Тема 6: Методы классификации

6.5. Деревья принятия решений

Метод деревьев решений (*decisiontrees*) является одним из наиболее популярных методов решения задач классификации и прогнозирования. Иногда этот метод называют деревьями решающих *правил*, деревьями классификации и регрессии.

Как видно из последнего названия, при помощи данного метода решаются задачи классификации и прогнозирования.

Если зависимая, т.е. целевая переменная принимает дискретные значения, при помощи метода дерева решений решается *задача классификации*.

Если же зависимая переменная принимает непрерывные значения, то дерево решений устанавливает зависимость этой переменной от независимых переменных, т.е. решает *задачу численного прогнозирования*.

Впервые деревья решений были предложены Ховилендом и Хантом (Hoveland, Hunt) в конце 50-х годов прошлого века.

В наиболее простом виде дерево решений – это способ представления правил в иерархической, последовательной структуре. Основа такой структуры – ответы "Да" или "Нет" на ряд вопросов.

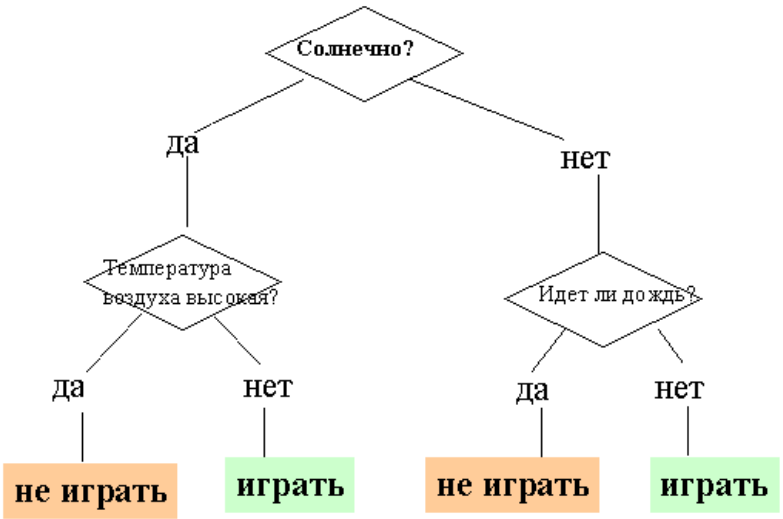


Рис. 6.4. Дерево решений "Играть ли в гольф?"

На рисунке 6.4 приведен **пример** дерева решений, задача которого – ответить на вопрос: "Играть ли в гольф?". Чтобы решить задачу, т.е.

принять *решение, играть* ли в гольф, следует отнести текущую ситуацию к одному из известных классов (в данном случае – "играть" или "не играть"). Для этого требуется ответить на ряд вопросов, которые находятся в узлах этого дерева, начиная с его корня.

Первый узел нашего дерева "Солнечно?" является *узлом проверки*, т.е. условием. При положительном ответе на вопрос осуществляется переход к левой части дерева, называемой левой ***ветвью***, при отрицательном – к правой части дерева. Таким образом, ***внутренний узел*** дерева является ***узлом проверки*** определенного условия. Далее идет следующий вопрос и т.д., пока не будет достигнут ***конечный узел*** дерева, являющийся *узлом решения*. Для нашего дерева существует два типа *конечного узла*: "играть" и "не играть" в гольф.

В результате прохождения от корня дерева (иногда называемого корневой вершиной) до его вершины решается задача классификации, т.е. выбирается один из классов – "играть" и "не играть" в гольф.

Целью построения дерева решения в нашем случае является определение значения категориальной зависимой переменной.

Итак, для нашей задачи основными элементами дерева решений являются:

– корень дерева: "Солнечно?";

– внутренний узел дерева или узел проверки: "Температура воздуха высокая?", "Идет ли дождь?";

– лист – конечный узел дерева – узел решения или вершина: "Играть", "Не играть";

– ветвь дерева (случаи ответа): "Да", "Нет".

В рассмотренном примере решается задача бинарной классификации, т.е. создается дихотомическая классификационная модель. Пример демонстрирует работу так называемых бинарных деревьев.

В узлах бинарных деревьев *ветвление* может вестись только в двух направлениях, т.е. существует возможность только двух ответов на поставленный вопрос ("да" и "нет").

Бинарные деревья являются самым простым, частным случаем деревьев решений. В остальных случаях, ответов и, соответственно, ветвей дерева, выходящих из его внутреннего узла, может быть больше двух.

Рассмотрим более сложный **пример**. База данных, на основе которой должно осуществляться прогнозирование, содержит следующие ретроспективные данные о клиентах банка, являющиеся ее атрибутами: возраст, наличие недвижимости, образование, среднемесячный доход, вернул ли клиент вовремя кредит. Задача состоит в том, чтобы на основании перечисленных выше данных (кроме последнего атрибута) определить, стоит ли выдавать кредит новому клиенту.

Как мы уже рассматривали выше, такая задача решается в два этапа: *построение классификационной модели* и *ее использование*.

На этапе построения модели строится дерево классификации или создается набор неких правил. На этапе использования модели построенное дерево, или путь от его корня к одной из вершин, являющийся набором правил для конкретного клиента, используется для ответа на поставленный вопрос "Выдавать ли кредит?". Правилом является логическая конструкция, представленная в виде "если : то :".

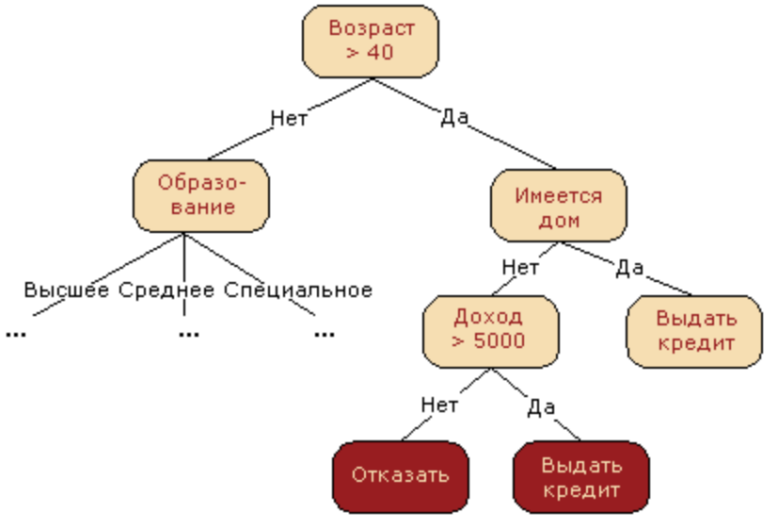


Рис. 6.5. Дерево решений "Выдавать ли кредит?"

На рисунке 6.5. приведен пример дерева классификации, с помощью которого решается задача "Выдавать ли *кредит* клиенту?". Она является

типичной задачей классификации, и при помощи деревьев решений получают достаточно хорошие варианты ее решения.

Как мы видим, *внутренние узлы* дерева (возраст, наличие недвижимости, доход и образование) являются атрибутами описанной выше *базы данных*. Эти атрибуты называют прогнозирующими, или *атрибутами расщепления* (*splitting attribute*). *Конечные* узлы дерева, или листы, именуются метками класса, являющимися значениями зависимой категориальной переменной "выдавать" или "не выдавать" *кредит*.

Каждая ветвь дерева, идущая от внутреннего узла, отмечена предикатом расщепления. Последний может относиться лишь к одному атрибуту расщепления данного узла. Характерная особенность предикатов расщепления: каждая запись использует уникальный путь от корня дерева только к одному узлу-решению. Объединенная информация об атрибутах расщепления и предикатах расщепления в узле называется критерием расщепления (*splitting criterion*).

На рисунке 6.5. изображено одно из возможных деревьев решений для рассматриваемой *базы данных*. Например, *критерий расщепления* "Какое образование?", мог бы иметь два *предиката расщепления* и выглядеть иначе: образование "высшее" и "не высшее". Тогда *дерево* решений имело бы другой вид.

Таким образом, для данной задачи (как и для любой другой) может быть построено множество деревьев решений различного качества, с различной прогнозирующей точностью.

Качество построенного дерева решения зависит от правильного выбора критерия расщепления.

Метод деревьев решений часто называют "наивным" подходом. Но благодаря целому ряду преимуществ, данный метод является одним из наиболее популярных для решения задач классификации.

6.5.1. Преимущества деревьев решений

Интуитивность деревьев решений. Классификационная модель, представленная в виде дерева решений, является интуитивной и упрощает

понимание решаемой задачи. Результат работы алгоритмов конструирования деревьев решений, в отличие, например, от нейронных сетей, представляющих собой "черные ящики", легко интерпретируется пользователем. Например, по схеме на рисунке 6.5 можно объяснить заемщику, почему ему было отказано в кредите. Скажем, потому, что у него нет дома и доход меньше 5000.

Это свойство деревьев решений не только важно при отнесении к определенному классу нового объекта, но и полезно при интерпретации модели классификации в целом. *Дерево* решений позволяет понять и объяснить, почему конкретный *объект* относится к тому или иному классу.

Деревья решений дают возможность извлекать *правила* из *базы данных* на *естественном языке*. Пример *правила*: Если Возраст > 35 и Доход > 200, то выдать *кредит*.

Деревья решений позволяют создавать классификационные модели в тех областях, где аналитику достаточно сложно формализовать знания.

Алгоритм конструирования дерева решений не требует от пользователя выбора входных атрибутов (независимых переменных). На вход алгоритма можно подавать все существующие атрибуты, алгоритм сам выберет наиболее значимые среди них, и только они будут использованы для построения дерева. В сравнении, например, с нейронными сетями, это значительно облегчает пользователю работу, поскольку в нейронных сетях выбор количества входных атрибутов существенно влияет на время обучения.

Точность моделей, созданных при помощи деревьев решений, сопоставима с другими методами построения классификационных моделей (*статистические методы*, нейронные сети).

Разработан ряд **масштабируемых алгоритмов**, которые могут быть использованы для построения деревьев решения на сверхбольших базах данных; *масштабируемость* здесь означает, что с ростом числа примеров или записей *базы данных* время, затрачиваемое на обучение, т.е. построение деревьев решений, растет линейно. Примеры таких алгоритмов: SLIQ, SPRINT.

Быстрый процесс обучения. На построение классификационных моделей при помощи алгоритмов конструирования деревьев решений требуется значительно меньше времени, чем, например, на обучение нейронных сетей.

Большинство алгоритмов конструирования деревьев решений имеют возможность специальной обработки **пропущенных значений**.

Многие классические *статистические методы*, при помощи которых решаются задачи классификации, могут работать только с числовыми данными, в то время как деревья решений работают и с числовыми, и с **категориальными** типами данных.

Многие *статистические методы* являются параметрическими, и *пользователь* должен заранее владеть определенной информацией, например, знать вид модели, иметь гипотезу о виде зависимости между переменными, предполагать, какой вид распределения имеют данные. Деревья решений, в отличие от таких методов, строят непараметрические модели. Таким образом, деревья решений способны решать задачи, в которых отсутствует априорная *информация* о виде зависимости между исследуемыми данными.

6.5.2. Процесс построения дерева решений

В примере "Выдавать ли кредит?" мы видели, что решение о выдаче кредита принималось на основе возраста, наличия недвижимости, дохода и других. Но какой признак выбрать первым?

Рассмотрим другой **пример**, где все признаки бинарные.

Один человек загадывает знаменитого человека, а второй пытается отгадать, задавая вопросы, на которые можно ответить только "Да" или "Нет" (опустим варианты "не знаю" и "не могу сказать"). Какой вопрос отгадывающий задаст первым делом? Конечно, такой, который в значительной степени уменьшит количество оставшихся вариантов. К примеру, вопрос "Это Анджелина Джоли?" в случае отрицательного ответа оставит миллионы вариантов для дальнейшего перебора. А вот вопрос "Это женщина?" отсечет уже около половины знаменитостей. То есть, признак "пол" намного лучше разделяет выборку людей, чем признак "Это Анджелина Джоли", "национальность – испанец" или "любит

футбол". Это интуитивно соответствует понятию прироста информации, основанного на энтропии.

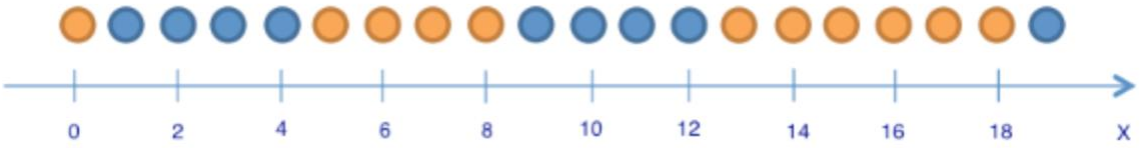
Энтропия

Энтропия Шеннона определяется для системы с N возможными состояниями следующим образом:

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

где p_i – вероятности нахождения системы в i-ом состоянии. Это очень важное понятие, используемое в физике, теории информации и других областях. Отметим, что, интуитивно, *энтропия соответствует степени хаоса в системе*. Чем выше энтропия, тем менее упорядочена система и наоборот. Энтропия поможет нам формализовать "эффективное разделение выборки".

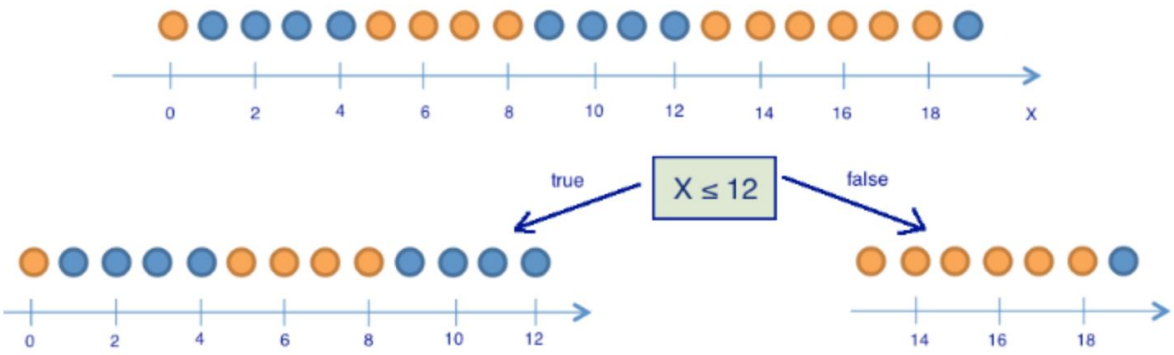
Пример. Для иллюстрации того, как энтропия поможет определить хорошие признаки для построения дерева, приведем пример с цветными шарами. Будем предсказывать цвет шарика по его координате. Данный пример позволит показать, как энтропия используется для построения дерева решений.



На рисунке 9 синих шариков и 11 желтых. Если мы наудачу вытащим шарик, то он с вероятностью $p_1 = 9/20$ будет синим и с вероятностью $p_1 = 11/20$ – желтым. Значит, энтропия состояния

$$S_0 = -\frac{9}{20} \cdot \log_2 \frac{9}{20} - \frac{11}{20} \cdot \log_2 \frac{11}{20} \approx 1.$$

Само это значение пока ни о чем нам не говорит. Теперь посмотрим, как изменится энтропия, если разбить шарики на две группы – с координатой меньше либо равной 12 и больше 12.



В левой группе оказалось 13 шаров, из которых 8 синих и 5 желтых. Энтропия этой группы равна

$$S_1 = -\frac{5}{13} \cdot \log_2 \frac{5}{13} - \frac{8}{13} \cdot \log_2 \frac{8}{13} \approx 0.96.$$

В правой группе оказалось 7 шаров, из которых 1 синий и 6 желтых. Энтропия правой группы равна

$$S_2 = -\frac{1}{7} \cdot \log_2 \frac{1}{7} - \frac{6}{7} \cdot \log_2 \frac{6}{7} \approx 0.6.$$

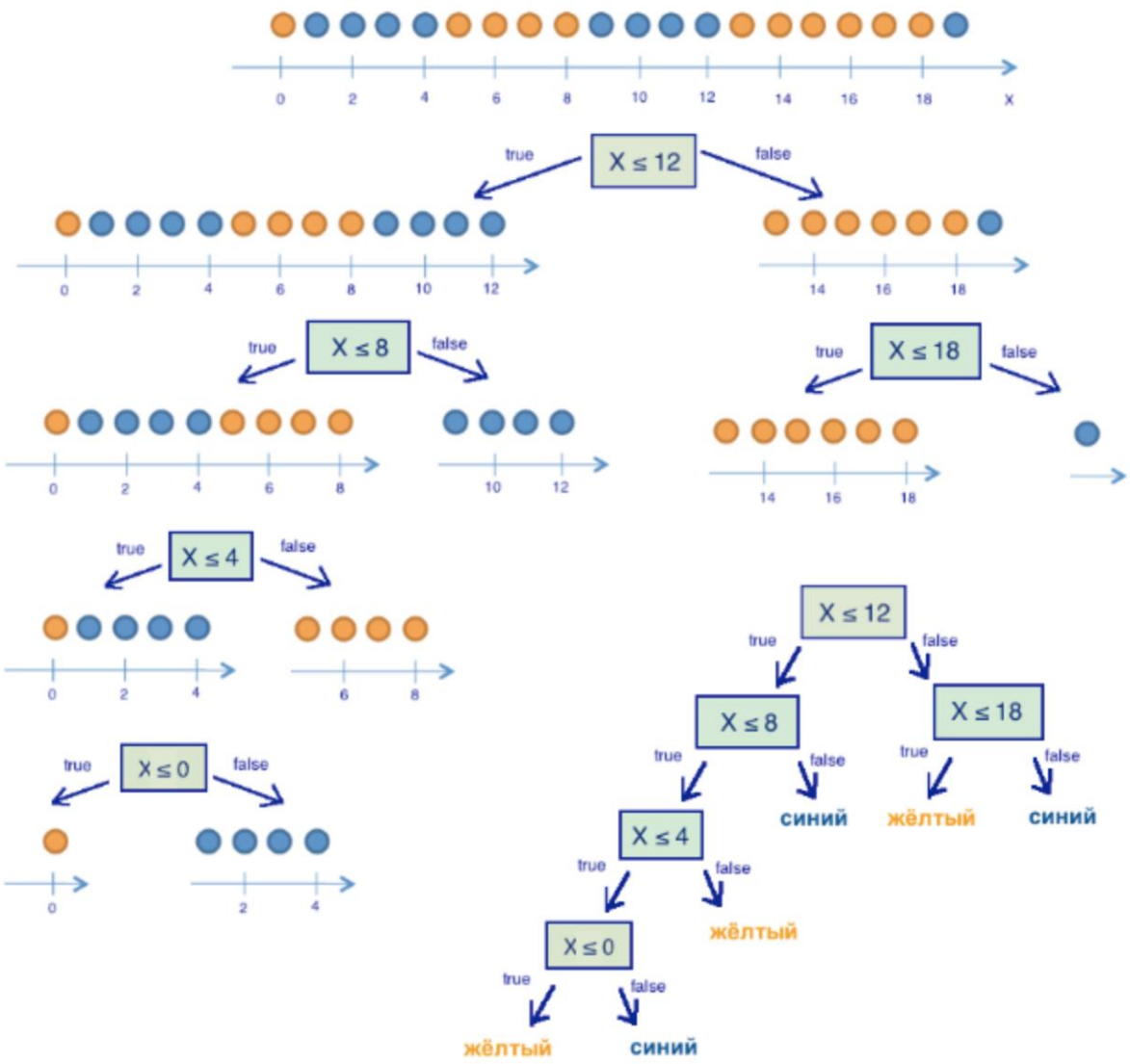
Как видим, энтропия уменьшилась в обеих группах по сравнению с начальным состоянием, хоть в левой и не сильно. Поскольку энтропия – по сути, степень хаоса (или неопределенности) в системе, уменьшение энтропии называют приростом информации. Формально прирост информации (information gain, IG) при разбиении выборки по признаку Q (в нашем примере это признак "$x \le 12$") определяется по формуле:

$$IG(Q) = S_0 - \sum_{i=1}^{q} \frac{N_i}{N} S_i,$$

где q – число групп после разбиения, N_i – число элементов выборки, у которых признак Q имеет i-ое значение. В нашем случае после разделения получилось две группы (q=2) – одна из 13 элементов (N_1=13), вторая – из 7 (N_2=7). Вычислим прирост информации:

$$IG(x \le 12) = S_0 - \frac{13}{20} S_1 - \frac{7}{20} S_2 \approx 0.16.$$

Получили, что разделив шарики на две группы по признаку "координата меньше либо равна 12", мы имеем более упорядоченную систему, чем в начале. Продолжим деление шариков на группы до тех пор, пока в каждой группе шарики не будут одного цвета.



Для правой группы потребовалось всего одно дополнительное разбиение по признаку "координата меньше либо равна 18", для левой – еще три. Очевидно, энтропия группы с шариками одного цвета равна 0 ($\log_2 1 = 0$), что соответствует представлению, что группа шариков одного цвета – упорядоченная.

В итоге мы построили дерево решений, предсказывающее цвет шарика по его координате. Отметим, что такое дерево решений может плохо работать для новых объектов (определения цвета новых шариков), поскольку оно идеально подстроилось под обучающую выборку (изначальные 20 шариков). Для

классификации новых шариков лучше подойдет дерево с меньшим числом "вопросов", или разделений, пусть даже оно и не идеально разбивает по цветам обучающую выборку. Эту проблему переобучения мы рассмотрим далее.

6.5.3. Алгоритм построения дерева

Напомним, что рассматриваемая нами задача классификации относится к стратегии *обучения с учителем*, иногда называемого индуктивным обучением. В этих случаях все объекты тренировочного набора данных заранее отнесены к одному из предопределенных классов.

Можно убедиться в том, что построенное в предыдущем примере дерево является в некотором смысле оптимальным – потребовалось только 5 "вопросов" (условий на признак *x*), чтобы "подогнать" дерево решений под обучающую выборку, то есть, чтобы дерево правильно классифицировало любой обучающий объект. При других условиях разделения выборки дерево может получиться глубже.

В основе популярных алгоритмов построения дерева решений, таких как ID3 и C4.5, лежит принцип жадной максимизации прироста информации – на каждом шаге выбирается тот признак, при разделении по которому прирост информации оказывается наибольшим. Дальше процедура повторяется рекурсивно, пока энтропия не окажется равной нулю или какой-то малой величине (если дерево не подгоняется идеально под обучающую выборку во избежание переобучения).

Алгоритмы конструирования деревьев решений состоят из этапов "*построение*" или "*создание*" дерева (*tree building*) и "*сокращение*" дерева (*tree pruning*). В ходе *создания* дерева решаются вопросы выбора *критерия расщепления* и остановки обучения (если это предусмотрено алгоритмом). В ходе этапа *сокращения* дерева решается вопрос отсечения некоторых его *ветвей*.

В разных алгоритмах применяются разные эвристики для "ранней остановки" или "отсечения", чтобы избежать построения переобученного дерева.

Рассмотрим эти вопросы подробней.

Критерий расщепления

Процесс *создания* дерева происходит сверху вниз, т.е. является нисходящим. В ходе процесса алгоритм должен найти такой *критерий расщепления*, иногда также называемый критерием разбиения, чтобы разбить множество на подмножества, которые бы ассоциировались с данным *узлом проверки*. Каждый *узел проверки* должен быть помечен определенным атрибутом. Существует правило выбора атрибута: он должен разбивать исходное множество данных таким образом, чтобы объекты подмножеств, получаемых в результате этого разбиения, являлись представителями одного класса или же были максимально приближены к такому разбиению. Последняя фраза означает, что количество объектов из других классов, так называемых "примесей", в каждом классе должно стремиться к минимуму.

Существуют различные критерии расщепления. Наиболее известные – *мера энтропии* и *индекс Gini*.

В некоторых методах для выбора атрибута расщепления используется так называемая мера информативности подпространств атрибутов, которая основывается на энтропийном подходе и известна под названием "мера информационного выигрыша" (information gain measure) или мера энтропии.

Другой *критерий расщепления*, предложенный Брейманом (Breiman) и др., реализован в алгоритме *CART* (Classification and Regression Tree) и называется неопределенностью Джини (Gini impurity) или **индексом Gini**. При помощи этого индекса атрибут выбирается на основании расстояний между распределениями классов.

Если дано множество T, включающее примеры из n классов, индекс Gini(T) определяется по формуле:

$$G = 1 - \sum_{k=1}^{n} (p_k)^2,$$

где T – текущий узел, p_k– вероятность класса k в узле T, n – количество классов.

Другим критерием качества разбиения в задаче классификации может служить *ошибка классификации* (misclassification error): $E = 1 - \max_k p_k$.

На практике ошибка классификации почти не используется, а неопределенность Джини и прирост информации работают почти одинаково.

В случае задачи бинарной классификации (p_+ – вероятность объекта имеет метку +) энтропия и неопределенность Джини примут следующий вид:

$$S = -p_+ \log_2 p_+ - p_- \cdot \log_2 p_- = -p_+ \log_2 p_+ - (1-p_+)\log_2(1-p_+);$$

$$G = 1 - p_+^2 - p_-^2 = 1 - p_+^2 - (1-p_+)^2 = 2p_+(1-p_+).$$

Когда мы построим графики этих двух функций от аргумента p_+, то увидим, что график энтропии очень близок к графику удвоенной неопределенности Джини (рис. 6.6), и поэтому на практике эти два критерия "работают" почти одинаково.

Пример:

Импортируем библиотеки:

```
from __future__ import division, print_function
# отключим всякие предупреждения Anaconda
import warnings
warnings.filterwarnings('ignore')
import numpy as np
import pandas as pd
%matplotlib inline
import seaborn as sns
from matplotlib import pyplot as plt
```

Рисуем графики:

```
plt.rcParams['figure.figsize'] = (6,4)
xx = np.linspace(0,1,50)
plt.plot(xx, [2 * x * (1-x) for x in xx], label='gini')
plt.plot(xx, [4 * x * (1-x) for x in xx], label='2*gini')
plt.plot(xx, [-x * np.log2(x) - (1-x) * np.log2(1 - x) for x in xx],
         label='entropy')
plt.plot(xx, [1 - max(x, 1-x) for x in xx], label='missclass')
plt.plot(xx, [2 - 2 * max(x, 1-x) for x in xx], label='2*missclass')
plt.xlabel('p+')
plt.ylabel('criterion')
plt.title('Критерии качества как функции от p+ (бинарная классификация)')
plt.legend()
```

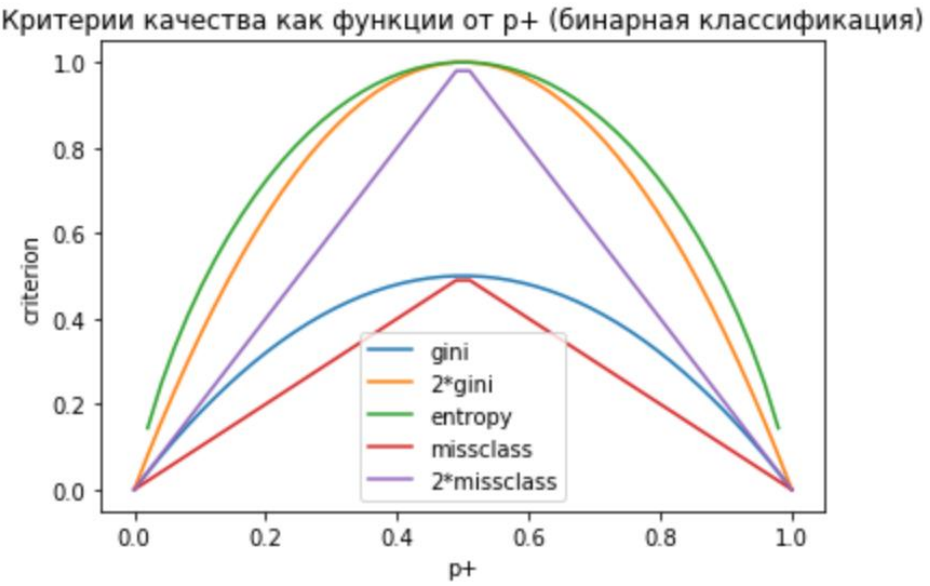


Рисунок 6.6. Графики энтропии и неопределенности Джини

Пример. Рассмотрим применение дерева решений из библиотеки `Scikit-learn` для синтетических данных. Два класса будут сгенерированы из двух нормальных распределений с разными средними.

```
# Код для генерации данных
# первый класс
np.seed = 7
train_data = np.random.normal(size=(100, 2))
train_labels = np.zeros(100)
# добавляем второй класс
train_data = np.r_[train_data, np.random.normal(size=(100, 2), loc=2)]
train_labels = np.r_[train_labels, np.ones(100)]
```

Отобразим данные. Неформально, задача классификации в этом случае – построить какую-то "хорошую" границу, разделяющую 2 класса (красные точки от желтых, рис. 6.7). Если утрировать, то машинное обучение в этом случае сводится к тому, как выбрать хорошую разделяющую границу. Возможно, прямая будет слишком простой границей, а какая-то сложная кривая, огибающая каждую красную точку – будет слишком сложной и будем много ошибаться на новых примерах из того же распределения, из которого пришла обучающая выборка. Интуиция подсказывает, что хорошо на новых данных будет работать какая-то *гладкая* граница, разделяющая 2 класса, или хотя бы просто прямая (в *n*-мерном случае – гиперплоскость).

```
# Отрисовка картинки
plt.rcParams['figure.figsize'] = (10,8)
plt.scatter(train_data[:, 0], train_data[:, 1],
            c=train_labels, s=100, cmap='autumn', edgecolors='black',
            linewidth=1.5);
plt.plot(range(-2,5), range(4,-3,-1))
```

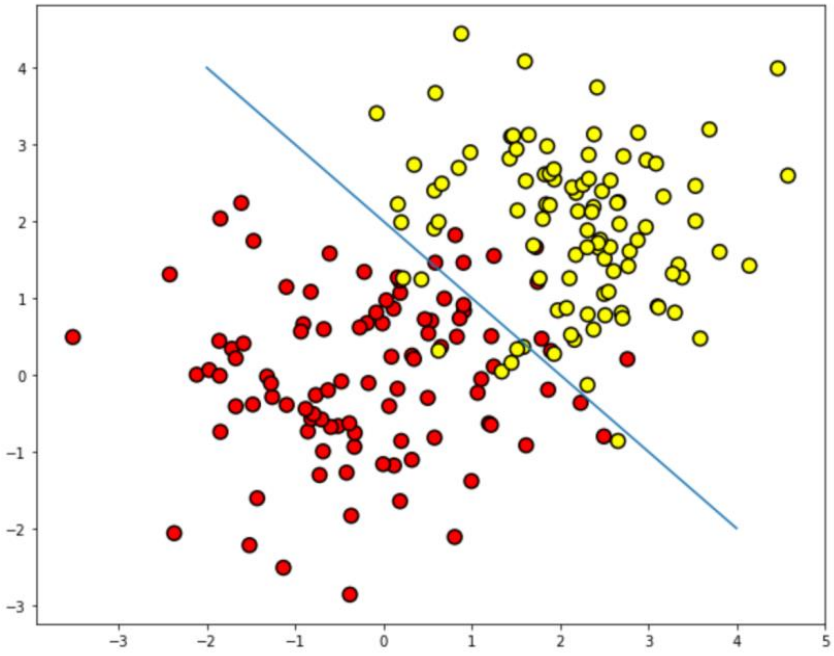


Рисунок 6.7. Граница, разделяющая два класса

Попробуем разделить эти два класса, обучив дерево решений. В дереве будем использовать параметр `max_depth`, ограничивающий глубину дерева. В библиотеке `scikit-learn` деревья решений реализованы в классах `DecisionTreeRegressor` и `DecisionTreeClassifier`.

Визуализируем полученную границу разделения классов (рис. 6.8).

```
# Код для обучения дерева и отрисовки его разделяющей границы
from sklearn.tree import DecisionTreeClassifier
# Напишем вспомогательную функцию, которая будет возвращать
# решетку для дальнейшей визуализации.
def get_grid(data):
    x_min, x_max = data[:, 0].min() - 1, data[:, 0].max() + 1
    y_min, y_max = data[:, 1].min() - 1, data[:, 1].max() + 1
    return np.meshgrid(np.arange(x_min, x_max, 0.01),
                       np.arange(y_min, y_max, 0.01))
# параметр min_samples_leaf указывает, при каком минимальном количестве
# элементов в узле он будет дальше разделяться
clf_tree = DecisionTreeClassifier(criterion='entropy',
                                  max_depth=3, random_state=17)
# обучаем дерево
clf_tree.fit(train_data, train_labels)
# немного кода для отображения разделяющей поверхности
xx, yy = get_grid(train_data)
predicted = clf_tree.predict(np.c_[xx.ravel(),
                                   yy.ravel()]).reshape(xx.shape)
plt.pcolormesh(xx, yy, predicted, cmap='autumn')
plt.scatter(train_data[:, 0], train_data[:, 1], c=train_labels, s=100,
            cmap='autumn', edgecolors='black', linewidth=1.5)
```

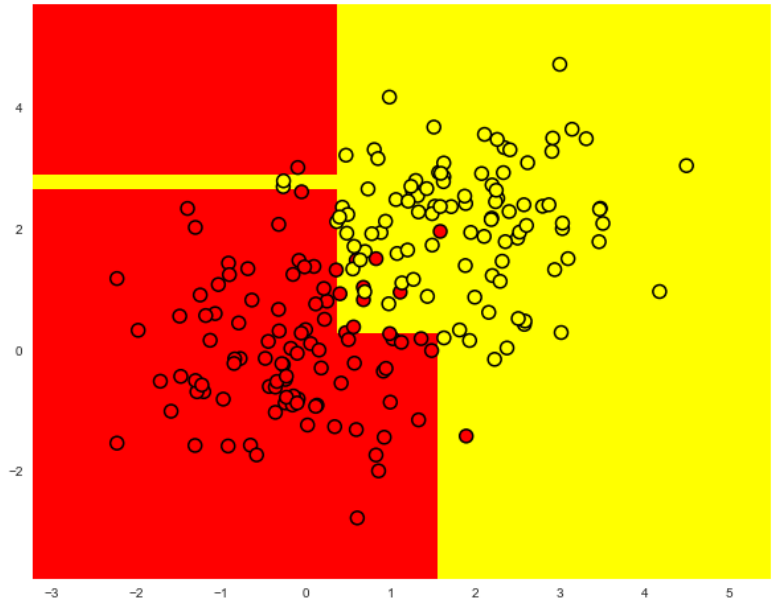


Рисунок 6.8

А как выглядит само построенное дерево? Видим, что дерево "нарезает" пространство на 7 прямоугольников (в дереве 7 листьев). В каждом таком прямоугольнике прогноз дерева будет константным, по превалированию объектов того или иного класса. Программный код для отображения дерева:

```
# используем .dot формат для визуализации дерева
from sklearn.tree import export_graphviz
export_graphviz(clf_tree, feature_names=['x1', 'x2'],
                out_file='../../img/small_tree.dot', filled=True)
# для этого понадобится библиотека pydot (pip install pydot)
!dot -Tpng '../../img/small_tree.dot' -o '../../img/small_tree.png'
```

Примечание: в данный программный код нужно внести очевидные изменения, чтобы получить изображение дерева решений.

Результатом выполнения данного программного кода является дерево решений (рис. 6.9).

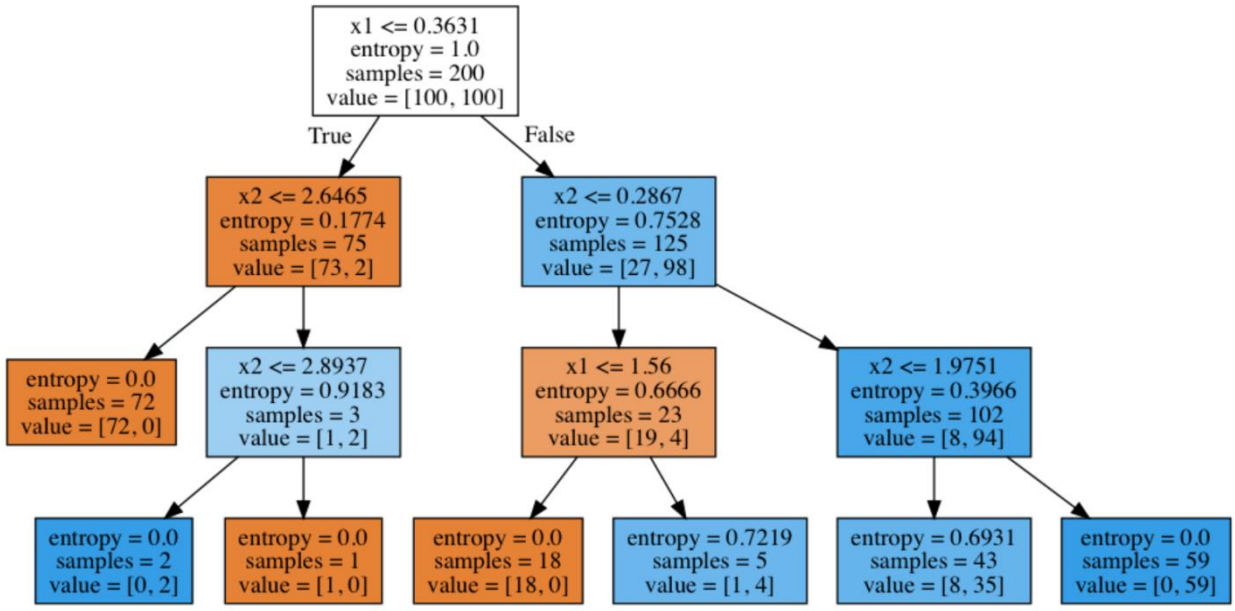


Рисунок 6.9. Дерево решений

Как "читается" такое дерево?

Вначале было 200 объектов, 100 – одного класса и 100 – другого. Энтропия начального состояния была максимальной – 1. Затем было сделано разбиение объектов на 2 группы в зависимости от сравнения признака x_1 со значением 0.3631 (найдите этот участок границы на рисунке выше, до дерева). При этом энтропия и в левой, и в правой группе объектов уменьшилась. И так далее, дерево строится до глубины 3. При такой визуализации чем больше объектов одного класса, тем цвет вершины ближе к темно-оранжевому и, наоборот, чем больше объектов второго класса, тем ближе цвет к темно-синему. В начале объектов одного класса поровну, поэтому корневая вершина дерева – белого цвета.