Деревья решений

Classification Tree (CART) в Python

Версия 3 25.01.2021

Classification and regression trees CART.
деревья классификации и регрессии

Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees.

В Python чаще всего используют процедуры

`sklearn.tree.`**`DecisionTreeClassifier`**
`sklearn.tree.`**`DecisionTreeRegressor`**

В R чаще всего используют пакеты **rpart** и **party**.

Термин CART защищен патентом!

Желаете написать процедуру с таким названием, платите деньги. Под другими названиями можно. Поэтому в R процедура называется ***rpart***, а в Python ***DecisionTreeClassifier***

Дервья классификации — слабый (weak) метод классификации (регрессии).

Но комбинируя деревья можно получить сильные классификаторы. Такие как случайный лес (Random Forest), gradient boosting machine и XGBoost.

Нейронные сети состоят из нейронов.

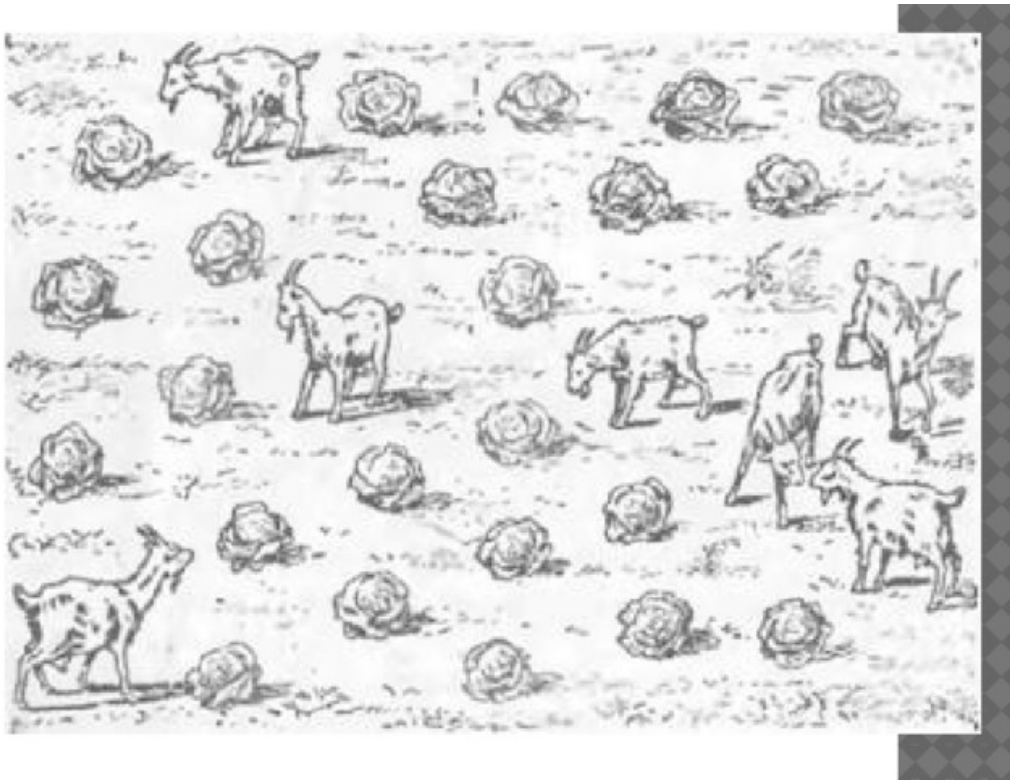
Случайный лес (Random Forest) и XGBoost состоят из деревьев.

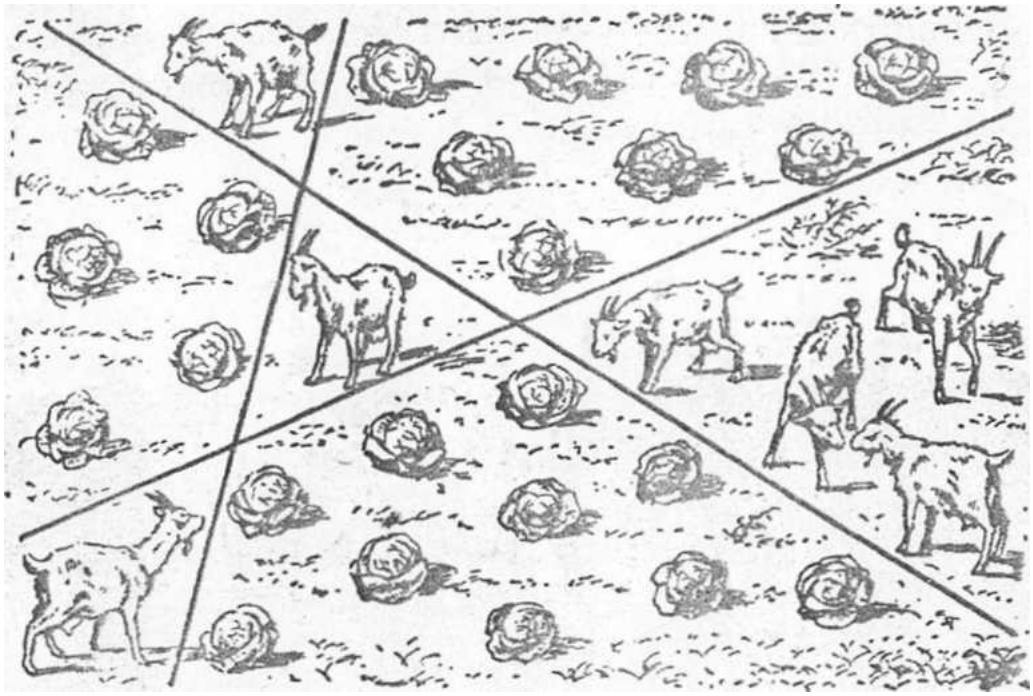
Поэтому без изучения деревьев классификации и регрессии не обойтись.

Деревья легко интерпретируются. В некоторых ситуациях ради понимания можно отказаться от других методов, жертвуя качеством результата.

Отделите коз от капусты тремя прямыми линиями.

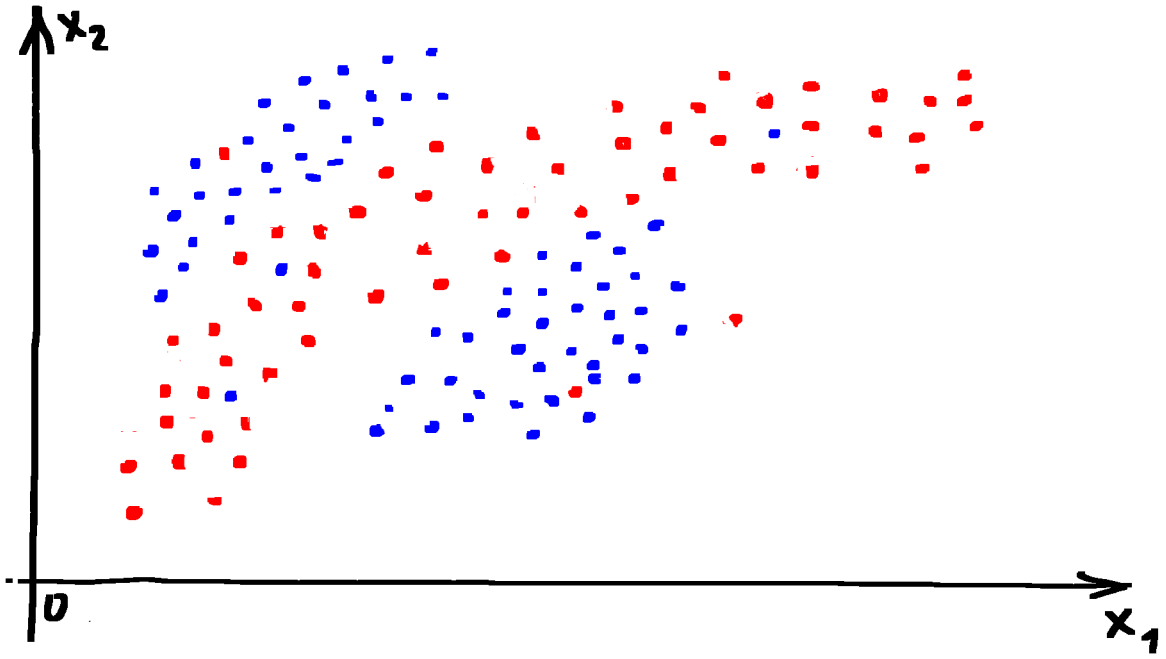
(2 класс ?)



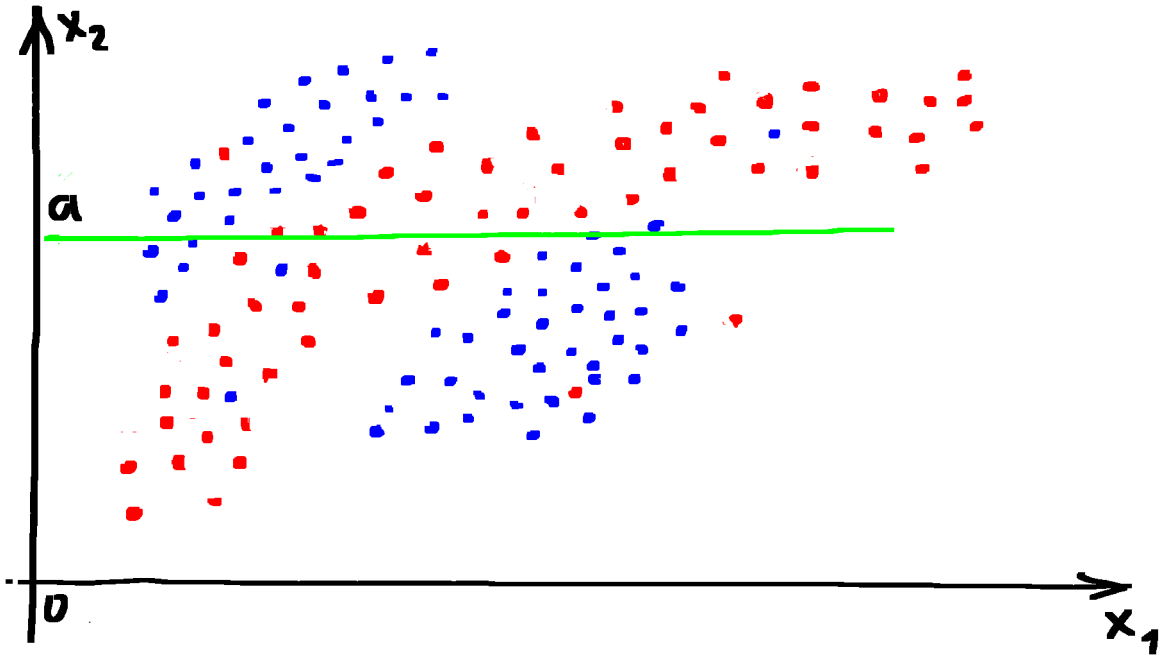


Алгоритм построения дерева классификации на примере.

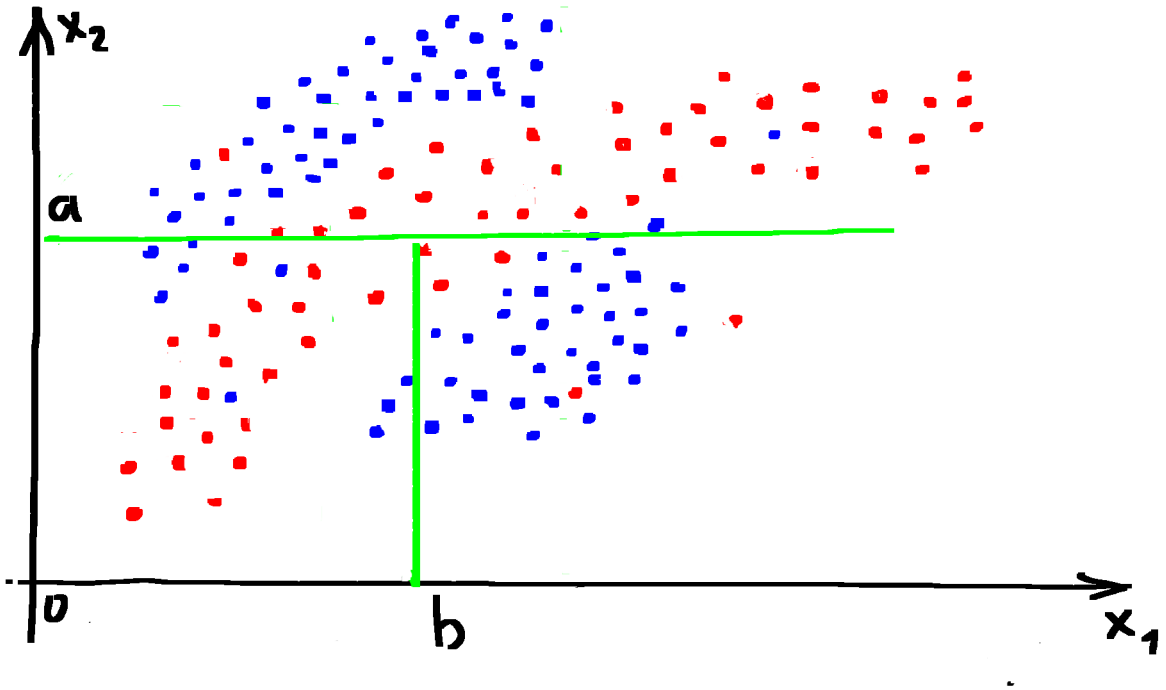
Первый вариант представления дерева классификации. Графический.



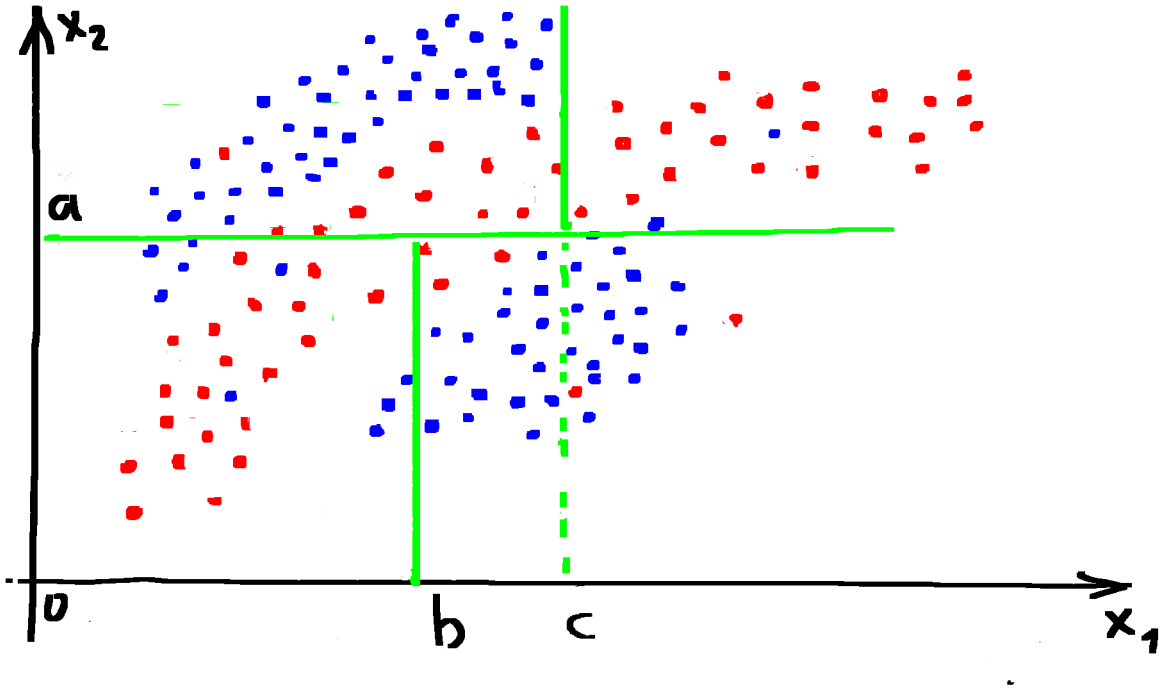
Шаг 0. Исходные данные, два класса



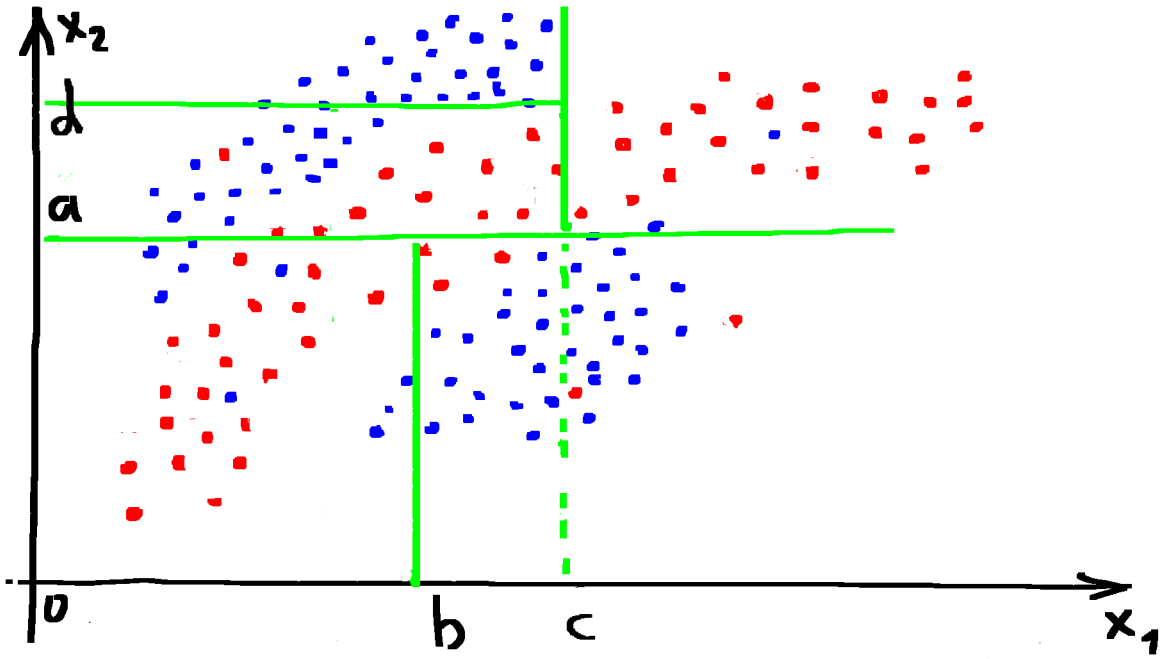
Шаг 1. Первое разделение (но процедура выберет другую)



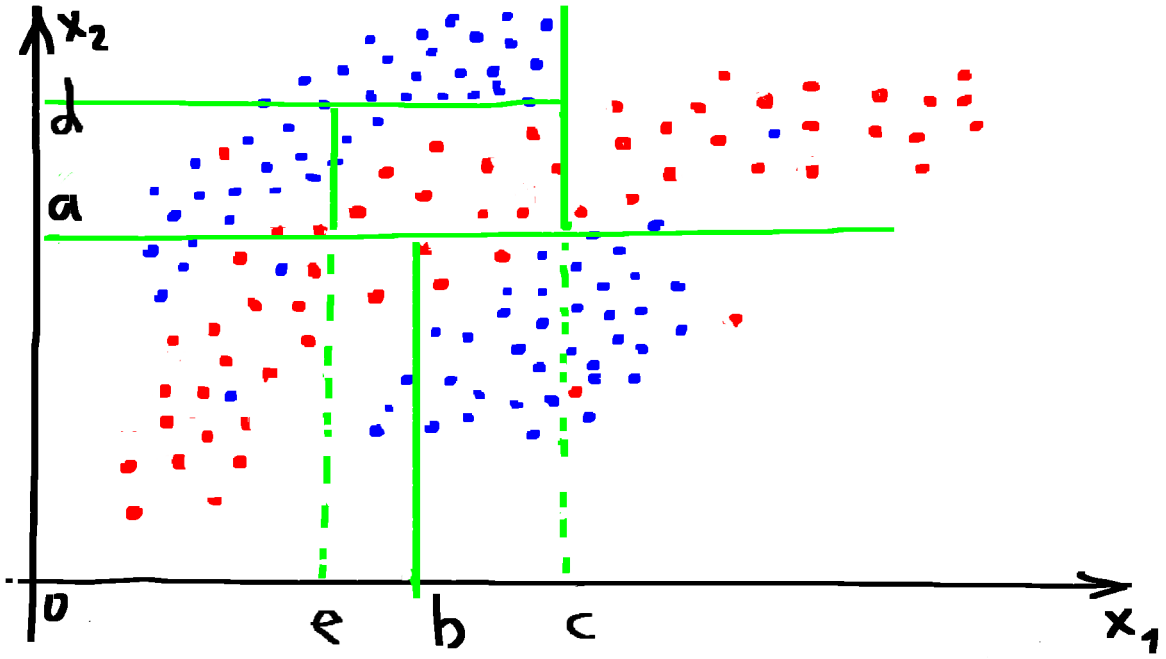
Шаг 2. Второе разделение. Делится только нижняя полуплоскость.



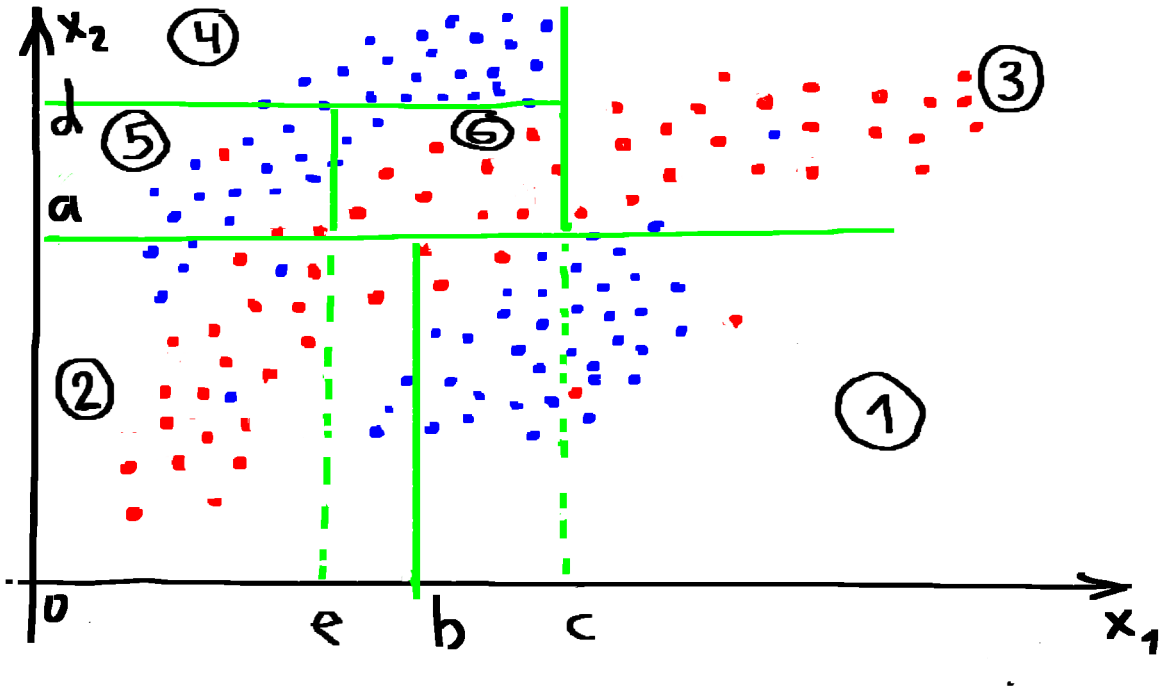
Шаг 3. Третье разделение. Делится только верхняя полуплоскость.



Шаг 4. Четвертое разделение.



Шаг 5. Пятое разделение. Пожалуй, дальнейшие разделения уже не нужны. (Почему?)



Шаг 6. Для дальнейшего удобно обозначить каждый прямоугольник.

Второй вариант представления дерева классификации. В виде логических правил.

1 $(x_1 > b) \cap (x_2 < a) \Rightarrow синий\ класс$

2 $(x_1 < b) \cap (x_2 < a) \Rightarrow красный\ класс$

3 $(x_1 > c) \cap (x_2 > a) \Rightarrow красный\ класс$

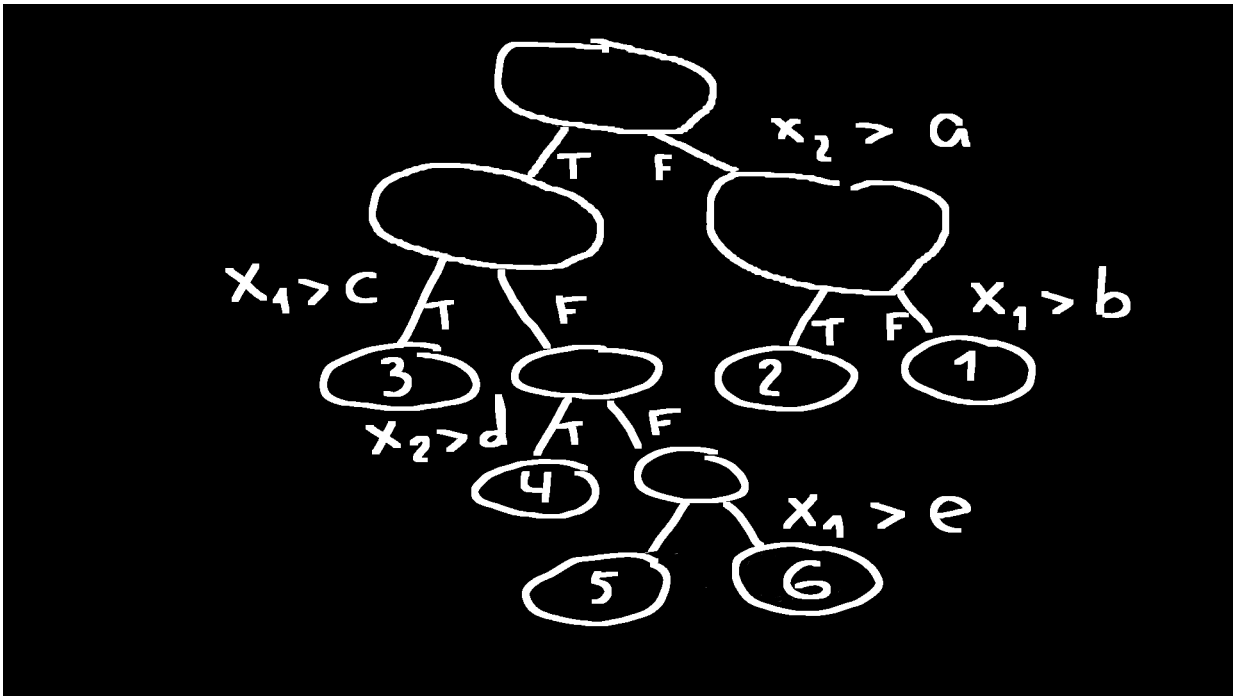
4 $(x_1 < c) \cap (x_2 > d) \Rightarrow красный\ класс$

5 $(x_1 < e) \cap (x_2 > a) \cap (x_2 < d) \Rightarrow красный\ класс$

Задача: Правило для шестого прямоугольника записать самостоятельно!

Третий вариант представления дерева классификации. Из которого становится понятно, причем тут деревья…

Дерево — это связный ациклический граф.



Основные понятия

Узел (node) — подмножество объектов из обучающей выборки.

родительский (parent node)
потомок (child node)
расщепление узла (node splitting)
конечный узел (лист дерева) (final node)
пороговое значение (threshold)

Алгоритм построения дерева классификации

Impurity measures (Меры загрязненности)

Обозначения
Пусть в задаче *k* классов.
Обозначим p_j - вероятность принадлежать классу *j*.
На практике p_j - доля объектов класса *j* в узле.

Графики функций вставить

Энтропия (entropy)

$$H_1 = -\sum_{j=1}^{k} p_j \cdot \log_2 p_j$$

Индекс Джини (Gini index)

$$H_2 = 1 - \sum_{j=1}^{k} p_j^{\,2} = \sum_{j=1}^{k} p_j \cdot (1 - p_j)$$

Ошибки классификации (classification error)

$$H_3 = 1 - max(p_j)$$

Когда узел родитель разделяется на 2 узла потомка.
Тогда очищение (увеличение чистоты узлов) измеряется посредством

$$\Delta H = H_{родителя} - \left(\frac{n_{левый}}{n_{родителя}} \cdot H_{левый} + \frac{n_{правый}}{n_{родителя}} \cdot H_{правый}\right)$$

Правила остановки обучения.
1. Ограничения на число наблюдений в родителе. Например, если в узле меньше 10 наблюдений, то такой узел запрещено расщеплять.
2. Ограничения на число наблюдений в потомке. Например, если в узле будет получено меньше 5 наблюдений, то расщепление с таким узлом запрещено
3. Если $\Delta H < \varepsilon$, то расщепление запрещено. Так как загрязнение уменьшается незначительно.
4. Ограничение на число слоев или на число конечных узлов. Например, требуем, чтобы в дереве было не больше шести слоев. Можно ограничивать число слоев, число расщеплений, число конечных узлов.

Более того, разработчики могут изменить характер этого ограничения в любой момент. Так пару раз было в R.

CART строит только бинарные деревья (что значит бинарное дерево?)

Другие способы строить деревья решений, конкуренты CART

На сегодня самый популярный метод CART. Но тучи сгущаются.
Фармацевтам и банкирам не нравится нестабильность деревьев CART (подробнее ниже)

C4.5 — алгоритм для построения деревьев решений, разработан Квинланом (John Ross Quinlan).
C4.5 улучшает алгоритм **ID3** того же автора.
Метод когда-то был самым популярным, но автор сделал процедуру платной…
Теперь метод снова бесплатный, но поезд ушел.

CHAID (Chi-square automatic interaction detection)
Метод допускает не бинарные расщепления, это хорошо. Расщепления часто оказываются несбалансированными (много узлов с малым числом элементов). Это плохо.

QUEST строит бинарные деревья классификации. Авторы Wei-Yin Loh и Yu-Shan Shih.
QUEST акроним для *Quick, Unbiased and Efficient Statistical Tree.*
Наименее популярный метод в списке.

Модификации деревьев решений.

oblique trees
пространство разделяется произвольными гиперплоскостями.
Снято требование, чтобы гиперплоскость была перпендикулярна одной из осей координат

Пакет R, реализующий процедуру oblique.tree был медленным и глючным, когда мы тестировали его 10 лет назад...
Пакет Python, реализующий процедуру oblique.tree отсутствует...

oblivious trees
В пределах одного слоя дерева все расщепления производятся по одной переменной.
Пакеты R и Python, реализующие процедуру: отсутствуют (?)
Говорят, раньше на их основе был реализован в Matrixnet…
Говорят, теперь поиск Яндекса теперь опирается на Deep Learning…

Обрезание деревьев (pruning).
Удаляем неэффективные узлы. Уменьшаем размер дерева и препятствуем переподгонке.

сначала обучаем дерево решений без ограничений, затем сокращаем (удаляем) ненужные узлы.

Версия 1
Узел родитель, все узлы потомки которого являются конечными, объявляется ненужным и удаляется, если уменьшение загрязнения, которое он обеспечивает, не является статистически значимым.
Стандартные статистические тесты, такие как критерий χ2, используются для оценки вероятности того, что улучшение является чисто случайным результатом (который называется нулевой гипотезой). Если эта вероятность, называемая p-значением, выше заданного порога (обычно 0.05), то узлы потомки считаются ненужными и удаляются. Сокращение продолжается до тех пор, пока не будут удалены все ненужные узлы.

Узел родитель, все узлы потомки которого являются конечными, объявляется ненужными удаляется, если уменьшение загрязнения, которое он обеспечивает, не подтверждается на выборке валидации.

Проблемы

1 Несбалансированные классы в выборке. Надо балансировать...

Достоинства

0. **Результаты работы метода хорошо интерпретируются.**

1. Метод CART не требует никаких предположений о вероятностном распределении переменных. Вообще никаких предположений нет! Если предположений нет, то их нельзя нарушить…

2. Метод CART самостоятельно отбирает информативные переменные. В частности, если переменная не используется при построении дерева, то она неинформативна. Хотя все не так просто.

3. Дерево строится быстро. Особенно при сравнении с другими подходами.

Но есть исключение. Если предиктор измерен в номинальной шкале и может принимать в узле k значений, то этот узел можно разбить на две части $2^{(k-1)} - 1$ способами. Поэтому большие значения k недопустимы на практике. Если переменные измерены в порядковой или количественной шкале, то ограничений нет.

Построение дерева можно еще ускорить за счет binning(?), другое название гистограмма. Но это ускорение скорее важно для комбинаций деревьев (XGBoost, Light GBM), когда строятся сотни или тысячи деревьев.

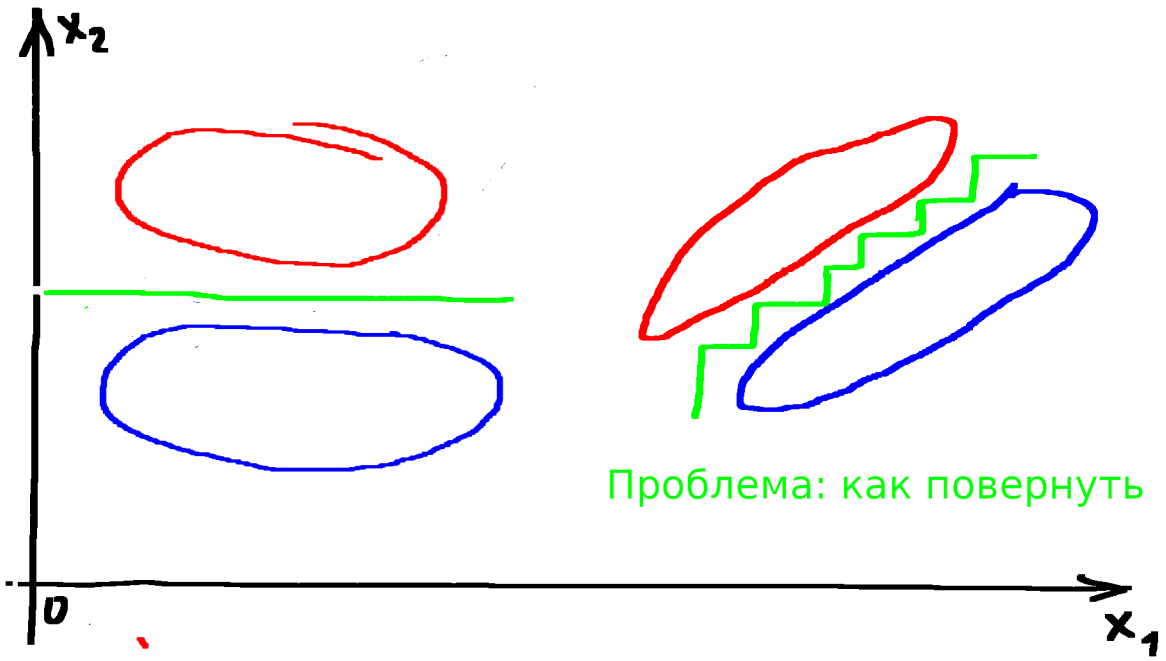
4. Метод устойчив (robust) к монотонным преобразованиям предикторов. Как следствие, стандартизация предикторов не нужна.

5. Метод устойчив (robust) к выбросам.

Недостатки

1. **CART Нестабилен к изменениям в обучающей выборке.** Деревья, построенные на разных выборках могут внешне сильно отличаться (дерево может увеличиваться, уменьшаться, включать другие предикторы и т.д..) Нестабильность осложняет валидацию модели. Хотя качество этих разных моделей будет примерно одинаковым. Поэтому в некоторых ситуациях CART “запрещен”
Как такое может случаться? Когда два расщепления очищают почти одинаково. Для одной выборке лучше окажется одно расщепление, для другой — второе.

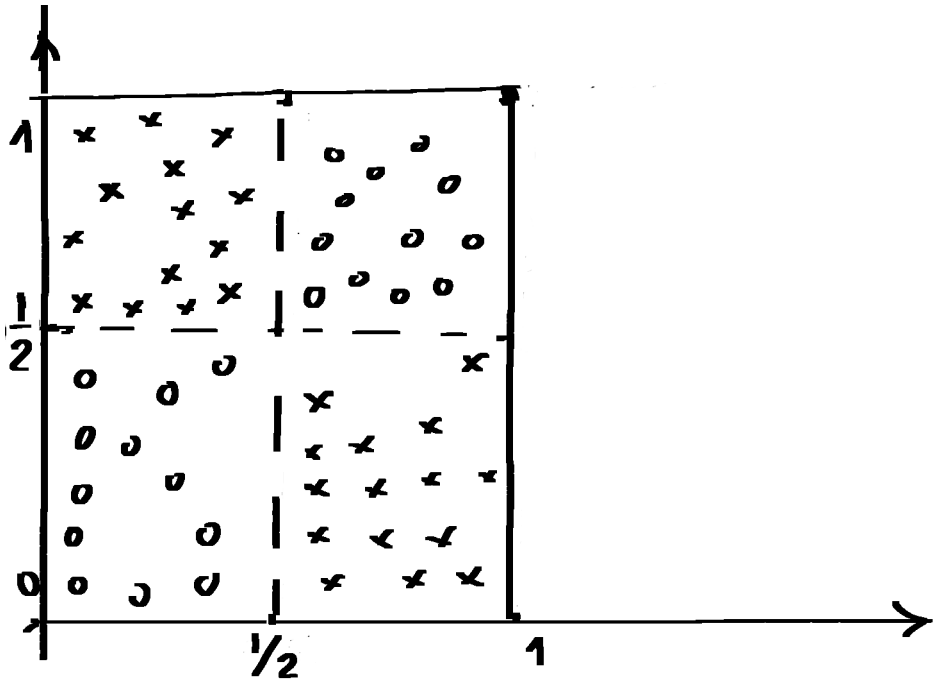
2. CART Неустойчив к поворотам данных



3. Процедуры питона DecisionTreeRegressor не умеют работать с пропусками (используйте rpart из R)

4. CART жадный алгоритм. Но нахождение оптимального дерева является NP-полной задачей: его построение требует O(exp(m)) времени, где m – число узлов. Заметим, что при

анализе данных используют множество жадных алгоритмов, у которых хорошая репутация. Например кластеризация методом к-средних (k-means)



Технические детали

1. Как изменить алгоритм, чтобы метод определял не код класса, а вероятность принадлежать классу?

2. Зачем в питоновских реализациях CART задают зерно датчика случайных чисел?

Во первых, может быть так, что два расщепления одинаково уменьшают загрязненность. Тогда выбираем расщепление «подбрасывая монетку»

Расщепление делаем случайным. См. Параметр процедуры

splitter : ***string, optional (default="best")***

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the ***best random split****.*

Моя рекомендация:используйте значение опции best. Если обучается долго, предварительно проведите binning.

Regression Tree (CART) в Python

Деревья регрессии — приближение кривой кусочно постоянной функцией

Обозначим
m – номер узла

N_m - число наблюдений в узле m

y_i - значения отклика для наблюдений, попавших в узел

Новому наблюдению, попавшему в узел, сопоставляется не класс, а значение отклика

$$\bar{y}_m = \frac{1}{N_m} \cdot \sum_{i \in узел_m} y_i$$

Impurity measures (Меры загрязненности) в задачах регрессии

$$H_2(узел_m) = \frac{1}{N_m} \cdot \sum_{i \in узел_m} (y_i - \bar{y}_m)^2$$

$$H_1(узел_m) = \frac{1}{N_m} \cdot \sum_{i \in узел_m} |y_i - \bar{y}_m|$$