Деревья решений

1.1 Предварительные сведения по деревьям решений

Деревья решений позволяют решать как задачи классификации (когда отклик является качественной переменной), в этом случае деревья называются деревьями классификации, так и задачи регрессии (когда отклик – количественная переменная), в этом случае говорят о регрессионных деревьях. В данном пособии будут рассматриваться только деревья классификации. В большинстве случаев методы, основанные на построении деревьев решений, не требуют предположений о статистическом распределении значений признаков.

В основе деревьев решений лежат правила вида «если …то…», которые могут быть сформулированы на естественном языке.

Термины, используемые в деревьях решений приведены в таблице 1.

Таблица 1 – Термины, используемые в деревьях решений

Термин	Значение
объект	пример, шаблон, наблюдение, запись
атрибут	признак, описывающий классифицируемые объекты, входное поле
метка класса	целевая переменная, выходное поле, отклик
узел	внутренний узел дерева, здесь содержатся правила, с помощью которых производится проверка атрибутов и множество объектов в данном узле разбивается на подмножества
лист	конечный узел дерева, узел решения, в нем содержатся

	подмножества объектов, ассоциированные с классами. В листе не производится проверка атрибутов и не производится разбиение на подмножества

В основе работы деревьев решений лежит процесс рекурсивного[1] разбиения исходного множества наблюдений или объектов на подмножества, ассоциированные с классами. Разбиение производится с помощью решающих правил, в которых осуществляется проверка значений атрибутов по заданному условию.

Например, требуется предсказать возврат или невозврат кредита с помощью набора решающих правил на основе единственного атрибута **Возраст клиента** (исходные данные приведены в таблице 2).

Таблица 2 – Выборка клиентов по возрасту и факту возврата (невозврата кредита)

Возраст клиента, лет	Возврат кредита
22	Нет
23	Нет
24	Нет
28	Нет
32	Да
34	Да
45	Да
46	Да
47	Да
51	Нет
56	Нет
58	Нет
62	Да

Возьмем некоторое значение возрастного порога, например, 50 лет и разобьем исходное множество на два подмножества в соответствие с условием **Возраст >50**. Прямоугольниками обозначим клиентов, отдавших кредит (принадлежащих классу С1), кругами – не отдавших (класс С2)

[1] Рекурсивными называются алгоритмы, которые работают в пошаговом режиме, при этом на каждом последующем шаге используются результаты, полученные на предыдущем шаге

(внутри фигур указывается возраст). Выбор возрастного порога 50 не позволил получить подмножества, содержащие только объекты одного класса, поэтому для решения задачи применяется разбиение уже полученных подмножеств (рисунок 1).

Для подмножества 1 используем порог 60 лет, а для подмножества 2 – 30 лет. Результаты повторного разбиения на рисунке 2. Из рисунка 2 видно, что исходное множество удалось разбить на чистые подмножества, содержащие только наблюдения одного класса. Дерево, реализующее эту процедуру, представлено на рисунке 3. Подмножества 4 и 6 ассоциированы с добросовестными клиентами, а 3 и 5 – с классом клиентов, не отдавших кредит. Применяя построенную модель к новым клиентам, можно предсказать риск, связанный с выдачей им кредита, в зависимости от того, в какое подмножество модель поместит соответствующую запись.

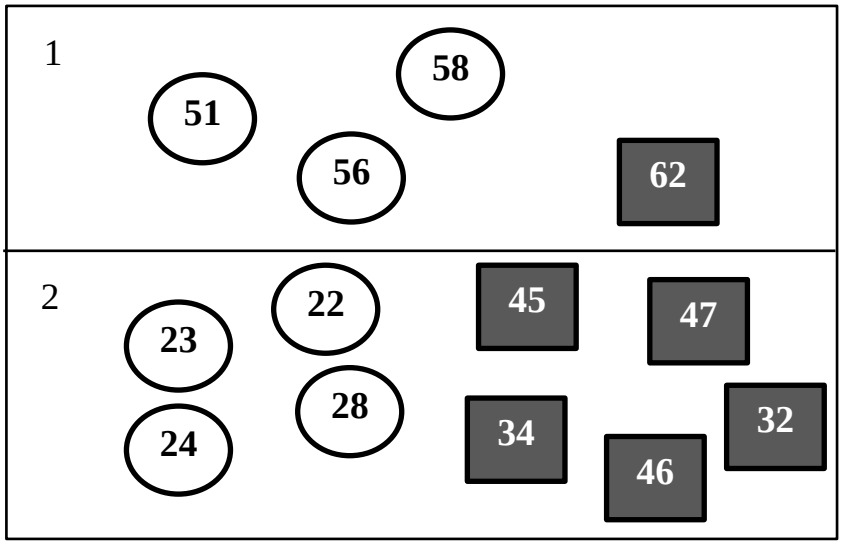


Рисунок 1 – Разбиение на два подмножества исходного множества клиентов

Из рисунка 3 видно, что в состав дерева решений входят два вида объектов – узлы и листья, объекты с номерами 1 и 2 – узлы, а с номерами 3,4,5,6 – листья. Основным отличием листа от узла является то, что в листе не производится проверка, разбивающая ассоциированное с ним подмножество, и соответственно, нет ветвления. Узлы и листья,

подчиненные узлу более высокого иерархического уровня, называются потомками, а тот узел, по отношению к ним – предком.

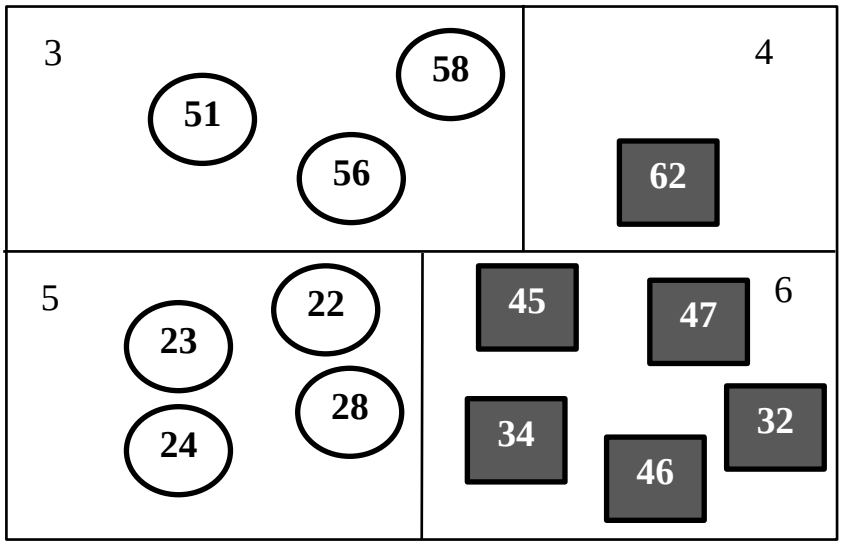


Рисунок 2 – Последующее разбиение множества клиентов на подмножества

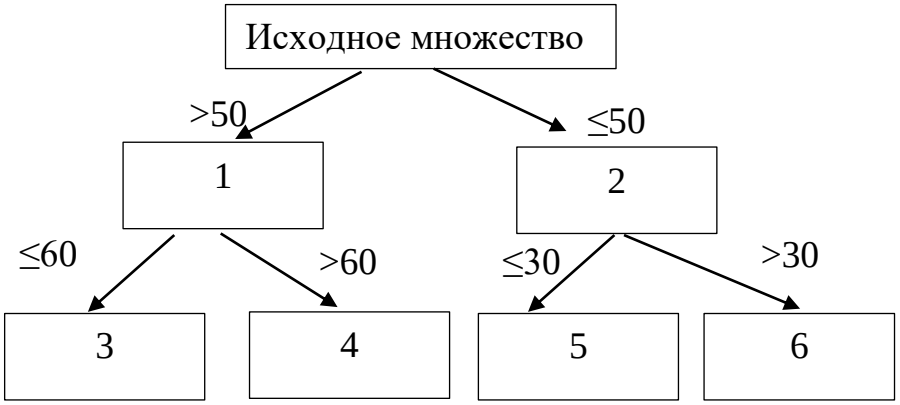


Рисунок 3 – Итоговое дерево решений

В реальности задача классификации по одному атрибуту встречается редко, для эффективного разделения на классы требуется несколько атрибутов. Например, кроме возраста, в рассмотренном примере, атрибутами могут быть доход, образование и т.д.

В процессе построения дерева формируются решающие правила и для каждого из них создается узел. Для каждого узла нужно выбрать атрибут, по которому будет производится дальнейшее ветвление. От того, насколько

удачно он выбран, зависит классифицирующая сила правила. Очередной выбранный атрибут должен обеспечивать наилучшее разбиение в узле. Наилучшим разбиением считается то, которое позволяет классифицировать наибольшее число примеров и создавать максимально чистые подмножества, в которых примесь объектов другого класса (не ассоциированного с данным узлом или листом) минимальна. Метод, в соответствии с которым осуществляется выбор атрибута ветвления на каждом шаге, называется ***алгоритмом построения дерева решений***.

Алгоритмы построения деревьев решений являются «**жадными**»[2]. Они развиваются путем рассмотрения каждого входного атрибута по очереди и оценивают увеличение чистоты, которое обеспечило разбиение с помощью данного атрибута.

Метки класса (целевая переменная) объектов могут быть как ***числовыми*** (непрерывными), так и ***категориальными*** (дискретными). В зависимости от того, какого типа для узла является метка класса, используются разные критерии разбиения. На рисунке 4 показаны виды критериев разбиения в зависимости от того, непрерывной или категориальной является целевая переменная.

[2]«Жадными» называются алгоритмы, которые на каждом шаге делают локально оптимальный выбор, допуская, что итоговое решение также окажется оптимальным. Для большого числа алгоритмических задач «жадные» алгоритмы действительно дают оптимальное решение. При построении деревьев решений «жадность алгоритма »заключается в том, что для каждого узла ищется оптимальный атрибут разбиения и предполагается, что дерево в целом тоже будет оптимальным.

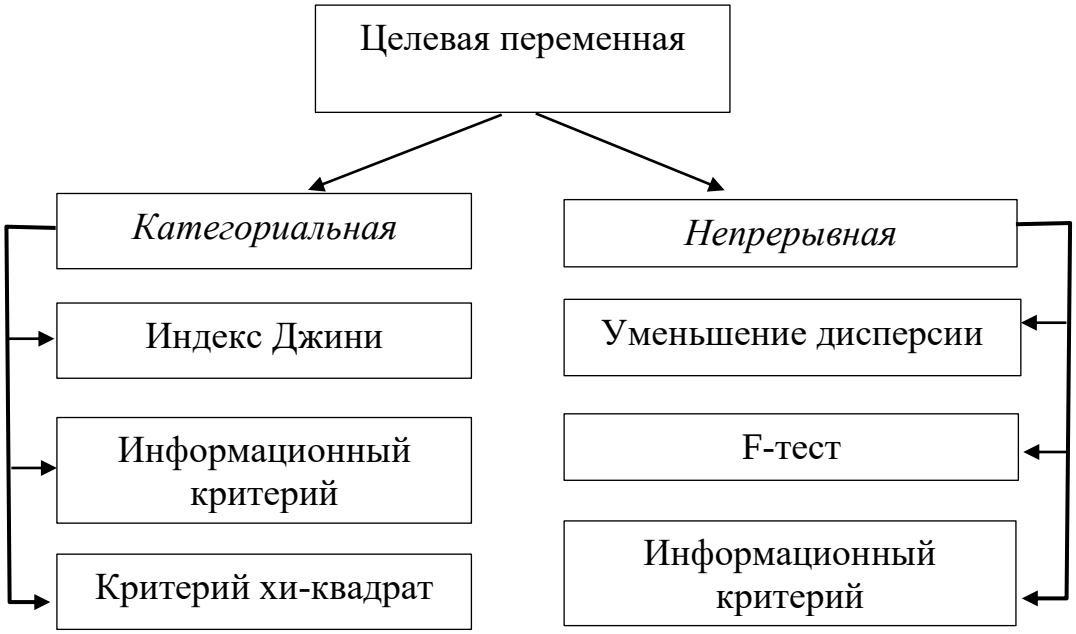


Рисунок 4

В настоящее время разработано значительное число алгоритмов обучения деревьев решений: ID3, CART, C4.5, C5.0, NewId, ITrule, CHAID, CN2 и т.д. Но наибольшее распространение и популярность получили следующие:

- **ID3** (Iterative Dichotomizer 3) – алгоритм позволяет работать только с дискретной целевой переменной, поэтому деревья решений, построенные с помощью данного алгоритма, являются классифицирующими. Число потомков в узле дерева не ограничено. Не может работать с пропущенными данными.
- **C5.0** – усовершенствованная версия алгоритмов ID3 и C4.5, в которую добавлена возможность работы с пропущенными значениями атрибутов.
- **CART** (Classification and Regression Tree) – алгоритм обучения деревьев решений, позволяющий использовать как дискретную, так и непрерывную целевую переменную, то есть решать как задачи классификации, так и регрессии. Алгоритм строит деревья, которые в каждом узле имеют только два потомка.

В алгоритмах ID3, C4.5, C5.0 для определения атрибута, по которому должно вестись разветвление, используется понятие энтропии. Эта мера, основана на исследовании разнообразия совокупности.

Например, пусть задано обучающее множество, содержащее K объектов (примеров) и N классов, $C = \{C_1, C_2, \ldots, C_N\}$ – множество меток классов, $A = \{A_1, A_2, \ldots, A_m\}$ – множество атрибутов.

Пусть $f(C_j, T)$ - количество объектов из некоторого множества T, принадлежащих одному классу C_j. Тогда вероятность того, что случайно выбранный объект из множества T принадлежит классу C_j:

$$P_j = \frac{f(C_j, T)}{K}. \quad (1)$$

В этом случае энтропия множества T имеет вид

$$H(T) = -\sum_{j=1}^{N} P_j \log_2(P_j). \quad (2)$$

Если количества объектов, принадлежащих разным классам, равные, то энтропия будет максимальной, если во множестве T все элементы принадлежат одному классу, то энтропия будет минимальной и равна 0. И множество T будет максимально чистым.

По каждому атрибуту A_i, $i = \overline{1,m}$ можно разбить множество T на подмножества T_1, T_2, …, T_n.

Условная энтропия множества объектов при рассматриваемом атрибуте A_i есть

$$H(T / A_i) = \sum_{j=1}^{n} \frac{K_j}{K} H(T_j), \quad (3)$$

где K_j – количество объектов, попавших в подмножество T_j, $j = \overline{1,n}$.

Далее для каждого из атрибутов вычисляется прирост информации

$$I(A_i) = H(T) - H(T / A_i). \quad (4)$$

Для ветвления выбирается тот атрибут, при котором прирост информации больше.

Пример 1 **(предсказание кредитного риска потенциального банка (задача кредитного скоринга))**

Имеется 8 объектов (потенциальных клиентов), каждый из которых характеризуется атрибутами: сбережения (низкие, средние, высокие), наличие других активов (недвижимость, автомобиль и т.д.), годовой доход (тыс. у.е.). Исходные данные представлены в таблице 1. Первые два атрибута – категориальные, третий – непрерывный. Требуется разбить совокупность объектов (клиентов) на два класса: с низким кредитным риском (C_1) и высоким (C_2).

Таблица 3 – Исходные данные для примера 1

№ клиента	Сбережения	Другие активы	Годовой доход, тыс. у.е.	Кредитный риск
1	Средние	Высокие	75	Низкий
2	Низкие	Низкие	50	Высокий
3	Высокие	Средние	25	Высокий
4	Средние	Средние	50	Низкий
5	Низкие	Средние	100	Низкий
6	Высокие	Высокие	25	Низкий
7	Низкие	Низкие	25	Высокий
8	Средние	Средние	75	Низкий

Решение: здесь $K = 8, N = 2$ (два класса C_1 – низкий кредитный риск и C_2 – высокий кредитный риск)

Определим энтропию для исходного множества объектов:

$$H(T) = -\frac{5}{8} \cdot \log_2 \frac{5}{8} - \frac{3}{8} \cdot \log_2 \frac{3}{8} = 0.954.$$

Выберем в качестве атрибута Сбережения (A_1), разобьем сначала исходное множество примеров по признаку Сбережения= «Низкие» и Сбережения = «Средние или Высокие».

В первое подмножество T_1 попадают примеры 2, 5, 7 с принадлежностью к классам C_2, C_1, C_2.

Во второе T_2 попадают примеры 1, 3, 4, 6, 8 с принадлежностью к классам C_1, C_2, C_1, C_1, C_1

Тогда

$$H(T/A_1^1)=\frac{3}{8}\cdot H(T_1)+\frac{5}{8}\cdot H(T_2)=\frac{3}{8}\cdot\left(-\frac{1}{3}\log_2\frac{1}{3}-\frac{2}{3}\log_2\frac{2}{3}\right)+$$

$$+\frac{5}{8}\cdot\left(-\frac{4}{5}\log_2\frac{4}{5}-\frac{1}{5}\log_2\frac{1}{5}\right)=0.796.$$

Разобьем теперь исходное множество примеров по тому же признаку Сбережения, но к T_1 отнесем множество примеров со Сбережениями= «Низкие», к T_2: Сбережения = «Средние», к T_3: Сбережения = «Высокие».

В первое подмножество T_1 попадают примеры 2, 5, 7 с принадлежностью к классам C_2, C_1, C_2.

Во второе T_2 попадают примеры 1, 4, 8 с принадлежностью к классам C_1, C_1, C_1

В третье T_3 попадают примеры 3 и 6 с принадлежностью к классам C_2, C_1

Тогда

$$H(T/A_1^2)=\frac{3}{8}\cdot H(T_1)+\frac{3}{8}\cdot H(T_2)+\frac{2}{8}\cdot H(T_3)=$$

$$=\frac{3}{8}\cdot\left(-\frac{1}{3}\log_2\frac{1}{3}-\frac{2}{3}\log_2\frac{2}{3}\right)+\frac{2}{8}\cdot\left(-\frac{3}{3}\log_2 1\right)+\frac{2}{8}\cdot\left(-\frac{1}{2}\log_2\frac{1}{2}-\frac{1}{2}\log_2\frac{1}{2}\right)=$$

$$=0{,}594.$$

В первом случае $I(A_1^1)=0{,}954-0{,}796=0{,}158$,

во втором случае $I(A_1^2)=0{,}954-0{,}594=0{,}36$.

Выбираем второе разбиение, так как прирост информации больше.

Для оценки качества классификации построенного дерева применяется два показателя – поддержка и достоверность.

Поддержка (S) – отношение числа правильно классифицированных примеров в данном узле или листе к общему числу попавших в него примеров, $0\le S\le 1$.

Достоверность – отношение числа правильно классифицированных примеров к числу ошибочно классифицированных. Чем больше число правильно классифицированных примеров в узле, тем выше достоверность.

Поддержка и достоверность могут использоваться в качестве параметров построения дерева решений. Например, можно задать, что разбиение должно производиться до тех пор, пока в узле не будет достигнут заданный порог поддержки.

Алгоритм CART

Деревья решений, построенные с помощью алгоритма CART, являются бинарными, то есть содержат только два потомка в каждом узле. Для определения атрибута, по которому должно вестись разветвление, используется понятие индекса Джини. Мера Джини показывает меру чистоты узла (преобладание примеров, принадлежащих одному какому-то классу).

Пусть задано обучающее множество, содержащее K примеров и N классов. Тогда мера Джини будет равна вероятности того, что два случайным образом выбранных объекта будут принадлежать одному классу. Поскольку классов N, то эта вероятность равна

$$\sum_{j=1}^{N} p_j^2,$$

где p_j – вероятность принадлежности одного примера классу j, $j = \overline{1,N}$ (ее можно оценить с помощью доли).

Индекс Джини – вероятность того, что два случайным образом выбранных объекта будут принадлежать разным классам. Он будет определяться по формуле

$$G = 1 - \sum_{j=1}^{N} p_j^2 .$$

Индекс Джини $0 \le G \le 1 - 1/N$, чем ближе он к $1 - 1/N$, тем более «загрязнена» совокупность, чем ближе к 0 тем больше «чистота», то есть тем

большее преобладание объектов, относящихся к одному определенному классу.

Если имеется всего два класса, то индекс Джини примет вид

$$G = 1 - p_1^2 - p_2^2 = 1 - p_1^2 - (1 - p_1)^2 = 1 - p_1^2 - 1 + 2p_1 - p_1^2 = 2p_1 - 2p_1^2 = \\ = 2p_1(1 - p_1)$$

График индекса Джини в зависимости от вероятности принадлежности к первому классу – парабола, с вершиной в точке (1/2; 1/2) (см. рисунок 5)

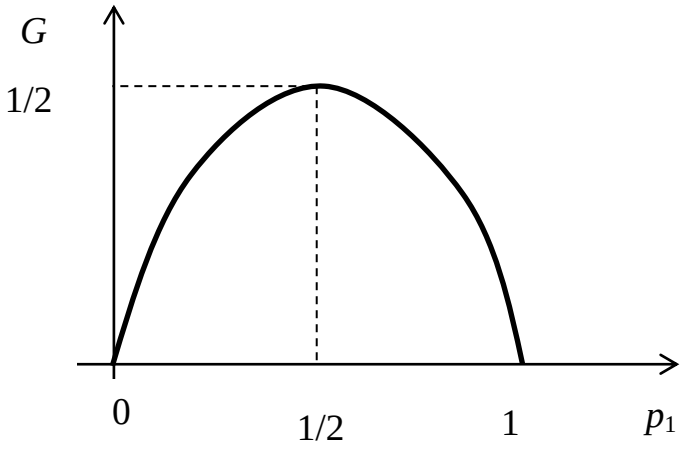


Рисунок 5

На практике для оценки эффективности разбиения, полученного на основе конкретного атрибута, часто вместо индекса Джини используется следующий показатель:

$$Q(s/t) = 2P_L P_R \sum_{j=1}^{N} \left| P(j/t_L) - P(j/t_R) \right|,$$

где t_L и t_R – левый и правый потомки узла t соответственно;

$P_L = \frac{K_L}{K}$ – отношение числа объектов в левом потомке узла t к общему числу объектов;

$P_R = \frac{K_R}{K}$ – отношение числа объектов в правом потомке узла t к общему числу объектов;

$P(j/t_L)=\frac{K_j^L}{K_t}$ – отношение числа объектов j-го класса в t_L к общему числу объектов в t_L;

$P(j/t_R)=\frac{K_j^R}{K_t}$ – отношение числа объектов j-го класса в t_R к общему числу объектов в t_R.

Наилучшим разбиением в узле t будет то, которое максимизирует показатель $Q(s/t)$.

Рассмотрим алгоритм CART на примере предсказания кредитного риска потенциального банка (задача кредитного скоринга).

Пример 2. Воспользуемся исходными данными из таблицы 3. Требуется, как и раньше, разбить совокупность объектов (клиентов) на два класса: с низким кредитным риском (C_1) и высоким (C_2).

Решение: Все восемь примеров обучающего множества поступают в корневой узел дерева. Поскольку алгоритм CART создает только бинарные разбиения, возможные кандидаты, которые будут оцениваться на начальном этапе, представлены в таблице 4.

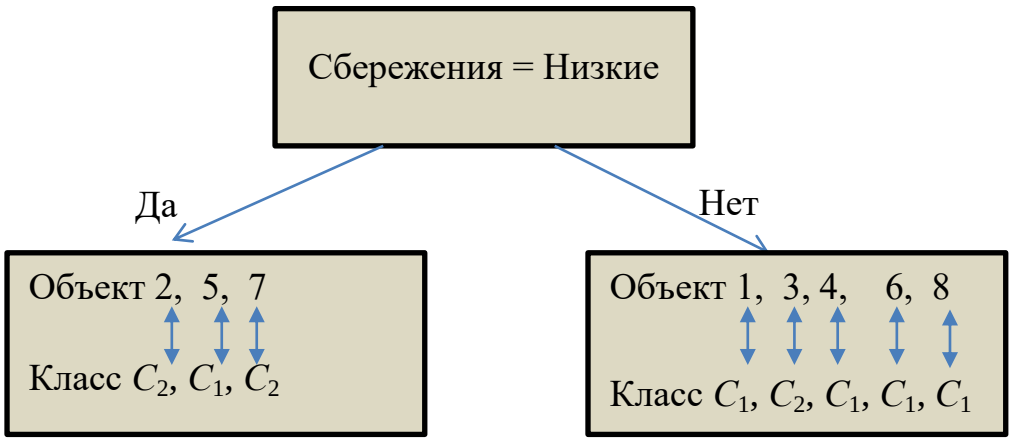
Непрерывный атрибут Доход был предварительно квантован с помощью порогов 25, 50 и 75 тыс.

Таблица 4 Разбиение исходных данных

Разбиение	**Левый потомок, t_L**	**Правый потомок, t_R**
1	Сбережения=Низкие	Сбережения∈{Средние, Высокие}
2	Сбережения=Средние	Сбережения∈{Низкие, Высокие}
3	Сбережения=Высокие	Сбережения∈{Низкие, Средние}
4	Активы=Низкие	Активы∈{Средние, Высокие}
5	Активы=Средние	Активы∈{Низкие, Высокие}
6	Активы=Высокие	Активы∈{Низкие, Средние}
7	Доход<=25	Доход >25
8	Доход<=50	Доход >50
9	Доход<=75	Доход >75

Найдем для каждого разбиения значение $Q(s/t)$.

Например, для разбиения 1 по значению атрибута Сбережения=Низкие:



Получаем, что потомок t_L содержит 3 объекта, то есть K_L=3, а потомок t_R содержит 5 объектов из 8.

Тогда $P_L = \frac{3}{8} = 0{,}375$, а $P_R = \frac{5}{8} = 0{,}625$.

В t_L содержится один объект, принадлежащий классу C_1 и два объекта (2 и 7), принадлежащих классу C_2, тогда

$$P(j=1/t_L) = \frac{1}{3} = 0{,}333,\ P(j=2/t_L) = \frac{2}{3} = 0{,}667.$$

В t_R содержится четыре объекта, принадлежащий классу C_1 (это 1, 4, 6 и 8-й объекты) и один объект (3-й), принадлежащий классу C_2, тогда

$$P(j=1/t_R) = \frac{4}{5} = 0.8,\ P(j=2/t_R) = \frac{1}{5} = 0.2.$$

Тогда

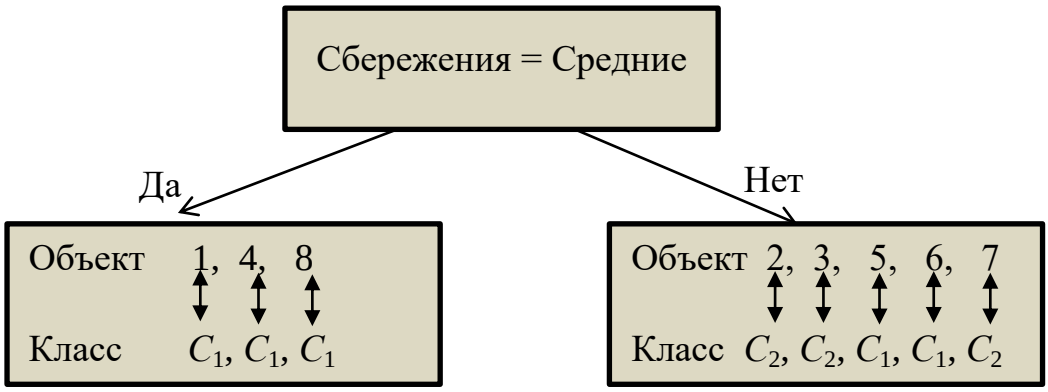
$$W(s/t) = \sum_{j=1}^{N} \left| P(j/t_L) - P(j/t_R) \right| = \sum_{j=1}^{2} \left| P(j/t_L) - P(j/t_R) \right| =$$

$$= \left|0{,}333 - 0.8\right| + \left|0{,}667 - 0.2\right| = 0{,}934;$$

$$2P_L P_R = 2 \cdot 0{,}375 \cdot 0{,}667 = 0{,}46875;$$

$$Q(s/t) = 2P_L P_R \sum_{j=1}^{N} \left| P(j/t_L) - P(j/t_R) \right| = 0{,}46875 \cdot 0{,}934 = 0{,}4378.$$

Для разбиения 2 по значению атрибута Сбережения=Средние:



Получаем, что потомок t_L содержит 3 объекта, то есть K_L=3, а потомок t_R содержит 5 объектов из 8.

Тогда $P_L = \frac{3}{8} = 0{,}375$, а $P_R = \frac{5}{8} = 0{,}625$.

В t_L содержится три объекта, принадлежащих классу C_1, тогда

$$P(j=1/t_L) = \frac{3}{3} = 1,\ P(j=2/t_L) = 0.$$

В t_R содержится два объекта, принадлежащих классу C_1 (это 5 и 6-й объекты) и три объекта, принадлежащих классу C_2, тогда

$$P(j=1/t_R) = \frac{2}{5} = 0.4,\ P(j=2/t_R) = \frac{3}{5} = 0.6.$$

$$W(s/t) = \sum_{j=1}^{N} \left| P(j/t_L) - P(j/t_R) \right| = \sum_{j=1}^{2} \left| P(j/t_L) - P(j/t_R) \right| =$$

$$= \left|1-0.4\right| + \left|0-0.6\right| = 1{,}2;$$

$$2P_L P_R = 2 \cdot 0{,}375 \cdot 0{,}667 = 0{,}46875;$$

$$Q(s/t) = 2P_L P_R \sum_{j=1}^{N} \left| P(j/t_L) - P(j/t_R) \right| = 0{,}46875 \cdot 1{,}2 = 0{,}5625$$

и т.д.

Занесем рассчитанные значения в таблицу 5.

Таблица 5 –

№	P_L	P_R	$P(j=1/t_L)$		$P(j=1/t_R)$		$2P_LP_R$	$W(s/t)$	$Q(s/t)$
			Низкий	Высокий	Низкий	Высокий			
1	0,375	0,625	0,333	0,667	0,8	0,2	0,46875	0,934	0,4378
2	0,375	0,625	1	0	0,4	0,6	0,46875	1,2	0,5625
3	0,25	0,75	0,5	0,5	0,667	0,333	0,375	0,334	0,1253
4	0,25	0,75	0	1	0,833	0,167	0,375	1,667	0,6248
5	0,5	0,5	0,75	0,25	0,5	0,5	0,5	0,5	0,25
6	0,25	0,75	1	0	0,5	0,5	0,375	1	0,375
7	0,375	0,625	0,333	0,667	0,8	0,2	0,46875	0,934	0,4378
8	0,625	0,375	0,4	0,6	1	0	0,46875	1,2	0,5625
9	0,875	0,125	0,571	0,429	1	0	0,21875	0,858	0,1877

Из таблицы 5 видно, что максимальное значение $Q(s/t)$=0,6248 достигается при 4-м разбиении, поэтому для начального разбиения CART выберет условие, являются ли активы клиента низкими. Тогда в результате разбиения будут созданы два потомка, в одном окажутся записи, в которых Активы принимают значения Низкие, во втором записи – в которых атрибут Активы принимает значения Высокие и Средние. Полученное в результате данного разбиения дерево представлено на рисунке 6 .

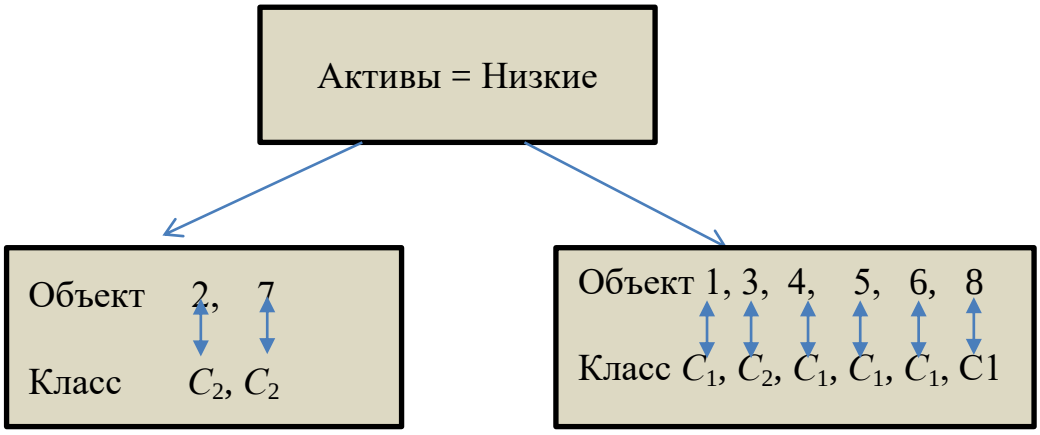


Рисунок 6

Обе записи, в которых активы клиента являются низкими и по этой причине оказавшиеся в левом узле, принадлежат одному классу C_2 (с высоким риском), поэтому этот узел является чистым и становится листом. Дальнейшее разбиение по данной ветви производиться не будет. Записи в

правом узле относятся к разным классам. Потребуется дальнейшее разбиение. Поэтому опять надо будет рассчитать значение $Q(s/t)$, при этом разбиение 4 мы далее не рассматриваем. Не рассматриваем также объекты 2 и 7.

Задание: самостоятельно продолжить построение дерева.

Замечание: поскольку алгоритм построения дерева решений является «жадным», то дерево решений может расти бесконечно, выбирая признаки разделения и ветвясь на все меньшие сегменты, пока не будут полностью классифицированы все примеры или пока в алгоритме не исчерпаются возможности для разделения. Но, когда дерево становится слишком большим, многие принимаемые им решения оказываются чрезмерно конкретными, и модель становится переобученной. Процесс сокращения дерева решений означает уменьшение его размера таким образом, чтобы дерево лучше обобщало новые данные.

Одним из решений этой проблемы является остановка роста дерева при достижении определенного числа точек принятия решений или когда узлы принятия решений будут содержать лишь небольшое количество примеров. Это называется ранней остановкой или ранним сокращением дерева решений (pre pruning). Поскольку при построении дерева алгоритм стремится избегать ненужной работы, это привлекательная стратегия. Одним из недостатков такого подхода является то, что не существует способа узнать, не будут ли при этом пропущены неявные, но важные паттерны, которые были бы изучены, если бы дерево выросло до большего размера.

Альтернатива, называемая поздним сокращением (post pruning), подразумевает построение намеренно слишком большого дерева и сокращение концевых узлов, чтобы уменьшить размер дерева до приемлемого. Такой подход часто оказывается более эффективным, чем ранняя остановка, поскольку бывает довольно сложно определить оптимальную глубину дерева решений заранее, без предварительного

увеличения. Последующее сокращение дерева позволяет алгоритму гарантировать, что были обнаружены все важные структуры данных.

В алгоритме CART методика отсечения ветвей следующая. Отсекаются ветви, имеющие наименьшую классификационную способность, то есть ветви, которые классифицируют малое число примеров в расчете на лист. Используется так называемая скорректированная ошибка, которая увеличивает показатель, отражающий количество ошибок классификации каждого узла на обучающем множестве, налагая штраф, учитывающий сложность дерева на основе количества листьев в нем. Она используется для определения самых «слабых» ветвей, то есть тех, в которых количество неправильно классифицированных примеров недостаточно мало, чтобы компенсировать штраф. Затем такие ветви помечаются как удаляемые. Скорректированая ошибка рассчитывается по формуле

$$R = E + \alpha \cdot |T|,$$

где $|T|$ – число листов дерева, E – ошибка классификации дерева, равная отношению числа неправильно классифицированных примеров к числу примеров в обучающей выборке, α – корректирующий параметр.

Решение задачи классификации с помощью деревьев решений в R

Рассмотрим инструменты, с помощью которых можно строить деревья решений в *R*.

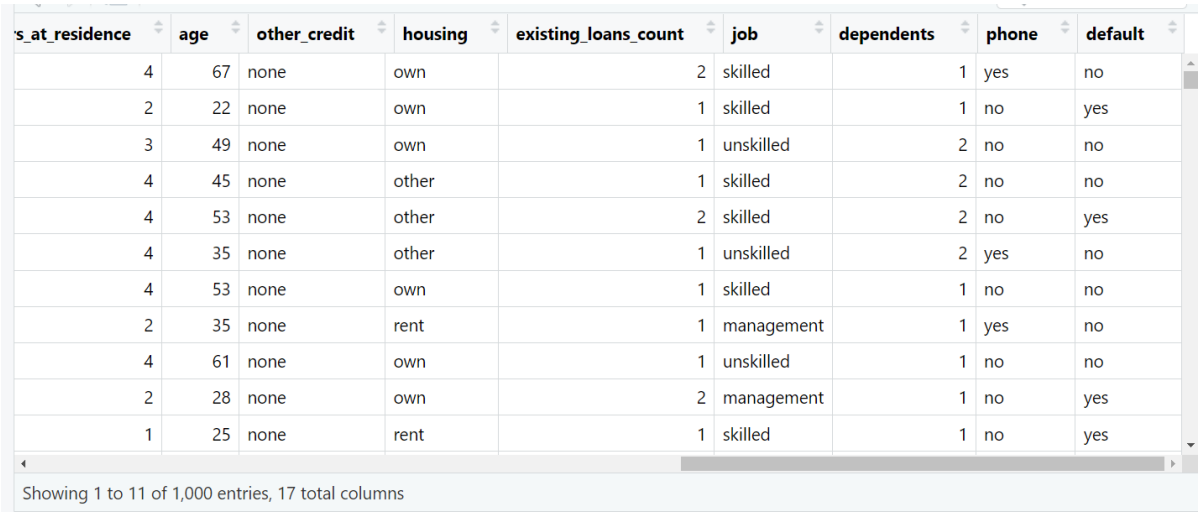
Рассмотрим задачу скоринга. Цель – выявить факторы, связанные с более высоким риском невозвращения кредита. Для этого необходимо получить данные о большом количестве предыдущих банковских кредитов и информацию о получателях этих кредитов. Данные с такими характеристиками собраны в файле credits.csv (ссылка для скачивания: Machine-Learning-with-R-datasets/credit.csv at master stedy/Machine-Learning-with-R-datasets GitHub). Этот набор данных содержит информацию о кредитах, полученных от кредитного агентства в Германии.

Загрузите этот файл в R-studio с помощью команды:

credit<-read.csv("Указываем путь к файлу/credits.csv", stringsAsFactors=TRUE)

В результате этот набор данных будет храниться в датафрейме с именем credit.

Фрагмент таблицы выглядит как на рисунке 7.

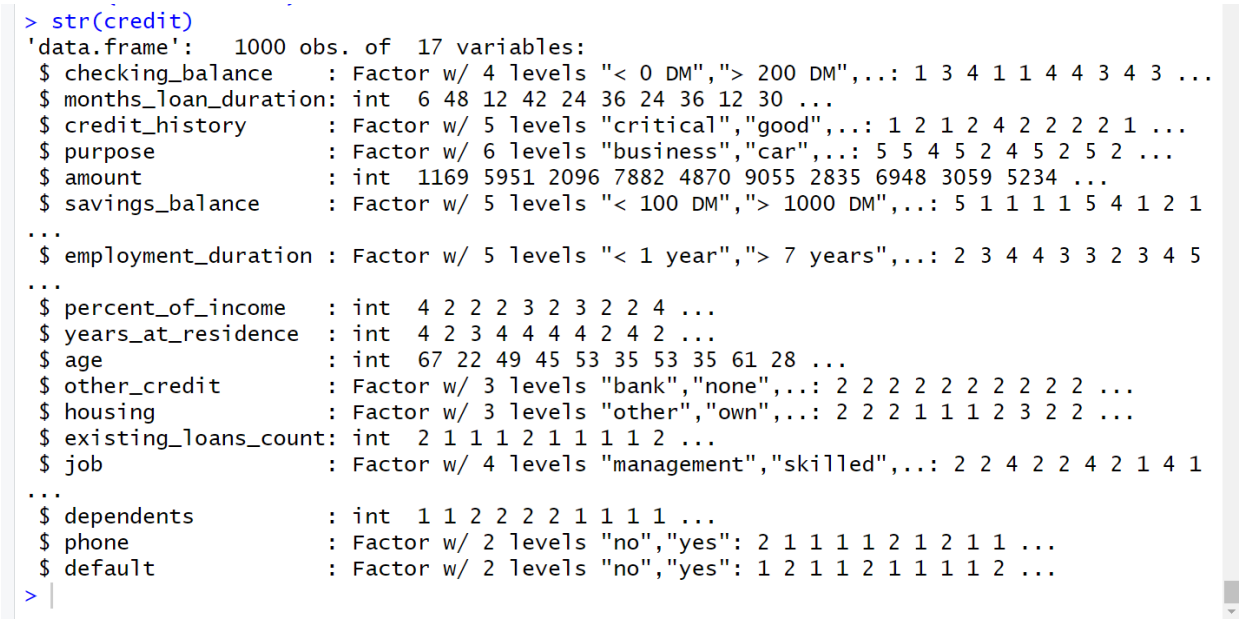


s_at_residence	age	other_credit	housing	existing_loans_count	job	dependents	phone	default
4	67	none	own	2	skilled	1	yes	no
2	22	none	own	1	skilled	1	no	yes
3	49	none	own	1	unskilled	2	no	no
4	45	none	other	1	skilled	2	no	no
4	53	none	other	2	skilled	2	no	yes
4	35	none	other	1	unskilled	2	yes	no
4	53	none	own	1	skilled	1	no	no
2	35	none	rent	1	management	1	yes	no
4	61	none	own	1	unskilled	1	no	no
2	28	none	own	2	management	1	no	yes
1	25	none	rent	1	skilled	1	no	yes

Showing 1 to 11 of 1,000 entries, 17 total columns

Рисунок 7

Как видно из рисунка 7, таблица содержит 17 атрибутов (переменных) и 1000 объектов (наблюдения за 1000 клиентами). Можно с помощью команды str(credit) вывести имена атрибутов и их типы (рискунок 8).



```
> str(credit)
'data.frame':   1000 obs. of  17 variables:
 $ checking_balance    : Factor w/ 4 levels "< 0 DM","> 200 DM",..: 1 3 4 1 1 4 4 3 4 3 ...
 $ months_loan_duration: int  6 48 12 42 24 36 24 36 12 30 ...
 $ credit_history      : Factor w/ 5 levels "critical","good",..: 1 2 1 2 4 2 2 2 2 1 ...
 $ purpose             : Factor w/ 6 levels "business","car",..: 5 5 4 5 2 4 5 2 5 2 ...
 $ amount              : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
 $ savings_balance     : Factor w/ 5 levels "< 100 DM","> 1000 DM",..: 5 1 1 1 1 5 4 1 2 1
...
 $ employment_duration : Factor w/ 5 levels "< 1 year","> 7 years",..: 2 3 4 4 3 3 2 3 4 5
...
 $ percent_of_income   : int  4 2 2 2 3 2 3 2 2 4 ...
 $ years_at_residence  : int  4 2 3 4 4 4 4 2 4 2 ...
 $ age                 : int  67 22 49 45 53 35 53 35 61 28 ...
 $ other_credit        : Factor w/ 3 levels "bank","none",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ housing             : Factor w/ 3 levels "other","own",..: 2 2 2 1 1 1 2 3 2 2 ...
 $ existing_loans_count: int  2 1 1 1 2 1 1 1 1 2 ...
 $ job                 : Factor w/ 4 levels "management","skilled",..: 2 2 4 2 2 4 2 1 4 1
...
 $ dependents          : int  1 1 2 2 2 2 1 1 1 1 ...
 $ phone               : Factor w/ 2 levels "no","yes": 2 1 1 1 1 2 1 2 1 1 ...
 $ default             : Factor w/ 2 levels "no","yes": 1 2 1 1 2 1 1 1 1 2 ...
>
```

Рисунок 8

Переменная default указывает на то, был ли возвращен кредит (1 – не возвращен, 2 – возвращен) – это выходная переменная. Переменная

checking_balance указывает баланс текущего счета, saving_balance – баланс накопительного счета. DM здесь означает немецкую марку, так как данные здесь за период до введения евро. Переменная months_loan_duration – содержит значения, равные продолжительности кредита в месяцах, переменная amount – сумму кредита.

Предварительно необходимо разбить наблюдения на две части: обучающую и контрольную (тестовую). Код на R показан на рисунке 9.

```
credit <- read.csv("~/R/examples/credits.csv", stringsAsFactors=TRUE)
View(credit)
str(credit)
# Устанавливаем начальную точку для запуска алгоритма генерации
# псевдослучайных чисел
set.seed(123)
#Формируем последовательность 900 целых случайных чисел из 1000 (от 1 до 1000)
train_index<-sample(1000,900)
#Формируем выборку для обучения с индексами из train_index
credit_train<-credit[train_index,]
#Формируем контрольную выборку, исключая из таблицы credit наблюдения
# с индексами из train_index
credit_test<-credit[-train_index,]
# проверяем, насколько обучающая и контрольная выборки сбалансированы
# (процент отдавших кредит в обущающей и контрольной выборках должны
# быть примерно равны)
prop.table(table(credit_train$default))
prop.table(table(credit_test$default))
```

Рисунок 9

Воспользуемся алгоритмом CART. В R он реализован с помощью пакета **rpart**. Его необходимо загрузить с помощью команды install.packages и подключить соответствующую библиотеку. Код представлен на рисунке 10

```
#загружаем и подключаем rpart
install.packages("rpart")
library("rpart")
```

Рисунок 10

С помощью функции rpart для обучающей выборки строим дерево (см. рисунок 11), а с помощью библиотеки rpart.plot и соответствующей команды рисуем дерево.

```
24  # строим дерево классификации
25  credit_tree<-rpart(default~., data=credit_train, method="class",
26                     control=rpart.control(minsplit=10, minbucket = 5,
27                                           maxdepth = 6))
28
29  #рисуем дерево
30  library("rpart.plot")
31  rpart.plot(credit_tree, type=2, extra=1)
32  print(credit_tree, digits = 1)
33
```

Рисунок 11

Нарисованное дерево изображено на рисунке 12.

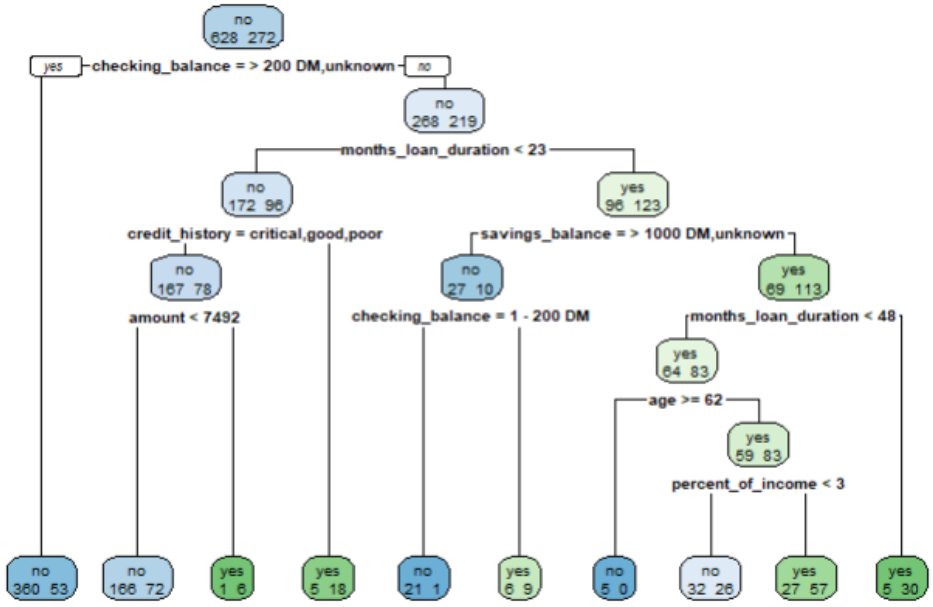


Рисунок 12

Опишем основные параметры функции rpart()

формула

default~.

означает, что default – это выходная переменная, а точка справа от тильды, что в качестве атрибутов рассматриваются все остальные переменные. Если в качестве атрибутов мы берем лишь некоторые переменные, то формулу можно записать, например, так

default~ saving_balance+ checking_balance

data – это таблица с данными;

method=”class” – указывает, что строится дерево классификации (если method=”anova”, то будет строиться дерево регрессии);

control – список параметров, управляющих подробностями алгоритма (например условиями остановки работы алгоритма):

minsplit – минимальное число наблюдений, которое должно существовать в узле для того, чтобы была предпринята попытка разделения (по умолчанию minsplit = 20)

minbucket – минимальное количество наблюдений в любом терминальном узле (листе). Если не указано явно числовое значение, то рассчитывается через minsplit по формуле minbucket = round(minsplit/3). Если указано значение minbucket, а значение minsplit нет, то оно восстанавливается через minbucket по приведенной формуле;

maxdepth – максимальная глубина любого узла конечного дерева, причем корневой узел будет считаться глубиной (по умолчанию равен 30);

Основные параметры функции rpat.plot()

x – дерево

type – тип дерева.

Возможные значения:

0 рисует в каждом узле условие разбиения, а в каждом листе метку предсказанного класса;

1 – помечает меткой класса все узлы, а не только листья;

2 – (по умолчанию) как и 1, но рисует условия разбиения под узлами;

3 – рисует условия разбиения по отдельности для левого и правого разветвления, метки классов указаны в листьях;

4 – как и 3, но метки классов указаны не только в листах, но и в узлах

5 – показывает имя переменной разделения во внутренних узлах.

extra – отображает дополнительную информацию в узлах.

Некоторые возможные значения:

«авто» – для деревьев классификации выбирается по умолчанию 106;

0 – никакой лишней информации;

1 – отображает количество наблюдений, попадающих в узел

2 –отображает степень классификации в узле, выраженную как количество правильных классификаций к количеству наблюдений в узле

3 –отображает уровень ошибочной классификации в узле, выраженный как количество неправильных классификаций и количество наблюдений в узле.

tweak – размер шрифта по сравнению с тем, что по умолчанию. По умолчанию tweak=1, но чтобы увеличить размер шрифта на 20%, можно указать tweak=1,2 .

Так же для построения дерева можно воспользоваться функцией prp(), например, команда

prp(credit_tree, extra = 6, box.palette = "auto"),

выдаст дерево, изображенное на рисунке 13

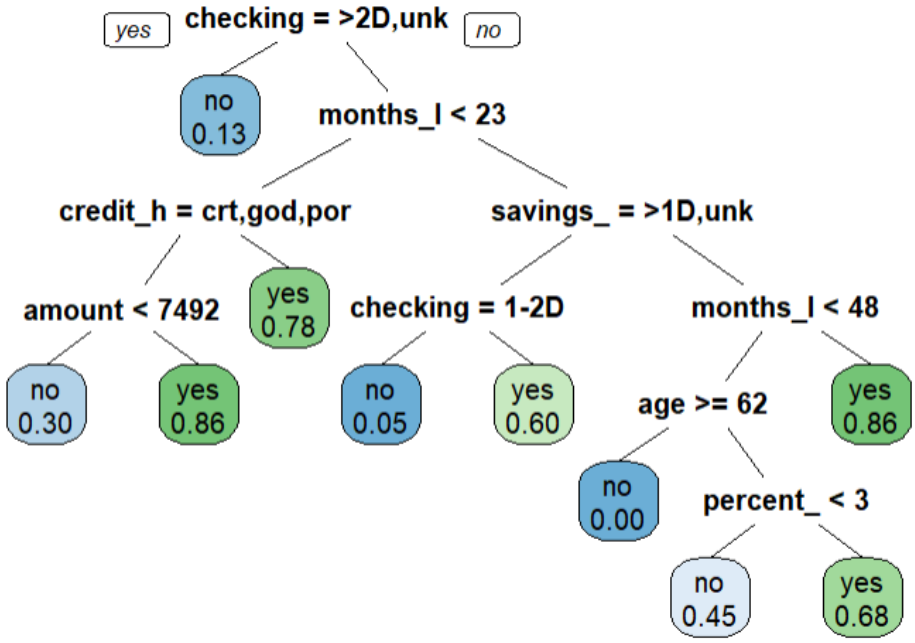


Рисунок 13

Можно нарисовать дерево с помощью функции fancyRpartPlot из пакета rattle (предварительно его надо инсталировать)

```
18  #рисуем дерево 2
19  library(rattle)
20  fancyRpartPlot(credit_tree)
```

Результат показан на рисунке 14

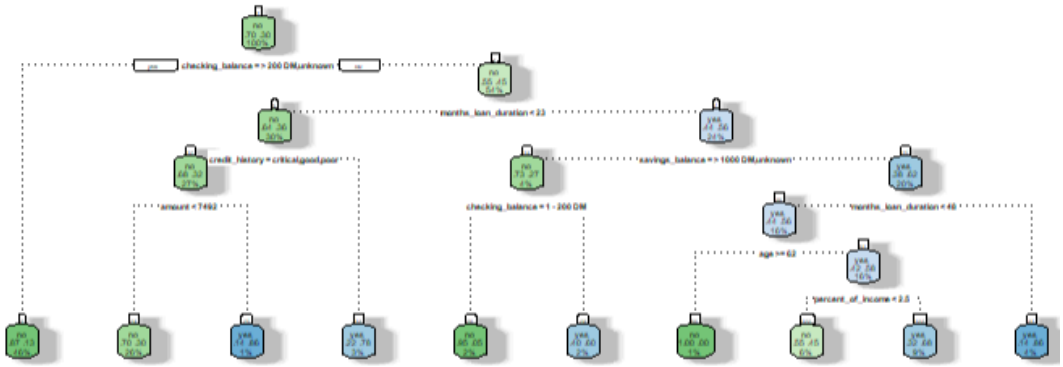


Рисунок 14

С помощью функции predict() (код на рисунке 15) можно прогнозировать Функция predict() возвращает вектор спрогнозированных значений классов или необработанных вероятностей, в зависимости от значения параметра type.

predict (m, test, type = "class")

• m — модель, обученная с помощью функции rpart();

• test — фрейм данных, содержащий тестовые данные с теми же признаками, что и тренировочные данные, использованные для построения классификатора;

• type — переменная, принимающая одно из двух значений, "class" или "prob", и определяющая, следует ли спрогнозировать наиболее вероятное значение класса или же необработанную вероятность.

Проведем классификацию с помощью полученного дерева на данных таблицы credit_test. Построим таблицу сопряженности, в которой будет показано количество правильно проклассифицированных меток и количество ошибок (см. код на рисунке 15).

```
39  # выдает результат в виде спрогнозированного класса
40  pr<-predict(credit_tree, credit_test[,-17], type="class")
41
42  #выводит результат в виде вероятности принадлежности к классам
43  pr<-predict(credit_tree, credit_test[,-17], type="prob")
44  #таблица сопряженности
45  table(credit_test[,17], predict(credit_tree, credit_test[,-17],
46                                  type="class"))
```

Рисунок 15

Построенная таблица сопряженности показана на рисунке 16.

```
      no yes
  no  64   8
  yes 17  11
```

Видим, что в 75 примерах из 100 из тестирующей выборки модель дает правильный результат.