第5章

决策树

5.1 决策树算法概述

5.1.1 决策树算法的基本原理

决策树(Decision Tree)是一种特别简单的机器学习分类算法。决策树的想法来源于人类的决策过程,是在已知各种情况发生概率的基础上通过构成决策树来评价项目风险,判断其可行性的决策分析方法,是直观运用概率分析的一种图解法。由于这种决策分支画成图形很像一棵树的枝干,故称决策树。在机器学习中,决策树是一个预测模型,其代表的是对象属性与对象值之间的一种映射关系。

决策树可看作一个树状预测模型,它是由节点和有向分支组成的层次结构。树中包含3种节点,即根节点、内部节点、叶子节点。决策树只有一个根节点,是全体训练数据的集合。树中每个内部节点都是一个分裂问题:指定了对实例的某个属性的测试,它将到达该节点的样本按照某个特定的属性进行分割,并且该节点的每一个后继分支对应于该属性的一个可能值。每个叶子节点是带有分类标签的数据集合,即为样本所属的分类[1,2,3]。

为了便于读者理解,下面用实例的方法解释各概念及决策树算法的流程。假设一个应用为推断某个孩子是否出门玩耍,其相应的样本属性包括是否晴天、湿度大小、是否刮风,通过前期统计,带标签的数据如表5.1所示,序号1～6的数据为样本数据,序号为7的数据为待分类数据,即判断在该属性数据情况下是否出门。

表 5.1 孩子出门情况统计表

序 号	是否晴天	湿度大小	是否刮风	是否出门(标签)
1	是	大	否	不出门
2	是	小	否	出门
3	是	小	是	不出门
4	否	小	是	不出门
5	否	大	否	出门
6	否	大	是	不出门
7	是	小	否	?

通过表 5.1 建立决策树模型，如图 5.1 所示，从该图中可以看出，首先对数据整体样本(即根节点处)按照某一属性进行决策分支，形成中间节点，之后递归分支，直到样本划分到一类中，即形成叶子节点。对于表 5.1 中序号为 7 的待分类样本，将其代入决策树中，首先按是否晴天进行分支，当其属性值为“是”时，再依据其湿度值为“小”，判断是否刮风为“否”，可判断该数据划分到“出门”这一类中。

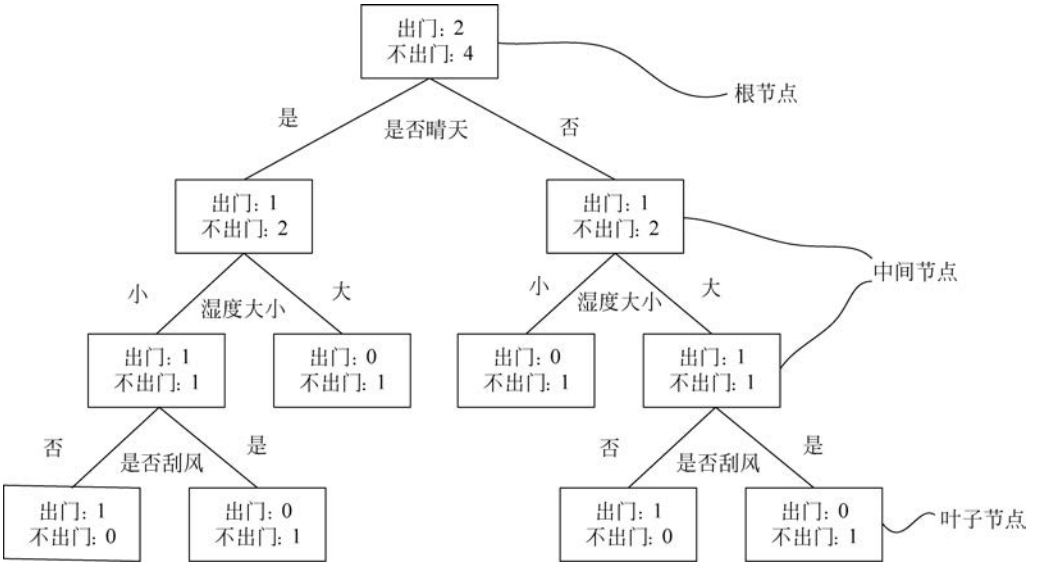

图 5.1 孩子出门决策树

决策树是一种十分常用的分类方法。其通过样本数据学习得到一个树形分类器，对于新出现的待分类样本能够给出正确的分类。对于创建决策树的过程，其步骤如下：

(1) 检测数据集中的每个样本是否属于同一分类。

(2) 如果是，则形成叶子节点，跳转到步骤(5)；如果否，则寻找划分数据集的最好特征(5.2 节将介绍方法)。

(3) 依据最好的特征划分数据集，创建中间节点。

(4) 对每一个划分的子集循环步骤(1)、(2)、(3)。

(5) 直到所有的最小子集都属于同一类时即形成叶子节点，则决策树建立完成。

5.1.2 决策树算法的特点

决策树算法的优点如下：

(1) 决策树易于理解和实现，用户在学习过程中不需要了解过多的背景知识，其能够直接体现数据的特点，只要通过适当的解释，用户就能够理解决策树所表达的意义。

(2) 速度快，计算量相对较小，且容易转化成分类规则。只要沿着根节点向下一直走到叶子节点，沿途分裂条件是唯一且确定的。

决策树算法的缺点主要是在处理大样本集时易出现过拟合现象，降低分类的准确性。

5.1.3 决策树剪枝

决策树是一种分类器，使用 ID3、C4.5 和 CART 等方法(5.2 节介绍)可以通过训练数据构建一个决策树。但是，算法生成的决策树非常详细并且庞大，每个属性都被详细地加以考虑，决策树的叶子节点所覆盖的训练样本都是绝对分类的。因此用决策树对训练样本进行分类时，会发现对于训练样本而言，这个树表现完好，误差率极低，且能够正确地对训练样本集中的样本进行分类。但是，训练样本中的错误数据也会被决策树学习，成为决策树的部分，并且由于过拟合，对于测试数据的表现并不佳，或者极差。

为解决上述出现的过拟合问题，需要对决策树进行剪枝处理。根据剪枝所出现的时间点不同，分为预剪枝和后剪枝。预剪枝是在决策树生成过程中进行的；后剪枝是在决策树生成之后进行的。后者的应用较广泛，而预剪枝具有使树的生长可能过早停止的缺点，因此应用较少。

1. 预剪枝(Pre-Pruning)

预剪枝指在构造决策树的同时进行剪枝。所有决策树的构建方法都是在无法进一步分支的情况下才会停止创建分支的过程，为了避免过拟合，可以设定一个阈值，当信息熵[①]减小到小于这个阈值时，即使还可以继续降低熵，也停止继续创建分支，而将其作为叶子节点。

2. 后剪枝(Post-Pruning)

后剪枝指在决策树构造完成后进行剪枝。剪枝的过程是对拥有同样父节点的一组节点进行检查，依据熵的增加量是否小于某一阈值决定叶子节点是否合并。后剪枝是目前最普遍的做法。

后剪枝的剪枝过程是删除一些子树，然后用其叶子节点代替，这个叶子节点所标识的类别通过大多数原则确定。所谓大多数原则，是指在剪枝过程中将一些子树删除而用叶子节点代替，这个叶子节点所标识的类别用这棵子树中大多数训练样本所属的类别(Majority Class)来标识。

---

① 信息熵，信息论之父克劳德·艾尔伍德·香农用数学语言阐明概率与信息冗余度的关系，5.2 节将详细介绍。

比较常见的后剪枝方法有代价复杂度剪枝(Cost Complexity Pruning,CCP)、错误率降低剪枝(Reduced Error Pruning,REP)、悲观剪枝(Pessimistic Error Pruning,PEP)、最小误差剪枝(Minimum Error Pruning,MEP)等。下面介绍前两种剪枝方法,为读者提供一定的剪枝思路。

1) 代价复杂度剪枝(CCP)

CCP 方法的基本思想是从决策树 T_0 通过剪枝的方式不断地修剪决策树,其形成一个子树的序列$\{T_0,T_1,\cdots,T_n\}$。其中 T_{i+1} 是 T_i 通过修剪关于训练数据集误差增加率最小的分支得来。对于决策树 T,假设其误差为 $R(T)$,叶子节点数为 $L(T)$,在节点 t 处修剪后,误差为 $R(T_t)$,叶子节点数为 $L(T_t)$,修剪前后误差增加 $R(T_t)-R(T)$,误差增加率为

$$\alpha=\frac{R(T_t)-R(T)}{L(T)-L(T_t)} \tag{5.1}$$

决策树经过不断修剪,直到误差增加率大于某一设定阈值,则修剪结束。下面利用具体实例进行讲解,假设依靠样本数据形成的决策树如图 5.2 所示。其中 A、B 为样本类,x、y、z 为属性,用 t_i 表示节点位置。

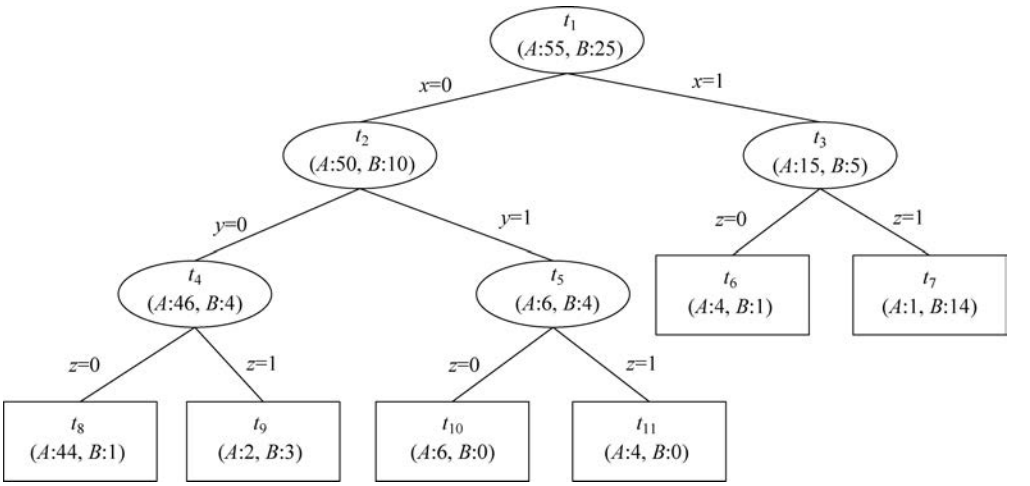

图 5.2 决策树

表 5.2 为图 5.2 所示决策树的剪枝数据的计算过程及结果。

表 5.2 决策树剪枝 α 计算值

T_0	$\alpha(t_4)=0.0125$	$\alpha(t_5)=0.050$	$\alpha(t_2)=0.0292$	$\alpha(t_3)=0.0375$
T_1	$\alpha(t_5)=0.050$	$\alpha(t_2)=0.0292$	$\alpha(t_3)=0.0375$	
T_2	$\alpha(t_3)=0.0375$			

从表 5.2 中可以看出,在原始决策树的 T_0 行,4 个非叶子节点中 t_4 的 α 值最小,因此裁剪 t_4 节点的分支,得到 T_1;在 T_1 行中,虽然 t_2 和 t_3 的 α 值相同,但是裁剪 t_2 能够得到更小的决策树,因此 T_2 是 T_1 裁剪 t_2 分支得到的。当然,假设误差增加率的阈值为

0.03，裁剪节点 t_4，形成决策树 T_1 后，裁剪结束。

2）错误率降低剪枝（REP）

REP 方法是根据错误率进行剪枝，如果决策树修剪前后子树的错误率没有下降，就可以认为该子树是可以修剪的。REP 方法需要用新的数据集进行效果验证。原因是如果用旧的数据集，不可能出现修剪后决策树的错误率比修剪前错误率高的情况。由于使用新的数据集没有参与决策树的构建，因此能够降低训练数据的影响，降低过拟合的程度，提高预测的准确率。

5.1.4 分类决策树与回归决策树

通过上述决策树的讲解，读者对于利用决策树进行分类问题的解决比较容易理解，对于利用决策树处理回归问题往往存在疑惑，下面通过两者的对比理解回归决策树。

以 C4.5 分类决策树为例，C4.5 分类决策树在每次分支时是穷举每个属性的每一个阈值，找到使得按照属性值≤阈值和属性值＞阈值分成的两个分支的熵最大的阈值，按照该标准分支得到两个新节点，用同样的方法继续分支，直到所有样本数据都被分入唯一的叶子节点，或达到预设的终止条件，若最终叶子节点中的标签类别不唯一，则以多数人的性别作为该叶子节点的性别。分类决策树常使用信息增益或增益比率来划分节点；每个节点样本的类别情况通过多数原则决定。

回归决策树的总体流程类似，区别在于，回归决策树的每个节点（不一定是叶子节点）都会得到一个预测值，以年龄为例，该预测值等于属于这个节点的所有人年龄的平均值。分支时穷举每个属性值的每一个阈值找最好的分割点，但衡量最好的标准不再是最大熵，而是最小化均方差。也就是被预测出错的人数越多，错的越离谱，均方差越大，通过最小化均方差能够找到最可靠的分支依据。分支，直到每个叶子节点上人的年龄都唯一或者达到预设的终止条件（例如叶子个数的上限），若最终叶子节点上人的年龄不唯一，则以该节点上所有人的平均年龄作为该叶子节点的预测年龄。回归决策树使用最大均方差划分节点；每个节点样本的均值作为测试样本的回归预测值。

5.2 基于决策树算法的算法改进

在 5.1 节介绍决策树的形成步骤时提到一个概念——数据集的最好特征，它是决策树形成时进行逐层划分的依据。为了描述这个最好特征，需要引入一个重要的概念——信息熵[4]。它是在 1948 年由香农提出的，用于表征信息量大小与其不确定性之间的关系。假设当前样本集合 D 中共包含 n 类样本，其中，第 k 类样本所占的比例为 p_k（$k=1,2,3,\cdots,n$），则 D 的信息熵的定义为：

$$\mathrm{Info}(D)=-\sum_{k=1}^{n}p_k\log_2 p_k \tag{5.2}$$

对于信息熵 $\mathrm{Info}(D)$，也可以称为信息的混乱程度，其数值越大，表示不确定性越大。

5.2.1 ID3 决策树

ID3 决策树算法是指依据上述的信息熵 $\mathrm{Info}(D)$ 进行分支的算法。为了表征决策树

在分支时属性选择的好坏，通过信息增益量(Information Gain)进行表示：

$$\text{Gain}(A)=\text{Info}(D)-\text{Info\_A}(D) \tag{5.3}$$

在该式中，Info(D)表示数据集 D 的信息量，Info_A(D)表示以属性 A 进行划分时得到的节点关于分类标签的信息量。一般而言，信息增益量越大，意味着使用属性 A 进行划分所获得的“纯度提升”越大。因此，可以用信息增益量来进行决策树的划分属性的选择。著名的 ID3 决策树学习算法就是以信息增益量为准则来选择划分属性的[5]。

同样用简单的实例进行讲解，以便读者能够具体地理解 ID3 算法的执行方法。假设样本数据如表 5.3 所示。

表 5.3　客户购买计算机的数据统计表

序　号	年　龄	收　入	学　生	信　用	购　买
1	青年	高	否	差	否
2	青年	高	否	优	否
3	中年	高	否	差	是
4	老年	中	否	差	是
5	老年	低	是	差	是
6	老年	低	是	优	否
7	中年	低	是	优	是
8	青年	中	否	差	否
9	青年	低	是	差	是
10	老年	中	是	差	是
11	青年	中	是	优	是
12	中年	中	否	优	是
13	中年	高	是	差	是
14	老年	中	否	优	否

依据上表数据进行决策树的建立，初学者开始一定迷惑，决策树第一次分支是选择年龄、收入、学生还是信用等级呢？读者在上文中了解了信息增益量这一概念，认为信息增益量大的属性越应该作为样本分支的属性。下面分别计算样本的信息熵 Info(D)以及以某属性进行划分时得到的节点的信息量 Info_A(D)。

$$\text{Info}(D)=-\frac{9}{14}\log_2\frac{9}{14}-\frac{5}{14}\log_2\frac{5}{14}=0.940$$

$$\text{Info\_年龄}(D)=\frac{5}{14}\left(-\frac{2}{5}\log_2\frac{2}{5}-\frac{3}{5}\log_2\frac{3}{5}\right)+\frac{4}{14}\left(-\frac{4}{4}\log_2\frac{4}{4}-\frac{0}{4}\log_2\frac{0}{4}\right)+$$
$$\frac{5}{14}\left(-\frac{3}{5}\log_2\frac{3}{5}-\frac{2}{5}\log_2\frac{2}{5}\right)=0.694$$

用相同的方法计算 Info_收入(D)=0.911，Info_学生(D)=0.798，Info_信用(D)=0.892，相应的信息增益量分别为 Gain(年龄)=0.246，Gain(收入)=0.029，Gain(学生)=0.151，Gain(信用)=0.048。通过比较大小，可知年龄属性的信息增益量最大，因此这次分支属性选择年龄属性。分支后形成的节点包含的数据作为新的数据集，依据上述方法，以此类推，即可建立整个决策树。

5.2.2 C4.5 决策树

C4.5 是机器学习算法中的一个分类决策树算法，它是决策树核心算法 ID3 的改进算法。

C4.5 决策树在实现决策树分支时，属性的选择是依靠参数“信息增益率”进行选择。信息增益率使用“分类信息值”将信息增益规范化[6]。对于属性 A，信息增益率通过下式进行计算：

$$\text{GainRatio}(A)=\frac{\text{Gain}(A)}{\text{SplitInfo}(A)} \tag{5.4}$$

式中，Gain(A)可通过 5.2.1 节的介绍进行计算，SplitInfo(A)表示分类信息值，其公式如下：

$$\text{SplitInfo}(A)=-\sum_{j=1}^{n}\frac{D_j}{D}\times\log_2\frac{D_j}{D} \tag{5.5}$$

式中，D 表示数据集中的样本个数，n 表示数据集中属性 A 所具有的属性值的个数，j 表示数据集中属性 A 所具有的属性值的标号；D_j 表示数据集中属性 A 的值等于编号 j 对应值的样本的个数。

下面借助 5.2.1 节中表 5.3 的数据对 C4.5 算法进行实例讲解。首先，依据式(5.5)计算。

$$\text{SplitInfo}(\text{年龄})=-\frac{5}{14}\log_2\frac{5}{14}-\frac{4}{14}\log_2\frac{4}{14}-\frac{5}{14}\log_2\frac{5}{14}=1.5774$$

$$\text{SplitInfo}(\text{收入})=-\frac{4}{14}\log_2\frac{4}{14}-\frac{6}{14}\log_2\frac{6}{14}-\frac{4}{14}\log_2\frac{4}{14}=1.5567$$

$$\text{SplitInfo}(\text{学生})=-\frac{7}{14}\log_2\frac{7}{14}-\frac{7}{14}\log_2\frac{7}{14}=1.0$$

$$\text{SplitInfo}(\text{信用})=-\frac{6}{14}\log_2\frac{6}{14}-\frac{8}{14}\log_2\frac{8}{14}=0.9852$$

依据 5.2.1 节计算的 Gain(A)以及式(5.4)，可计算 GainRatio(年龄)=0.156、GainRatio(收入)=0.0186、GainRatio(学生)=0.151、GainRatio(信用)=0.049。GainRatio(年龄)的信息增益率最大，所以选择这一属性作为进行分支的属性。

C4.5 算法继承了 ID3 算法的优点，并在以下几个方面对 ID3 算法进行了改进：

(1) 用信息增益率来选择属性，克服了用信息增益选择属性时偏向选择取值多的属性的不足。

(2) 在树构造过程中进行剪枝。

(3) 能够完成对连续属性的离散化处理。

(4) 能够对不完整数据进行处理。

C4.5 算法的优点是产生的分类规则易于理解，准确率较高；缺点是在构造树的过程中需要对数据集进行多次顺序扫描和排序，因而导致算法低效。此外，C4.5 算法只适合能够驻留于内存的数据集，当训练集大得无法在内存中容纳时程序无法运行。

5.2.3 分类回归树

分类回归树(Classification And Regression Tree,CART)也属于一种决策树,CART模型最早由 Breiman 等人提出,已经在统计领域和数据挖掘技术中普遍使用。CART 决策树是通过引入 GINI 指数(与信息熵的概念相似)增益 GINI_Gain(A)进行分支时属性的选择[7]。

同样,以表 5.2 中的数据为例进行计算。先以年龄属性为例介绍计算过程,其中,青年群体中有 3 个未购买,两个购买,得到

$$\text{GINI}(\text{年龄:青年})=1-\left[\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right]=0.48$$

中年群体中 4 个都购买了,得到

$$\text{GINI}(\text{年龄:中年})=1-\left(\frac{4}{4}\right)^2=0$$

老年群体中有 3 个购买,两个未购买,得到

$$\text{GINI}(\text{年龄:老年})=1-\left[\left(\frac{3}{5}\right)^2+\left(\frac{2}{5}\right)^2\right]=0.48$$

对于年龄属性,GINI 指数增益为

$$\text{GINI\_Gain}(\text{年龄})=\frac{5}{14}\times 0.48+\frac{4}{14}\times 0+\frac{5}{14}\times 0.48=0.3429$$

利用同样的方法得到 GINI_Gain(收入)=0.4405,GINI_Gain(学生)=0.3673,GINI_Gain(信用)=0.4048。选择最小的 GINI 指数增益作为分支属性,即选择年龄属性进行分支。

5.2.4 随机森林

随机森林指的是利用多棵决策树(类似一片森林)对样本进行训练并预测的一种分类器。该分类器最早由 Leo Breiman 和 Adele Cutler 提出,并被注册成了商标。在机器学习中,随机森林是一个包含多个决策树的分类器,并且其输出的类别是由个别树输出的类别的众数而定。这个方法则是结合 Breimans 的“Bootstrap aggregating”想法和 Ho 的“random subspace method”来建造决策树的集合。

5.3 决策树算法的 Python 实现

下面通过 Python 的实例方法演示决策树算法的具体应用。本章例子为鸢尾花的分类问题。鸢尾花有 3 种不同的类别,分别为 Iris Setosa、Iris Versicolour、Iris Virginica,可以根据它们的 4 个特征进行区分,分别为萼片长度、萼片宽度、花瓣长度、花瓣宽度。

读者可以通过 UCI 机器学习数据集网站下载鸢尾花数据集,网址为 https://archive.ics.uci.edu/ml/datasets/iris。为了方便,本节直接使用 scikit-learn 导入该数据集。scikit-learn 是 Python 语言中专门针对机器学习应用发展起来的一款开源框架。用户不仅可以通过 scikit-learn 导入数据集,还可以直接调用该框架集成好的决策树算法函

数——DecisionTreeClassifier()。在 DecisionTreeClassifier()函数中包含了 4 个参数,分别如下。

- criterion = gini/entropy:可以选择用基尼指数或者熵来做损失函数。
- splitter = best/random:用来确定每个节点的分裂策略,支持“最佳”或者“随机”。
- max_depth = int:用来控制决策树的最大深度,防止模型出现过拟合。
- min_samples_leaf = int:用来设置叶子节点上的最少样本数量,用于对树进行修剪。

具体代码如下(Dectree.py):

```
#采用决策树算法对鸢尾花数据集进行分类训练和测试
#使用 scikit-learn 框架导入数据集
from sklearn import datasets
#直接导入的数据集分布不均,是按顺序排列的,这会影响训练效果,因此用 train_test_split
#将数据的顺序打乱并拆分成训练集和测试集
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
#数据初始化函数,即数据的获取以及训练集和测试集的划分
def init_data():
  #加载 iris 数据集
  iris = datasets.load_iris()
  #获得 iris 的特征数据,即输入数据
  iris_feature = iris.data
  #获得 iris 的目标数据,即标签数据
  iris_target = iris.target
  #将数据集打乱并划分测试集和训练集,test_size=0.3 表示将整个数据的 30% 作为测试
  #集,其余 70% 作为训练集; random_state 为数据集打乱的程度
  feature_train, feature_test, target_train, target_test = train_test_split(iris_
feature, iris_target, test_size=0.3, random_state=42)
  return feature_train, feature_test, target_train, target_test
#主函数
if __name__ == '__main__':
  #获得数据
  feature_train, feature_test, target_train, target_test = init_data()
  #导入决策树分类函数
  from sklearn.tree import DecisionTreeClassifier
  #所有分类参数均设置为默认
  dt_model = DecisionTreeClassifier()
  #使用训练集训练模型
  dt_model.fit(feature_train,target_train)
  #用训练好的模型对测试集进行预测
  predict_results = dt_model.predict(feature_test)
  #导入评估方法
  from sklearn.metrics import accuracy_score
  #对测试的准确率进行评估
  print(accuracy_score(predict_results, target_test))
```

最后在终端输入“python Dectree.py”。

得到如下输出：

```
0.98
```

即测试的鸢尾花识别率为98%。输出结果可能是0.95～1.00的任意值，多次操作可能得到不同的结果，这取决于打乱数据以及划分训练和测试数据的随机性。

本章参考文献

[1] 刘小虎，李生.决策树的优化算法[J].软件学报，1998，9(10)：797-800.

[2] 栾丽华，吉根林.决策树分类技术研究[J].计算机工程，2004，30(9)：94-96.

[3] 周志华.机器学习[M].北京：清华大学出版社，2016.

[4] 冯少荣.决策树算法的研究与改进[J].厦门大学学报：自然科学版，2007，46(4)：496-500.

[5] 曲开社，成文丽，王俊红.ID3算法的一种改进算法[J].计算机工程与应用，2003，39(25)：104-107.

[6] 冯帆，徐俊刚.C4.5决策树改进算法研究[J].电子技术，2012，39(6)：1-4.

[7] 张立彬，张其前.基于分类回归树（CART）方法的统计解析模型的应用与研究[J].浙江工业大学学报，2002，30(4)：315-318.