Введение в деревья решений [M.151]

Деревья решений (decision trees) относятся к числу самых популярных и мощных инструментов Data Mining, позволяющих эффективно решать задачи классификации и регрессии. В отличие от методов, использующих статистический подход, таких как классификатор Байеса, линейная и логистическая регрессия, деревья решений основаны на машинном обучении и в большинстве случаев не требуют предположений о статистическом распределении значений признаков. В основе деревьев решений лежат решающие правила вида «если… то…», которые могут быть сформулированы на естественном языке. Поэтому деревья решений являются наиболее наглядными и легко интерпретируемыми моделями.

Для удобства приведем базовые понятия теории деревьев решений в таблице 1.

Таблица 1 – Понятия, встречающиеся в теории деревьев решений

Название	Описание
Объект	Пример, шаблон, наблюдение, запись
Атрибут	Признак, независимая переменная, свойство, входное поле
Метка класса	Зависимая переменная, целевая переменная, выходное поле
Узел	Внутренний узел дерева
Лист	Конечный узел дерева, узел решения
Проверка	Условие в узле

Атрибутами в теории деревьев решений называются признаки, описывающие классифицируемые объекты.

В основе работы деревьев решений лежит процесс рекурсивного 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. Рекурсивными называются алгоритмы, которые работают в пошаговом режиме, при этом на каждом последующем шаге используются результаты, полученные на предыдущем шаге.

Рассмотрим главную идею алгоритмов построения деревьев решений на примере. Пусть требуется предсказать возврат или невозврат кредита с помощью набора решающих правил на основе единственного атрибута Возраст клиента. Для этого будем использовать множество наблюдений, в каждом из которых указывается возраст, а также факт возврата/невозврата займа. Графически такое множество наблюдений представлено на рисунке 1. Условно примем, что объект в форме круга указывает на невозврат, в форме прямоугольника — на дефолт по займу, а внутри каждого объекта указан возраст. Необходимо разбить множество объектов на подмножества таким образом, чтобы в каждое из них попали объекты только одного класса.

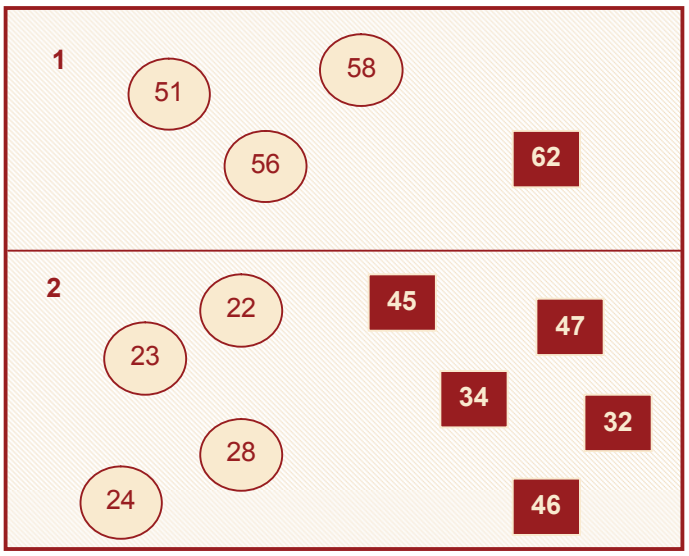

Рисунок 1 – Разделение на классы

Выберем некоторое значение возрастного порога, например равное 50, и разобьем исходное множество на два подмножества в соответствии с условием Возраст > 50. В результате разбиения в одном подмножестве окажутся все записи, для которых значение атрибута Возраст больше 50, а во втором — меньше 50. На рисунке 1 данные подмножества обозначены номерами 1 и 2 соответственно. Легко увидеть, что выбор возрастного порога 50 не позволил получить подмножества, содержащие только объекты одного класса, поэтому для решения задачи применяется разбиение полученных подмножеств. Поскольку для этого имеется только один атрибут – Возраст, мы будем использовать его и в дальнейшем, но в условиях выберем другой порог. Например, для подмножества 1 применим порог 60, а для подмножества 2 — 30. Результаты повторного разбиения представлены на рисунке 2.

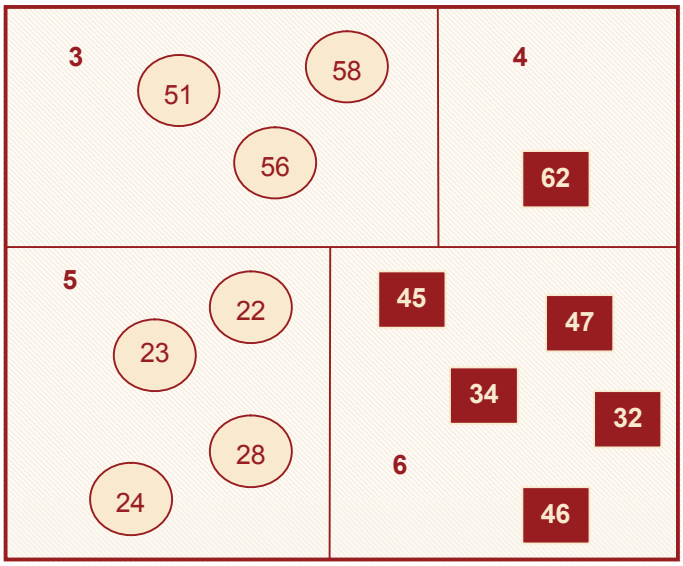

Рисунок 2 – Продолжение деления на классы

На рисунке 2 можно увидеть, что задача решена: исходное множество удалось разбить на чистые подмножества, содержащие только наблюдения одного класса. Дерево, реализующее данную процедуру, представлено на рисунке 3.

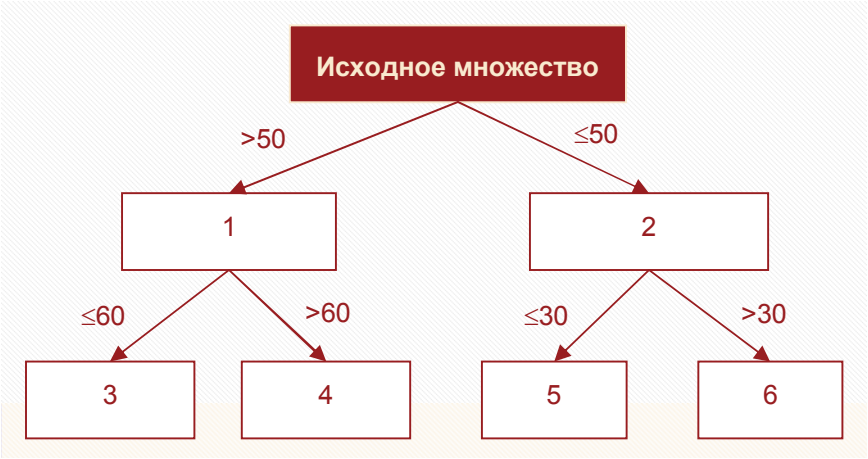

Рисунок 3 – Сформированное дерево решений

Сопоставление рисунке 2 и 3 показывает, что подмножества 3 и 5 ассоциированы с классом клиентов, не вернувших кредит, а подмножества 4 и 6 — с классом погасивших заем. Применяя построенную модель к новым клиентам, мы можем предсказать риск, связанный с выдачей им кредита, на основе того, в какое из подмножеств модель поместит соответствующую запись.

Конечно, приведенный пример тривиален. В реальности задача классификации по одному атрибуту встречается очень редко, поскольку для эффективного разделения на классы требуется несколько атрибутов. Но даже этот пример позволяет увидеть причину привлекательности деревьев решений. Они не только классифицируют объекты и наблюдения, но и объясняют, почему объект был отнесен к данному классу. Так, если с помощью дерева решений было предсказано, что вероятность возврата кредита данным клиентам слишком мала, и на этом основании в выдаче кредита было отказано, то можно не только принять решение, но и объяснить его причину. В нашем случае это возраст клиента.

Обладая высокой объясняющей способностью, деревья решений могут использоваться и как эффективные классификаторы, и как инструмент исследования предметной области.

Таким образом, мы получили систему правил вида «если… то…», которые позволяют принять решение относительно принадлежности объекта к определенному классу. Решающие правила образуют иерархическую древовидную структуру, дающую возможность выполнять классификацию объектов и наблюдений. Эта структура и называется деревом решений.

Определение

> *Деревья решений — иерархические древовидные структуры, состоящие из решающих правил вида «если… то…» и позволяющие выполнять классификацию объектов. В дереве каждому объекту соответствует единственный узел, дающий решение.*

Деревья решений стали одним из наиболее популярных методов Data Mining, используемых при решении задач классификации. Это обусловлено следующими факторами.

- Деревья решений — это модели, основанные на обучении. Процесс обучения сравнительно прост в настройке и управлении.
- Процесс обучения деревьев решений быстр и эффективен.
- Деревья решений универсальны — способны решать задачи как классификации, так и регрессии.
- Деревья решений обладают высокой объясняющей способностью и интерпретируемостью.

Замечание

Термин «дерево решений» используется не только в Data Mining, но и в смежных областях анализа и обработки данных, искусственном интеллекте. Например, в задачах алгоритмизации дерево решений — это способ представления процесса принятия решения, имеющий вид ответов на серию вопросов, образующих древовидную структуру, каждая конечная вершина которой представляет элементарное решение. В менеджменте под деревом решений понимают графическое изображение альтернативных действий и их последствий. Однако между всеми этими толкованиями термина много общего.

Для эффективного построения дерева решений должны выполняться следующие условия.

- *Описание атрибутов*. Анализируемые данные должны быть представлены в виде структурированного набора, в котором вся информация об объекте или наблюдении должна быть выражена совокупностью атрибутов.
- *Предварительное определение классов.* Категории, к которым относятся наблюдения (метки классов), должны быть заданы предварительно, то есть имеет место обучение с учителем.
- *Различимость классов.* Должна обеспечиваться принципиальная возможность установления факта принадлежности или непринадлежности примера к определенному классу. При этом количество примеров должно быть намного больше, чем количество классов.
- *Полнота данных.* Обучающее множество должно содержать достаточно большое количество различных примеров. Необходимая численность зависит от таких факторов, как количество признаков и классов, сложность классификационной модели и т. д.

Структура дерева решений

Как можно увидеть на рисунке 3, структура деревьев решений проста и в целом аналогична древовидным иерархическим структурам, используемым в других областях Data Mining, например *деревьям ассоциативных правил*. В состав деревьев решений входят два вида объектов — узлы (node) и листья (leaf). В узлах содержатся правила, с помощью которых производится проверка атрибутов и множество объектов в данном узле разбивается на подмножества. Листья — это конечные узлы дерева, в которых содержатся подмножества, ассоциированные с классами. Основным отличием листа от узла является то, что в листе не производится проверка, разбивающая ассоциированное с ним подмножество и, соответственно, нет ветвления. В принципе, листом может быть объявлен любой узел, если принято решение, что множество в узле достаточно однородно в плане классовой принадлежности объектов и дальнейшее разбиение не имеет смысла, поскольку не приведет к значимому увеличению точности классификации, а только усложнит дерево.

В дереве, представленном на рисунке 3, объекты с номерами 1 и 2 — узлы, а с номерами 3, 4, 5 и 6 — листья. Обратим внимание на то, что в дереве имеется по два листа, ассоциированных с одним классом. В этом нет никакого противоречия, просто для классификации объектов в них использовались различные способы проверки. Для каждого листа в дереве имеется уникальный путь. Начальный узел дерева является входным: через него проходят все объекты, предъявляемые дереву. Обычно входной узел называют *корневым узлом* (root node). Следовательно, дерево растет сверху вниз. Узлы и листья, подчиненные узлу более высокого иерархического уровня, называются *потомками*, или *дочерними узлами*, а тот узел по отношению к ним — *предком*, или *родительским узлом*. Обобщенная структура дерева проиллюстрирована на рисунке 4.

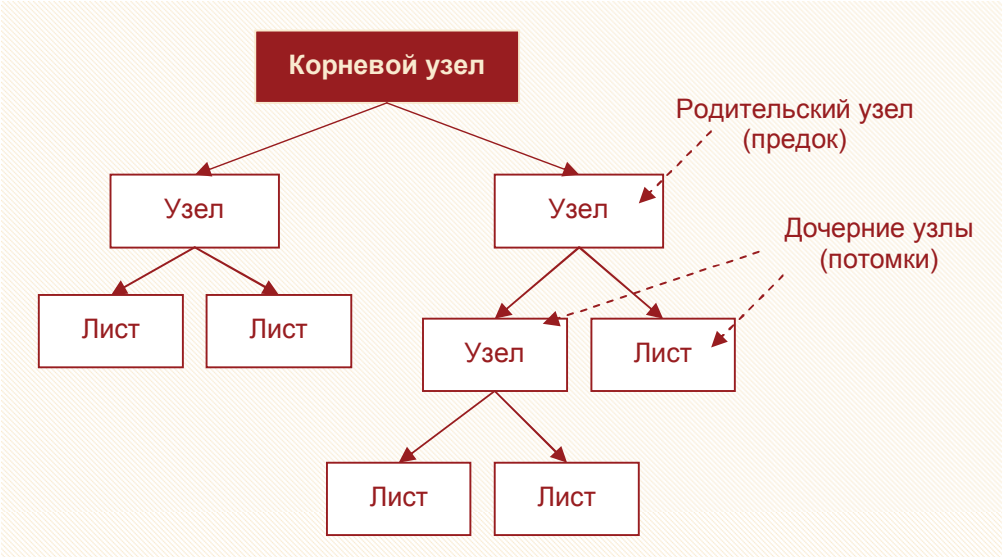

Рисунок 4 – Узлы и листья в дереве решений

Как и любая модель Data Mining, дерево решений строится на основе обучающего множества. Атрибуты могут быть как числовыми (непрерывными), так и категориальными (дискретными). Одно из полей обязательно должно содержать независимую переменную — метку класса. Иными словами, для каждой записи в обучающем множестве должна быть задана метка класса, определяющая классовую принадлежность связанного с ней объекта.

В процессе построения дерева решений формируются решающие правила и для каждого из них создается узел. Для каждого узла нужно выбрать атрибут, по которому будет производиться проверка правила. Его принято называть *атрибутом ветвления*, или *атрибутом разбиения* (splitting attribute), и от того, насколько удачно он выбран, зависит классифицирующая сила правила. Метод, в соответствии с которым осуществляется выбор атрибута ветвления на каждом шаге, называется *алгоритмом построения дерева решений*. Разработано достаточно много таких алгоритмов. Сформулируем общую цель, которая должна преследоваться при выборе атрибута ветвления: 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 (то есть не ассоциированного с данным узлом или листом) минимальна.

Хотя между алгоритмами построения деревьев решений имеются существенные различия, все они основаны на одной и той же процедуре — рекурсивном разбиении данных на все более малые группы таким образом, чтобы каждое новое поколение узлов содержало больше примеров одного класса, чем родительский.

Выбор атрибута разбиения в узле

Процесс создания дерева начинается с подготовки обучающего множества. В итоге будет построено дерево, которое назначает класс (или вероятность принадлежности к классу) для выходного поля новых записей на основе значений входных переменных.

Мерой оценки возможного разбиения является так называемая *чистота* (purity), под которой понимается *отсутствие примесей*. Существует несколько способов определения чистоты, но все они имеют один и тот же смысл. Низкая чистота означает, что в подмножестве представлены объекты, относящиеся к различным классам. Высокая чистота свидетельствует о том, что члены отдельного класса доминируют. Наилучшим разбиением можно назвать то, которое дает наибольшее увеличение чистоты дочерних узлов относительно родительского. Кроме того, хорошее разбиение должно создавать узлы примерно одинакового размера или как минимум не создавать узлы, содержащие всего несколько записей. Рассмотрим рисунок 5.

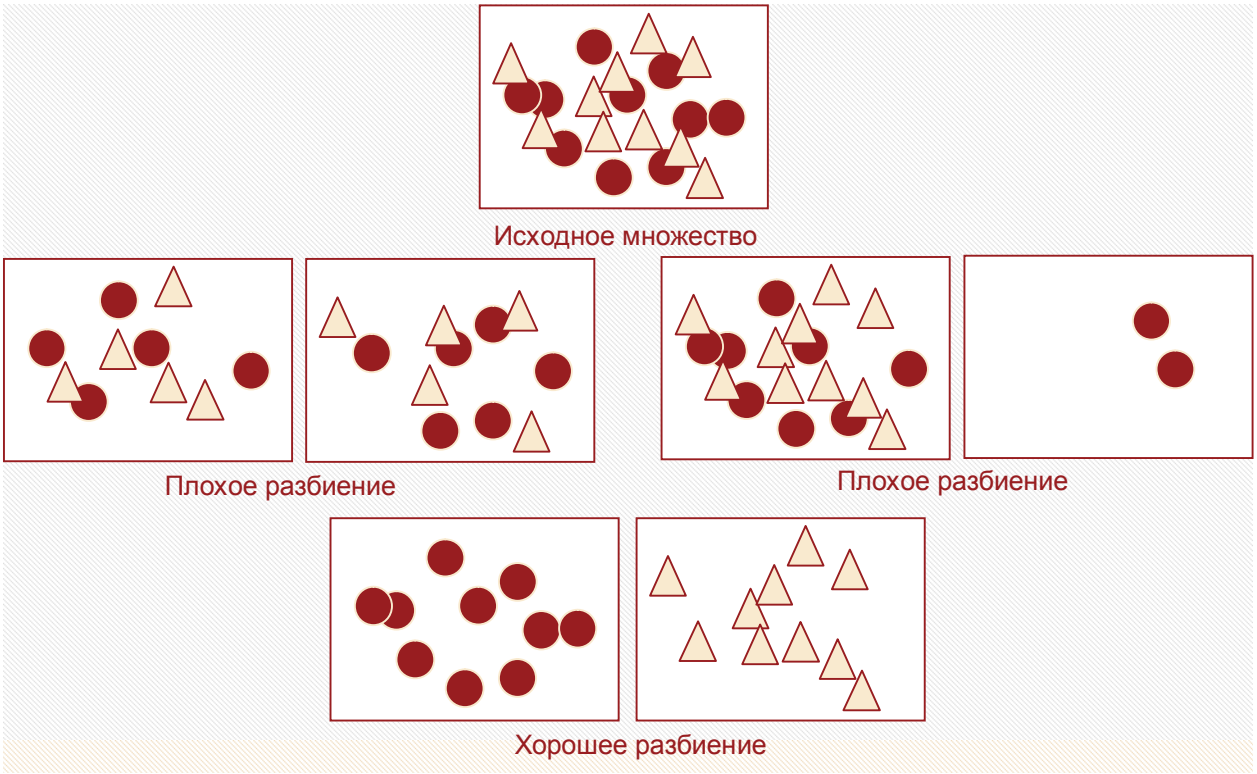

Рисунок 5 – Различные варианты разбиений

В исходном множестве представлена смесь объектов различной формы (треугольники и круги) в пропорции 1 : 1 (10 кругов и 10 треугольников). Разбиение слева признано плохим, потому что оно не увеличивает чистоту результирующих узлов: пропорция объектов обоих классов в них сохраняется и также составляет 1 : 1 (5 кругов и 5 треугольников). Во втором случае плохого разбиения (справа) с помощью условия в родительском узле удалось добиться доминирования класса в одном из дочерних узлов (круги), но к классу было отнесено только два объекта. Такое разбиение неудачно по двум причинам. Во-первых, правило, с помощью которого было получено это разбиение, имеет очень низкую значимость, то есть относится к малому числу примеров. Во-вторых, чистота другого узла осталась низкой: соотношение объектов в нем — 0,8 : 1 (8 кругов и 10 треугольников). И хотя в целом чистота относительно родительского узла улучшилась (класс треугольников стал слабо доминирующим), это не позволяет говорить о положительном результате. Наконец, случай хорошего разбиения обеспечивает абсолютную чистоту обоих дочерних узлов, поскольку в каждый из них распределены объекты только одного класса.

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

Замечание

«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. Однако вопрос о применимости жадного алгоритма в каждом классе задач решается отдельно. При построении деревьев решений жадность алгоритма заключается в следующем: для каждого узла ищется оптимальный атрибут разбиения и предполагается, что дерево в целом также будет оптимальным.

После того как все потенциальные атрибуты будут проверены, тот из них, который обеспечит наилучшее разбиение, будет использован для начального разбиения в корневом узле, в результате чего будут созданы дочерние узлы. Если дальнейшее разбиение невозможно (например, из-за того, что в узле осталось всего два примера) или ни одно возможное разбиение не может увеличить чистоту дочерних узлов, то алгоритм завершит работу, а текущий узел будет объявлен листом.

Замечание

*Еще одним важным свойством деревьев решений является **ацикличность**. Выбрав атрибут разбиения в определенном узле и выполнив соответствующее разбиение, алгоритм не может вернуться назад, если оно оказалось неоптимальным, и выполнить новое разбиение на основе другого атрибута. Иными словами, если по какой-либо причине на определенном шаге построения дерева решений было получено неудачное разбиение, алгоритм все равно будет продолжать строить дерево. Чтобы компенсировать неудачный выбор, алгоритму потребуется создать дополнительные узлы, что приведет к увеличению сложности дерева и временных затрат на его построение.*

Принцип «разделяй и властвуй»

Процесс рекурсивного разбиения подмножеств в узлах дерева решений получил название «разделяй и властвуй» (divide and conquer). В его основе лежит следующий принцип. Пусть задано множество T, на котором определены классы $\{C_1, C_2, \ldots, C_k\}$. Тогда существуют три возможных варианта разбиения.

1. Множество T содержит два примера или более, которые относятся к одному классу C_j. Деревом решений для множества T будет лист, идентифицирующий класс C_j. Это тривиальный случай, который на практике не представляет интереса. Графически он может быть интерпретирован, как показано на рисунке 6.

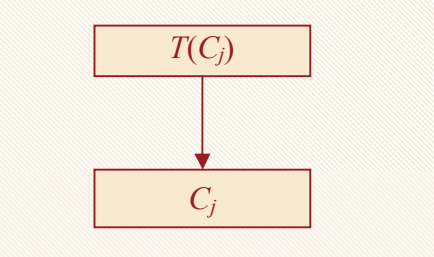

Рисунок 6 – Иллюстрация первого случая

2. Множество T не содержит примеров, то есть является пустым. Деревом решений также будет лист, но класс, ассоциированный с данным листом, должен быть определен из другого множества, например родительского. Этот случай иллюстрируется на рисунке 7. После разбиения по атрибуту Доход ветвление продолжилось по атрибуту Образование.

   Как должен вести себя алгоритм, если в обучающем множестве нет ни одного наблюдения, где Доход = Высокий и Образование = Среднее? Алгоритм должен создать три дочерних узла, и в T_3 окажется пустое множество. Тогда сгенерируется узел, который будет ассоциирован с классом, наиболее часто встречающимся в родительском множестве.

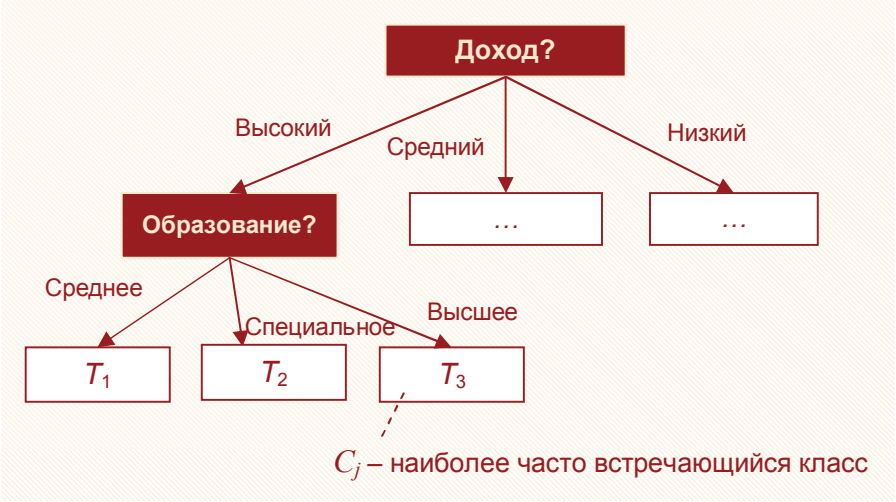

Рисунок 7 – Иллюстрация второго случая

3 Множество T содержит примеры, относящиеся к различным классам. Задача заключается в разделении T на подмножества, ассоциированные с классами. Выбирается один из входных атрибутов, принимающих два отличных друг от друга значения v_1, v_2, …, v_n или более, после чего T разбивается на подмножества $\{T_1(v_1), T_2(v_2), \ldots, T_n(v_n)\}$, где каждое подмножество T_i содержит все примеры из исходного множества, в которых выбранный атрибут принимает значение v_i. Данная процедура будет рекурсивно повторяться до тех пор, пока подмножества не будут содержать примеры только одного класса. Этот случай проиллюстрирован на рисунке 8.

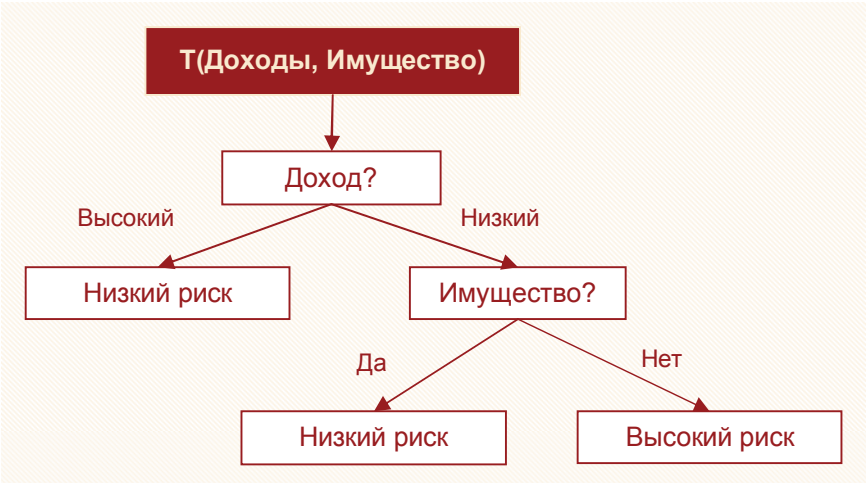

Рисунок 8 – Иллюстрация третьего случая

Пусть требуется определить класс риска, связанный с выдачей кредита клиенту, на основе двух атрибутов Доход и Имущество. Атрибуты принимают по два значения — Высокий и Низкий. На первом шаге алгоритм разобьет исходное множество на два подмножества, в одно из которых будут помещены клиенты с высокими доходами, а в другое — с низкими. Предположим, что все клиенты с высокими доходами относятся к классу низкого кредитного риска, поэтому соответствующее подмножество будет объявлено листом и дальнейшее разбиение в нем остановится. Другое подмножество, в котором остались клиенты с низким доходом, подвергается проверке с помощью атрибута Имущество. Логика проста: если клиент не имеет высокого дохода, но имеет достаточно имущества, чтобы покрыть кредит при невозможности его выплатить, то кредитный риск может быть определен как низкий.

Данный случай является наиболее распространенным и практически важным в процессе построения деревьев решений.

Процесс построения дерева решений не является однозначно определенным. Для различных атрибутов и даже для различного порядка их применения могут быть сгенерированы различные деревья решений. В идеальном случае разбиения должны быть такими, чтобы результирующее дерево оказалось наиболее компактным. Встает вопрос: почему бы тогда не исследовать все возможные деревья и не выбрать самое компактное? К сожалению, во многих реальных задачах перебор всех возможных деревьев решений приводит к комбинаторному взрыву. Даже для небольшой базы данных, содержащей всего 5 атрибутов и 15 примеров, возможное число деревьев превышает 106 в зависимости от количества значений, принимаемых каждым атрибутом.

Эффективность разбиения оценивается по чистоте полученных дочерних узлов относительно целевой переменной. От ее типа и будет зависеть выбор предпочтительного критерия разбиения. Если выходная переменная является категориальной, то необходимо использовать такие критерии, как индекс Джини, прирост информации или тест хи-квадрат. Если выходная переменная является непрерывной, то для оценки эффективности разбиения используются метод уменьшения дисперсии или F-тест (рисунок 9).

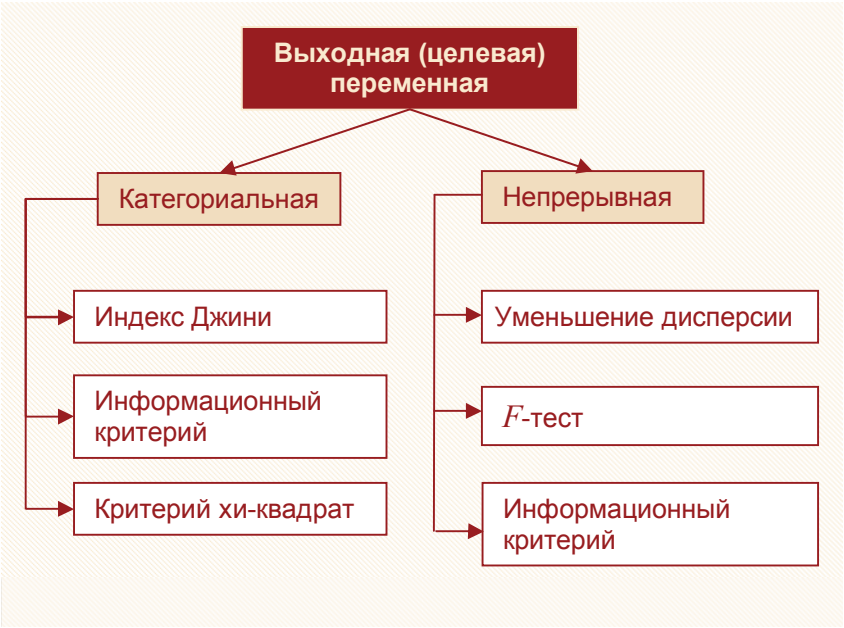

Рисунок 9 – Критерии разбиения