Классификация

Деревья классификации и регрессии (CART)

Храмов Д.А.

09.02.2020

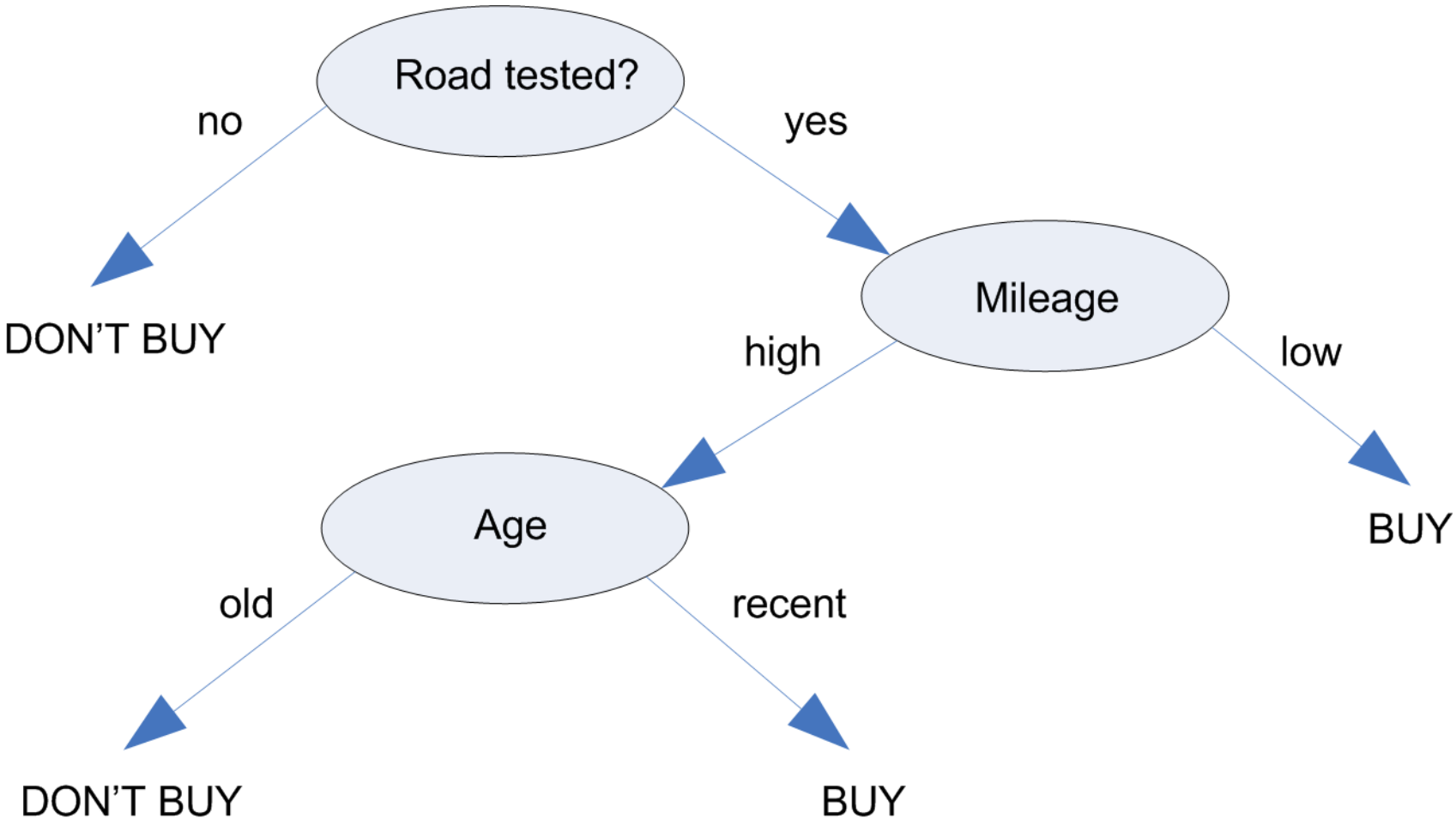
Содержание

- Деревья принятия решений.
- Построение деревьев классификации и регрессии (CART).
- Классификация CART в пакете `caret`.

Дерево обычное и дерево математическое

Деревья принятия решений

Дерево, используемое для классификации, называется — **дерево принятия решения** (decision tree).



Семейство методов

Семейство методов классификации при помощи одиночных деревьев:

- ID3 (Iterative Dichotomiser 3)
- C4.5 (развитие ID3)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector)
- MARS: улучшает обработку числовых данных
- Conditional Inference Trees (ctree): не требуют обрезки

Источник: https://en.wikipedia.org/wiki/Decision_tree_learning

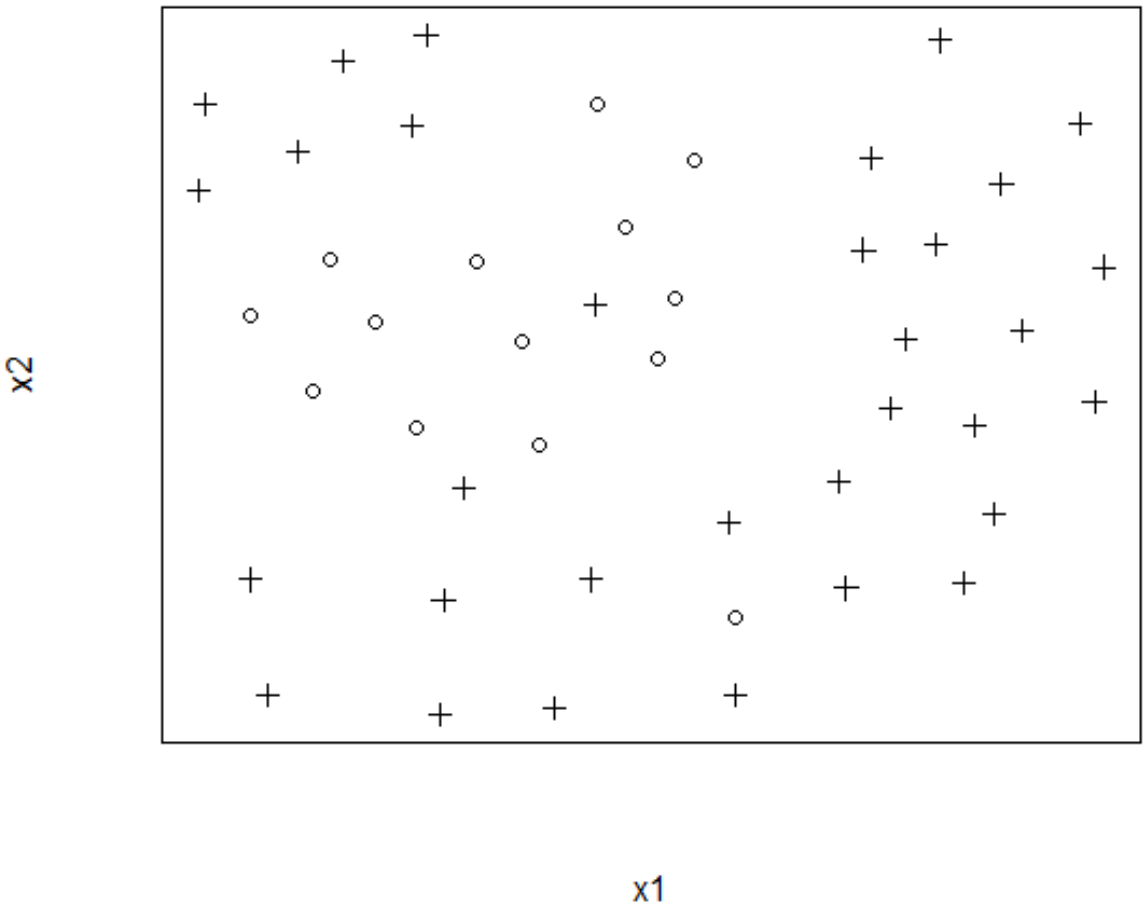
Рассмотрим метод CART (Classification and regression trees), предложенный в 1983 г. четверкой известных ученых в области анализа данных: Лео Брейманом (Leo Breiman), Джеромом Фридманом (Jerome Friedman), Ричардом Олшеном (Richard Olshen) и Чарлзом Стоуном (Charles Stone).

Особенности метода

- Его результаты удобно интерпретировать.
- Используются как составная часть более продвинутых методов (например, случайных лесов) и для оценочной классификации (разведки).

Методы классификации при помощи деревьев очень популярны.

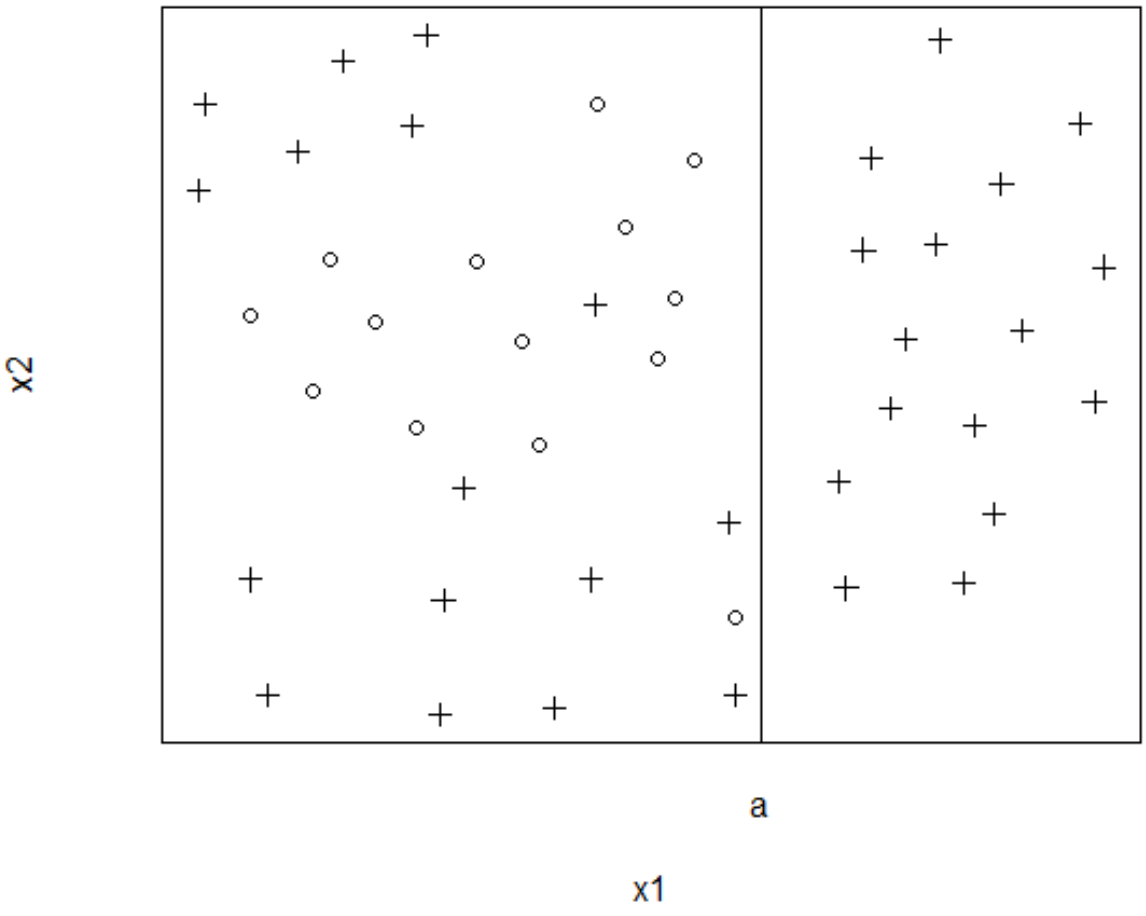
Бинарная классификация



Исходные данные составляют два класса

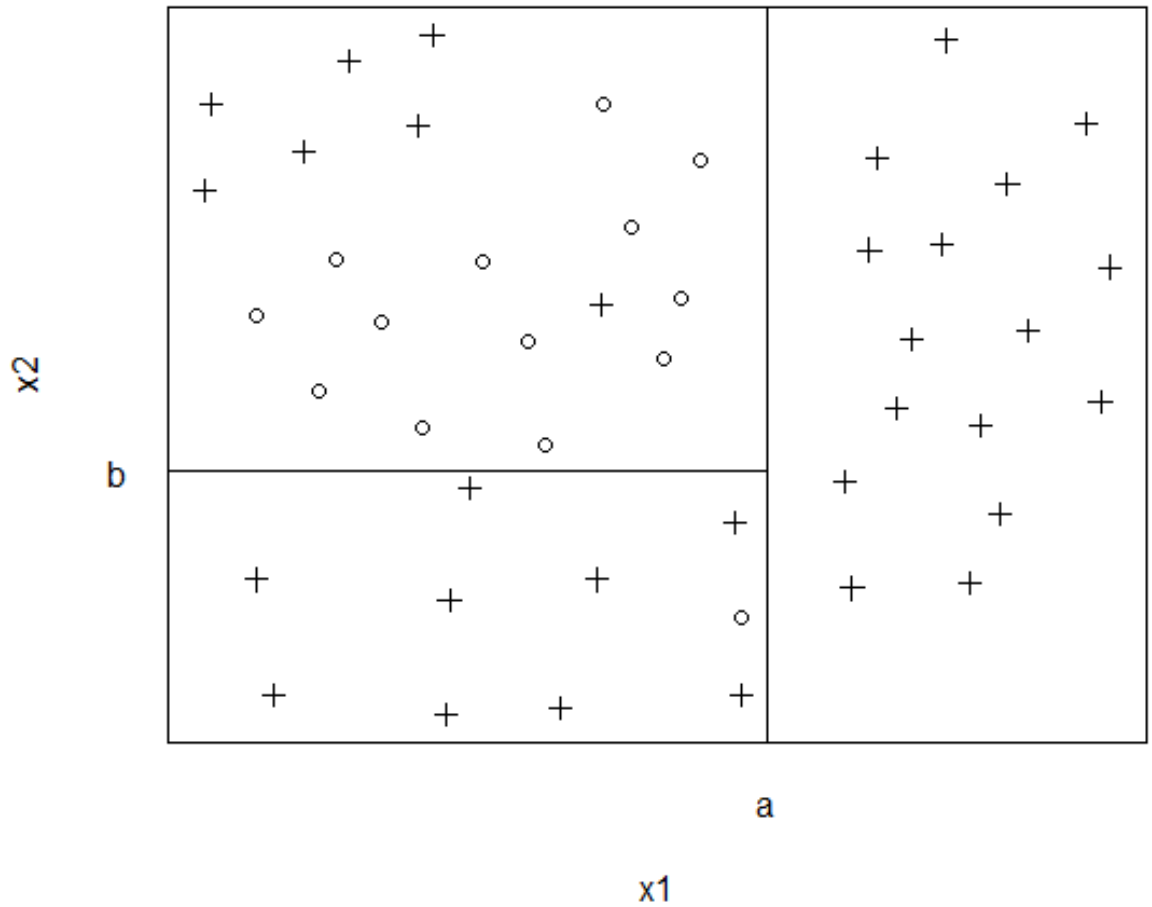
Алгоритм разбиения

1. Будем разделять пространство на части при помощи гиперплоскостей (в 2-мерном случае — это прямая, в 3-мерном — плоскость, то есть размерность на единицу меньше размерности пространства).
2. Разделяющая гиперплоскость должна быть перпендикулярна одной из осей координат.
3. Разделять будем так, чтобы в одной части оставалось больше объектов одного класса, а в другой — другого.

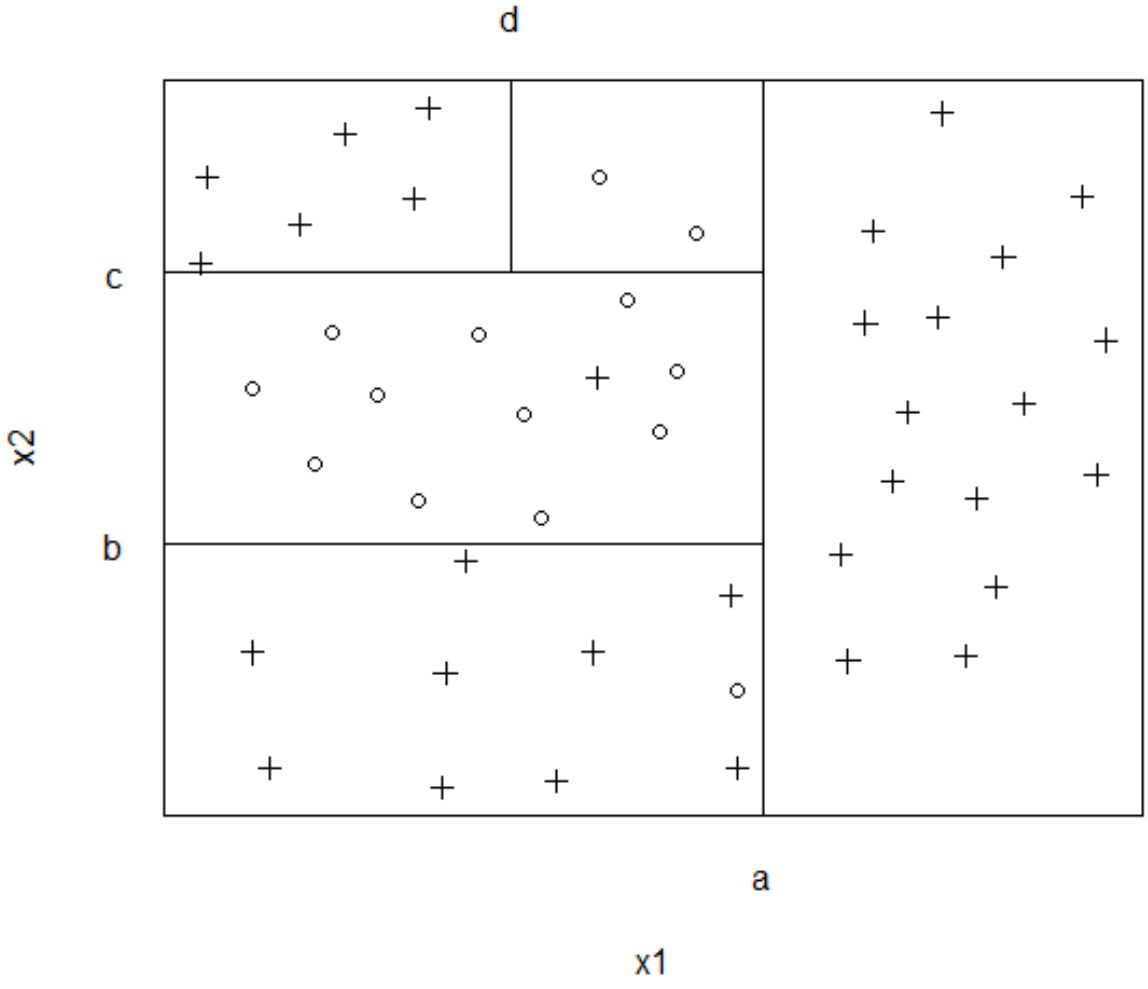


Первое разделение. Правило: $x_1 > a$ — “крестики”

Последующие прямые (гиперплоскости) будут делить одну из выделенных ранее полуплоскостей (полупространств)

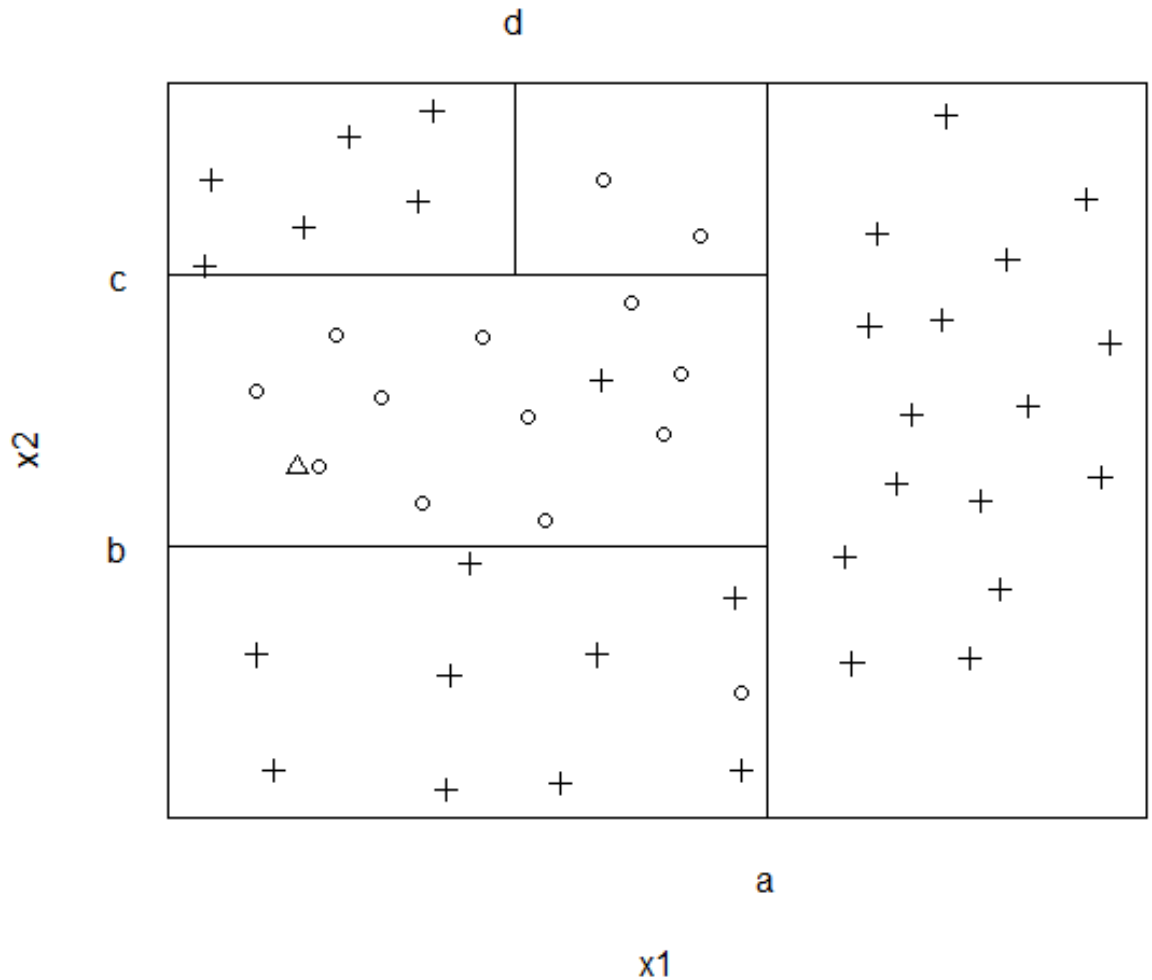


Второе разделение. Правило $x_2 < b$ — “крестики”



3-е и 4-е разделения: $x_2 > c$ и $x_1 > d$ — “нолики”

Новый объект будет относиться к классу, который составляет большинство в данной области.



Новый объект - “нолик”

Запись логических правил

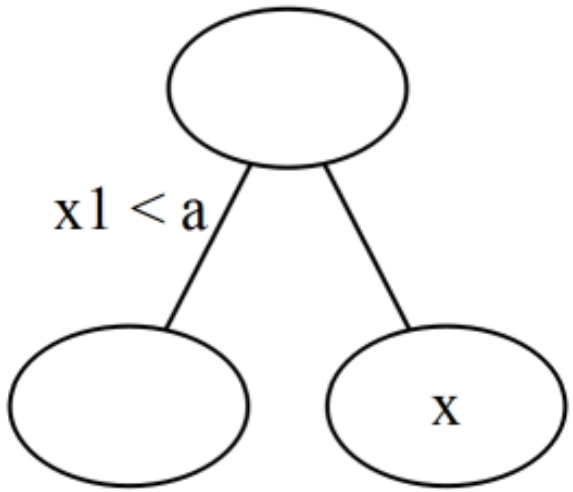
- 1 область: $(x_1 < a)$ AND $(x_2 < c)$ AND $(x_2 > b)$ -> “o”
- 2 область: $x_1 > a$ -> “+”
- ...

Всего 5 областей и 5 правил.

Такая запись правил очень удобна.

Первый взгляд на задачу — геометрический, второй — в виде формального логического правила. Где же тут деревья?

Начинаем строить дерево

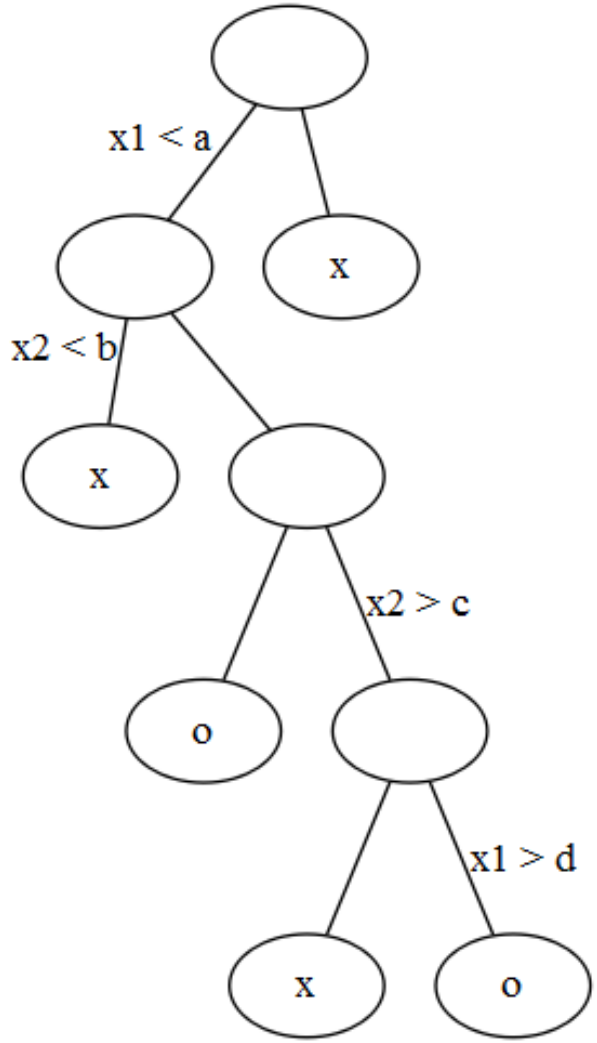


Графическое отображение правила $x_1 < a$

Основные понятия

- узел (node) — это множество наблюдений
  - родительский (parent node) — узел, который разделяется;
  - потомок (child node) — узел, возникший после деления родительского узла;
  - конечный узел (лист дерева) (final node, terminal node) — узел, который дальше не разделяется;
- пороговое значение — числа a, b, c, d в которых проходят сечения.

Запись в форме дерева принятия решений



Полученное дерево дало название методу

Для разметки нового объекта

1. Смотрим, в каком узле оказалась точка.
2. Объекты какого класса составляют в этом узле большинство?
3. Приписываем этому классу новый объект.

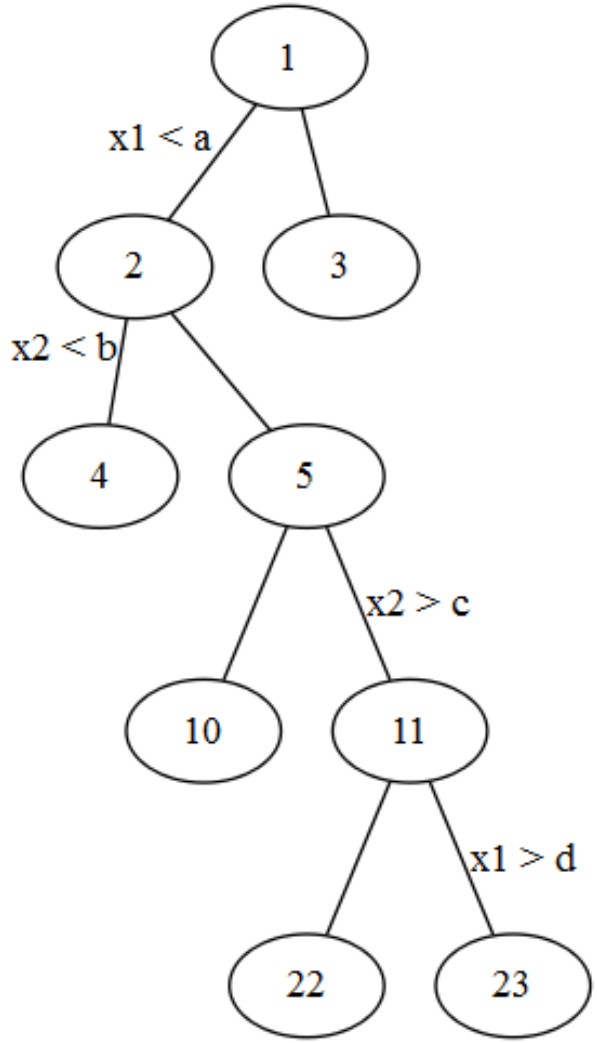
Метод хорошо поддается интерпретации. Если вы не хотите использовать “черный ящик” (который выдает результат, но непонятно как), то используйте этот метод.

Особенности построения дерева классификации в `rpart()`

Функция `rpart` реализует CART (Classification And Regression Trees). Слово CART защищено копирайтом, поэтому функцию пришлось назвать по-другому.

- Все узлы пронумерованы. Номера не идут подряд, а следуют правилу: узлы-потомки узла номер `n` имеют номера `2*n` и `2*n+1`. Это удобно, потому что по номеру узла можно определить номер его родителя.
- При расщеплении, когда условие выполнено, наблюдение попадает в левый узел.

Нумерация узлов в `rpart()`



Узлы-потомки узла номер `n` имеют номера `2*n` и `2*n+1`

Разновидность метода: oblique trees — косые или наклонные деревья

Отказываемся от требования перпендикулярности гиперплоскости осям координат.

- Для достижения цели понадобится меньше разбиений.
- Дает намного более сложные условия

Источник: http://research.ijcaonline.org/volume82/number13/pxc3892023.pdf

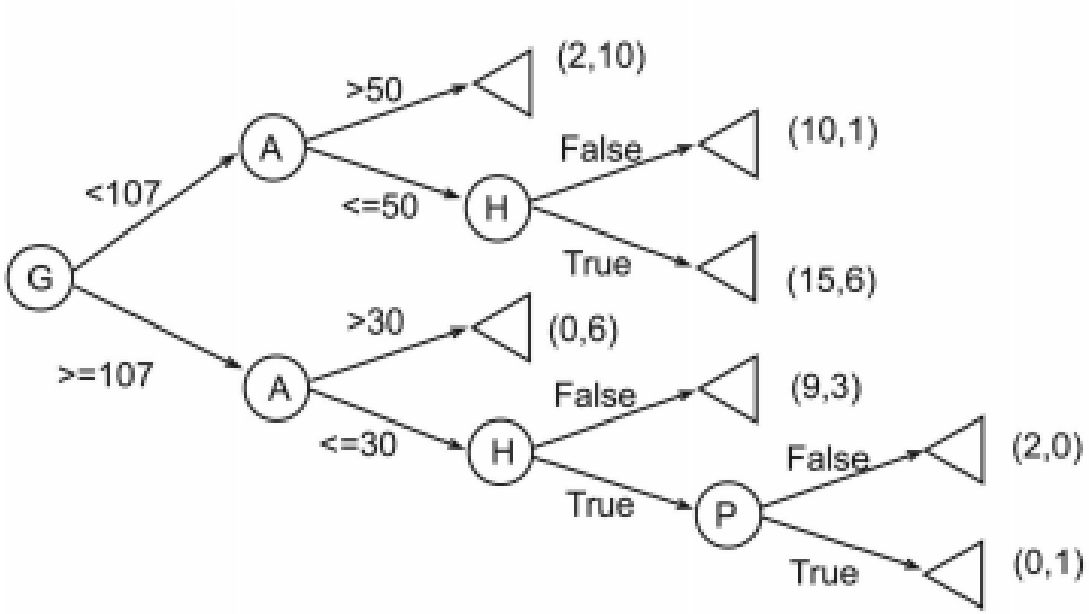
Разновидность: oblivious trees — забывающие деревья

Ужесточим условия:

1. Разделим дерево на слои по высоте.
2. Потребуем, чтобы на каждом слое разделение происходило только по одной переменной.

Выигрыш — в скорости выполнения расчетов.

Пример забывающего дерева решений



- Входные признаки: уровень глюкозы (G), возраст (A), артериальная гипертензия (H) и беременность (P).
- Бинарная целевая функция — страдает ли этот пациент диабетом.
- Числа в терминальных узлах указывает количество объектов класса, прошедших по этому пути. Так, для пациентов с уровнем глюкозы ниже 107 и возрастом старше 50, у 10 диагностирован диабет, а у 2 — не диагностирован.

Источник: Rokach L., Maimon O. Decision Trees

Как строить разделяющие гиперплоскости

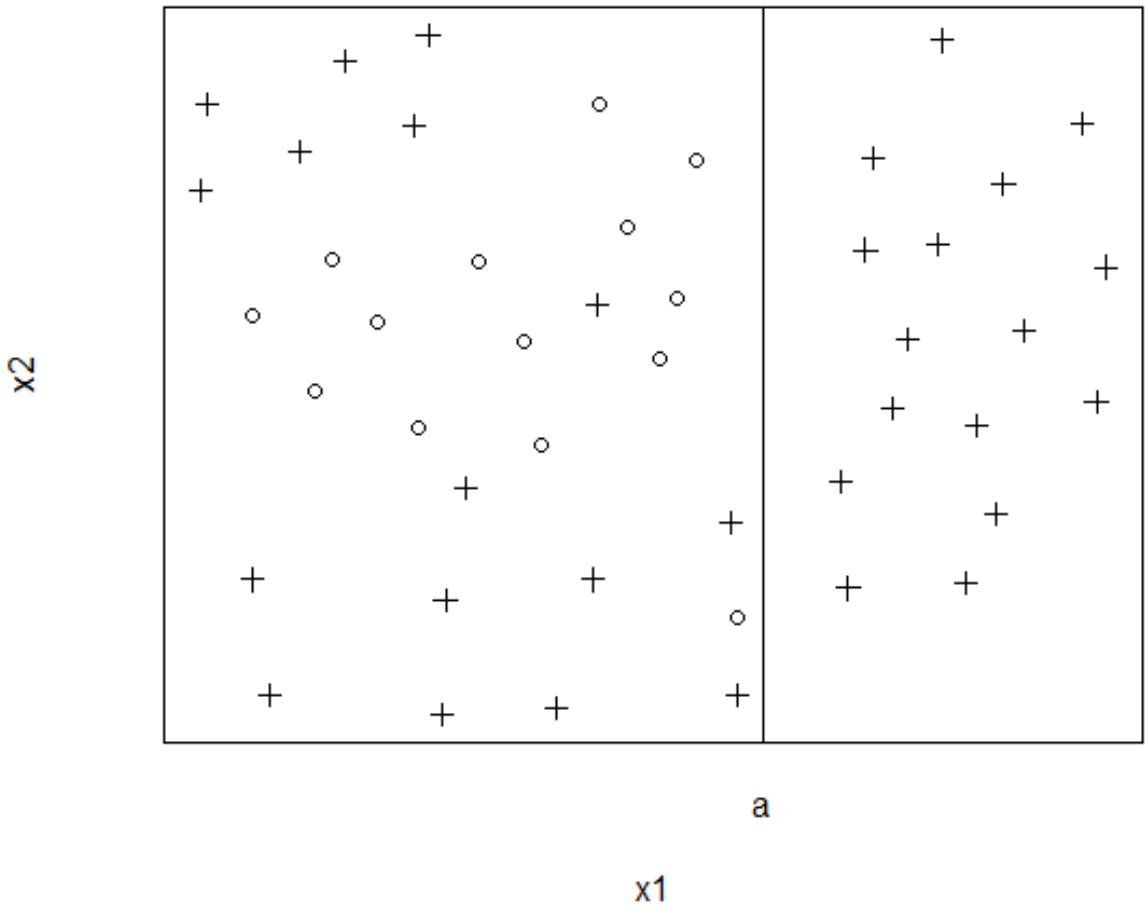
- Как искать пороговые значения?
- Как выбирать, по какой переменой разделять?

— как этот процесс автоматизировать, чтобы сделать из него функцию (потерь?).

Мы двигали разделяющую прямую так, чтобы с одной стороны было подавляющее число объектов одного класса, а с другой — другого.

Введем понятие **чистоты** (purity).

Разбиение увеличивает чистоту



$x_1 > a$ — чистое множество, в нем только “крестики”

Нужно получить такое разбиение, чтобы результат его стал чище исходного неразделенного множества.

p('x') — вероятностью "крестиков" — называется доля крестиков в каком-то конкретном узле.

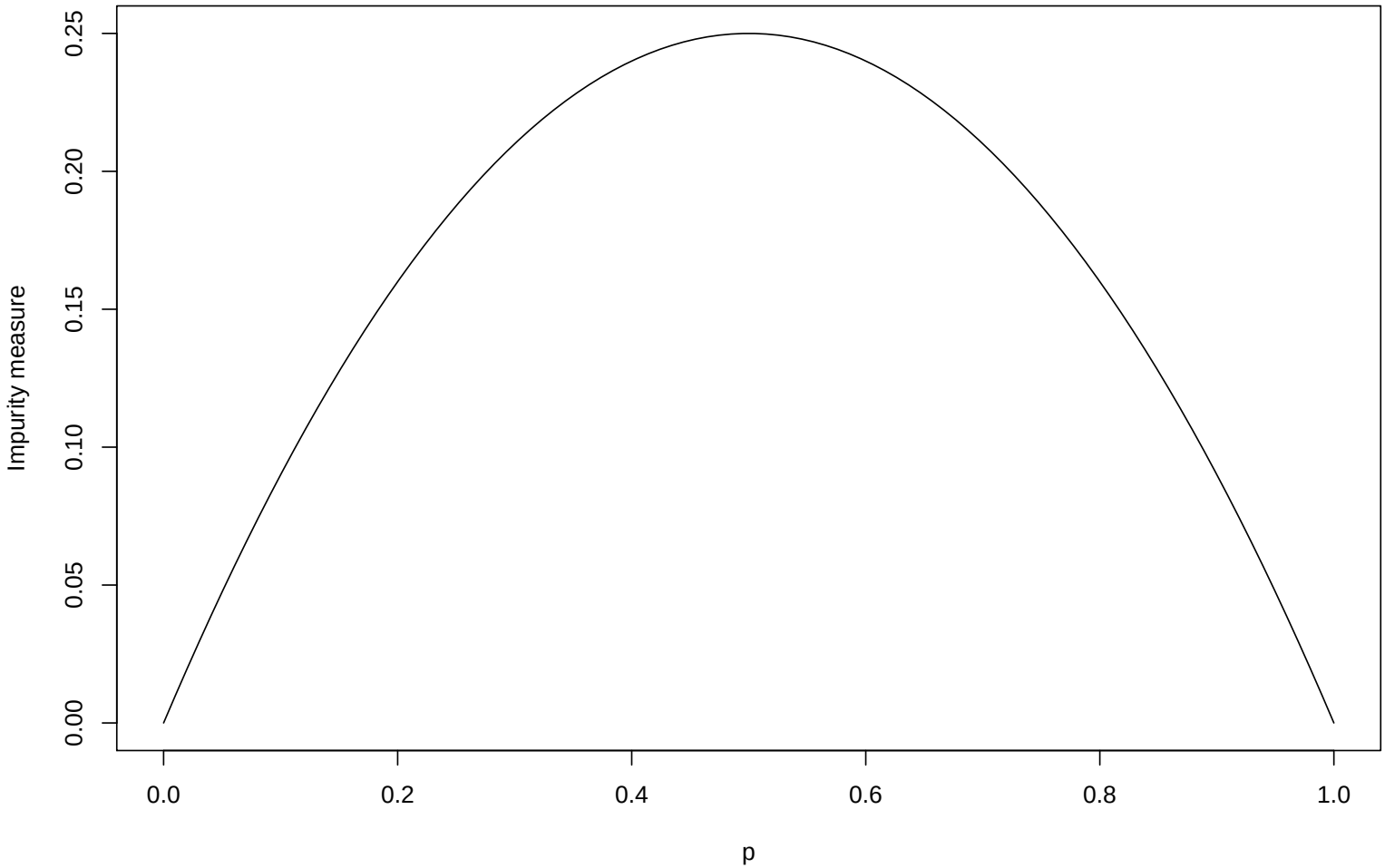
Узел чистый, когда $p_i = 1$ (в узле только "крестики") или $p_i = 0$ (только "нолики").

Максимально "грязный" (impurity) случай: $p_i = 0.5$.

Теперь нужна функция, которая будет измерять чистоту множества (узла).

Эта функция должна иметь минимум в точках 0 и 1 и максимум — где-то между ними, в идеале — в 0.5.

Вот какая функция нужна



Если у нас будет такая функция, то

. . . можно будет сравнить чистоту множества до разделения и чистоту двух множеств, полученных после разделения. Если после разделения стало чище, то это хорошо.

Затем из всех возможных разделений возьмем то, что приводит к наиболее чистым результатам (максимально сокращает степень загрязненности).

Способы измерения чистоты: 1) Энтропия

Понятие энтропии происходит из физики

$$H_1 = -\sum_{i=1}^{N} p_i \log_2 p_i$$

Рассмотрим некоторую систему, которая может принимать конечное множество состояний x_i с вероятностями p_i.
Энтропия — мера неопределенности состояния системы.

Энтропия 1) обращается в нуль, когда одно из состояний системы достоверно, а другие — невозможны; 2) при заданном числе состояний обращается в максимум, когда эти состояния равновероятны (с увеличением числа состояний энтропия увеличивается).

Знак минус перед суммой поставлен для того, чтобы энтропия была положительной (вероятности меньше единицы и их логарифмы отрицательны).

Способы измерения чистоты: 2) Индекс Джини (Gini index)

Функция $y = x(1 - x)$ имеет минимумы и максимум как раз там, где нам нужно. Поэтому

$$H_2 = \sum_{i=1}^{N} p_i(1 - p_i) = 1 - \sum_{i=1}^{N} p_i^2$$

По умолчанию для измерения загрязненности мы будем использовать индекс Джини.

Другие формулы для измерения загрязненности: https://en.wikipedia.org/wiki/Decision_tree_learning

Создатели метода CART разрабатывали его, чтобы отличать силуэты советских кораблей от американских. Оказалось, что лучше работает индекс Джини, хотя сперва применяли энтропию.

Это означает, что при решении задачи стоит поиграть с мерами загрязненности.

Какую переменную для разбиения следует выбрать и какое установить для нее пороговое значение?

Используем метод перебора.

1. Перебираем переменную за переменной (n — число переменных).
2. Для каждой переменной перебираем пороговые значение.
3. Для каждого порогового значения считаем ΔH — насколько сократится загрязненность (возрастет чистота) в результате разбиения.
4. Выбираем переменную и пороговое значение, соответствующие максимальной ΔH.

Зададимся некоторым разделением узла на 2 потомка. Тогда увеличение чистоты узлов измеряется как

$$\Delta H = H_{\text{родителя}} - \left(\frac{n_1}{n_{\text{родителя}}} H_{\text{потомок1}} + \frac{n_2}{n_{\text{родителя}}} H_{\text{потомок2}} \right)$$

Правило останова

Когда нужно прекратить расщеплять?

1. $\Delta H < \delta_{min}$ — улучшение слишком мало.
2. Число слоев превысило заданное значение (maxdepth — “глубина” дерева).
3. Размер узла-родителя — не дробить слишком маленькие множества (minsplit).
4. Размер конечного узла — конечные узлы должны быть достаточно представительными для принятия решения относительно класса нового элемента (minbucket).

Почему нужно прекратить расщеплять: мы опасаемся переобучения.

Кредитный скоринг

Моделируется задача кредитного скоринга. Нужно разработать процедуру, которая по данным о заемщике будет определять, надежен он или нет.

Данные о заемщике:

- `кредит` — кредитный рейтинг (низкий, высокий) — целевое значение.
- `класс` — информация о профессии (“Management”, “Professional”, “Clerical”, “Skilled Manual”, “Unskilled”).
- `з_плата` — периодичность выдачи заработной платы (еженедельно, ежемесячно).
- `возраст` — возраст (“Молод (< 25)”, “Средний(25-35)”, “Пожилой(> 35)”).
- `кр_карта` — наличие кредитной карты данного банка.

Чтение и проверка данных

```
# Шаг 0. Прочитаем данные
credit.01 <- read.table("data/credit.csv",
                        header=T, sep=";",
                        encoding = "UTF-8")
# Проверка: импортировали правильно?
credit.01[1:5,]
```

```
##   кредит класс з_плата возраст кр_карта
## 1      1     2       2       2         1
## 2      0     2       1       2         0
## 3      0     4       1       1         1
## 4      1     2       2       2         0
## 5      1     3       2       1         0
```

```
dim(credit.01)
```

```
## [1] 323   5
```

К каким классам относятся переменные?

```
class(credit.01[ , 1])
```

```
## [1] "integer"
```

```
class(credit.01[ , 2])
```

```
## [1] "integer"
```

```
class(credit.01[ , 3])
```

```
## [1] "integer"
```

```
class(credit.01[ , 4])
```

```
## [1] "integer"
```

```
class(credit.01[ , 5])
```

```
## [1] "integer"
```

Внимание аналитикам!

Теперь надо принимать решение: должны быть переменные факторами или нет?

То есть в какой из шкал — номинальной, порядковой или интервальной — измерены переменные.

Функция классификации

```
library(rpart)
credit.01.res <- rpart(кредит ~ класс+з_плата+возраст+кр_карта,
                                data = credit.01, method="class",
                                control=rpart.control(minsplit=10,
                                                      minbucket=5,
                                                      maxdepth=6)
                       )
```

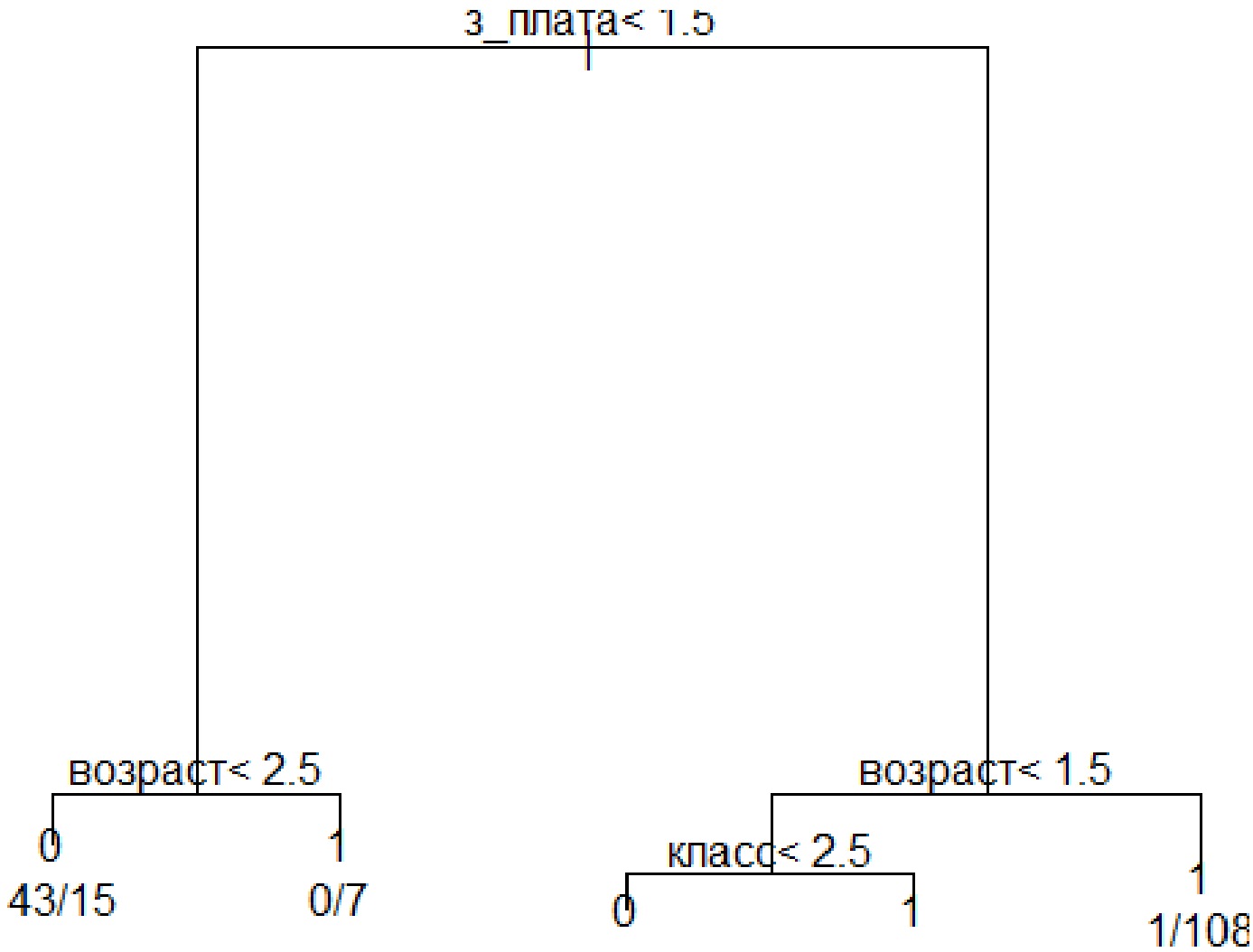
- `credit.01.res` — список с результатами (модель).
- `rpart` — функция, которая строит дерево.
- `data = credit.01` — анализируются переменные из таблицы `credit.01`.
- `method="class"` — строится дерево классификации, а не дерево регрессии.
- `model = TRUE` — сохранить копию данных внутри модели (списка credit.01.res).
- `control = rpart.control(minsplit=10, minbucket=5, maxdepth=6)` — какие правила останова использовать.

Функция классификации: экономная запись формулы

```
credit.01.res <- rpart(кредит ~ .,
                                data = credit.01, method="class",
                                control=rpart.control(minsplit=10,
                                                      minbucket=5,
                                                      maxdepth=6)
                       )
```

Рисование дерева: встроенный plot

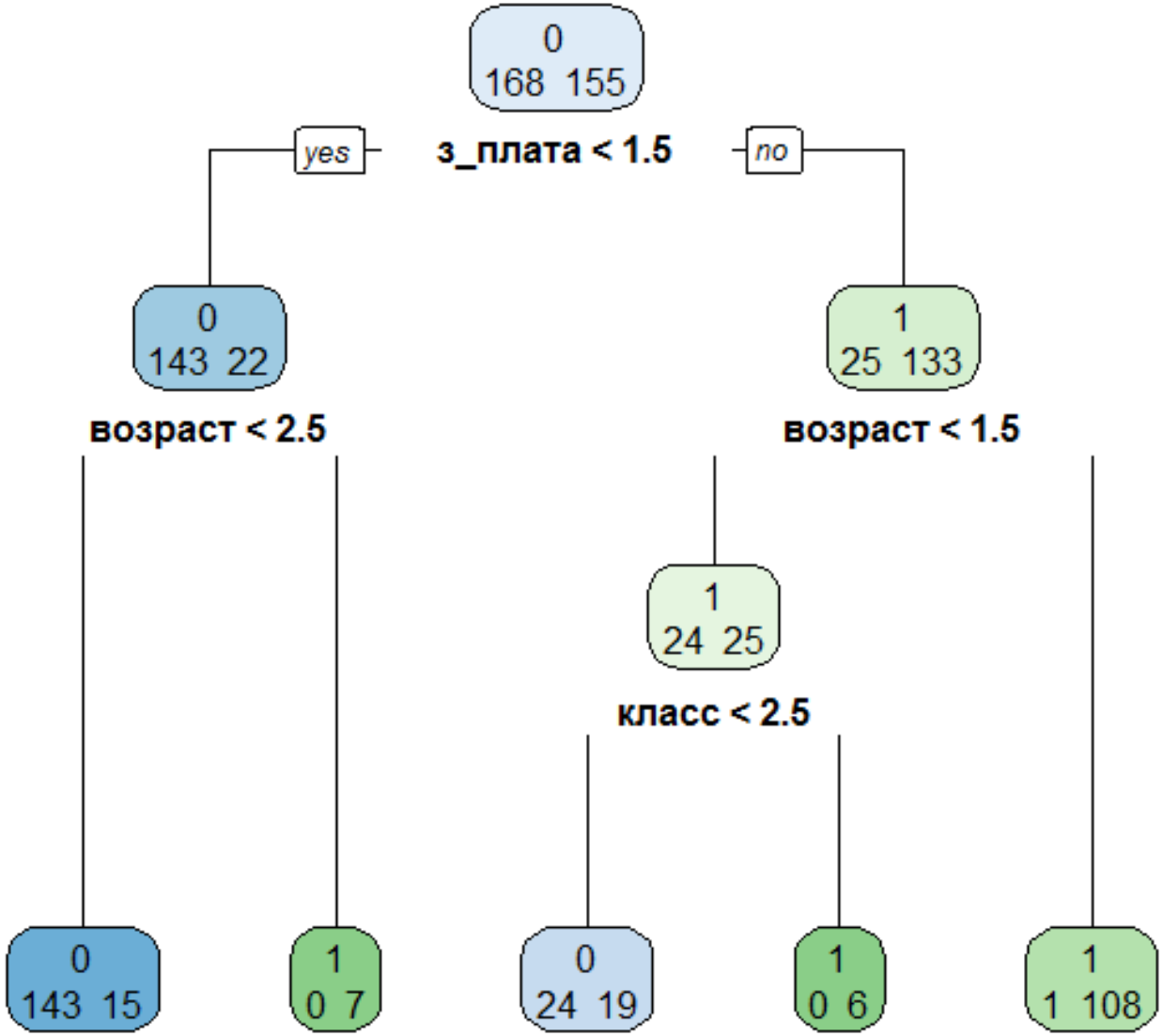
```
plot(credit.01.res)
text(credit.01.res, use.n=T)
```



Рисование дерева: библиотека rpart.plot

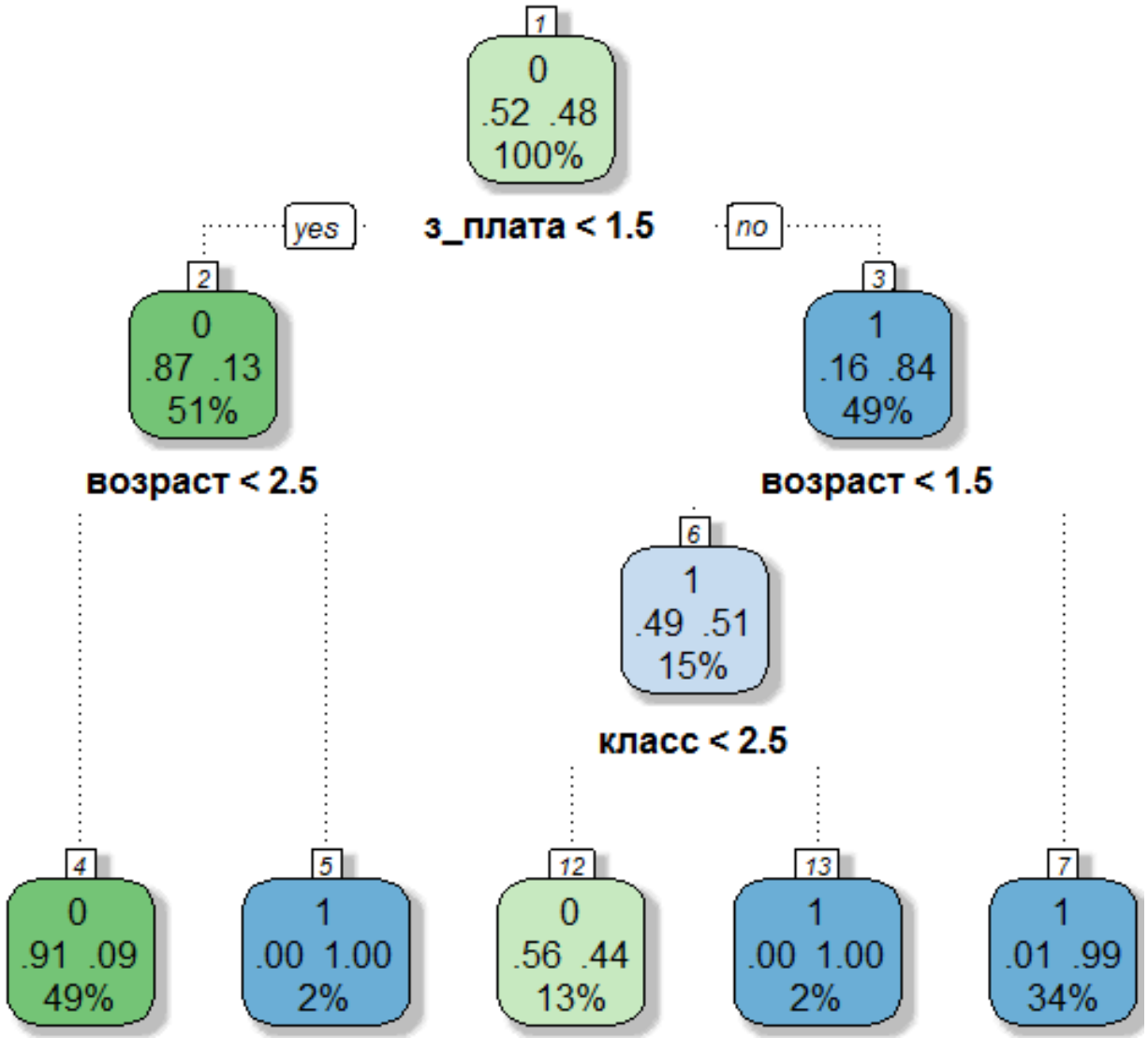
```
library(rpart.plot)
rpart.plot(credit.01.res, type=2, extra=1)
```

- `type` — где и какие делать подписи;
- `extra` — какую дополнительную информацию об узле следует отображать.



Как трактовать дробные пороговые значения?

Рисование дерева: rattle::fancyRpartPlot()



Печать результатов

```
credit.01.res
```

```
## n= 323
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 323 155 0 (0.520123839 0.479876161)
##    2) з_плата< 1.5 165  22 0 (0.866666667 0.133333333)
##      4) возраст< 2.5 158  15 0 (0.905063291 0.094936709) *
##      5) возраст>=2.5 7   0 1 (0.000000000 1.000000000) *
##    3) з_плата>=1.5 158  25 1 (0.158227848 0.841772152)
##      6) возраст< 1.5 49  24 1 (0.489795918 0.510204082)
##       12) класс< 2.5 43  19 0 (0.558139535 0.441860465) *
##       13) класс>=2.5 6   0 1 (0.000000000 1.000000000) *
##      7) возраст>=1.5 109   1 1 (0.009174312 0.990825688) *
```

Расшифровка информации

`node), split, n, loss, yval, (yprob)`

- `node` — номер узла. Номера следуют правилу: узлы-потомки узла номер n имеют номера `2*n` и `2*n+1`;
- `split` — условие, которое должно быть выполнено, чтобы наблюдение попало в узел при расщеплении узла-родителя;
- `n` — общее количество наблюдений, попавших в узел из обучающей выборки;
- `loss` — число наблюдений другого класса, попавших в узел (число ошибок);
- `yval` — какой класс приписывается наблюдению из данного узла. Определяется тем, наблюдения какого класса составляют в узле большинство;
- `(yprob)` — доля объектов из каждого класса в узле. Интерпретируется как вероятность того, что попавший в узел объект будет принадлежать классу.

Управление печатью результатов

```
print(credit.01.res, digits = 2)
```

```
## n= 323
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 323 160 0 (0.5201 0.4799)
##    2) з_плата< 1.5 165  22 0 (0.8667 0.1333)
##      4) возраст< 2.5 158  15 0 (0.9051 0.0949) *
##      5) возраст>=2.5 7   0 1 (0.0000 1.0000) *
##    3) з_плата>=1.5 158  25 1 (0.1582 0.8418)
##      6) возраст< 1.5 49  24 1 (0.4898 0.5102)
##       12) класс< 2.5 43  19 0 (0.5581 0.4419) *
##       13) класс>=2.5 6   0 1 (0.0000 1.0000) *
##      7) возраст>=1.5 109   1 1 (0.0092 0.9908) *
```

Подробный анализ результатов — summary()

```
summary(credit.01.res)
```

summary() позволяет проследить за “проигравшими” вариантами разбиений.

Фрагмент ее работы:

```
Node number 1: 323 observations,    complexity param=0.6967742
  predicted class=0  expected loss=0.4798762  P(node) =1
    class counts:   168   155
   probabilities: 0.520 0.480
  left son=2 (165 obs) right son=3 (158 obs)
  Primary splits:
      з_плата  < 1.5 to the left,  improve=81.0164500, (0 missing)
      возраст  < 1.5 to the left,  improve=73.3493500, (0 missing)
      класс    < 2.5 to the right, improve=53.7945900, (0 missing)
      кр_карта < 0.5 to the left,  improve= 0.1134861, (0 missing)
  Surrogate splits:
      класс    < 2.5 to the right, agree=0.842, adj=0.677, (0 split)
      возраст  < 1.5 to the left,  agree=0.765, adj=0.519, (0 split)
      кр_карта < 0.5 to the left,  agree=0.542, adj=0.063, (0 split)
```

Прогнозирование и подсчет ошибок

```
predicted <- predict(credit.01.res, credit.01[ ,-1],
                                              type="class")
table(credit.01[ ,1], predicted)
```

```
> predicted
 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
 1   0   0   1   1   0   1   0   0   0   0   0   0   0   0   0   0   0   0
20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
 0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
...

##
## predicted   0   1
##         0 167  34
##         1   1 121
```

Что плохо в выполненной проверке?

В скрипте `scoring.R` показано как строить деревья, если данные измеряются в номинальной и порядковой шкалах.

Прогноз кредитного рейтинга для новых данных

```
new <- data.frame(2,2,2,1)
names(new) <- names(credit.01)[-1]

predicted.new <- predict(credit.01.res, new, type="class")
```

Переобучение

Если ошибок на тестовой выборке существенно больше, чем на обучающей, то возможно мы имеем дело с переобучением.

Лечение:

- Дерево нужно сокращать, то есть обрезать ненужные ветки. Делается это настройкой параметров `rpart.control()`.
- В пакете `rpart` функция для обрезки дерева: `prune`.

Что приносит использование номинальных переменных?

Увеличение объема вычислений. Теперь нужно не просто делить упорядоченное множество на две части, а проверять условие для всех возможных разбиений неупорядоченного множества на две части.

Достоинства и недостатки

Достоинства

- результаты легко интерпретировать
- позволяет строить нелинейные модели
- не нуждается в стандартизации данных
- устраняет нерелевантные признаки

Недостатки

- Неустойчивость классификации — если слегка “пошевелить” данные, дерево классификации также изменится, т.е. деревья чувствительны к шуму в данных.
- Склонность к переобучению: большие деревья требуют обрезки.
- Границы раздела параллельны осям координат.

Лечение: вместо одного дерева использовать множество деревьев (метод “случайного леса”).

Ответ William Chen в топике “What are the disadvantages of using a decision tree for classification” на Quora

Классификация вин с помощью пакета caret

```r
library(caret)

wine <- read.table('../../data/wine.txt',
                   header=T, sep="\t")
# Проверка: что находится в наборе?
wine[1:5,]
summary(wine)
dim(wine)

# Тип вина Wine_type нужно переделать в фактор
wine$Wine_type <- as.factor(wine$Wine_type)
```

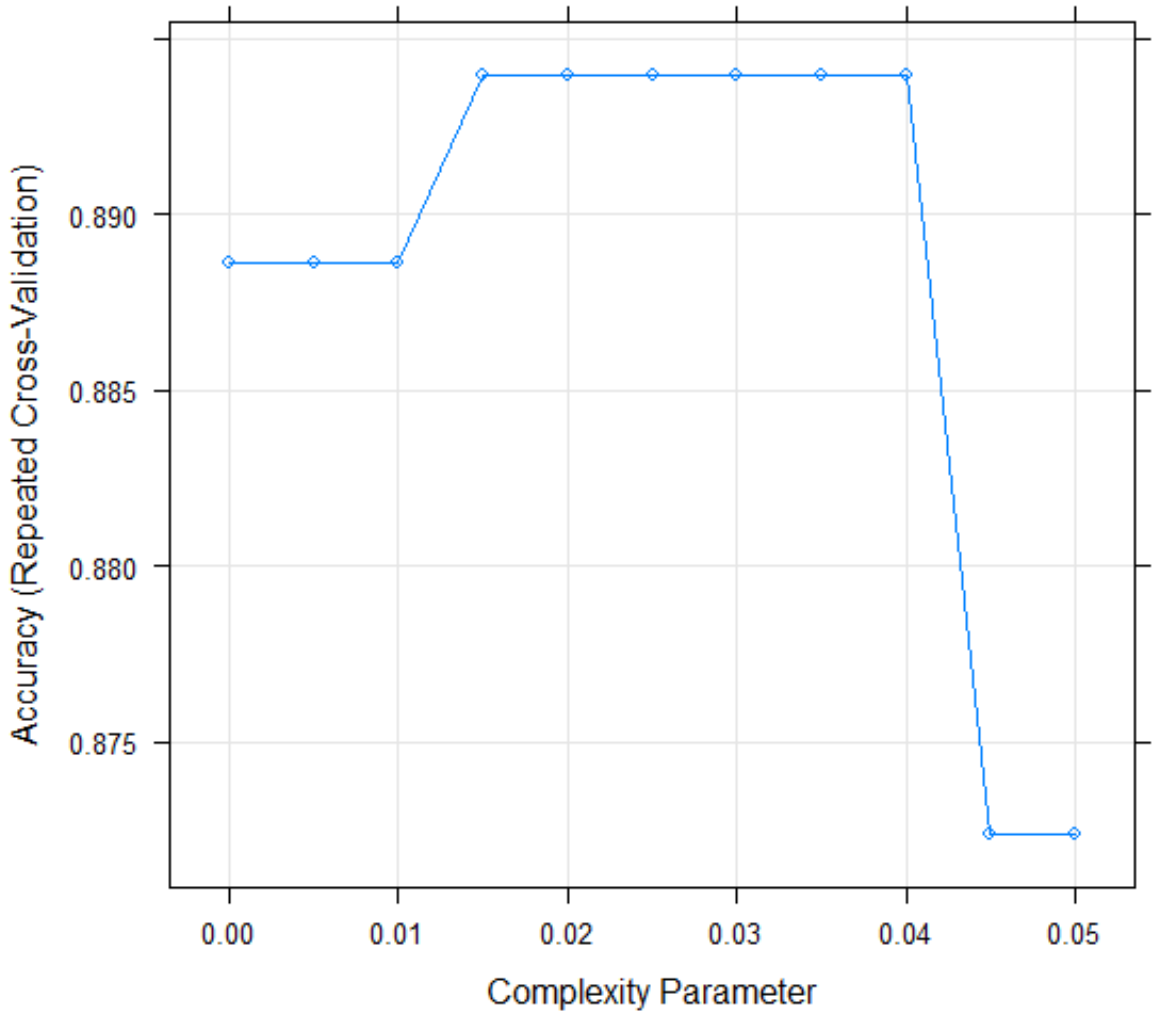
Деление выборки и управление обучением

```
set.seed(123)
ind_train <- createDataPartition(wine$Wine_type,
                                 p = 0.7, list = FALSE)
train <- wine[ind_train,]
test <- wine[-ind_train,]

ctrl <- trainControl(method = "repeatedcv",
                     number = 10, repeats = 3)

rpart_grid <- expand.grid(cp=seq(0, 0.05, 0.005))
```

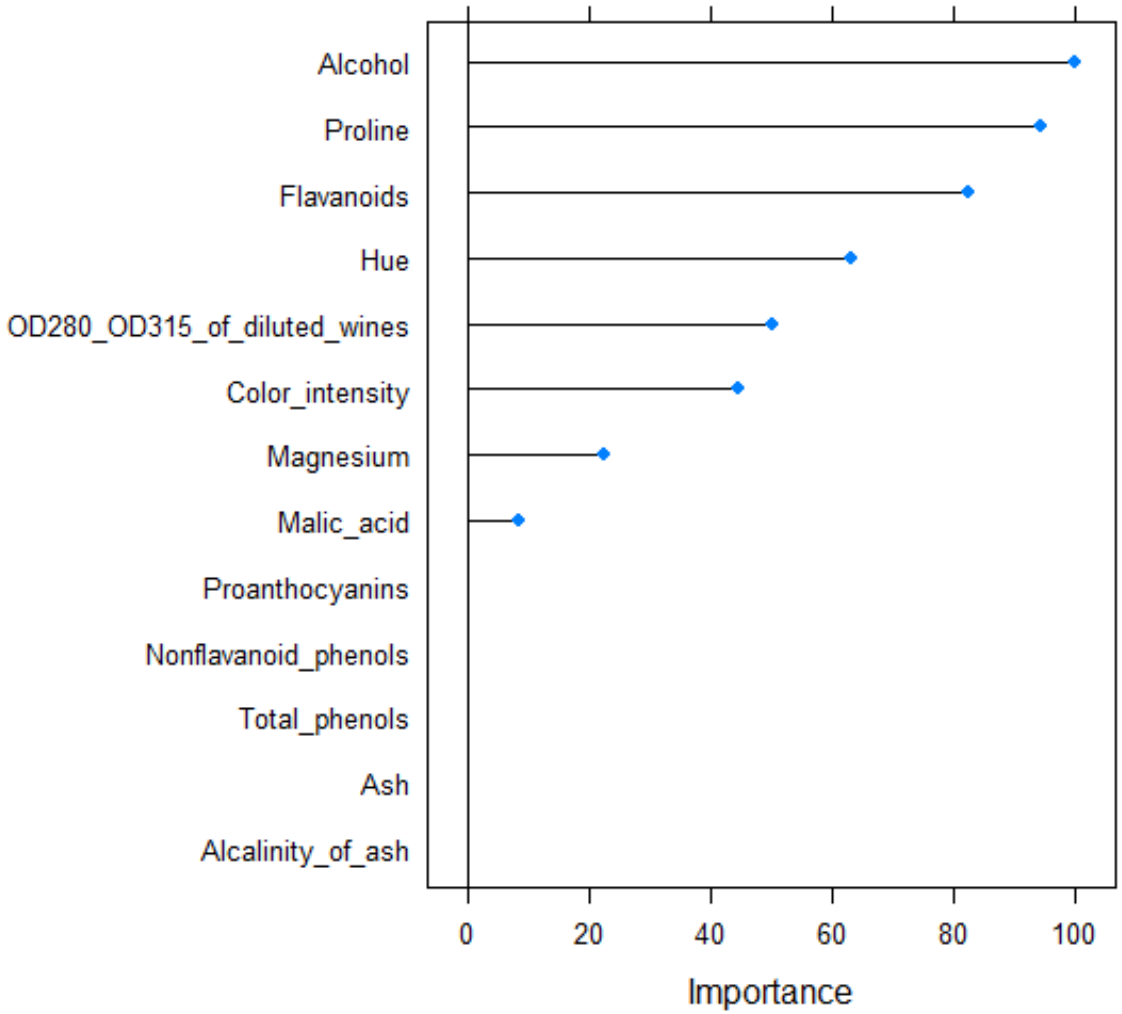
Обучение

```
# Используем information gain вместо gini
# для измерения загрязненности
cart_fit <- train(Wine_type ~., data = train,
                  method = 'rpart',
                  trControl = ctrl,
                  #preProcess = c('center', 'scale'),
                  parms = list(split='information'),
                  tuneGrid = rpart_grid
                  )
```

Настройка параметров



Влиятельность переменных



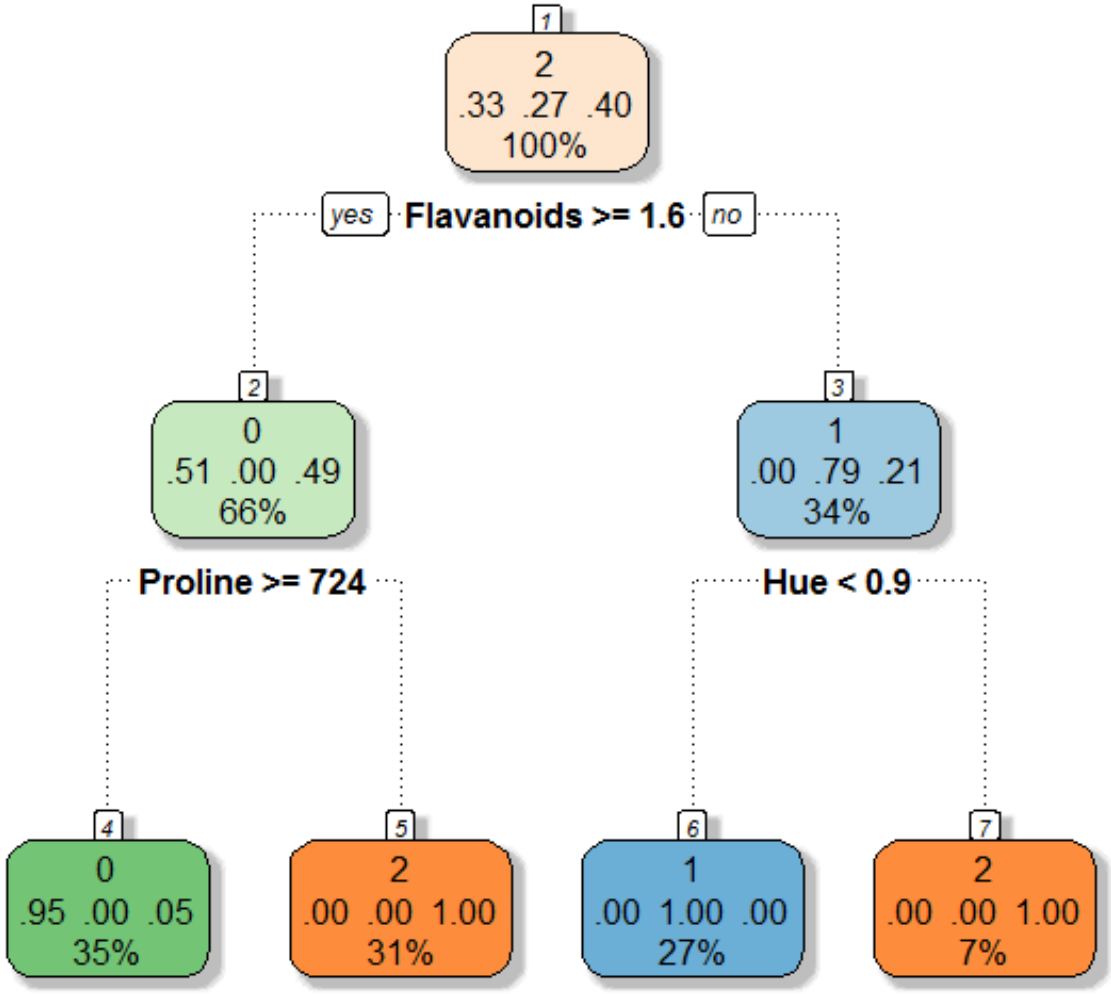
Контроль ошибок

```
confusionMatrix(prediction, test$Wine_type)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1  2
         0 16  0  2
         1  0 12  1
         2  1  2 18

Overall Statistics

               Accuracy : 0.8846
                 95% CI : (0.7656, 0.9565)
    No Information Rate : 0.4038
    P-Value [Acc > NIR] : 7.681e-13
```

Построим дерево с библиотекой rattle

```
library(rattle)
fancyRpartPlot(cart_fit$finalModel)
```



Дополнительная информация

- *Terry M. Therneau, Elizabeth J. Atkinson.* An Introduction to Recursive Partitioning Using the RPART Routines — в лекции мы ориентировались на это руководство по `rpart`.
- *Arthur Charpentier.* Classification From Scratch, Trees — крачайшее руководство. Читайте про обрезку деревьев и оценку влиятельности признаков.
- *Max Kuhn.* The caret Package — книга о пакете `caret`, написанная его разработчиком.