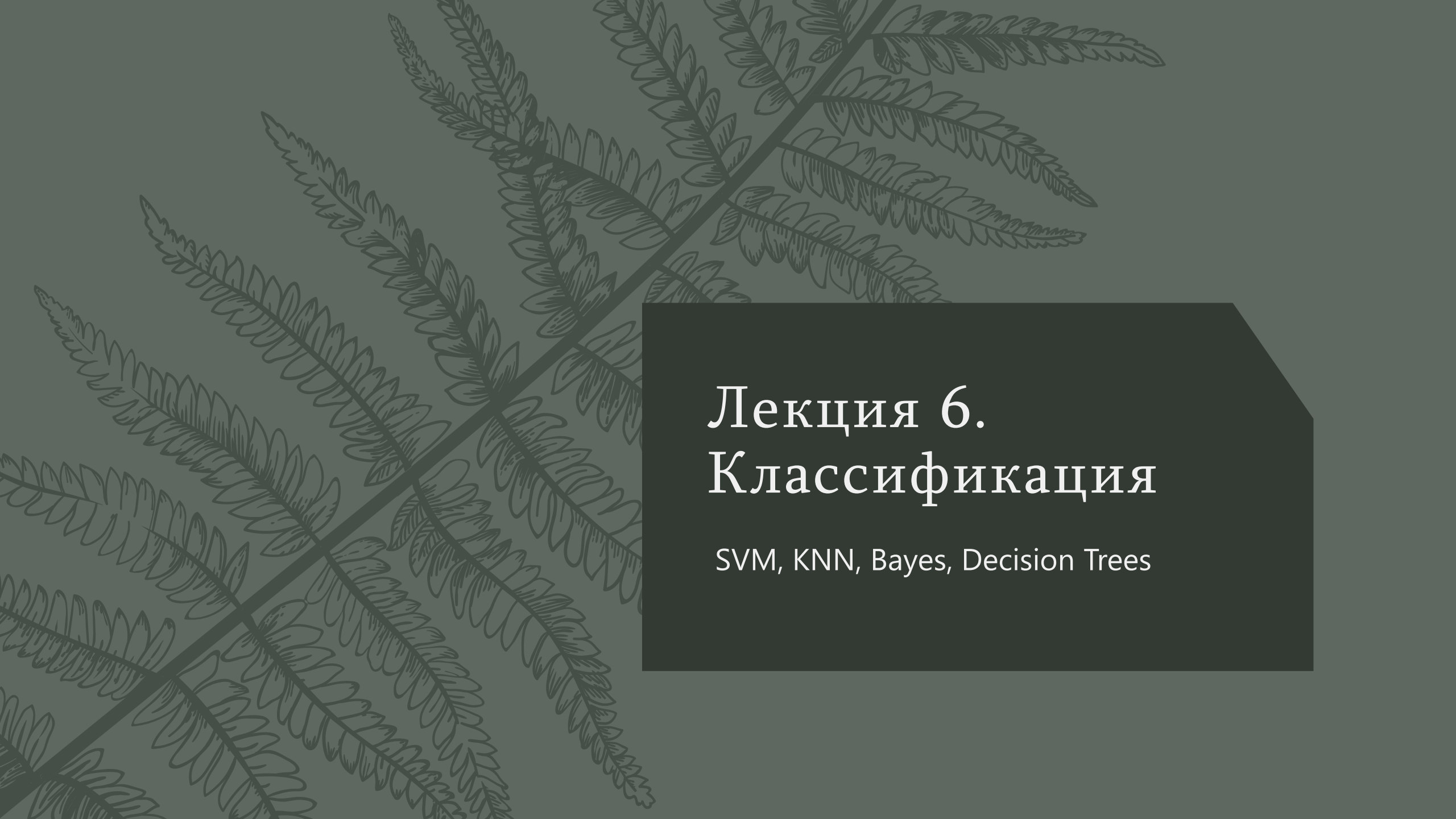


Лекция 6. Классификация

SVM, KNN, Bayes, Decision Trees

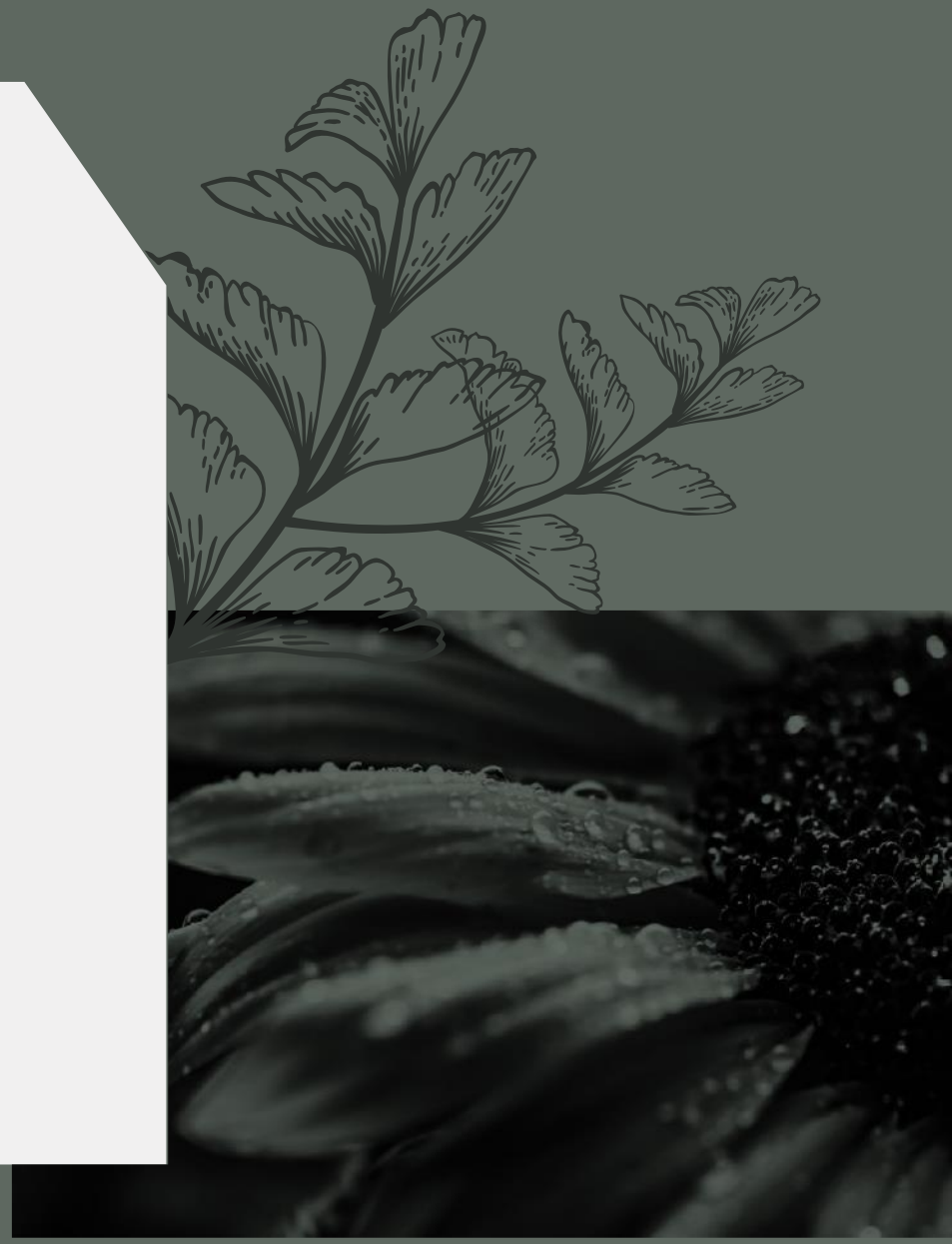
Линейные vs. нелинейные классификаторы

Линейные

- Логистическая регрессия
- Метод опорных векторов (SVM – support vector machine)

Нелинейные

- Наивный Байес
- Метод K ближайших соседей (KNN)
- Деревья решений



Линейные классификаторы

Логистическая регрессия

Мы с вами уже разбирали ее: в ее основе линейная регрессия + функция сигмоида.

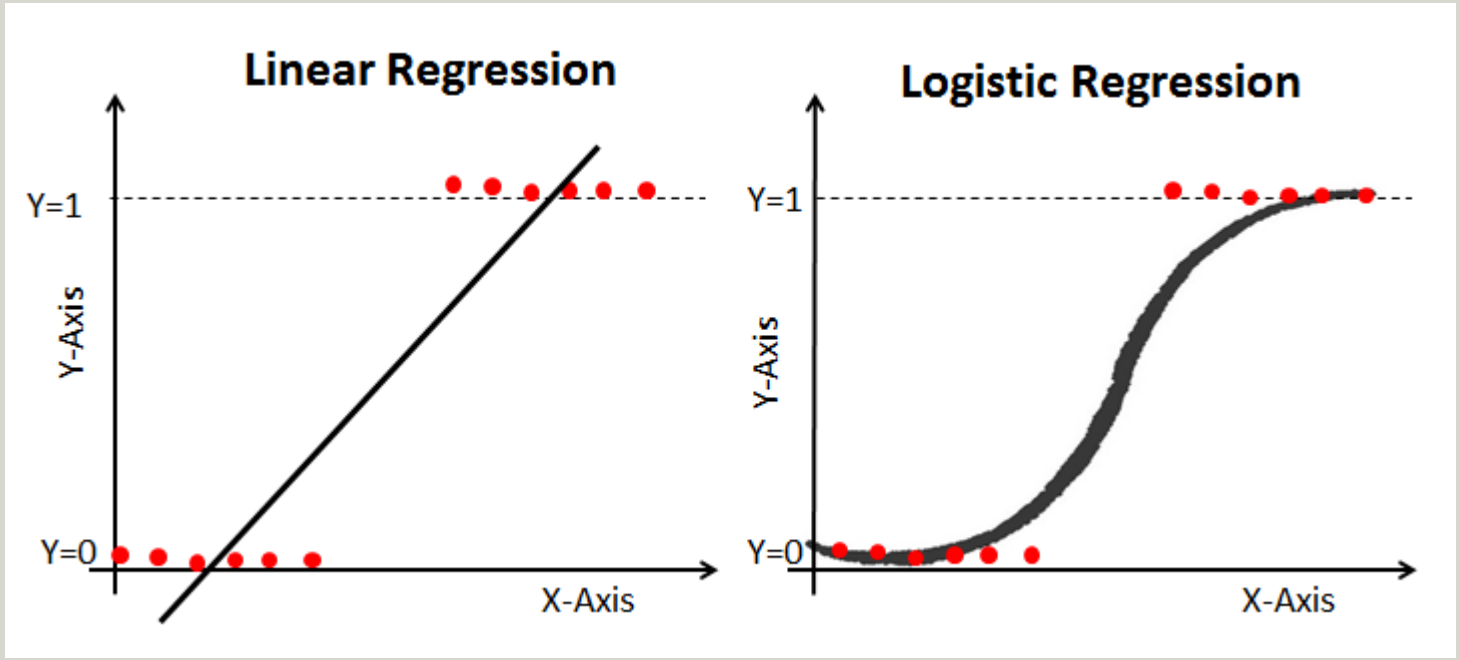
Метод опорных векторов

Строим разделяющую плоскость так, чтобы она была максимально далеко от точек разных классов

Логистическая регрессия

a(x, w) = g((x, w)), где g(z) – ***сигмоида***

$$g(z) = \frac{1}{1 + e^{-z}}$$



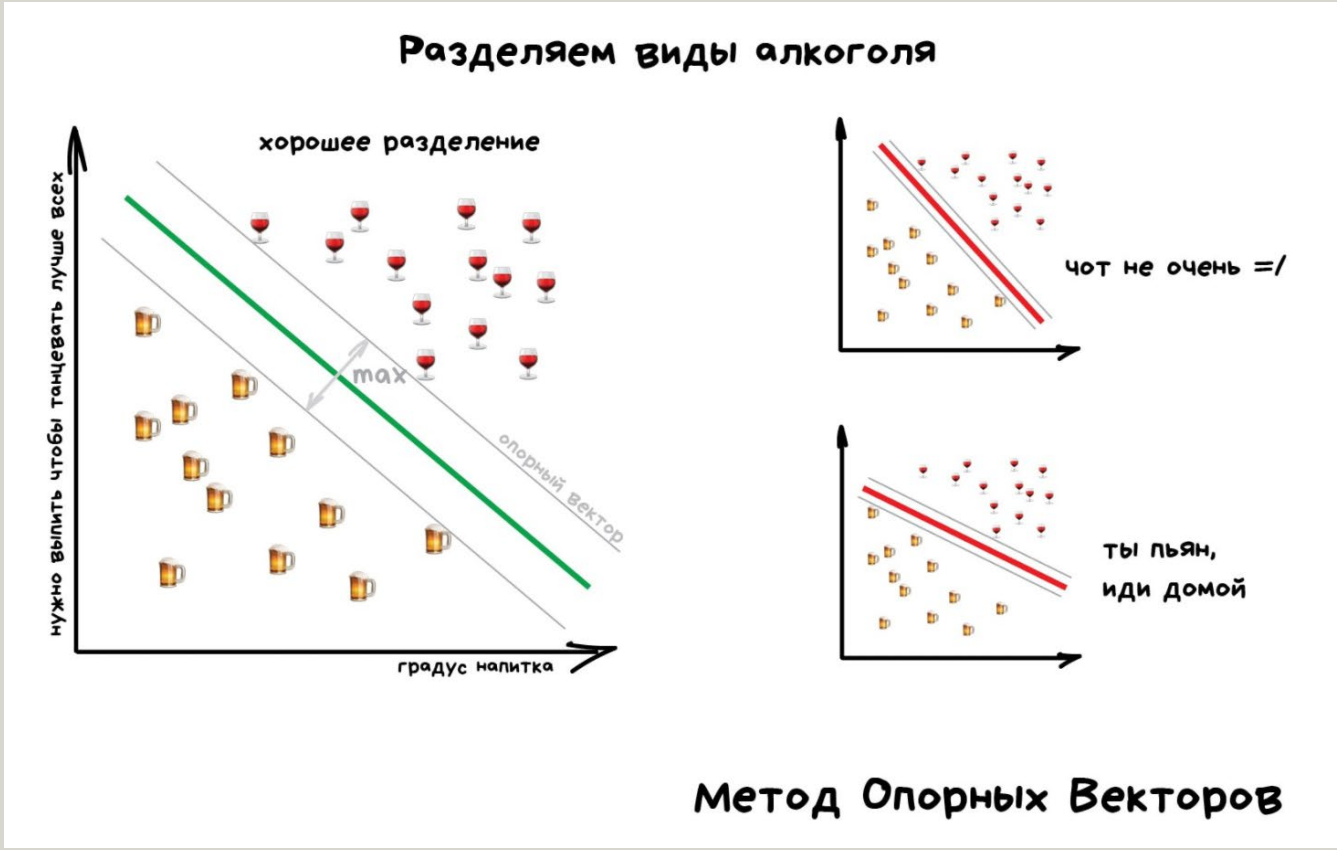
Логистическая регрессия

- Внутри – обычная множественная регрессия (уравнение прямой с кучей переменных)
- Чтобы предсказывала вероятность, загоняем при помощи сигмоиды ответ уравнения в интервал [0, 1]
- Вышенаписанное – для бинарной классификации (относится к классу 1 или к классу -1)
- Метрика качества – ROC-кривая (ROC-AUC) (Хорошая статья на тему)

Метод опорных векторов

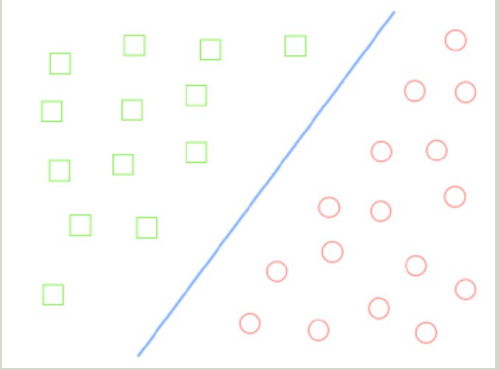
Проводим разделяющую линию таким образом, чтобы образовался максимальный зазор между этой линией ближайшими к ней объектами выборки – очевидно, нужно считать расстояние от точки до плоскости.

Может быть довольно медленным...



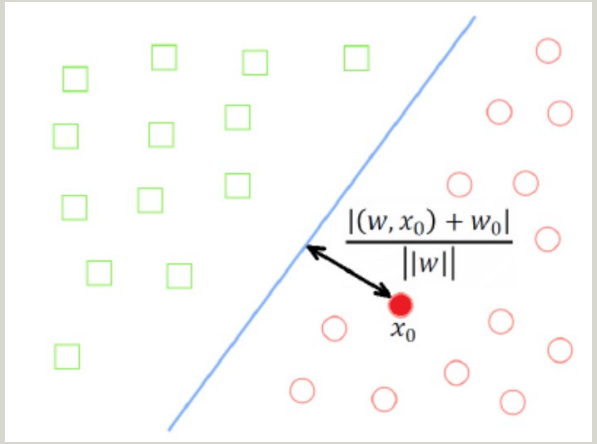
Линейно разделимая выборка

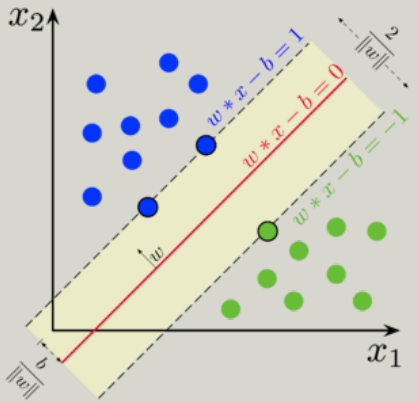
Выборка линейно разделима, если существует такой вектор параметров *w**, что соответствующий классификатор a(x) не допускает ошибок на этой выборке.



Метод опорных векторов: разделимый случай

- $a(x) = sign\big((w, x) + w_0\big)$
  Нормируем параметры w *и* w_0 так, что
  $$\min_{x \in X} |(w, x) + w_0| = 1\,.$$
- Расстояние от точки x_0 до разделяющей гиперплоскости, задаваемой классификатором:
  $\rho(x_0, a) = \frac{|(w, x_0) + w_0|}{\|w\|}$ *(*$\|w\|$ *- сумма квадратов коэффициентов плоскости)*
- Расстояние до ближайшего объекта $x \in X$:
  $$\min_{x \in X} |(w, x) + w_0| = \frac{1}{\|w\|} \min_{x \in X} |(w, x) + w_0| = \frac{1}{\|w\|}$$





Оптимизационная задача SVM для разделимой выборки

$$\begin{cases} \frac{1}{2}\|w\|^2 \to \min\limits_{w} \\ y_i\big((w, x_i) + w_0\big) \geq 1, i = 1, \dots, l \end{cases}$$

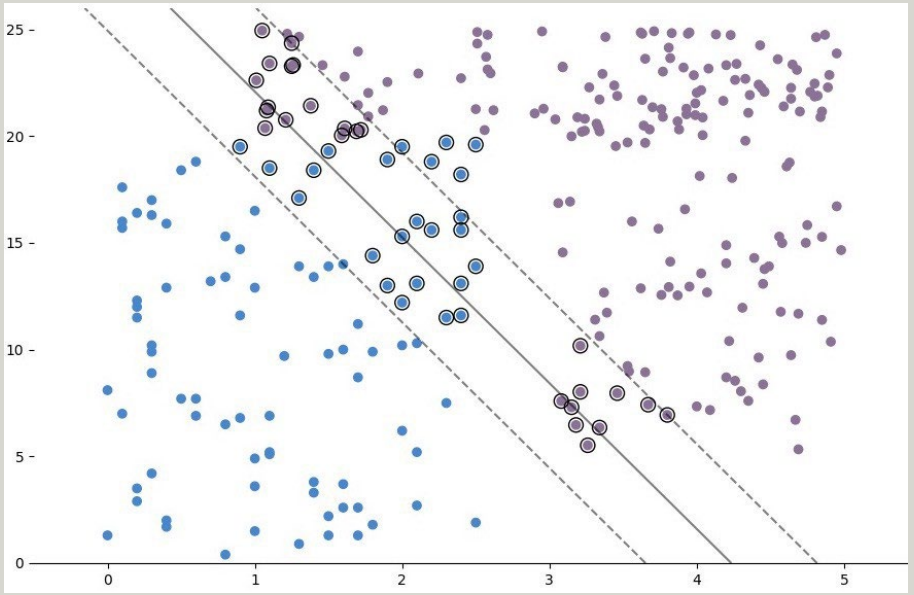
Данная оптимизационная задача имеет единственное решение (доказывать не будем…)

$\frac{1}{2}\|w\|^2 \to min$ – *максимизируем ширину разделяющей полосы (квадрат – чтобы удобнее было считать производную);*

$y_i\big((w, x_i) + w_0\big)$ - *отступ (margin), который >0, если правильная классификация; минимальное расстояние – 1, значит, все расстояния от точек до плоскости д.б. 1 и больше.*

Линейно неразделимая выборка

Существует хотя бы один объект $x \in X$ такой, что
$y_i\big((w, x_i) + w_0\big) < 1.$



Линейно неразделимая выборка

Существует хотя бы один объект $x \in X$ такой, что

$$y_i\big((w, x_i) + w_0\big) < 1.$$

Смягчим ограничения, введя штрафы $\xi_i \geq 0$:

$$y_i\big((w, x_i) + w_0\big) \geq 1 - \xi_i, i = 1, \dots, l$$

Хотим:

1. Минимизировать штрафы $\sum_{i=1}^{l} \xi_i$
2. Максимизировать отступ $\frac{1}{\|w\|}$

Задача оптимизации

$$\begin{cases} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \to \min\limits_{w,w_0,\xi_i} \\ y_i\big((w,x_i)+w_0\big) \geq 1-\xi_i, i=1,\dots,l \\ \xi_i \geq 0, i=1,\dots,l \end{cases}$$

Является выпуклой и тоже имеет единственное решение. Доказывать опять не будем. Более простой (ну кому как, конечно) вид этой задачи:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} max\Big(0, \quad 1-y_i\big((w,x_i)+w_0\big)\Big) \to \min_{w,w_0}$$

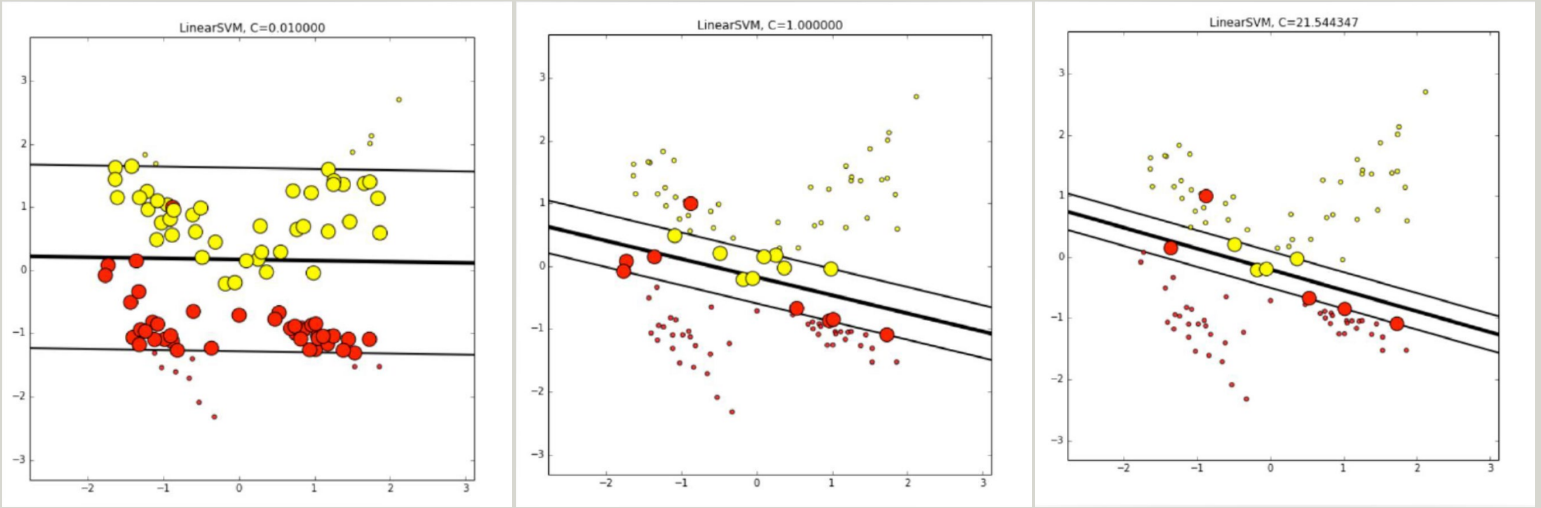
Значение константы C

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} max\left(0, \quad 1 - y_i\left((w, x_i) + w_0\right)\right) \to \min_{w,w_0}$$

Что за C?

Это положительная константа, которая является управляющим параметром метода и позволяет находить компромисс между максимизацией разделяющей полосы и минимизацией суммарной ошибки.

Значение константы C



Нелинейные классификаторы

Наивный Байес

Считаем, что признаки линейно независимые.

Метод К ближайших соседей

Ориентируемся на ближайших соседей объекта

Деревья решений

Строим деревья, которые ставят вопросеки

Наивный Байес

У нас есть некий набор признаков, и мы считаем, что они вносят независимый вклад в определение класса объекта.

Теорема Байеса

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

- 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

Наивный Байес

+ Самый простой и быстрый алгоритм эва
+ В случае, если признаки действительно не зависят друг от друга, классификатор показывает высокое качество
- Если в тестовых данных есть категория, которая не встречалась в обучающей выборке, классификатор присвоит ей нулевую вероятность

Метод ближайших соседей

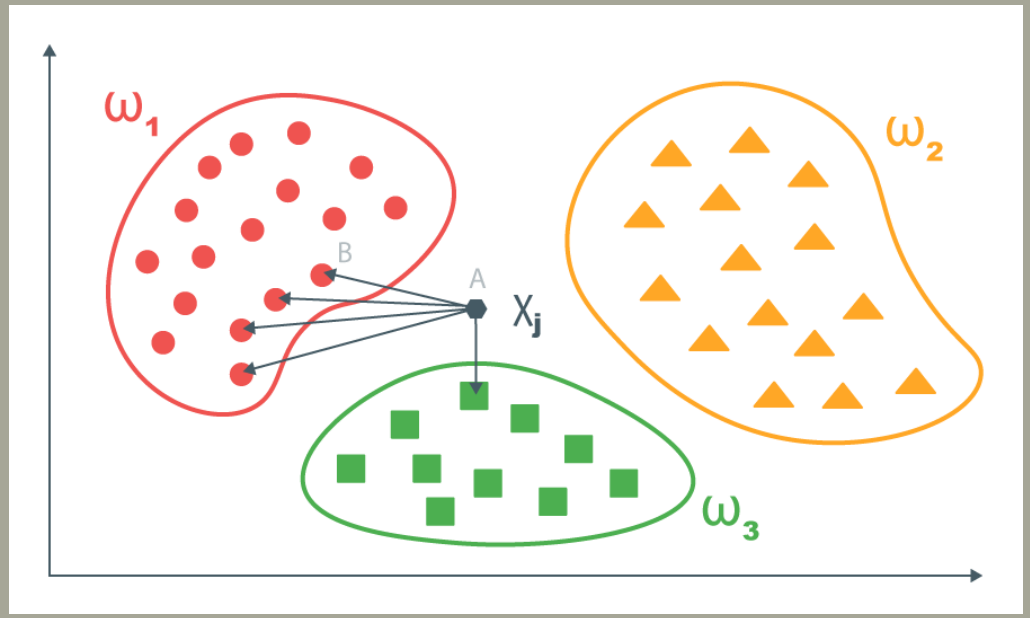
Предаемся конформизму

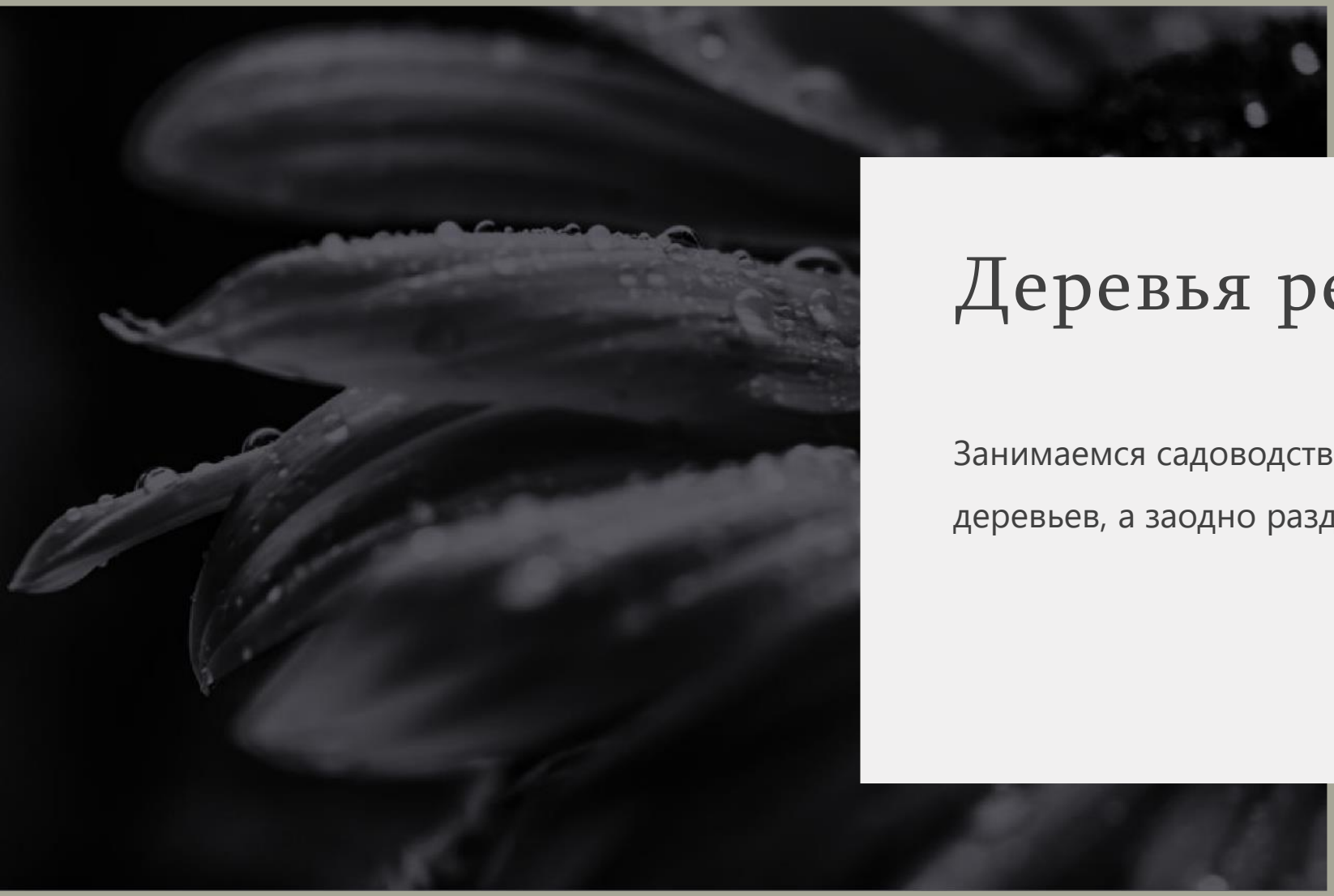
Метод ближайших соседей

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.



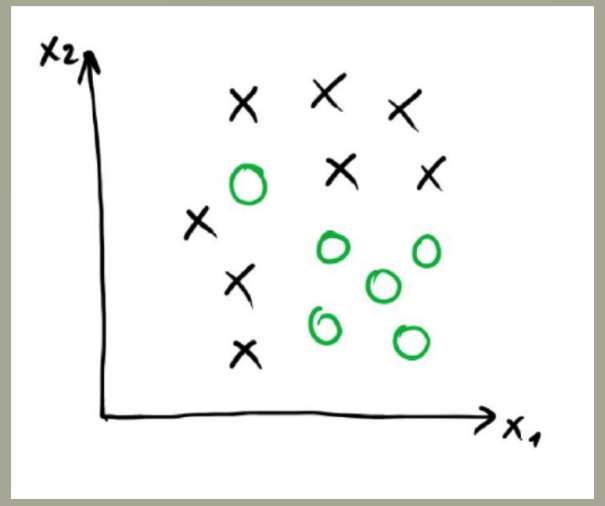


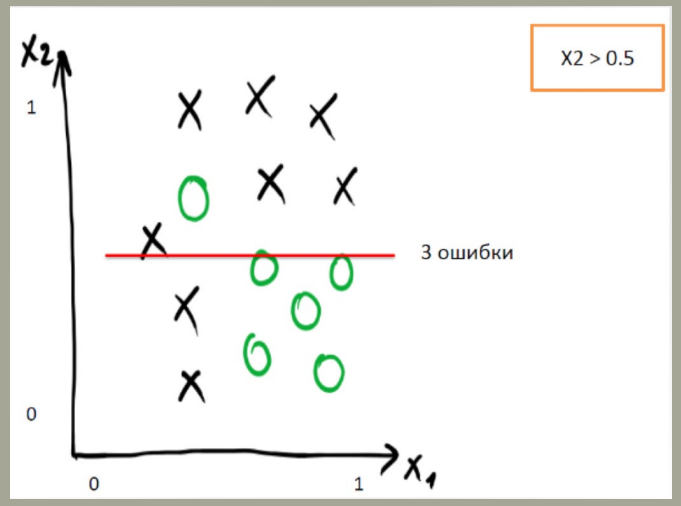
Деревья решений

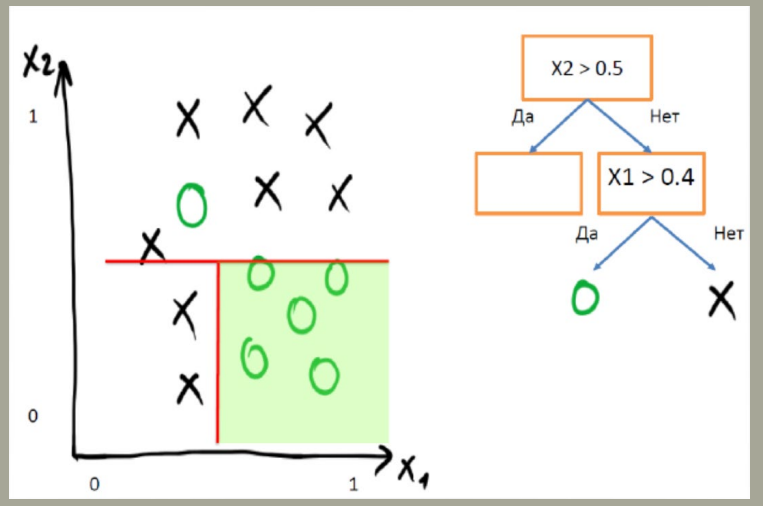
Занимаемся садоводством и стрижкой деревьев, а заодно раздаем кредиты

Решающие деревья

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



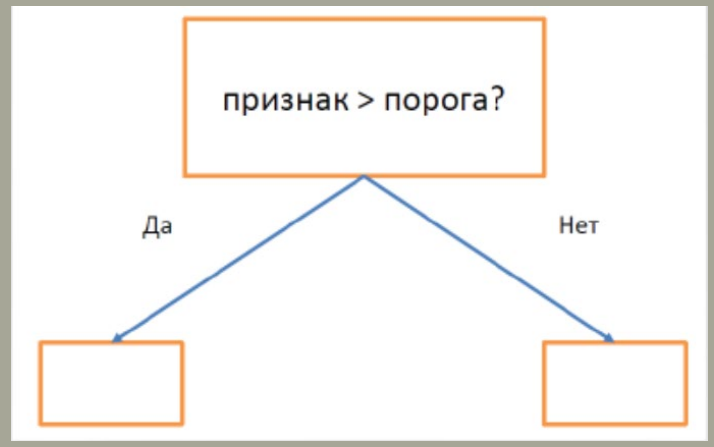




Решающие деревья

Решающее дерево – это бинарное дерево, в котором:

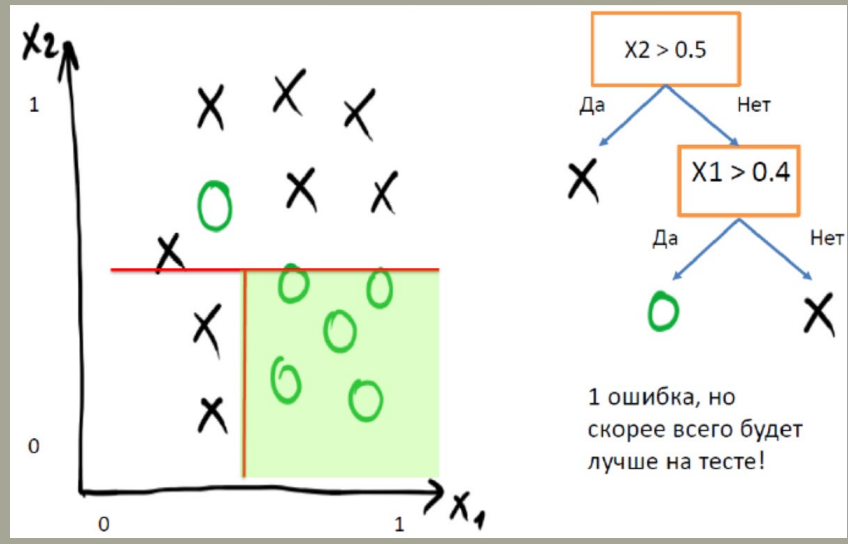
1) Каждой вершине v приписана функция (предикат) $\beta_v: X \to \{0, 1\}$



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

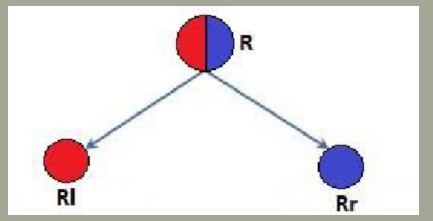
Решающие деревья

- 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 – когда отрезаем слишком низкие узлы



Критерии информативности

- 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 $H(R_l) \to min\, H(R_r) \to min$



Функционал качества

К чему все это?..

Нам же нужно что-то оптимизировать в нашем методе, вот и будем критерий информативности минимизировать (а функционал качества – максимизировать)

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

Q(R, j, t) – функционал качества, 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.

Решающие деревья и регрессия

В каждом листе дерева – некоторый набор возможных значений.

Если мы используем в качестве функционала ошибки RMSE, то в качестве ответа надо выдавать среднее значение целевых переменных всех объектов, попавших в лист.

Но если функционал ошибки другой, то и итоговое значение будет считаться как-то иначе!

Критерии останова

Как спастись от переобучения?

1) Использовать стрижку (сперва построить полностью, а потом похерить нижние листочки)
2) Использовать 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 в листе все объекты одного класса
   - Требование, чтобы функционал качества при дроблении увеличивался как минимум на s% (если слишком мало растет – останавливаем)

GridSearch и гиперпараметры

Соберем все выделенные красным параметры…

- Константа C в SVM
- Число соседей K в KNN
- max_depth, min_samples_leaf в деревьях

- alpha в Lasso и Ridge
- Также есть некоторые другие

Гиперпараметры и параметры

- *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.