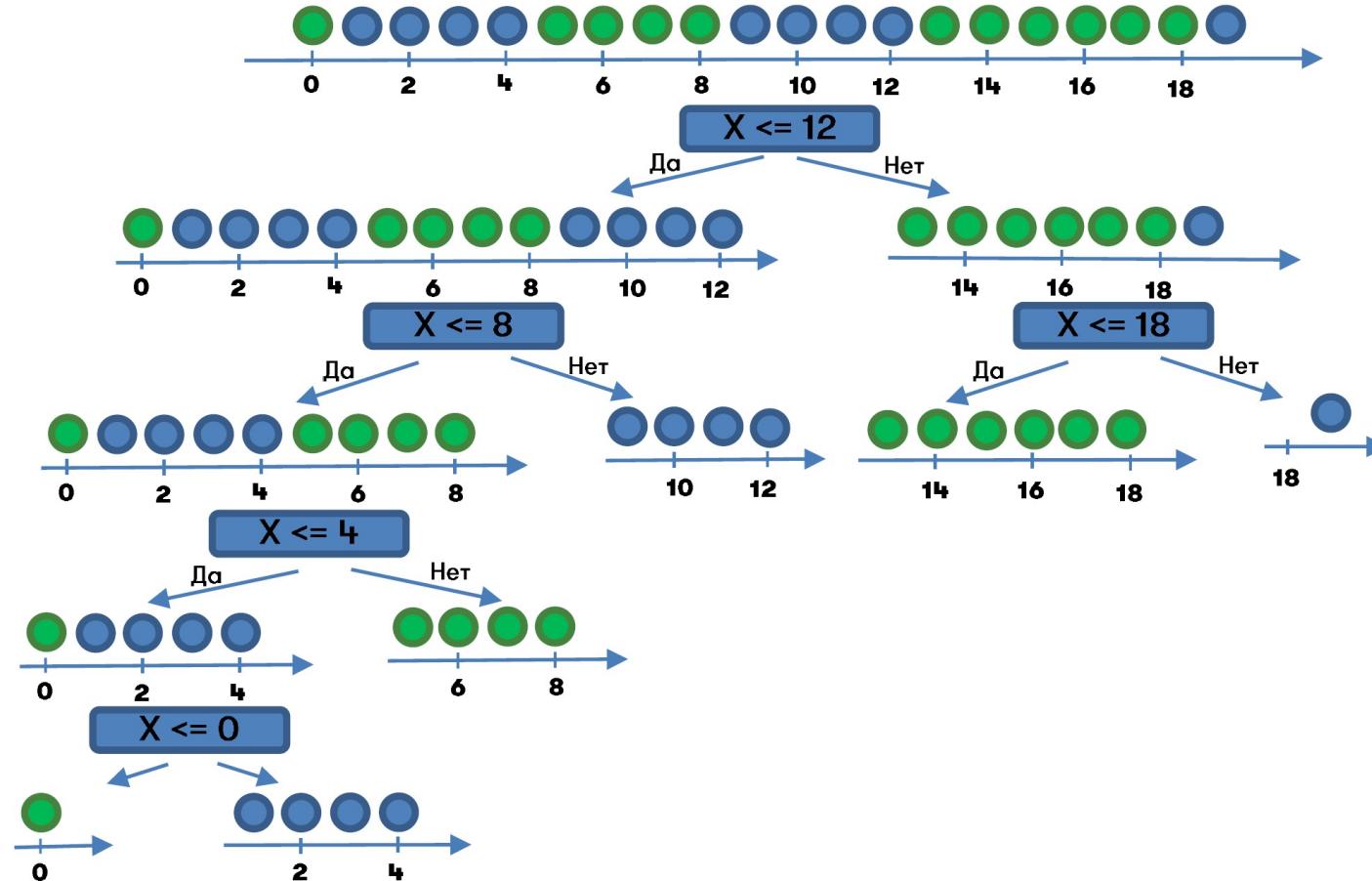
$$S_0 = -\frac{9}{20} \log_2 \frac{9}{20} - \frac{11}{20} \log_2 \frac{11}{20} \approx 1$$



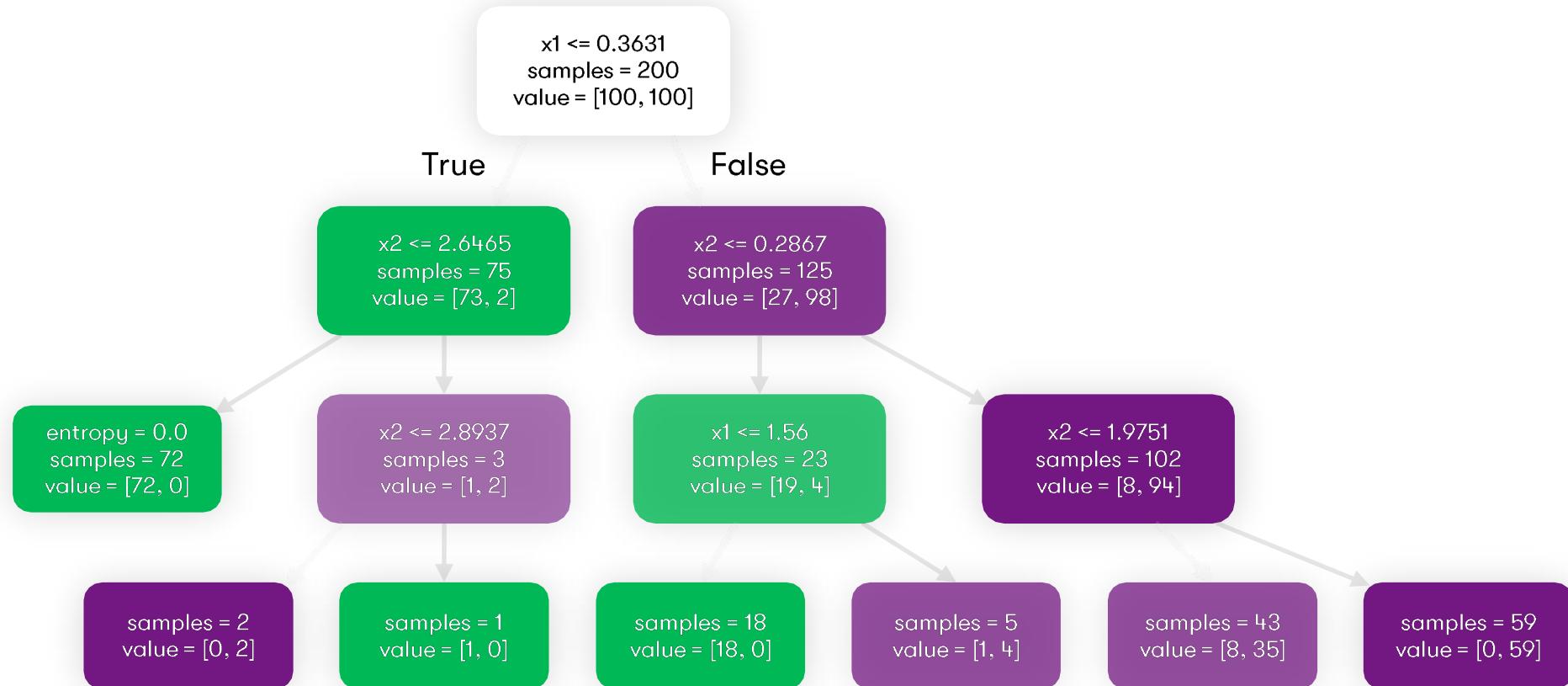
$$IG(Q) = S_0 - \sum_{i=1}^q \frac{N_i}{N} S_i, \quad \longrightarrow$$

$$IG(x \leq 12) = S_0 - \frac{13}{20} S_1 - \frac{7}{20} S_2 \approx 0.16.$$

# Пример построения

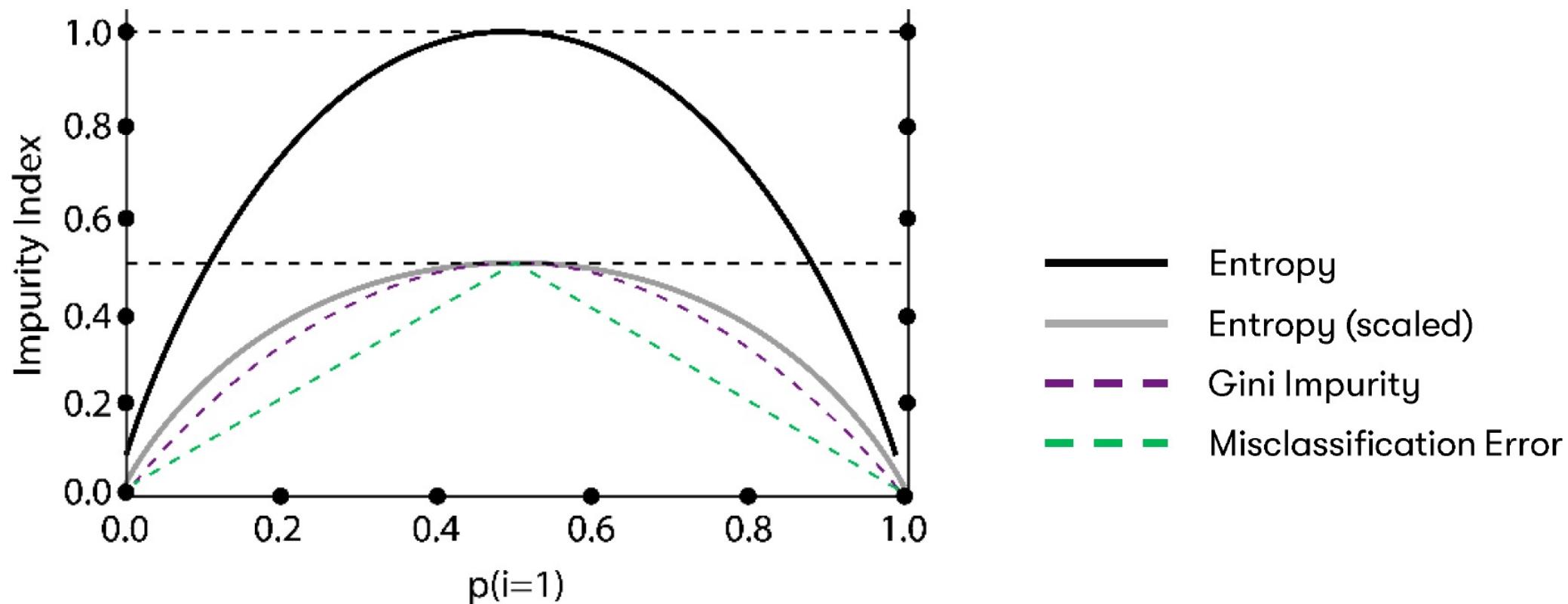


# Визуализация «боевого» дерева решений

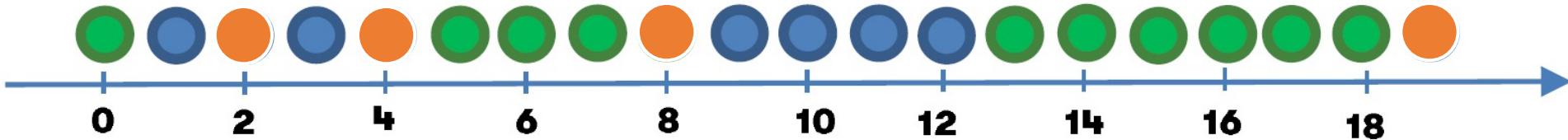


Последовательность логических правил  
Константа в листьях

# Не энтропией единой



Окей, а что с мультиклассовой классификацией?



Что тогда с формулой энтропии?

$$S = -\frac{10}{20} \log \frac{10}{20} - \frac{4}{20} \log \frac{4}{20} - \frac{6}{20} \log \frac{6}{20}$$

# А что с задачей регрессии?

$$D = \frac{1}{\ell} \sum_{i=1}^{\ell} (y_i - \frac{1}{\ell} \sum_{j=1}^{\ell} y_j)^2,$$

Variance (дисперсия)

где  $\ell$  - число семплов в листе,  
 $y_i$  значение таргет переменной.

Другими словами, мы теперь делим таким образом, чтобы дисперсия в обеих листах была примерно одинаковой.

# Плюсы

Учитывает Нелинейность в данных

Интерпретируемость и возможность визуализации

Интуитивная настройка гипер параметров  
(критерии разделения, остановки)

Может работать с разными типами данных  
(но не всегда из коробки)

Очень хороши при классификации



# Минусы

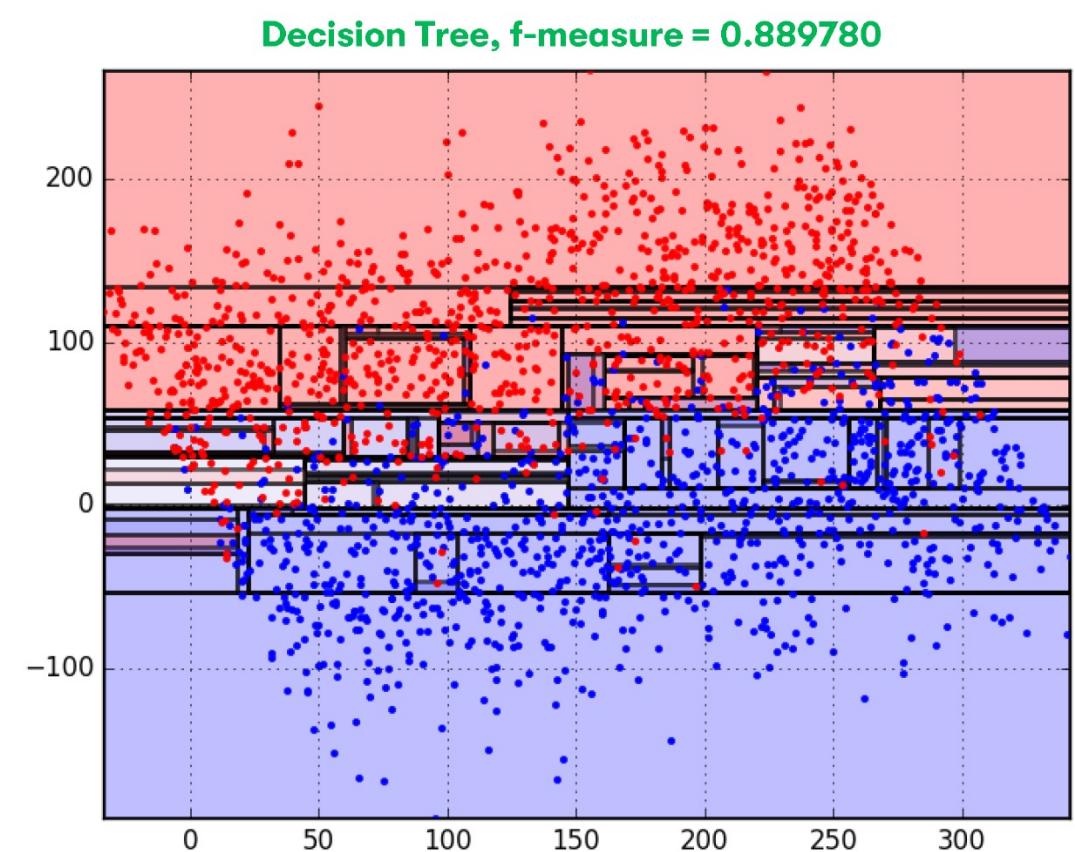
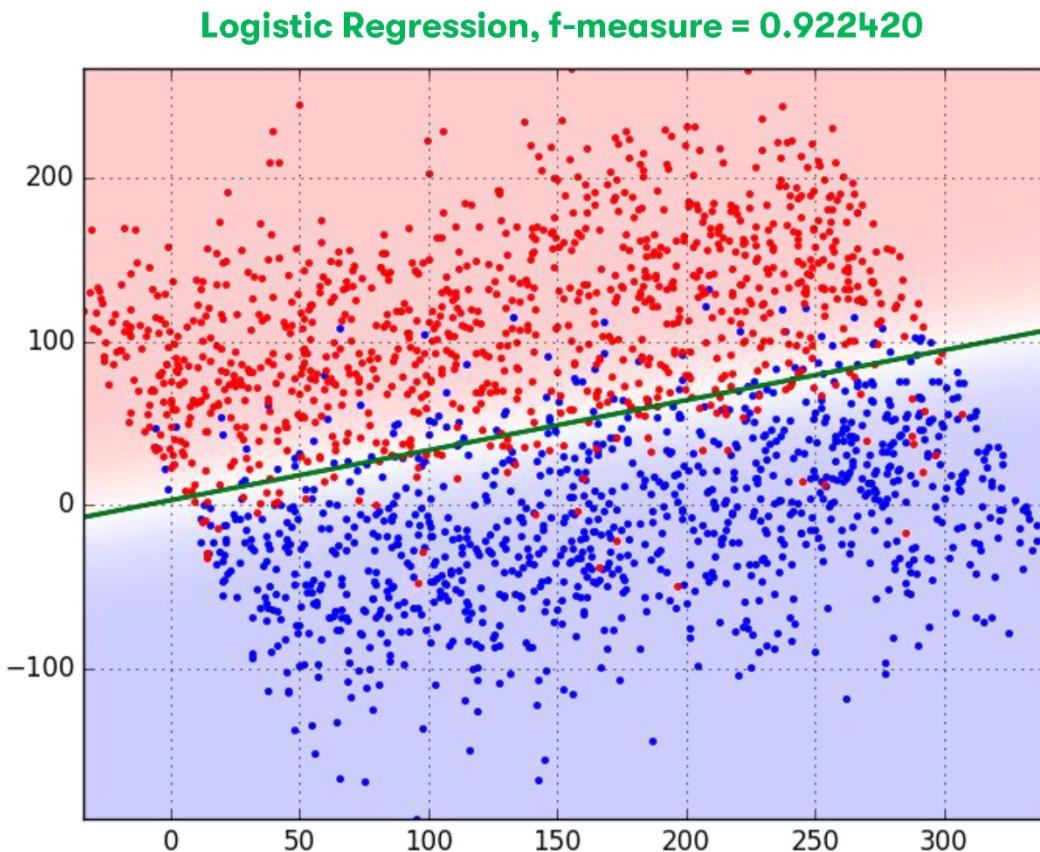
**Жадный** Алгоритм построения дерева пересужняет структуру дерева и, как следствие, переобучается

**Фрагментация выборки:** чем дальше от корня, тем меньше статистическая надежность

**Высокая чувствительность** к шуму к составу выборки и КИ



# Пример плохой классификации



# Ансамбли деревьев



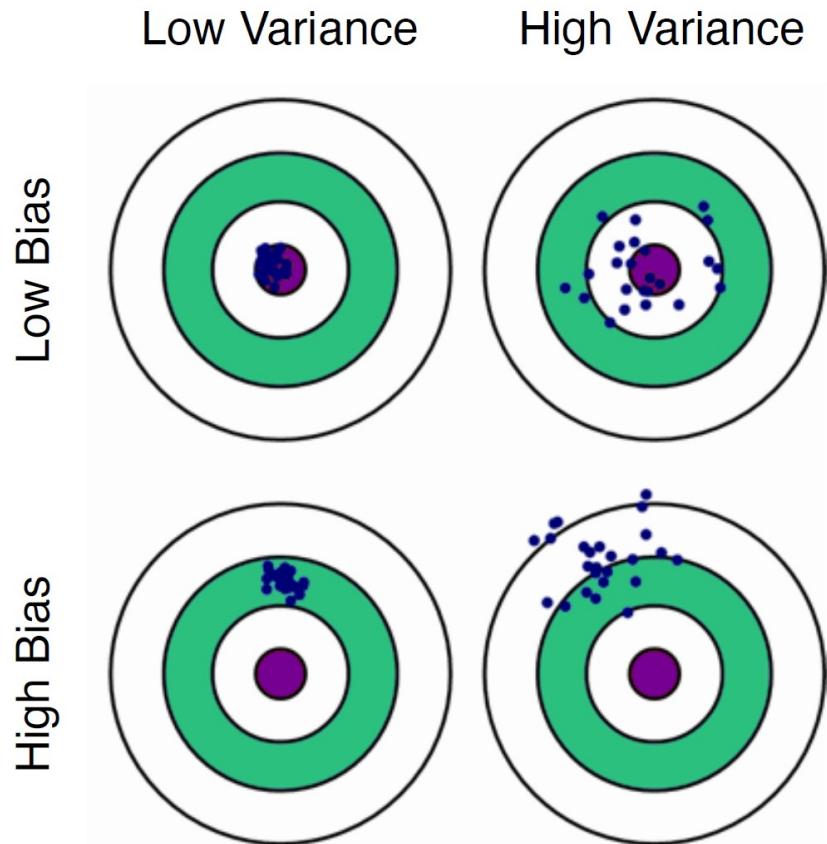
# Немного про ошибку алгоритмов

$$\text{Err}(\vec{x}) = \text{Bias}(\hat{f})^2 + \text{Var}(\hat{f}) + \sigma^2$$

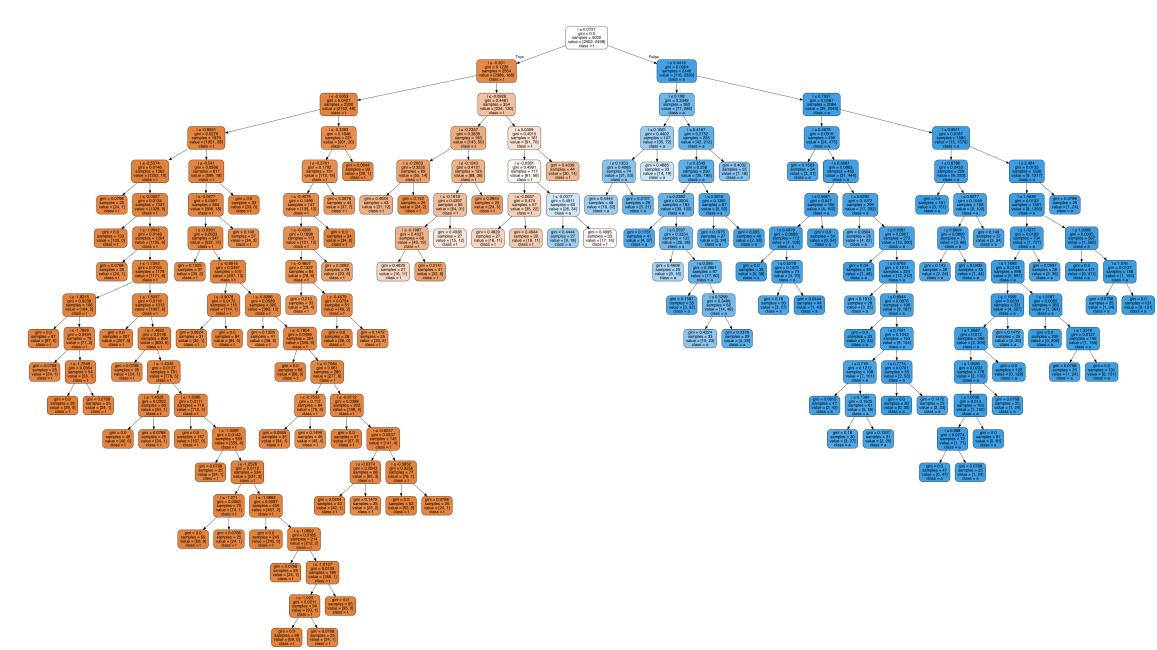
Ошибка состоит из:

- квадрата смещения:  $\text{Bias}(\hat{f})$  – средняя ошибка по всевозможным наборам данных;
- дисперсии:  $\text{Var}(\hat{f})$  – вариативность ошибки, то, на сколько ошибка будет отличаться, если обучать модель на разных наборах данных;
- неустранимой ошибки:  $\sigma^2$ . (шум)

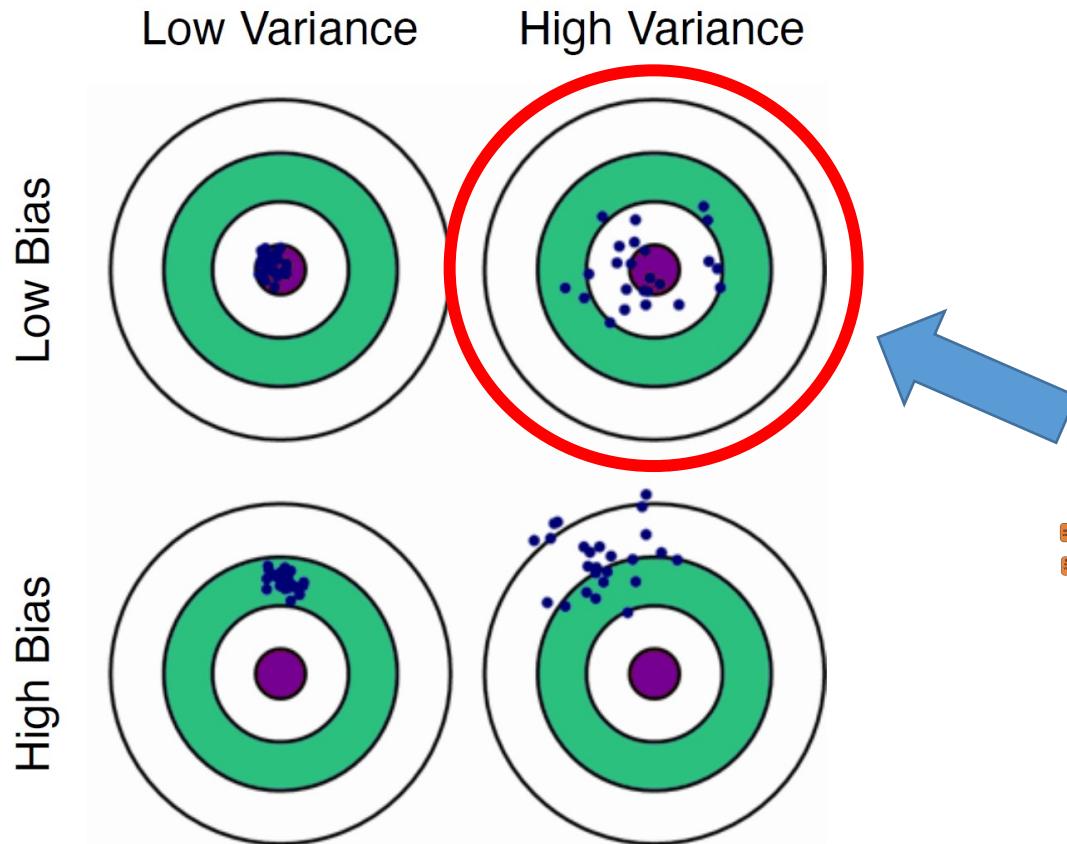
# Интуиция за ансамблями



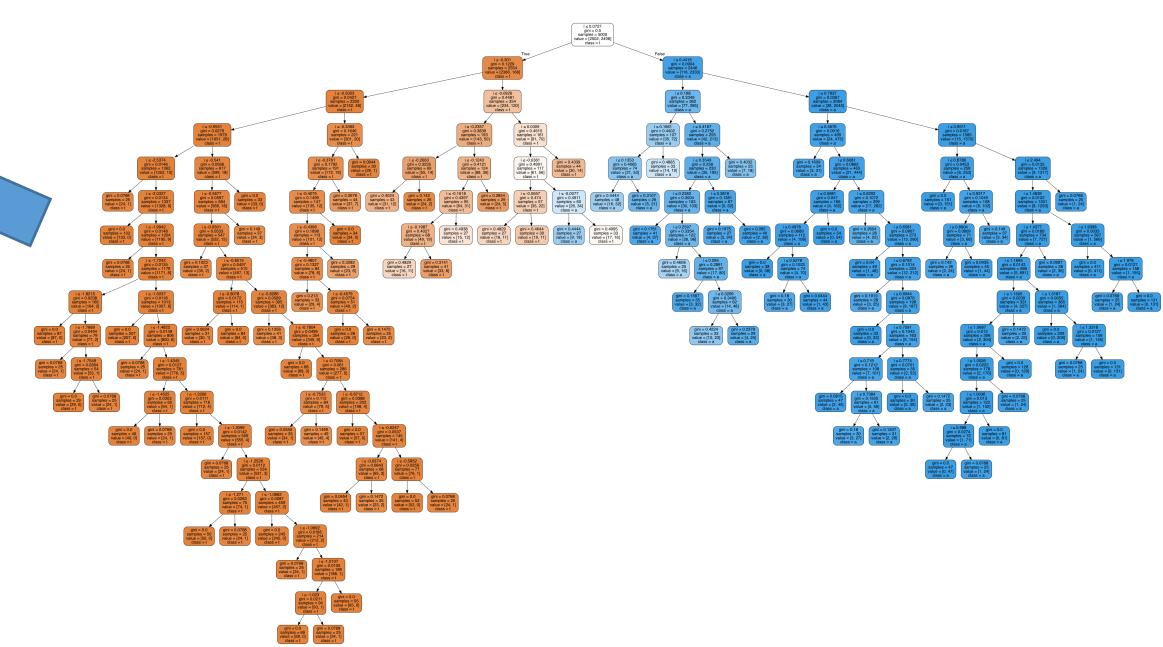
Угадайте, где здесь будет переобученное дерево  
Бесконечной глубины



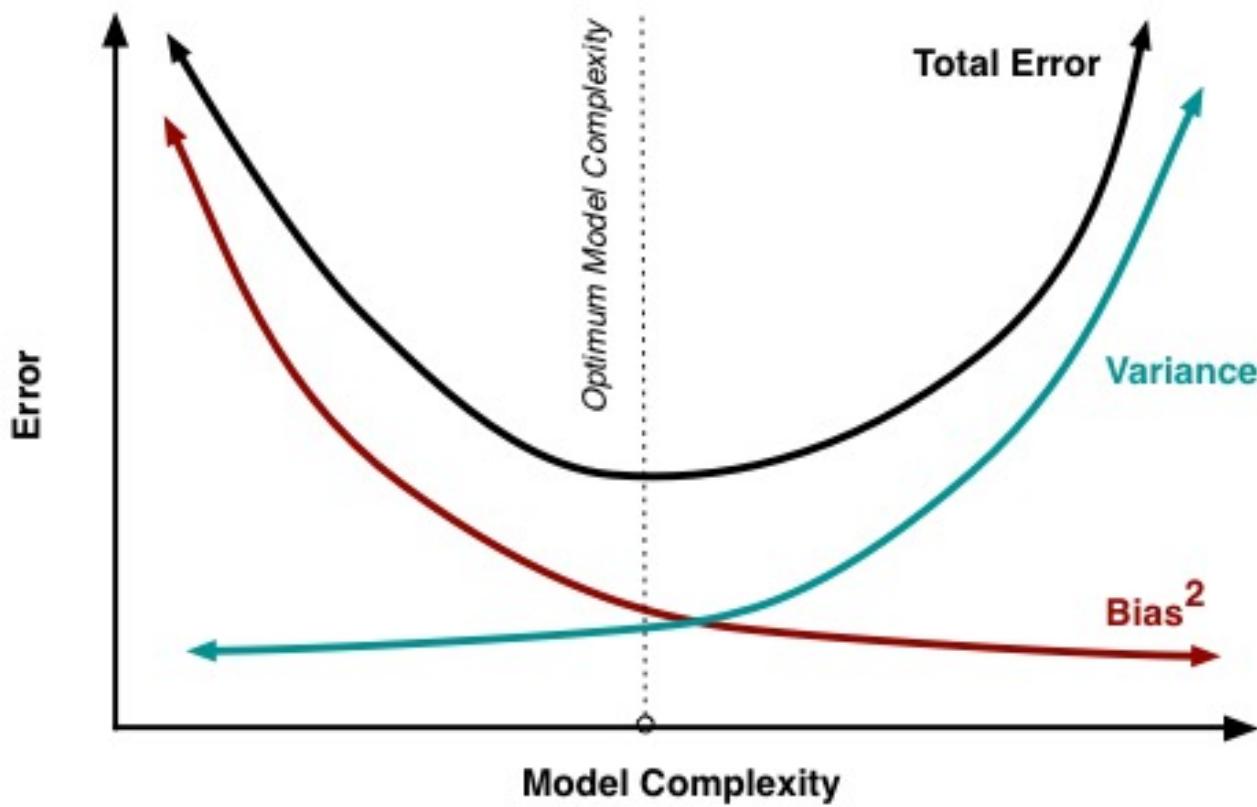
# Интуиция за ансамблями



Угадайте, где здесь будет переобученное дерево  
Бесконечной глубины



# Интуиция за ансамблями



Тоже самое, но по другому визуализированное

# Почему именно так?

- Если не ограничивать глубину дерева, то оно может найти разбиение под каждый элемент в обучающей выборке
- Но такие разбиения скорее всего будут ошибочными и учитывать даже малейшие изменения в признаках
- Как следствие, дерево идеально поняло, какие именно значения надо предсказывать (мин баес), но предсказания шатает из-за переобучения (макс вариенс)

Окей, а что если таких деревьев будет  $N$ , и они будут некоррелированные?

$$\begin{aligned}\mathbb{E}_x \varepsilon_i(x) &= 0, \\ \mathbb{E}_x \varepsilon_i(x) \varepsilon_j(x) &= 0, i \neq j.\end{aligned}$$

Ошибки не смещены (мин баес) и некоррелированные

$$\mathbb{E}_x (b_j(x) - y(x))^2 = \mathbb{E}_x \varepsilon_j^2(x).$$

Как выглядит ошибка в случае одного базового алгоритма

$$\mathbb{E}_x \left( \frac{1}{N} \sum_{j=1}^n b_j(x) - y(x) \right)^2 = \mathbb{E}_x \left( \frac{1}{N} \sum_{j=1}^N \varepsilon_j(x) \right)^2$$

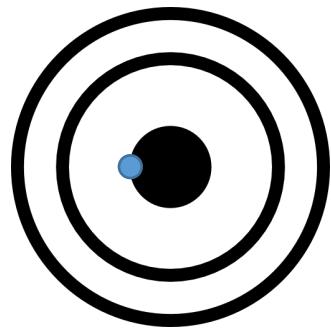
А вот так она выглядит, если их  $N$  штук

$$\mathbb{E}_x \left( \frac{1}{N} \sum_{j=1}^N \varepsilon_j(x) \right)^2 = \frac{1}{N^2} \mathbb{E}_x \left( \sum_{j=1}^N \varepsilon_j^2(x) + \underbrace{\sum_{i \neq j} \varepsilon_i(x) \varepsilon_j(x)}_{=0} \right) = \frac{1}{N} E_1.$$

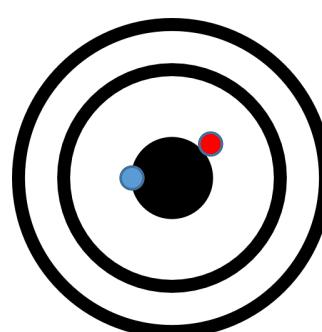
После Магии раскрытия скобок получаем уменьшенную в  $N$  раз ошибку

Здесь должна была быть большая выкладка про то, почему Variance тоже уменьшается в  $N$  раз...

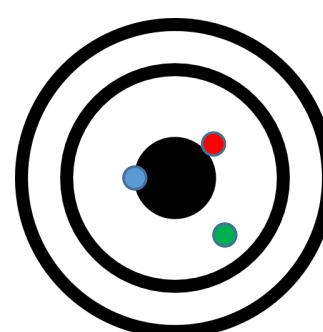
Но давайте лучше представим, почему  $N$  алгоритмов сработают лучше, чем 1



Предсказание  
первого дерева

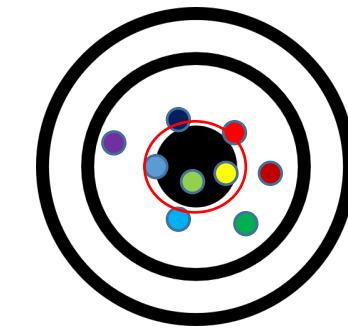


+ Предсказание  
второго



+ Предсказание  
третьего

• • •



+ Предсказание  
 $N$ -го дерева

Если взять  
Среднее, то  
получим  
мин варианс

# В итоге получаем Bagging

- смещение композиции, полученной с помощью бэггинга, совпадает со смещением одного базового алгоритма
- если базовые алгоритмы некоррелированы, то дисперсия композиции в  $N$  раз меньше дисперсии отдельных алгоритмов

Нужны слабо скоррелированные  
алгоритмы с маленьким смещением, при  
этом разброс мы уменьшим за счет  
bagging

# Random Forest

Давайте тогда возьмем за основу решающие деревья, потому что у них маленький Баес, но большой Варианс.  
В итоге получим...

## Случайный лес - bagging над решающими деревьями

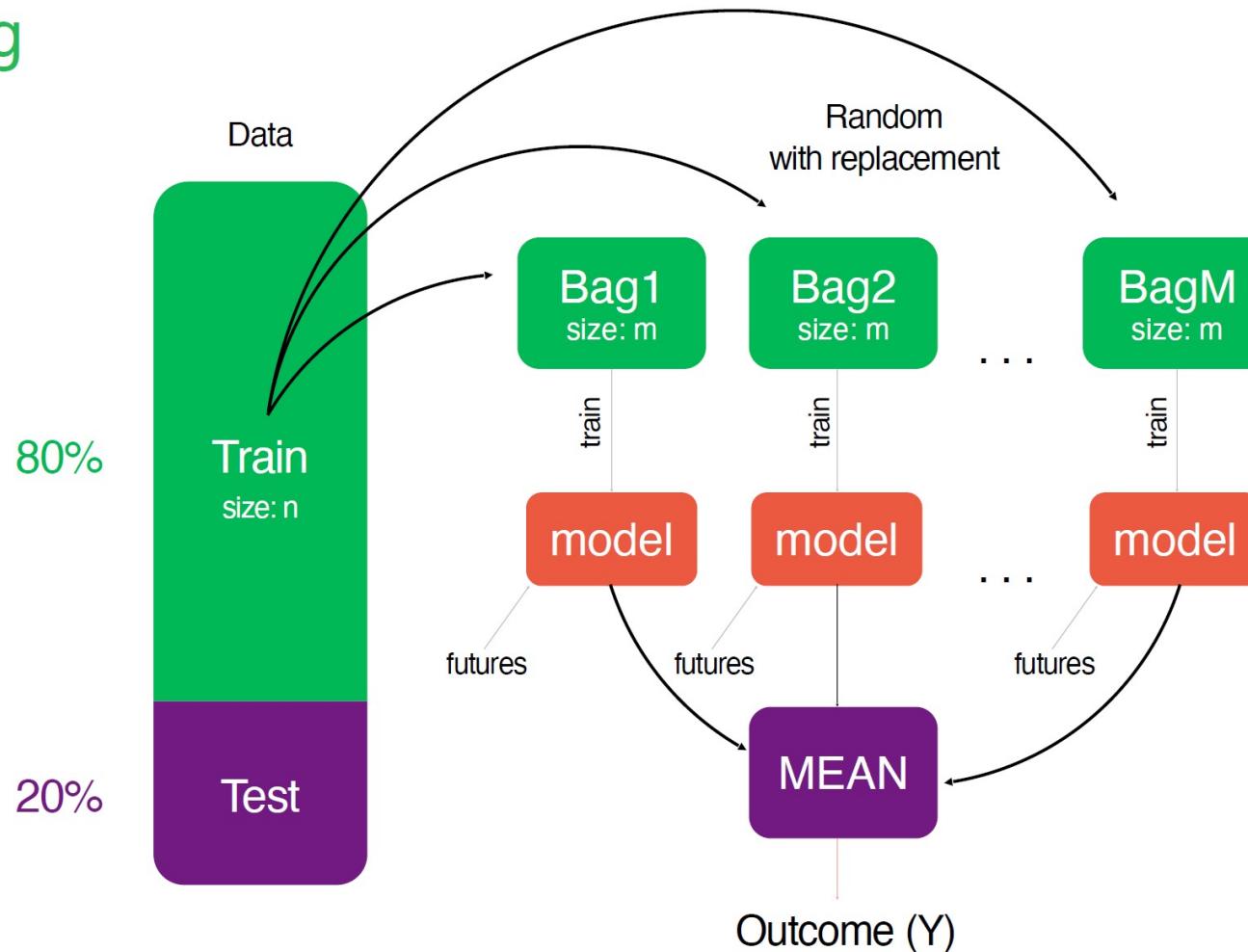
N раз:

1. Сгенерировать выборку X с помощью бутсрэпа
2. Выбрать m случайных признаков
3. Обучить классификатор

Ответом является композиция  $a_N(x) = \frac{1}{N} \sum_{n=1}^N b_n(x)$

# Random forest

## Bagging



Bagging = Bootstrap\* + Aggregation

\*Bootstrap - сэмплирование объектов из выборки с возвращением

# Финальная интуиция

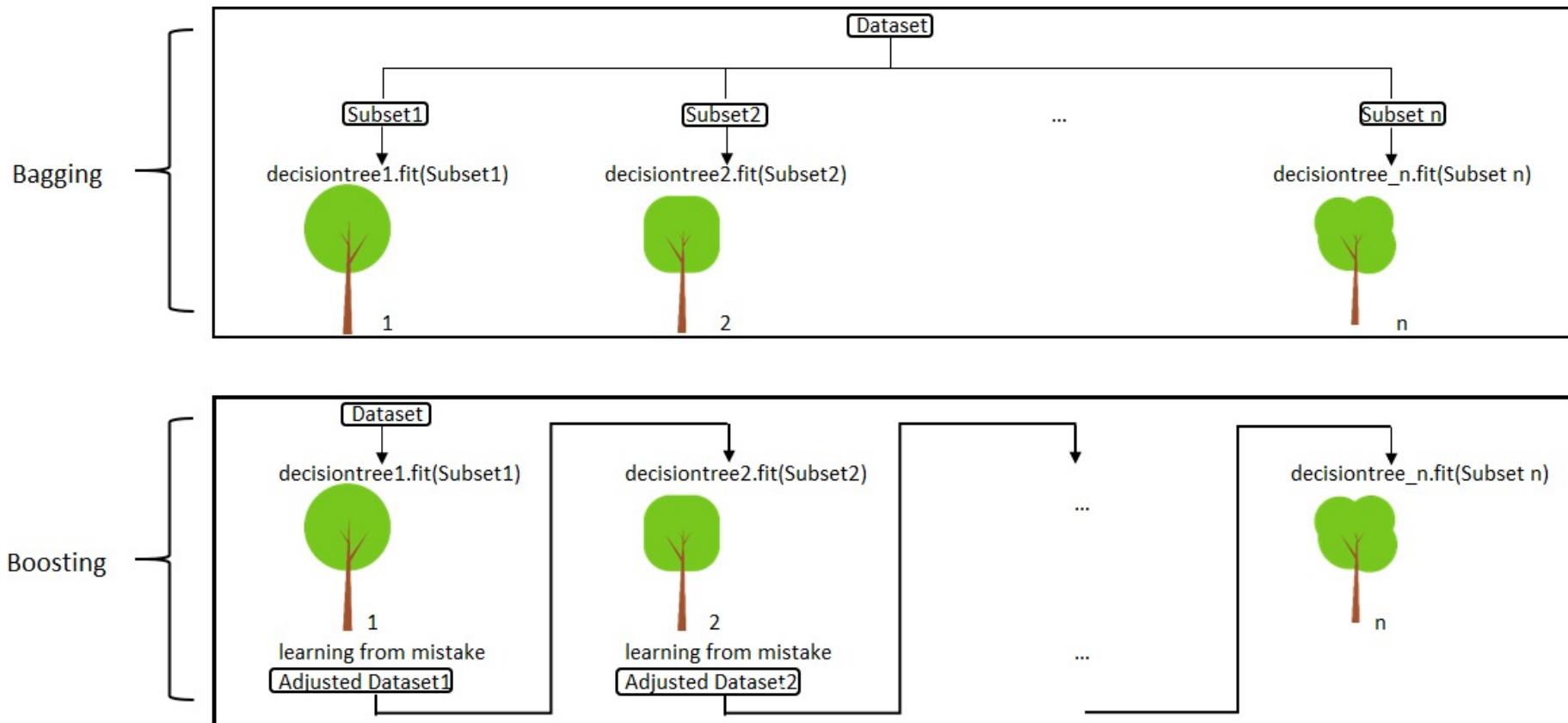


**Bootstrap** - сэмплирование объектов из выборки с возвращением

**Bagging = Bootstrap + Aggregation** - агрегирование ответов алгоритмов, обученных на бустрапированных выборках

**Random Forest = Bagging trees + sampling of features** -  
рандомизация и на уровне объектов, и на уровне признаков  
(чтобы алгоритмы не коррелировали)

# Ну и что там с вашим бустингом?



# Математика бустинга

Решаем задачу регрессии.

- Пусть есть модель,  $F_1(x) = y$
- Обучим базовый алгоритм на остатках от модели  $F_1$ ,  $h_1(x) = y - F_1(x)$
- Получим новую модель, сложив предсказания  $F_1$  и  $h_1$ ,  $F_2(x) = F_1(x) + h_1(x)$

$$F(x) = F_1(x) \mapsto F_2(x) = F_1(x) + h_1(x) \cdots \mapsto F_M(x) = F_{M-1}(x) + h_{M-1}(x)$$

Возможные твики:

- 1) Вместо  $F_1$  использовать какие-то наивные предсказания
- 2) Этот пример хорошо работает для MSE, но что если мы хотим минимизировать другую метрику?  
(или вообще, у нас задача классификации)

# Пример базовой реализации

PersonID	Age	Tree1 Prediction	Tree1 Residual	Tree2 Prediction	Combined Prediction	Final Residual
1	13	19.25	-6.25	-3.57	15.68	2.68
2	14	19.25	-5.25	-3.57	15.68	1.68
3	15	19.25	-4.25	-3.57	15.68	0.68
4	25	57.20	-32.20	-3.57	53.63	28.63
5	35	19.25	15.75	-3.57	15.68	-19.32
6	49	57.20	-8.20	7.13	64.33	15.33
7	68	57.20	10.80	-3.57	53.63	-14.37
8	71	57.20	13.80	7.13	64.33	-6.67
9	73	57.20	15.80	7.13	64.33	-8.67

MSE.Tree1	MSE.Combined
1993.55	1764.57

# Дополненная Математика бустинга

Окей, тогда будем обучаться на псевдо остатках от функций потерь,  
а  $F_0$  будем инициализировать константным прогнозом, который минимизирует нашу функцию потерь.

Инициализируем модель константным значением,  $F_0(x) = \arg \min_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$

Для  $M$  итераций:

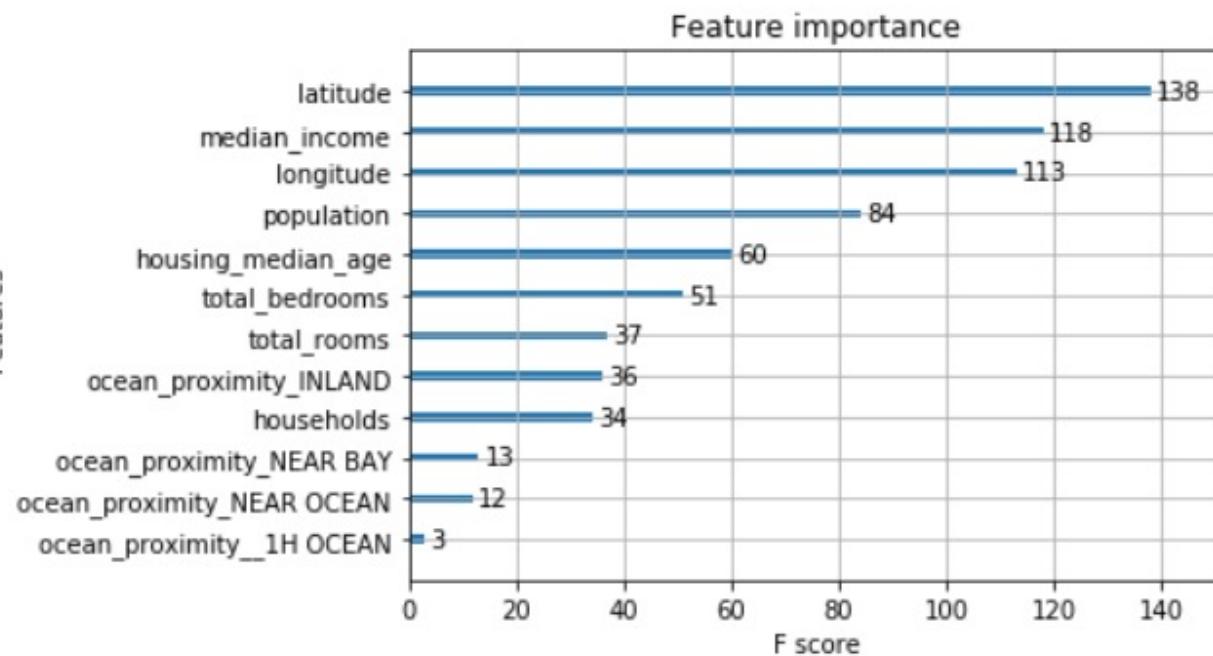
- 1) Считаем псевдо остатки  $r_{im} = - \left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(x)=F_{m-1}(x)}$  for  $i = 1, \dots, n$
- 2) Обучаем наш алгоритм  $h_m(x)$  на псевдо остатки
- 3) Считаем коэффициент  $\gamma_m$ , с которым обновим предсказания
- 4) Обновляем нашу модель  $F_m(x) = F_{m-1}(x) + \gamma_m h_m(x)$

# Пример допиленной реализации

Age	F0	PseudoR esidual0	h0	gamma0	F1	PseudoR esidual1	h1	gamma1	F2
13	40.33	-27.33	-21.08	1	19.25	-6.25	-3.57	1	15.68
14	40.33	-26.33	-21.08	1	19.25	-5.25	-3.57	1	15.68
15	40.33	-25.33	-21.08	1	19.25	-4.25	-3.57	1	15.68
25	40.33	-15.33	16.87	1	57.20	-32.20	-3.57	1	53.63
35	40.33	-5.33	-21.08	1	19.25	15.75	-3.57	1	15.68
49	40.33	8.67	16.87	1	57.20	-8.20	7.13	1	64.33
68	40.33	27.67	16.87	1	57.20	10.80	-3.57	1	53.63
71	40.33	30.67	16.87	1	57.20	13.80	7.13	1	64.33
73	40.33	32.67	16.87	1	57.20	15.80	7.13	1	64.33

# На последок про метрики качества...

Features



- “weight” Сколько раз фича появилась в дереве
- “gain” Сколько информации получаем, когда делим по этой фиче
- “cover” Сколько в среднем семплов мы «задеваем», когда используем фичу

Спасибо за внимание!