AI Community

5. Решающие деревья и случайный лес

План на сегодня

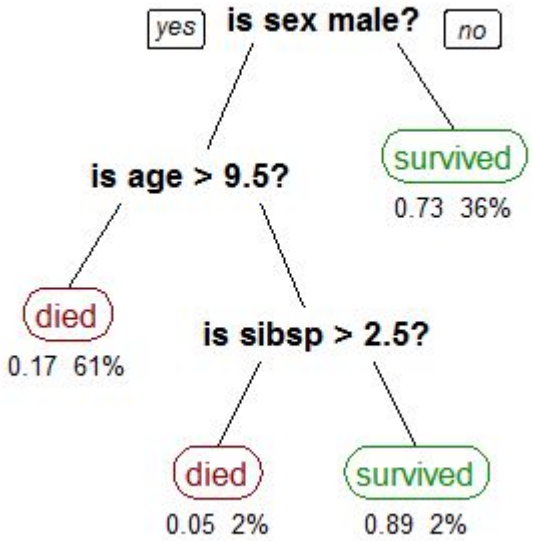
1. Решающие деревья
    a. Определение
    b. Примеры решающих деревьев
    c. Виды деревьев в различных ML библиотеках
2. Построение решающего дерева
    a. Определения
    b. Алгоритм C4.5
    c. Процедура выбора
3. Ансамбли
    a. Бэггинг
    b. Случайный лес
    c. Бустинг

Решающие деревья

Решающее дерево

Решающее дерево (дерево решений) — это алгоритм, принимающий решение посредством цепочки принятых решений.

Итоговое решение определяется путем прохода дерева от корня к листу. Переход по дереву происходит путем принятия решения в узле дерева.

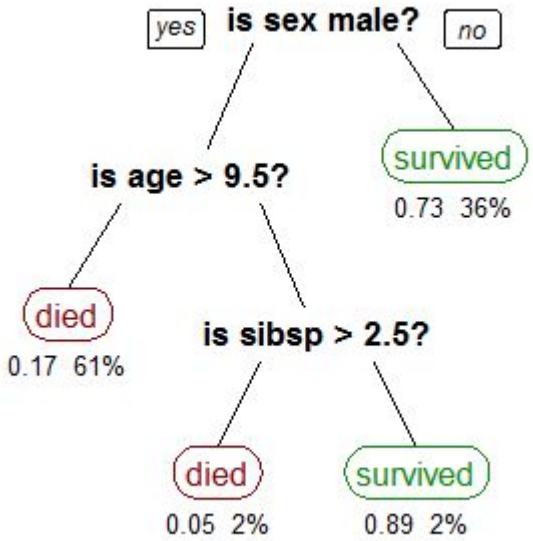


Решающее дерево

Решающее дерево (дерево решений) — это алгоритм, принимающий решение посредством цепочки принятых решений.

Принятие решения в каждом конкретном узле происходит вычислением логической функции.

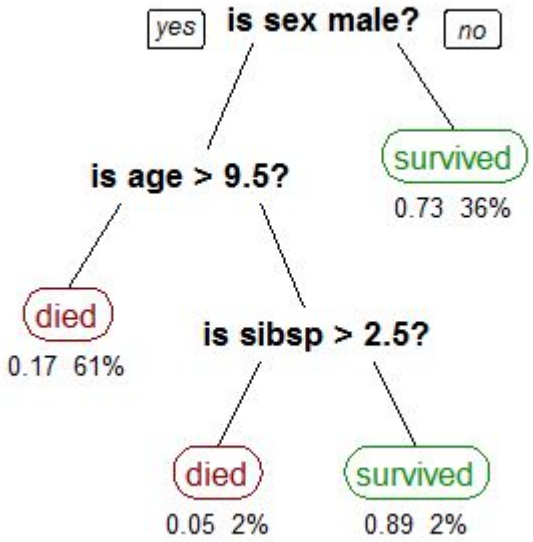
Ответ “да” — идем в одну сторону, ответ “нет” — в другую.



Решающее дерево

Решающее дерево (дерево решений) — это алгоритм, принимающий решение посредством цепочки принятых решений.

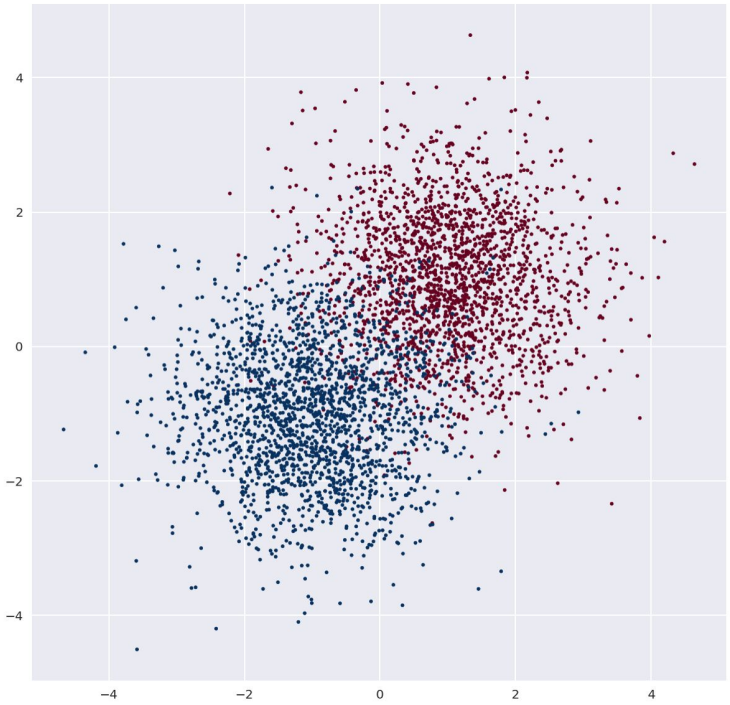
Дерево интерпретируется как логическая функция. Например, согласно дереву справа значение `Survived` принимается согласно логическому условию: `Sex == "female" | ((Age > 9.5) & (Sibsp > 25))`.



Решающее дерево на примере

Рассмотрим на примере:

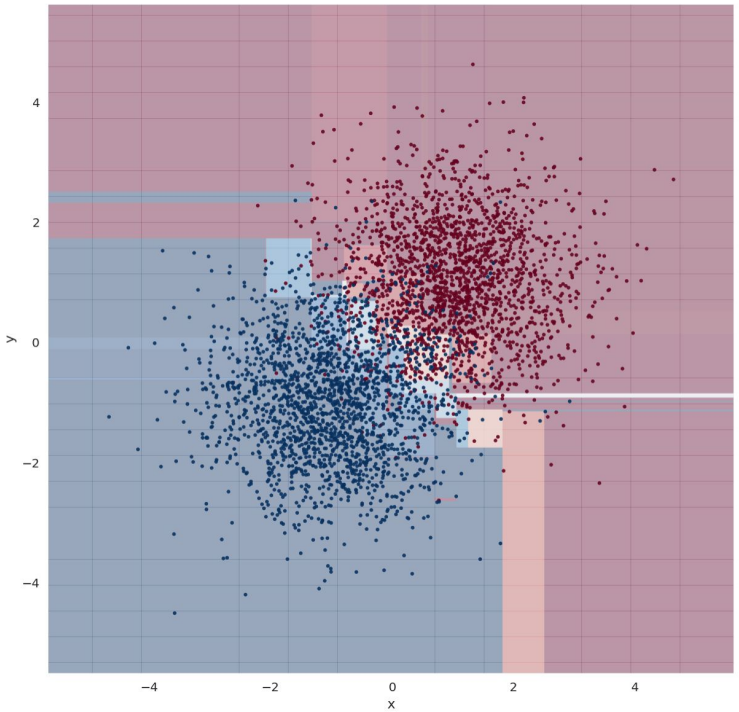
- Два класса объектов
- Объекты имеют два числовых признака — положение на плоскости
- Классы частично пересекаются



Решающее дерево на примере

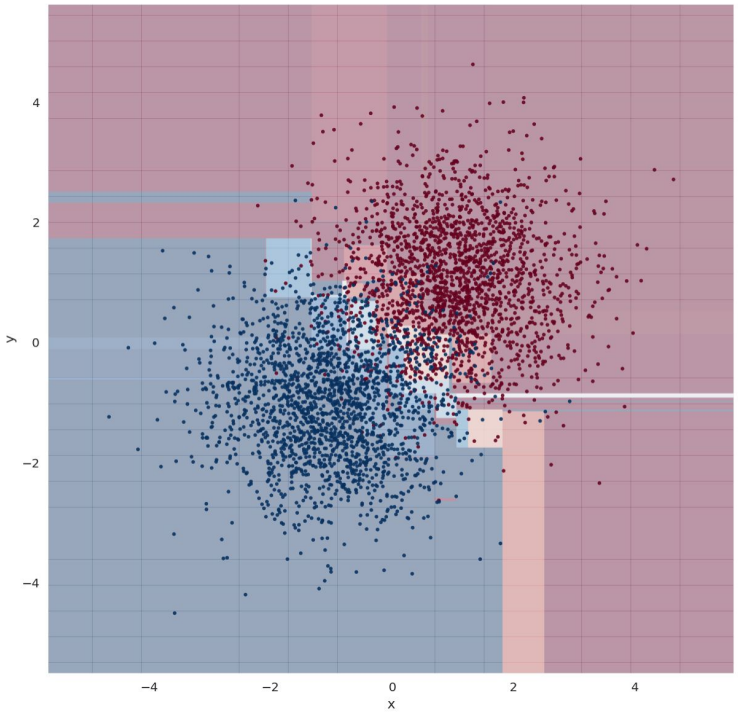
При построении дерево разделяет плоскость чередой прямых линий — такое поведение определяется тем, что логические условия для числовых переменных имеют вид “x < 5”. В этом случае 5 — **критическое значение** признака.

Яркость цвета описывает уверенность дерева в ответе — пропорцию классов объектов в соответствующем листе дерева.

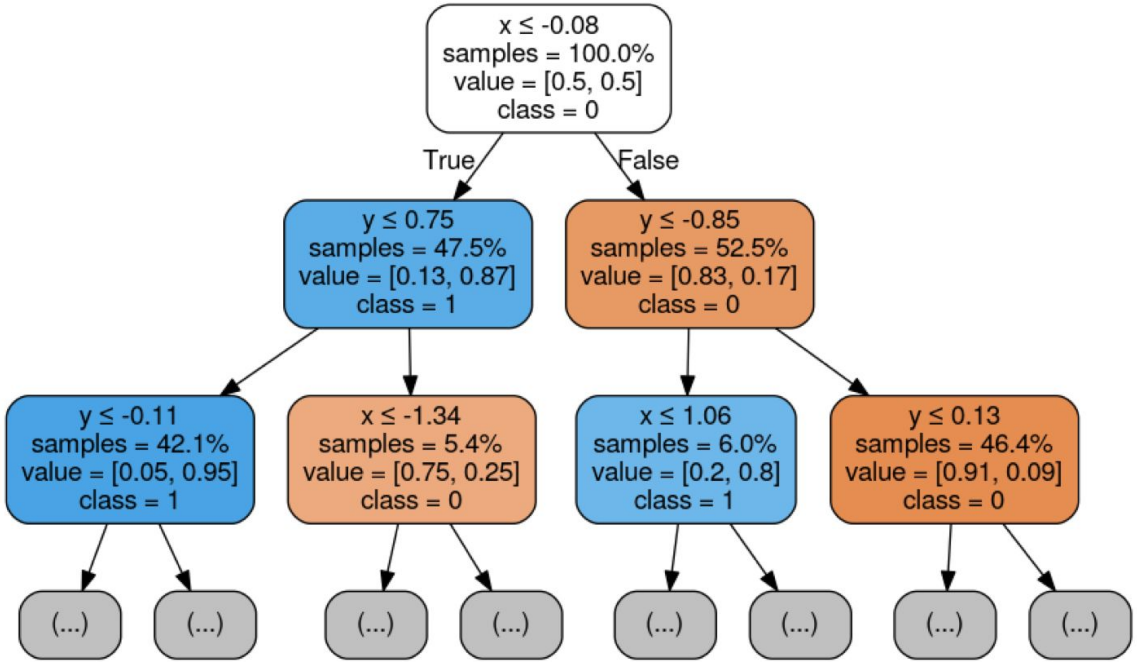


Решающее дерево на примере

Полученная функция является кусочно-непрерывной — представляет собой набор сегментов, в каждом из которых функция принимает константное значение.

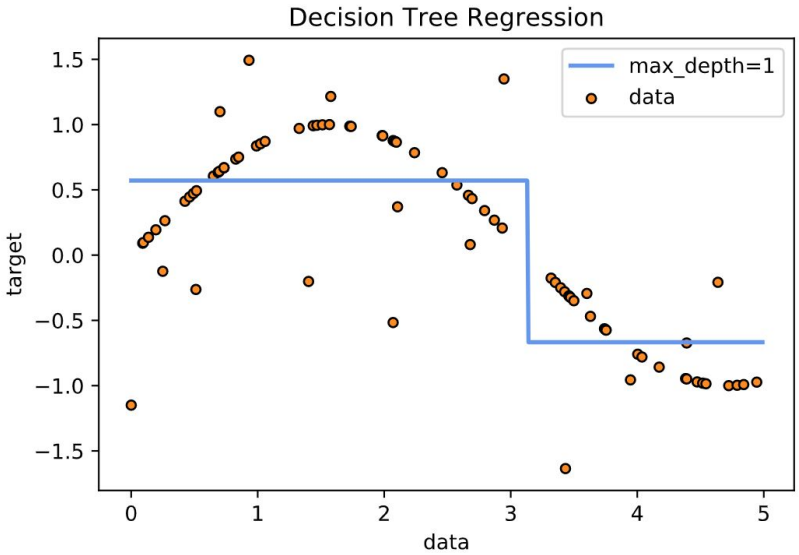


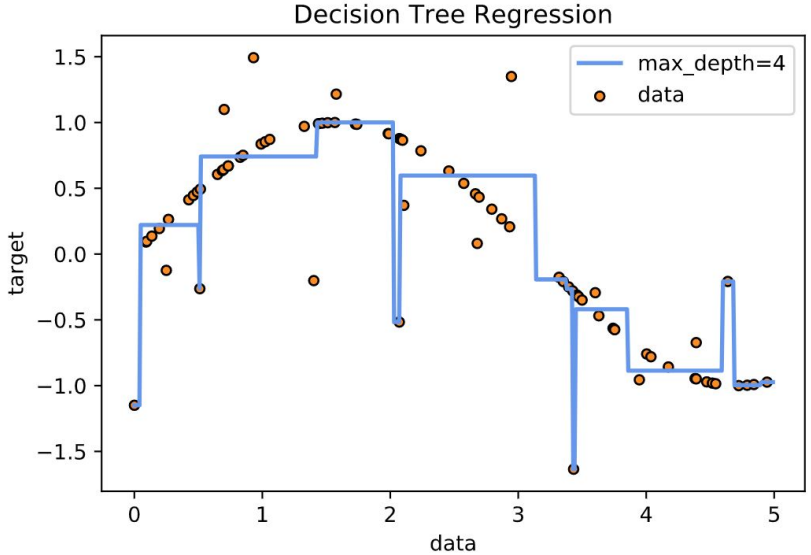
Решающее дерево на примере



Так выглядит дерево внутри, если раскрыть его на два уровня от корня.

Решающее дерево для регрессии





В случае регрессии дерево — кусочно-постоянная функция.

Проблема переобучения в решающих деревьях

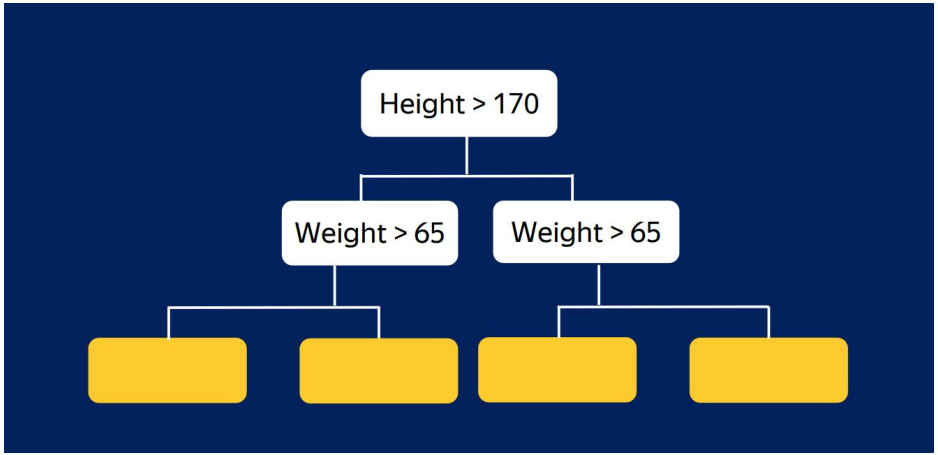
Решающие деревья крайне чувствительны к выбросам.
Для борьбы с этим применяют ряд эвристик, которые влияют на структуру дерева:

- Ограничение максимальной глубины дерева
- Ограничение на общее количество листьев в дереве
- Ограничение на минимальное число объектов в листе
- Усечение построенного дерева (pruning)
- Использование деревьев специального вида

Решающие деревья в CatBoost

В библиотеке CatBoost используются небрежные решающие деревья (Oblivious Trees).

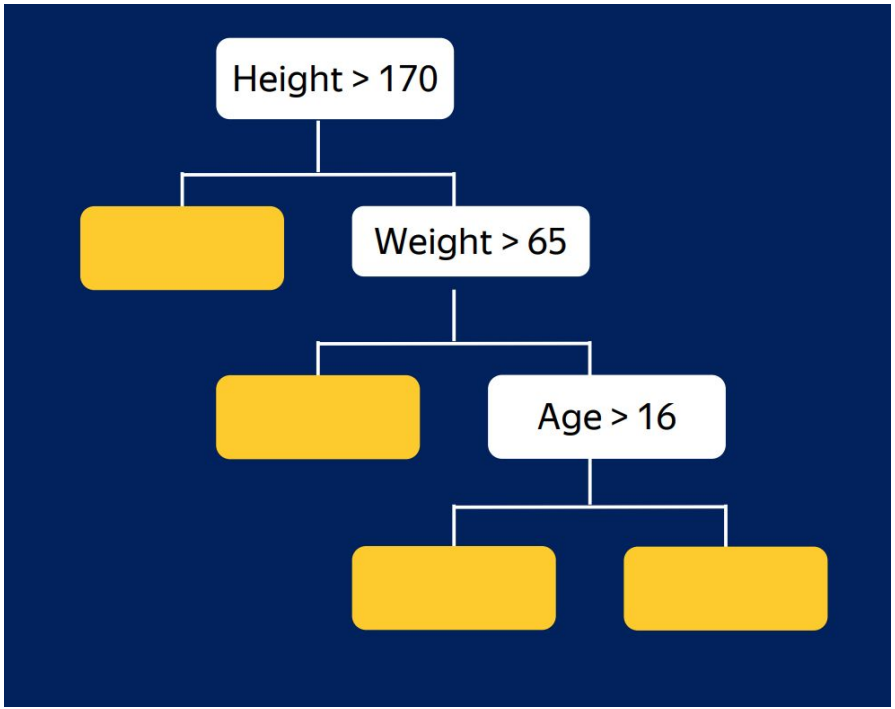
В них на каждом уровне дерева расположено гарантированно одно и то же условие. Это обеспечивает регуляризацию и позволяет хранить дерево в многомерном массиве.



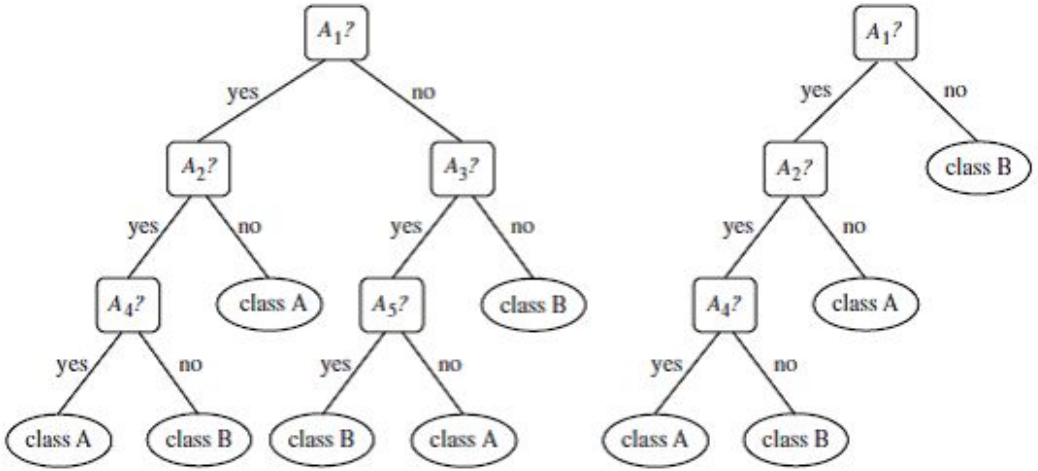
Решающие деревья в LightGBM

В библиотеке LightGBM деревья строятся в глубину, по одной вершине за раз.

Такие деревья можно представить как связный список.



Решающие деревья в XGBoost



В библиотеке XGBoost используется усечение (pruning).
В примере выше правое дерево является усеченной версией левого.

Построение решающего дерева

Определения: критерий информативности

Для описания алгоритма потребуется несколько определений.

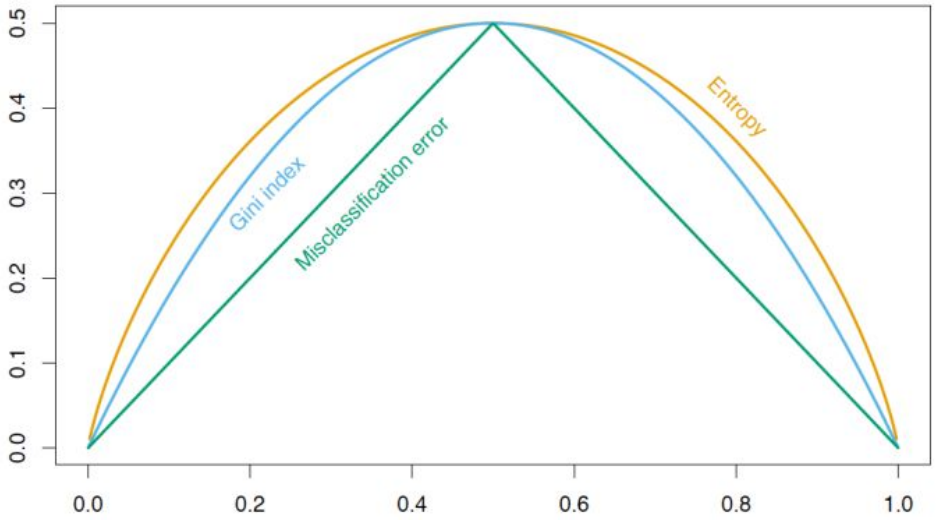
Критерий информативности — это величина, определенная для значений из обучающей выборки, попавших в лист дерева. Говорит о том, насколько хорошо в работает предикат в этом листе.

Рассмотрим на примерах.

Критерии информативности

Существует несколько критериев информативности:

1. Индекс Джини: 2p(1-p)
2. Ошибка классификации: 1-max(p, 1 - p)
3. Энтропия: -p*log(p)-(1-p)*log(1-p)



Энтропия

Энтропия — статистическая величина, измеряющая однорость данных. Низкая энтропия означает, что данные однородные, высокая — что разнообразные. Она считается по следующей формуле.

$$H(R) = \sum_{x \in X} -p_R(x) \log_2 p_R(x)$$

R — объекты, для которых строится решающее правило

X — множество возможных классов

$p_R(x)$ — вероятность класса x в R (частота)

Пример: Пусть в данных класс 0 встретился 1 раз, класс 1 встретился 99 раз. Тогда энтропия равна

$$H(R) = -0.01 \cdot \log_2 0.01 - 0.99 \cdot \log_2 0.99 = 0.0808$$

Энтропия используется в задачах классификации.

Information Gain

Information Gain (рус. прирост информации): Условимся что для каждого признака f можно получить разбиение множества R на семейство непересекающихся подмножеств R_i.
Тогда:

$$IG(R, f) = H(R) - \sum_{R_i \subset R} \frac{|R_i|}{|R|} H(R_i)$$

Другими словами, Information Gain показывает насколько выгодно разбить множество R с использованием признака f.

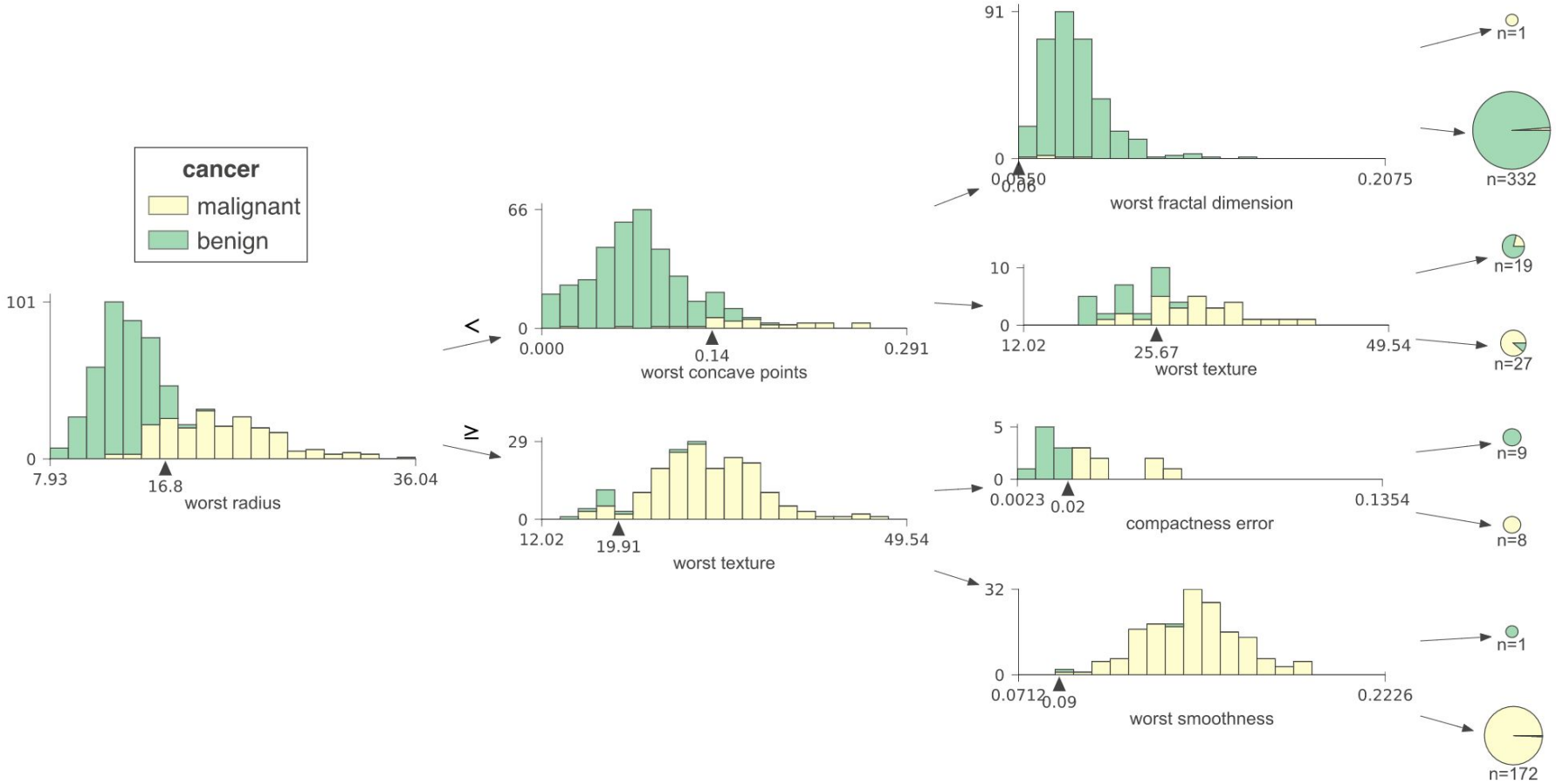
Алгоритм C4.5

Алгоритм представляет собой череду вызовов процедуры роста дерева для листьев.

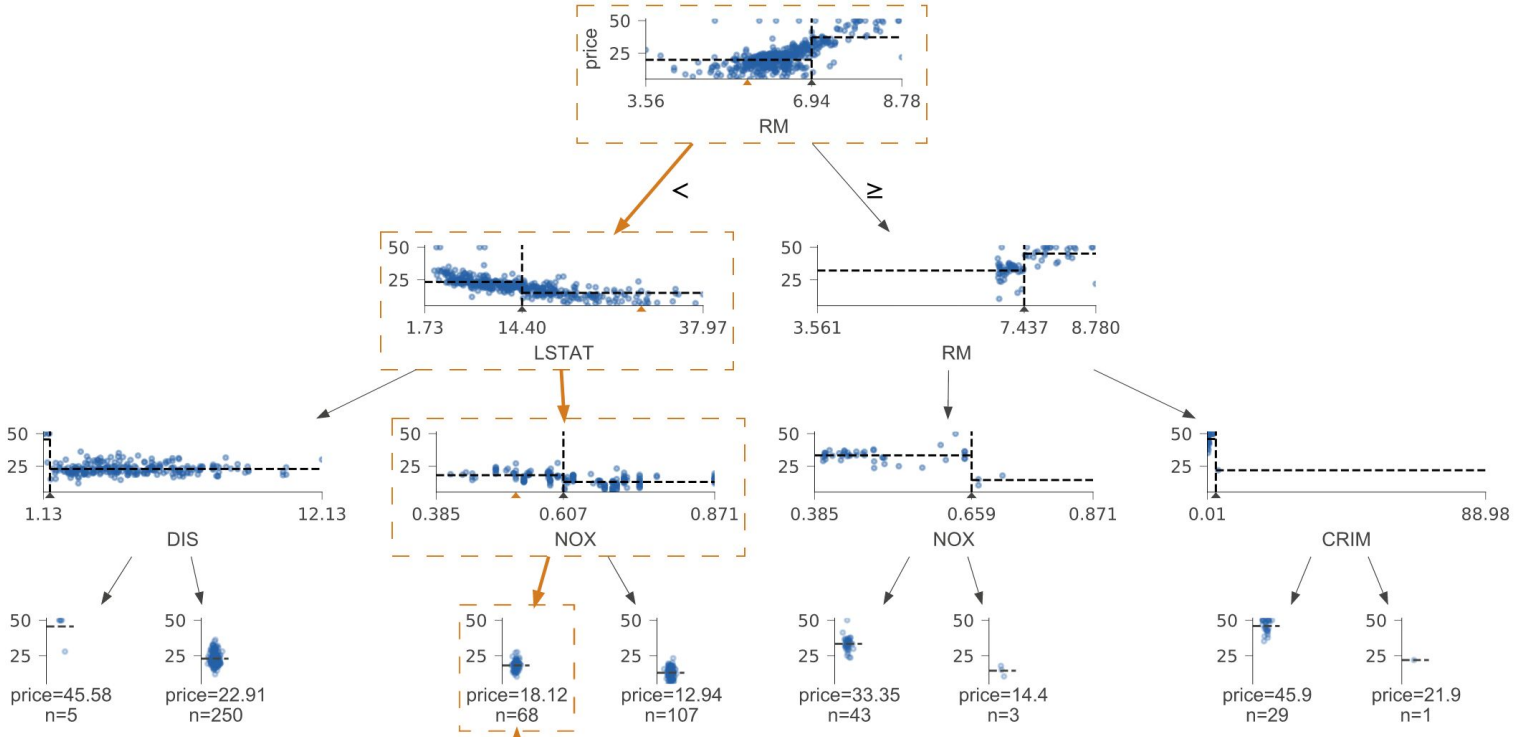
Процедура роста дерева выглядит так:

1. Найти признак и его критическое значение, которые обеспечат наибольший IG при выполнении разделения.
2. Разделить множество объектов по этому признаку (выполнить сплит).
3. Повторить для каждого подмножества, если его энтропия не равна нулю.

Для случая регрессии используется стандартное отклонение и выбирается признак, который обеспечит его минимум.



https://explained.ai/decision-tree-viz/



CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT
0.21	12.50	7.87	0.00	0.52	5.63	100.00	6.08	5.00	311.00	15.20	386.63	29.93

https://explained.ai/decision-tree-viz/

Процедура выбора признака для сплита

При выборе признака и его критического значения выполняют полный перебор значений всех возможных признаков. Это затратно, поэтому применяют ряд эвристик:

- Удаление признаков, принимающих константные значения
- Построение гистограммы значений признака таким образом, чтобы обеспечить равномерное количество объектов в каждой корзине
- Преобразование значений признака (например, взятие логарифма)
- Прекращение построения дерева если в текущей вершине находится недостаточно объектов

Ансамбли

Бутстреп

Бутстреп — это метод непараметрической статистики, моделирующий наличие большего объема данных, чем имеется.

Для этого из имеющегося набора данных генерируются новые:

1. Инициализировать новый набор данных.
2. Пока размер нового набора меньше старого:
    a. Выбрать случайный элемент из старого набора
    b. Положить его копию в новый набор

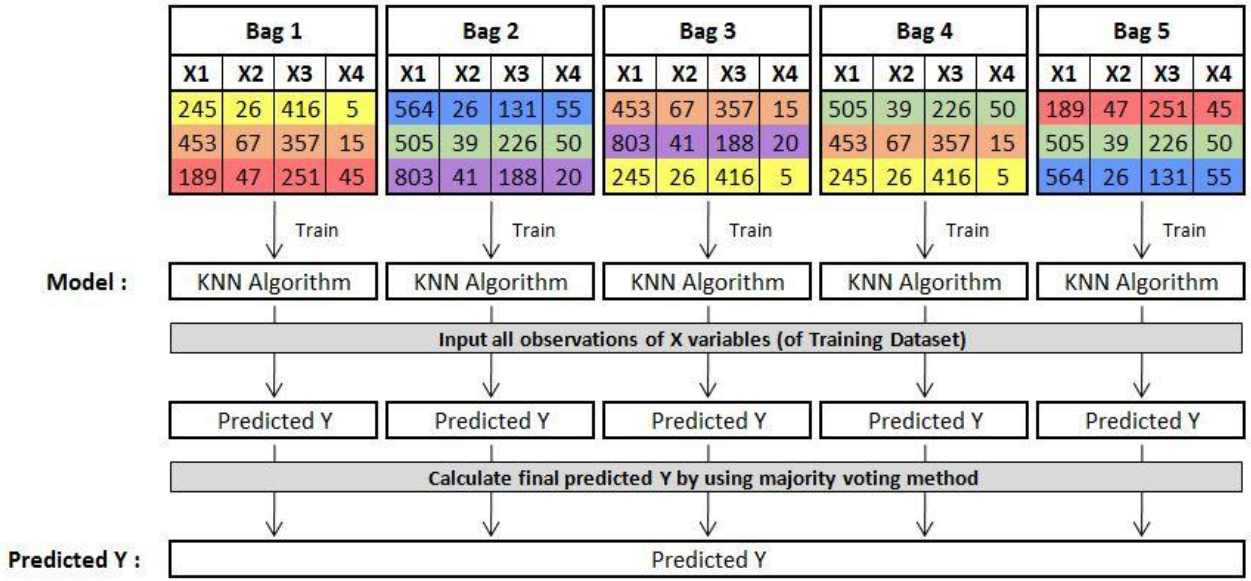
Таким образом может получится так, что в новом наборе данных будут копии объектов. Их рассматривают как разные объекты с одинаковым набором признаков.

Бэггинг

Бэггинг (bagging, bootstrap aggregating) — метод построения композиции алгоритмов, в котором каждый алгоритм строится независимо от других на подвыборках обучающей выборки.

Bag 1				Bag 2				Bag 3				Bag 4				Bag 5			
X1	X2	X3	X4	X1	X2	X3	X4	X1	X2	X3	X4	X1	X2	X3	X4	X1	X2	X3	X4
245	26	416	5	564	26	131	55	453	67	357	15	505	39	226	50	189	47	251	45
453	67	357	15	505	39	226	50	803	41	188	20	453	67	357	15	505	39	226	50
189	47	251	45	803	41	188	20	245	26	416	5	245	26	416	5	564	26	131	55

Бэггинг



Итоговый алгоритм принимает решения посредством голосования среди всех алгоритмов — например, возвращается самый частый ответ для классификации и среднее для задачи регрессии.

Бэггинг

Получаемый при агрегации результат — модель, которая сохраняет bias, но может снизить variance. На практике это позволяет улучшить качество модели, сделав ее более вычислительно сложной.

Важный момент: отдельные модели при голосовании должны быть попарно некоррелированными (т.е. делать различные ошибки).

Бэггинг

Композиция решающих деревьев на практике показывает себя лучше чем одно дерево.

Структура дерева сильно зависит от обучающей выборки, а значит что, если немного изменить обучающую выборку, то дерево сильно изменится.

Бэггинг идеально подходит для агрегации решающих деревьев, поскольку композиция алгоритмов при помощи голосования работает наилучшим образом, когда модели различны.

Random Subspaces

Помимо бэггинга, для построения набора различных моделей используется метода выбора случайных подвыборок признаков (Random Subspaces).

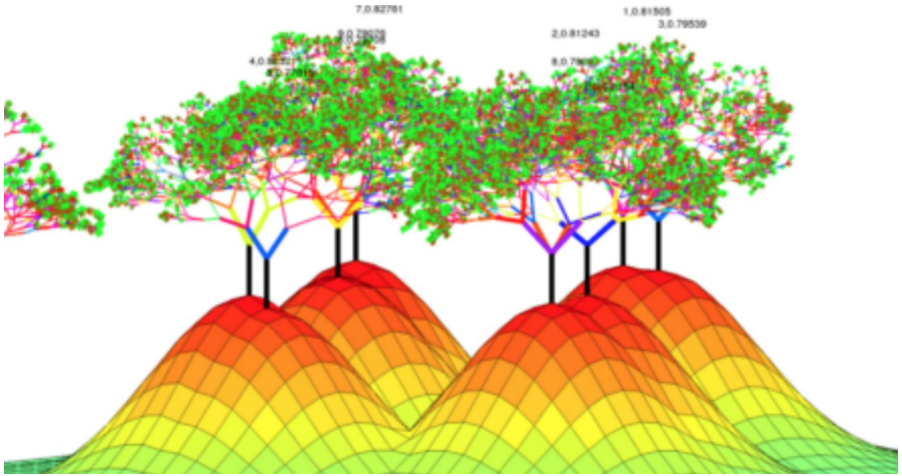
Метод обеспечивает устойчивость алгоритма к набору доступных признаков.

Bag 1			Bag 2			Bag 3			Bag 4		
X1	X2	X4	X2	X3	X4	X1	X2	X3	X1	X3	X4
245	26	5	26	416	5	245	26	416	245	416	5
505	39	50	39	226	50	505	39	226	505	226	50
453	67	15	67	357	15	453	67	357	453	357	15
189	47	45	47	251	45	189	47	251	189	251	45
564	26	55	26	131	55	564	26	131	564	131	55
803	41	20	41	188	20	803	41	188	803	188	20

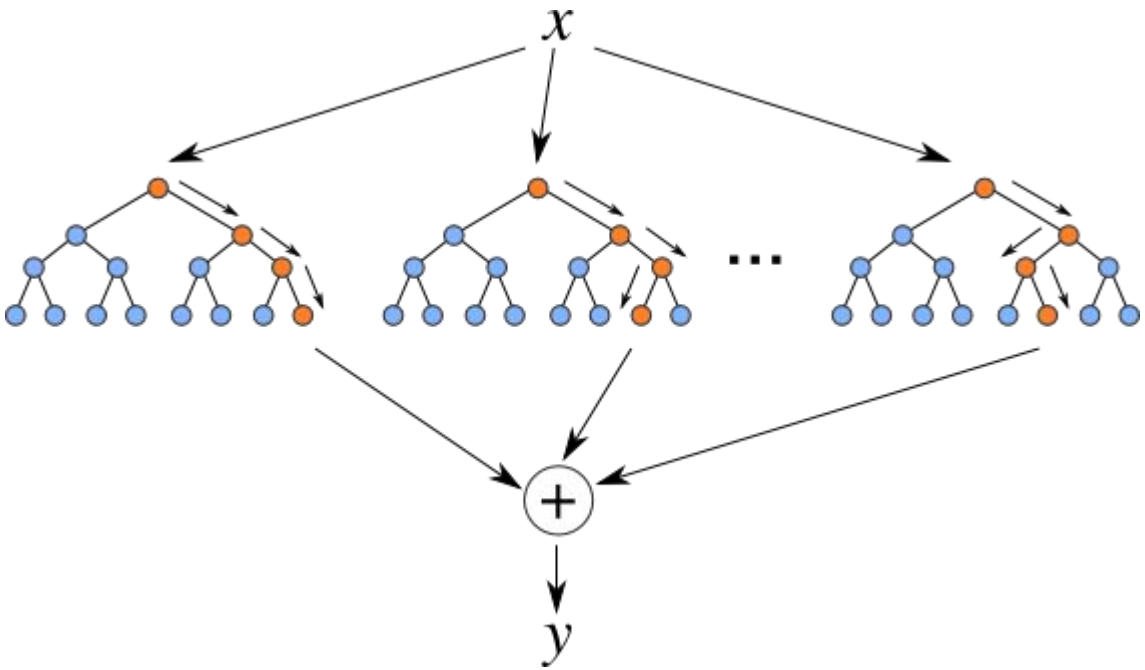
Случайный лес

Случайный лес (Random Forest) — это набор решающих деревьев, каждое из которых обучается на подмножестве признаков (Random Subspaces) и на подмножестве объектов (bagging).

Получается, что случайный лес случаен и по признакам, и по объектам.



Случайный лес



Ответы каждого дерева агрегируются при помощи голосования.

Оценка качества случайного леса

Для ансамбля, построенного бэггингом, есть возможность оценить его качество без отложенной выборки и кросс-валидации.

Out-of-bag score (OOB) — это метод оценивания качества ансамбля (строящегося бэггингом) во время его обучения. Для этого каждый обученный на подвыборке алгоритм оценивается на примерах, которые в подвыборку не попали.

Доверять такой оценке можно лишь при достаточном числе алгоритмов и при достаточном размере выборки.

Бустинг

Бустинг (boosting) — метод построения композиции алгоритмов, в котором алгоритмы строятся последовательно, и где каждый алгоритм старается исправить ошибки предыдущего.

Объединяя множество слабых алгоритмов, можно создать сильный алгоритм, относительно точно обобщающий данные.

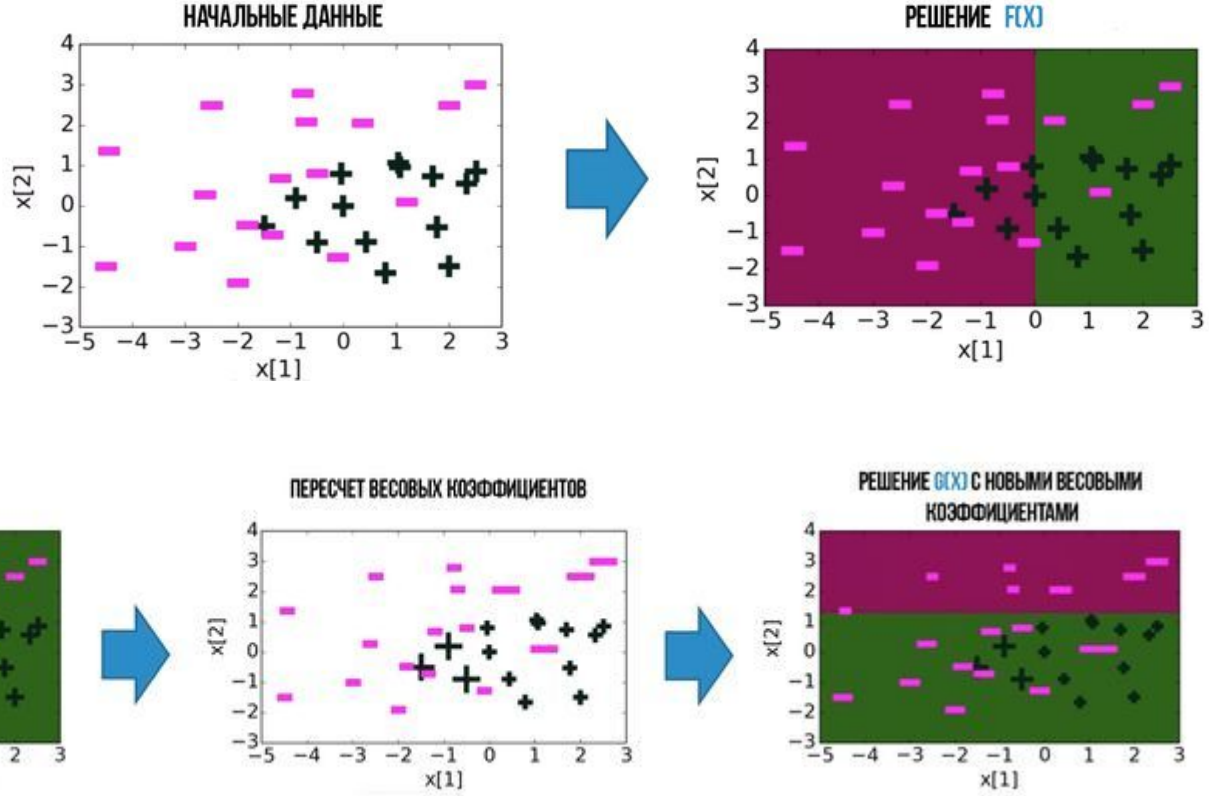
AdaBoost

AdaBoost циклически вызывает слабые классификаторы. После каждого вызова обновляется распределение весов Dt, которые отвечают важности каждого из объектов обучающего множества для классификации.

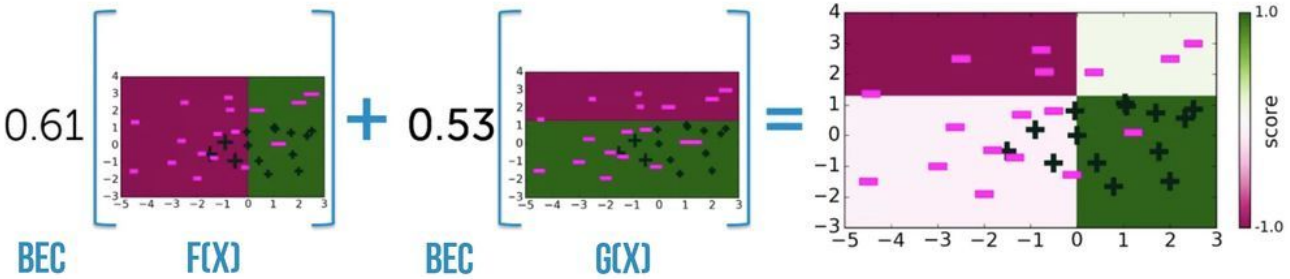
На каждой итерации веса каждого неверно классифицированного объекта возрастают, таким образом новый классификатор «фокусирует своё внимание» на этих объектах.

Рассмотрим набор данных, которые пометим как − и +.

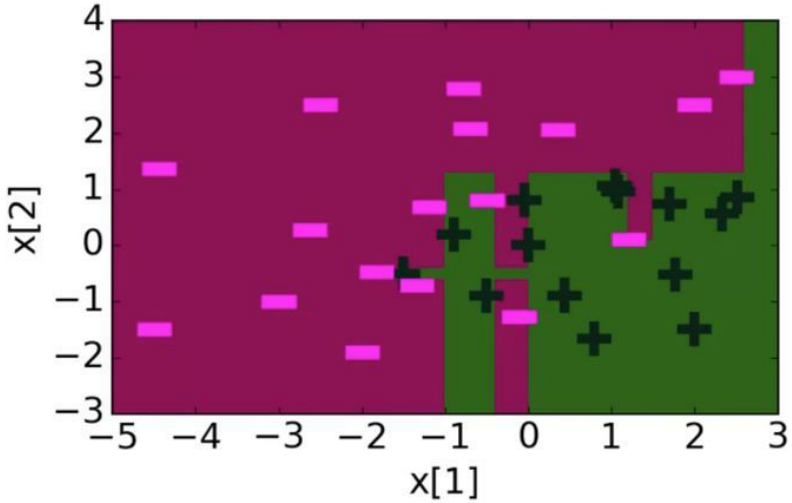
Из результата после первой итерации для всех ошибочно классифицированных объектов увеличим веса, а для верно классифицированных уменьшим



После второй итерации все, что находится в "цветной" зоне, можно однозначно классифицировать, но появляются ошибки и "белые" зоны, которые нельзя однозначно классифицировать



После 30-ти итерации все объекты классифицируются верно и число ошибок на выборке равно нулю



AdaBoost

Достоинства:

1. Простота реализации;
2. Хорошая обобщающая способность. В реальных задачах удаётся строить композиции, превосходящие по качеству базовые алгоритмы. Обобщающая способность может улучшаться по мере увеличения числа базовых алгоритмов;
3. Время построения композиции практически полностью определяется временем обучения базовых алгоритмов;
4. Возможность идентифицировать выбросы. Это наиболее «трудные» объекты x_i, для которых в процессе наращивания композиции веса w_i принимают наибольшие значения.

Недостатки:

1. Склонен к переобучению при наличии значительного уровня шума в данных;
2. Требует достаточно длинных обучающих выборок. Другие методы линейной коррекции, в частности, бэггинг, способны строить алгоритмы сопоставимого качества по меньшим выборкам данных.