Классификация и алгоритмы. Деревья решений

Мультиклассовая классификация

- Каждый объект выборки принадлежит одному из N классов
- Задача – сконструировать такую функцию, которая будет предсказывать класс
- Но мы умеем только в бинарную классификацию, как быть?

Мультиклассовая классификация

- Если бы мы могли узнать плотность распределения каждого класса $p_i(x)$, то мы могли бы предсказывать по формуле:

$$f(x) = \arg \max_{i \in 1,\dots,N} p_i(x)$$

- Мы можем вычислить плотности, но это сложно, + проблемы из-за многомерного пространства и ограниченности данных.
- Значит, как обычно, надо придумывать обходные пути

Мультиклассовая классификация

- Всегда можно разложить задачу на несколько не связанных между собой задач бинарной классификации!
- Два варианта:
    1. One-VS-All Classification
    2. All-VS-All Classification

Мультиклассовая классификация

One-VS-All

- Построим N бинарных классификаторов, у которых положительный класс – это объекты класса i, а отрицательный – все остальные объекты.
- $f_i(x)$ – наш i-тый классификатор
- Тогда итоговый классификатор:

$$f(x) = \arg\max_i f_i(x)$$

All-VS-All

- Построим N*(N – 1) классификаторов для каждой пары классов i, j.
- $f_{ij}(x)$ – классификатор, где i – положительный класс, а j – отрицательный.
- Заметим, что $f_{ij}(x) = -f_{ji}(x)$.
- Тогда:

$$f(x) = \arg\max_i(\sum_j f_{ij}(x))$$

Мультиклассовая классификация

- Используются оба варианта решения задачи (они различаются вычислительной сложностью и удобством для разных алгоритмов)
- В sklearn все это уже реализовано и нет нужды об этом даже думать – библиотека сама определяет по нашим данным, какая нам нужна классификация.

Нелинейные алгоритмы классификации

Наивный Байесовский классификатор

- Основан на теореме Байеса с допущением о независимости признаков (потому и наивный)
- Пример: фрукт может считаться яблоком, если он:
    1. красный
    2. круглый
    3. его диаметр составляет порядка 8 см
- Предполагаем, что признаки вносят независимый вклад в вероятность того, что фрукт является яблоком.

Наивный Байесовский классификатор

$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$

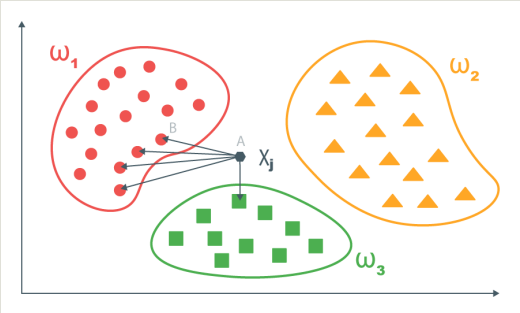
- P(c|x) – вероятность того, что объект со значением признака x принадлежит классу c
- P(c) – априорная вероятность класса c
- P(x|c) – вероятность того, что значение признака равно x, при условии, что объект принадлежит классу c
- P(x) – априорная вероятность значения признака x

Метод ближайших соседей

Идея: схожие объекты находятся близко друг к другу в пространстве признаков.

Как классифицировать новый объект?

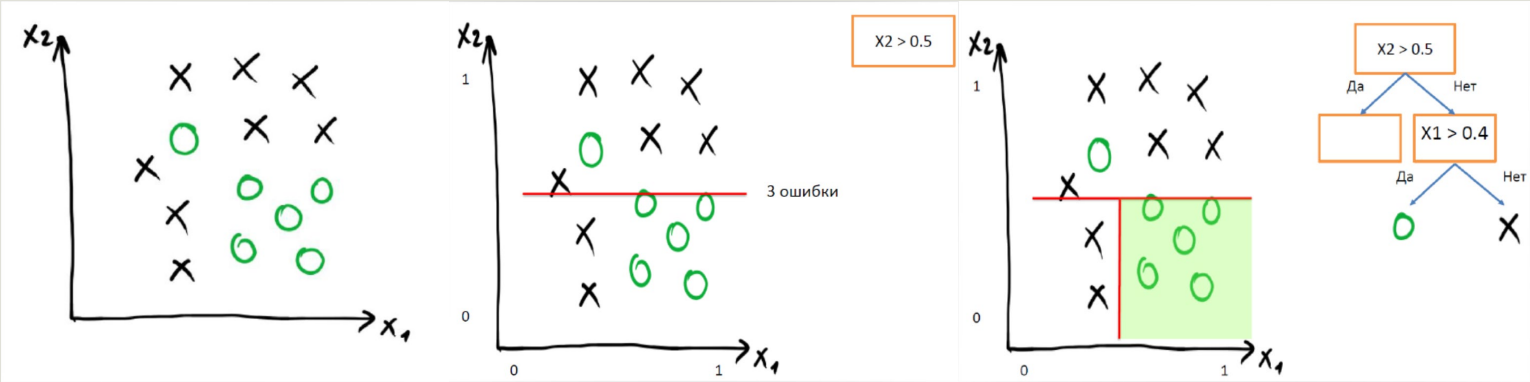
- Вычислить расстояние до каждого из объектов обучающей выборки
- Выбрать k объектов обучающей выборки, расстояние до которых минимально
- Соседи решают класс голосованием! Соседей какого класса больше, тот и наш.



Деревья решений

Деревья решений

Пробуем делить выборку таким образом, чтобы было как можно меньше ошибок. Выбираем всегда только один признак! Делим, пока не получим идеальный результат (или нет...)



Деревья решений

Решающее дерево – это бинарное дерево, в котором:

1) Каждой вершине v приписана функция (предикат) $\beta_v: X \to \{0, 1\}$

2) Каждой листовой вершине v приписан прогноз $c_v \in Y$ (для классификации – класс или вероятность класса, для регрессии – действительное значение целевой переменной

Деревья решений

Жадный алгоритм построения:

1. Найдем наилучшее разбиение выборки X на две части: $R_1(j,t) = \{x | x_j < t\}$ и $R_2(j,t) = \{x | x_j \geq t\}$ с точки зрения некоторого функционала $Q(X,j,t)$:
   - найдем лучшие j, t
   - создадим корень дерева, поставив в него предикат $[x_j < t]$.
2. Для каждой из полученных выборок R_1 и R_2 рекурсивно применим шаг 1.

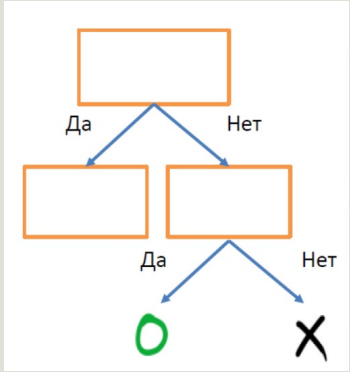
В каждой вершине на каждом шаге проверяем, не выполнилось ли условие останова.

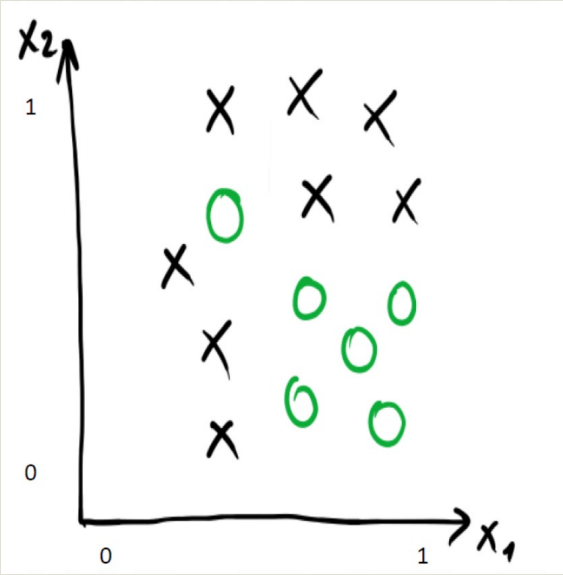
Если выполнилось, то объявляем вершину листом

и записываем в него предсказание.

Деревья решений

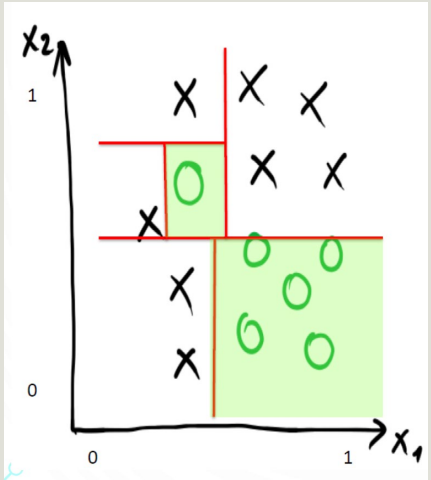
Строим дерево:





Деревья решений

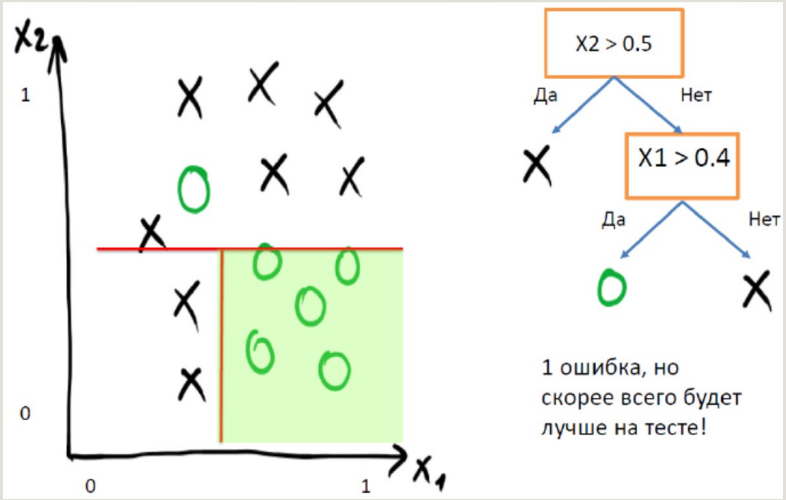
Построили все дерево:



Но хорошее ли это дерево?

Деревья решений

- На любой выборке можно построить такое дерево, которое на этой выборке будет делать 0 ошибок
- Но это будет означать, что дерево **подогналось под данные**
- Как с этим бороться? Стричь деревья!
- Стрижка деревьев (pruning) – когда отрезаем слишком низкие узлы



Что влияет на построение РД

- вид предикатов в вершинах
- функционал качества $Q(X, j, t)$
- критерий останова
- метод обработки пропущенных значений
- метод стрижки

Критерии останова

- Ограничение максимальной глубины дерева (max_depth)
- Ограничение минимального числа объектов в листьях (min_samples_leaf)
- Ограничение максимального числа листьев в дереве
- Останов в случае, если все объекты в листе из одного класса
- Требование, что функционал качества при дроблении увеличивался как минимум на $s\%$.

Стрижка дерева (Pruning)

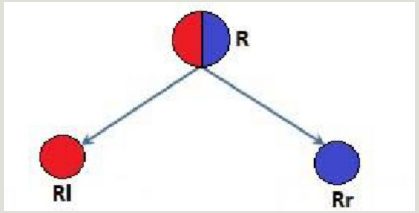
1. Строится переобученное дерево на максималках
2. Дерево оптимизируется с целью уменьшения переобучения.

Это альтернатива критериям останова

Критерии информативности

В каждой вершине оптимизируем функционал $Q(R, j, t)$. Что это такое?

- На каждом шаге в вершину попадает множество объектов R, причем в левую половину попадает R_l объектов, а в правую - R_r.
- Цель: хотим, чтобы в левой и правой половине были по возможности однородные по классу объекты.
- Функция H(R) - критерий информативности - оценивает меру неоднородности целевых переменных внутри R. Чем меньше неоднородность, тем меньше функция.
- То есть, хотим $H(R_l) \to min\, H(R_r) \to min$



Критерии информативности

- Определим функционал $Q(R, j, t)$ по формуле:

$$Q(R, j, t) = H(R) - \frac{|R_l|}{|R|} H(R_l) - \frac{|R_r|}{|R|} H(R_r) \to \max_{j,t}$$

(где R – множество объектов при заданном предикате, j – номер признака, t – порог (напомню, в каждом узле мы сравниванием значение признака j с пороговым значением t)).

- Что за критерий информативности и как его определить? Можно предложить оценивать однородность множества объектов R тем, насколько хорошо их целевые переменные предсказываются константой (при оптимальном выборе этой константы):

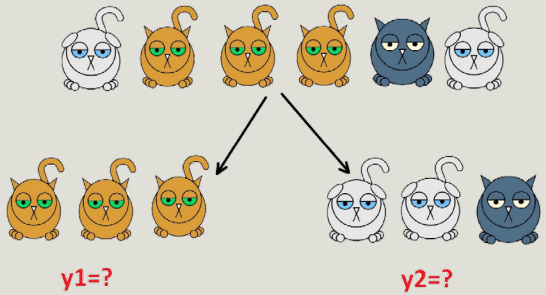
$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x_i, y_i \in R)} L(y_i, c)$$

где $L(y, c)$ – некоторая функция потерь (разная для разных задач)

Деревья решений

Классификация

Выбираем тот класс, представителей которого в листе больше:



Регрессия

Считаем среднее значение по листу:

/ 3

Деревья решений: классификация

- Для классификации предлагается использовать логарифм правдоподобия:

$$H(R) = \min_{\sum_k c_k = 1} \left(-\frac{1}{|R|} \sum_{(x_i, y_i \in R)} \sum_{k=1}^{K} [y_i = k] \log c_k\right)$$

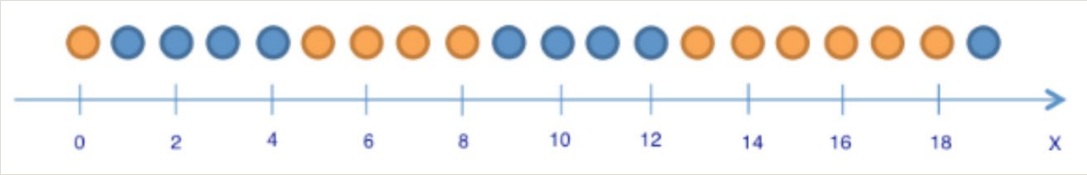
- Преобразованиями мучиться не будем, но в итоге можно получить это:

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

- А это у нас энтропия распределения классов. Минимальное значение энтропии – для вырожденных случаев, а значит, они нам выгоднее.

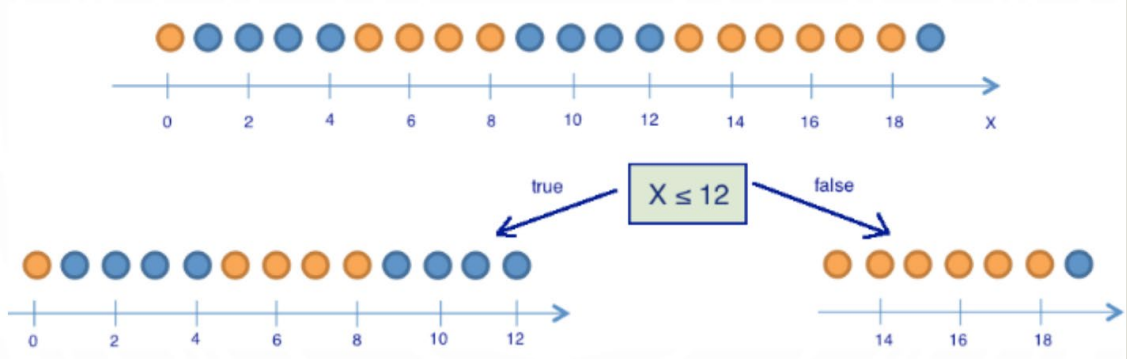
почитать про эти ужастики подробнее

Пример использования энтропийного критерия



$p_1 = \frac{9}{20}, p_2 = \frac{11}{20}$, энтропия $H_0 = -\frac{9}{20}\log\frac{9}{20} - \frac{11}{20}\log 11/20 \approx 1$

Пример использования энтропийного критерия



В левой части $H_l = -\frac{5}{13}\log\frac{5}{13} - \frac{8}{13}\log\frac{8}{13} \approx 0.96$

В правой части $H_r = -\frac{1}{7}\log\frac{1}{7} - \frac{6}{7}\log\frac{6}{7} \approx 0.6$

То есть $Q = H_0 - \frac{|R_l|}{|R|}H_l - \frac{|R_r|}{|R|}H_r = 1 - \frac{13}{20}\cdot 0.96 - \frac{7}{20}\cdot 0.6 \approx 0.16$

Деревья решений: регрессия

- В качестве функции потерь возьмем MSE и подставим в нашу формулу:

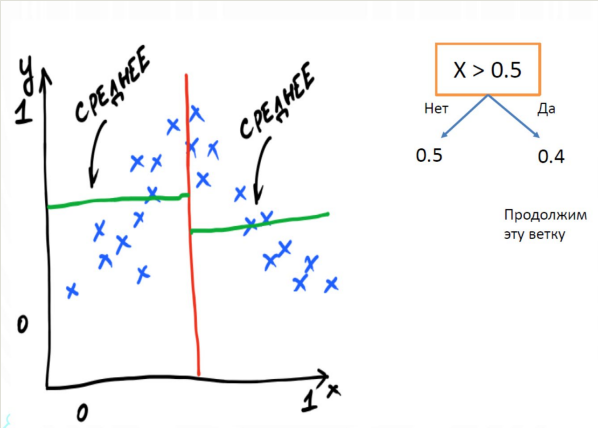
$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x_i, y_i \in R)} (y_i - \mathrm{c})^2$$

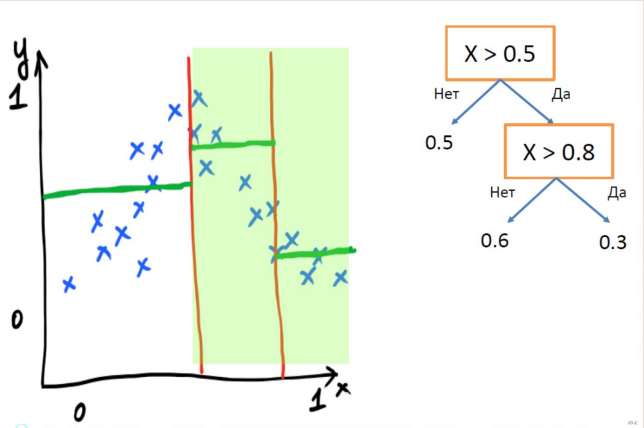
- Минимум в этом выражении будет достигаться при среднем значении целевой переменной c (мы его и хотим). Критерий можно переписать:

$$H(R) = \frac{1}{|R|} \sum_{(x_i, y_i \in R)} \left(y_i - \frac{1}{|R|} \sum_{(x_j, y_j \in R)} y_j \right)^2$$

- Это формула дисперсии: чем ниже разброс целевой переменной, тем лучше вершина.

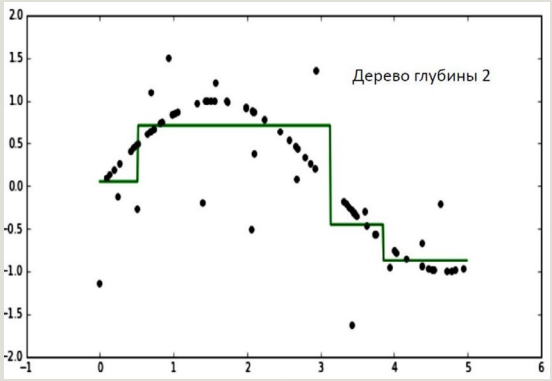
Пример дерева решений с регрессией



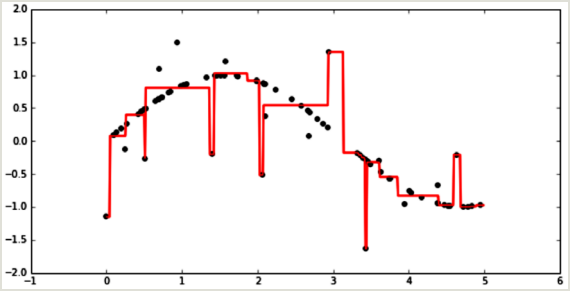


Пример дерева решений с регрессией

Дерево глубины 2



Дерево глубины 5



Деревья решений

Плюсы

- Четкие правила классификации (интерпретируемые предикаты, например, “возраст > 25”)
- Деревья решений легко визуализируются, то есть хорошо интерпретируются
- Быстро обучаются и выдают прогноз
- Малое число параметров

Минусы

- Очень чувствительны к шумам в данных, модель сильно меняется при небольшом изменении обучающей выборки
- Разделяющая граница имеет свои ограничения (состоит из гиперплоскостей)
- Необходимость борьбы с переобучением (стрижка или какой-либо из критериев останова)
- Проблема поиска оптимального дерева (NP-полная задача, поэтому на практике используется жадное построение дерева)

GridSearchCV

- *Параметры модели* – величины, настраивающиеся по обучающей выборке (например, веса в линейной регрессии)
- *Гиперпараметры модели* – величины, контролирующие процесс обучения. Они не могут быть настроены в процессе обучения.

- Как подбирать гиперпараметры?
- По кросс-валидации: главное – не использовать тестовую выборку!
- Для этого есть функция GridSearchCV.