Experiment 3 Decision Tree Learning for Classification

一、 Principle and Theory：

Decision trees classify instances by sorting them down the tree from root to leaf nodes. This tree-structured classifier partitions the input space of the data set recursively into mutually exclusive spaces. Following this structure, each training data is identified as belonging to a certain subspace, which is assigned a label, a value, or an action to characterize its data points. The decision tree mechanism has good transparency in that we can follow a tree structure easily in order to explain how a decision is made. Thus interpretability is enhanced when we clarify the conditional rules characterizing the tree.

Entropy of a random variable is the average amount of information generated by observing its value. Consider the random experiment of tossing a coin with probability of heads equal to 0.9, so that P(Head) = 0.9 and P(Tail) = 0.1. This provides more information than the case where P(Head) = 0.5 and P(Tail) = 0.5.

Entropy is used to evaluate randomness in physics, where a large entropy value indicates that the process is very random. The decision tree is guided heuristically according to the information content of each attribute. Entropy is used to evaluate the information of each attribute; as a means of classification. Suppose we have *m* classes, for a particular attribute, we denoted it by pi by the proportion of data which belongs to class *Ci* where *i = 1, 2, ... m.*

The entropy of this attribute is then:

$$Entropy = \sum_{i=1}^{m} -p_i \cdot \log_2 p_i$$

We can also say that entropy is a measurement of the impurity in a collection of training examples: larger the entropy, the more impure the data is. Based on entropy, Information Gain (IG) is used to measure the effectiveness of an attribute as a means of discriminating between classes.

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where all examples *S* is divided into several groups (i.e. *Sv* for *v* ∈ *Values(A)*) according to the value of *A*. It is simply the expected reduction of entropy caused by partitioning the examples according to this attribute.

二、**Objective**

The goals of the experiment are as follows:

(1) To understand why we use entropy-based measure for constructing a decision tree.

(2) To understand how Information Gain is used to select attributes in the process of building a decision tree.

(3) To understand the equivalence of a decision tree to a set of rules.

(4) To understand why we need to prune the tree sometimes and how can we prune? Based on what measure we prune a decision tree.

(5) To understand the concept of Soft Decision Trees and why they are important extensions to classical decision trees.

三、**Contents and Procedure**

(1) According to the above principle and theory , implement the code to calculate the information entropy of each attribute.

我们使用 sklearn.datasets 中的 iris 的鸢尾花数据集来学习决策树算法。

鸢尾花数据集是包含 150 个样本的三种不同种类的鸢尾花，每个鸢尾花有 4 个属性特征。

```
from sklearn.datasets import load_iris
iris = load_iris()
feature_names = ['花萼长度', '花萼宽度', '花瓣长度', '花瓣宽度']
class_names = ['山鸢尾花', '变色鸢尾花', '维吉尼亚鸢尾花']
```

根据信息熵的公式：

$$Entropy = \sum_{i=1}^{m} -p_i \cdot \log_2 p_i$$

```
# 计算每种鸢尾花的概率
iris_probability = np.divide(iris_count, 150)
# 根据公式算出鸢尾花的熵：
iris_h = -np.sum(iris_probability * np.log2(iris_probability))
```

我们可以得到：

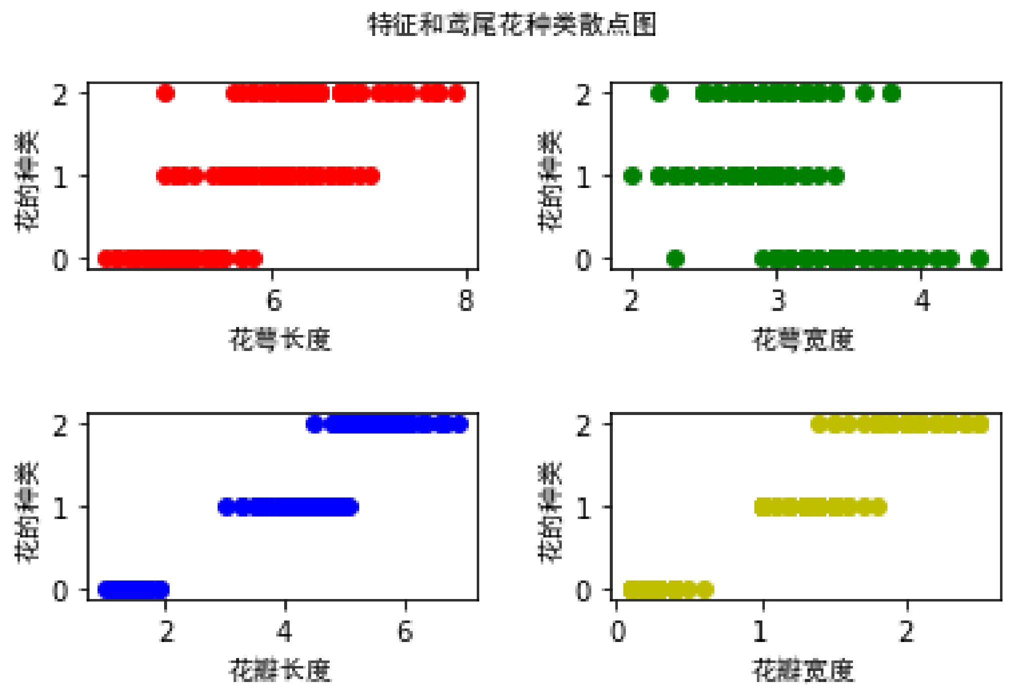
```
三种鸢尾花的数量分别为： [50. 50. 50.]
三种莺尾花的概率为： [0.33333333 0.33333333 0.33333333]
鸢尾花的熵为： 1.584962500721156
三种鸢尾花同等概率，熵为 1.584962500721156
```

(2) Select the most informative attribute from the Iris Dataset from the UCI Machine Learning Repository.

决策树通过引入鸢尾花的特征的信息来降低信息熵。我们画出鸢尾花的四个特征与鸢尾花种类的散点图来更好的识别出鸢尾花特征和种类之间的关系。

plt.scatter(features[:,i], target, c=colors[i])

如图：



我们可以从上述散点图看出，通过花瓣的宽度和花瓣的长度可以较好的识别出山鸢尾花。因此我们引入了花瓣宽度，可以降低鸢尾花的熵。

条件熵是已知鸢尾花特征的情况下，鸢尾花的种类的熵。条件熵数学公式如下：

$$\mathrm{H}(Y|X) \equiv \sum_{x\in\mathcal{X}} p(x)\,\mathrm{H}(Y|X=x)$$

以鸢尾花的花瓣宽度为例，我们以 0.8cm 为分界线，把鸢尾花花瓣宽度分为两类。样本中花瓣宽度小于 0.8 的有 50 个，大于 0.8 的有 100 个。

```
        #计算信息熵的函数
    def calcEntropy(target):
        entropy = entropy - p_i[i] * np.log2(p_i[i])
        # 计算条件熵的函数
    def calcConditionEntropy(feature, condition, target):
        # 每种特征类别的概率乘以该类别下的信息熵
        entropy = p_true * calcEntropy(target_true) + p_false *
calcEntropy(target_false)
        return entropy
    鸢尾花默认的信息熵 ： 1.584962500721156
    带花瓣宽度的条件熵 ： 0.6666666666666667
```

从上述结果可以看出，带花瓣宽度的条件熵的值比默认的信息熵更低一些，即不确定性更少了。

(3) Now consider the case of with continuous attributes or mixed attributes, how can we deal with the decision trees? Can you propose some approaches to do discretization?

当考虑连续属性的情况，需要根据如下方法划分分界点：

- 对特征的取值进行排序
- 将 target 有变动的地方作为可能的划分点
- 分界点定义为划分点的对应的两个特征取值的平均值
- 计算所有划分点的信息增益，选取最大的那个信息增益对应的值作为最终选择的分界点

```
# 生成特征的所有划分点，先对特征进行排序，然后将 target 有变动的地方作为分界点
def generate_feature_points(feature, target):
    split_value.append((f1[i] + f1[i - 1]) / 2)
    return np.array(split_value)
#计算特征的所有分界点的条件熵，返回最小的那个条件熵
def calc_feature_entropy(feature, target):
        entropy = calcConditionEntropy(feature, lambda f: f < p, target)
            if entropy < min_entropy:
                min_entropy = entropy
```

花瓣宽度的分界点与条件熵

```
In [22]: calc_feature_entropy(features[:,3], target)
Out[22]: (0.8, 0.6666666666666667)
```

花瓣长度的分界点与条件熵

```
In [23]: calc_feature_entropy(features[:,2], target)
Out[23]: (2.45, 0.6666666666666667)
```

(4) Find an appropriate data structure to represent a decision tree. Building the tree from the root to leaves based on the principal by using Information Gain guided heuristics.

信息增益表示不确定性减少的程度，其值等于信息熵减去条件熵。

$$IG(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

决策树的生成算法 ID3 就是根据信息增益的大小来选择节点的特征，用递归方法构建决策树的：

- 遍历所有特征的信息增益，选择信息增益最大的特征作为根节点的特征
- 由节点特征的不同分类建立子节点，再对各个子节点上述方法选择各子节点的特征
- 重复以上步骤直至所有特征的信息增益都很小或者没有特征可以选择为止

花瓣宽度的分界点为 0.8 时熵最小，此时的该特征的信息增益时最大的。我们根据 ID3 算法创建决策树。

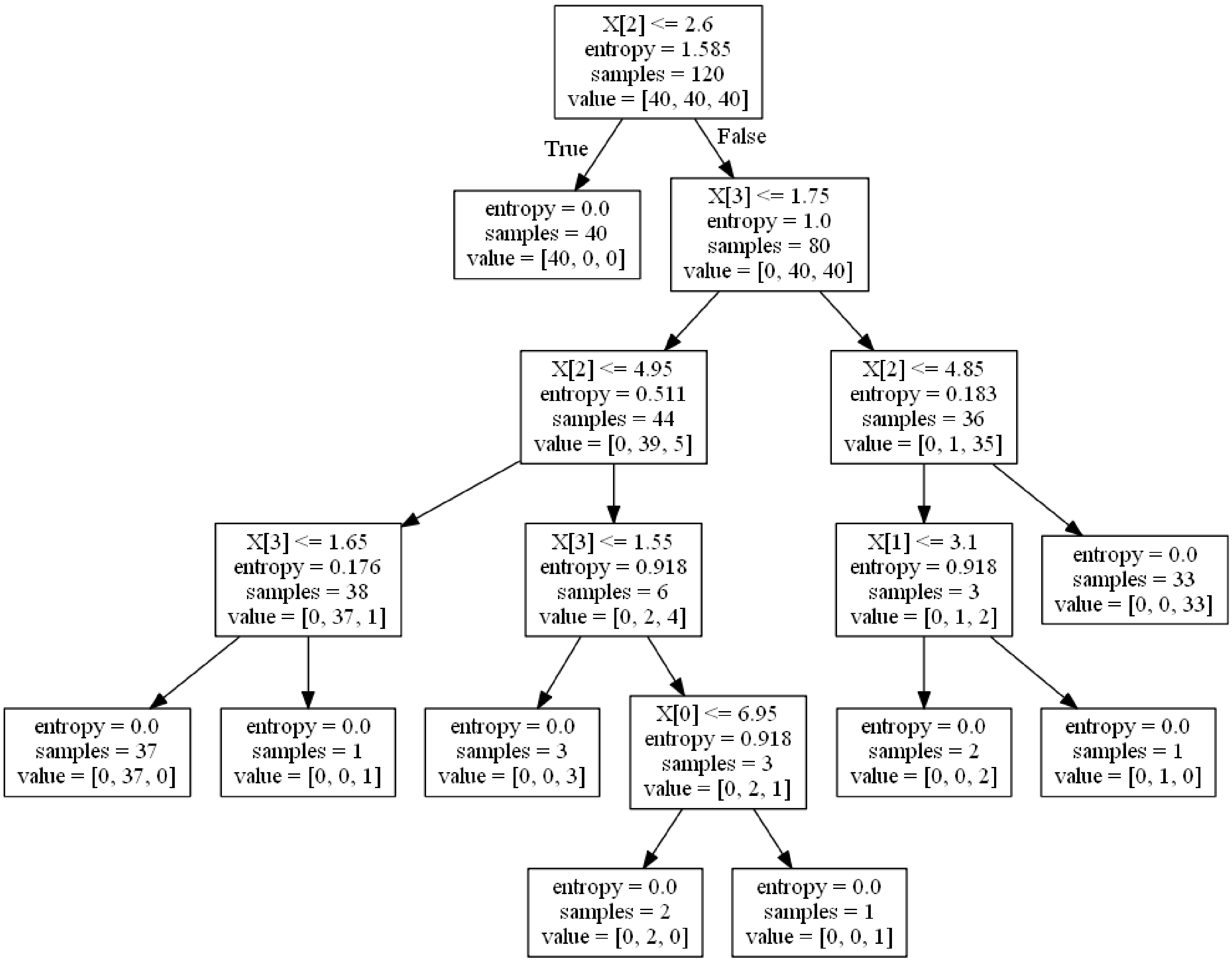
```
#从所有特征中选择出条件熵最小的特征
def select_feature(features, target):
    point, entropy = calc_feature_entropy(features[:, i], target)
        if entropy <= min_entropy:
            min_point = point
            min_entropy = entropy

 #递归构建决策树
def build_tree(features, target, idn):
    print('build tree node id %d, index %d, point %f, entropy %f, label %s ' %
          (idn, index, point, entropy, node.target_label))
```

决策树部分结果如下：

```
build tree node id 1, index 3, point 0.800000, entropy 0.666667, label 0
build tree node id 2, index 3, point 0.000000, entropy 0.000000, label 0
too low entropy :   0
```

可视化操作：

ID3 算法生成决策树：



叶子节点具有 value 属性。iris 有 3 个分类，故 value 有三个值。若第 1 个值比较大，认为是第 1 个分类；若第 2 个值比较大，认为是第 2 个分类；若第 3 个值比较大，认为是第 3 个分类。

(5) Is there a tradeoff between the size of the tree and the model accuracy? Is there existing an optimal tree in both compactness and performance?

决策树的深度过大，叶子节点过多，叶子节点含有的样本数过少，就需要想办法剪去这些树枝，从而得到一棵不高不胖的决策树。

节点依据信息增益和增益率扩展子节点，在这个过程中，验证集的整体准确率会随着节点的扩展发生变化。某次扩展如果导致验证集当前分类准确率降低，则舍弃，这便是预剪枝。

预剪枝不仅能降低过拟合风险，还大大降低了决策树的建树时间复杂度。但是，由于预剪枝使用贪心策略决定是否展开分支，不能全局权衡利弊，也给最终模型带来了欠拟合风险。

后剪枝是一种全局优化方法。当决策树建树完成也就是全部节点都展开后，后剪枝从下往上依次考查各个非叶子结点，考察方法和预剪枝一样，若当前节点的扩展在验证集上并未提高准确率，则舍弃。

后剪枝既降低了过拟合风险，又降低了欠拟合风险。但是，后剪枝由于是在生成完整决策树之后进行，因此模型的整体时间复杂度会比较高。

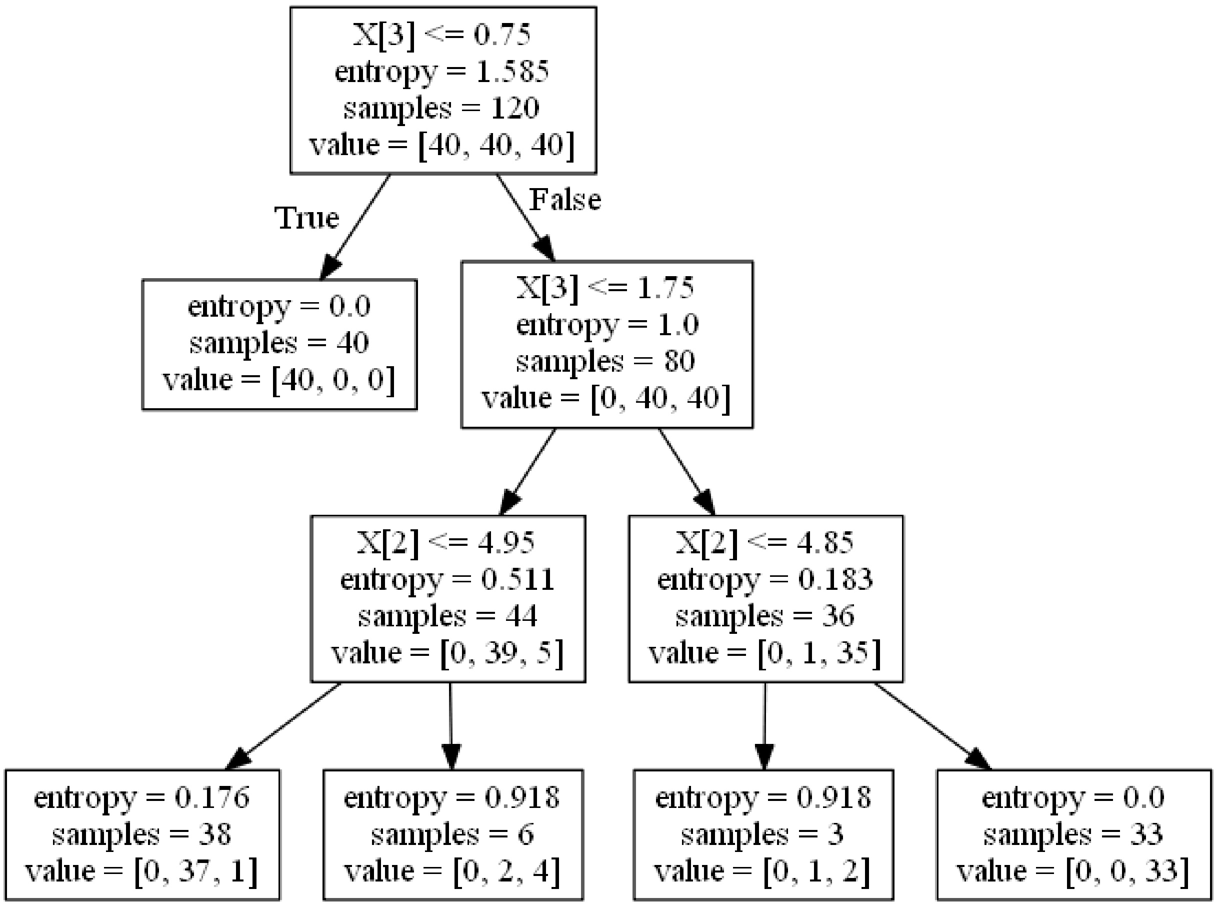
预剪枝决策树：

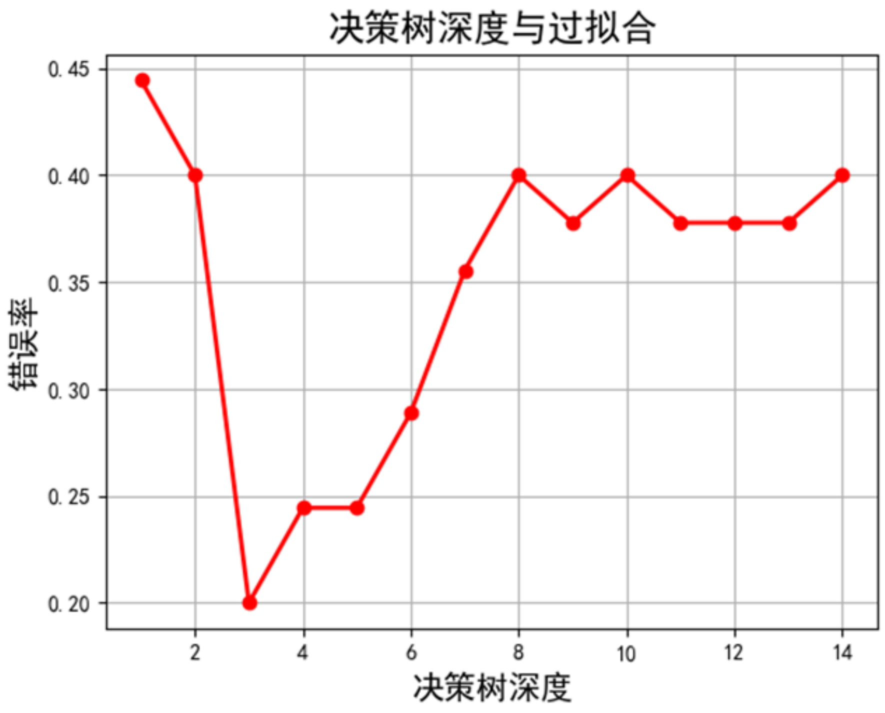
clf = tree.DecisionTreeClassifier()这个构建决策树的构造函数，带有参数常用的包括如下：

- criterion='entropy', 选用信息增益作为选择特征
- max_depth=3, 树的最大深度
- min_samples_split=2, 分裂点的样本个数
- min_samples_leaf =1, 叶子节点的样本个数
- max_leaf_nodes=None，最大的叶子节点数

可以看到这个构造函数的参数，都包括了以上阐述的预剪枝的策略，

如果参数的 max_depth = 3，那么得到的决策树如下所示：





我们可以看到，决策树深度为 3 时，决策树的紧凑性和性能最佳。

(6) For one data element, the classical decision tree gives a hard boundary to decide which branch to follow, can you propose a “soft approach” to increase the robustness of the decision tree?

决策树中每个节点表示某个对象，而每个分叉路径则代表某个可能的属性值。

随机森林由 N 棵决策树组成，每棵决策树都是一个基学习器，那么对于一个输入样本，N 棵树会有 N 个分类结果。而随机森林集成了所有的分类投票结果，将投票次数最多的类别指定为最终的输出。

```
#设定随机种子
np.random.seed(0)
#创建一个随机森林分类器
clf = RandomForestClassifier(n_jobs=2, random_state=0)
#训练分类器
clf.fit(train[features], y)
#创建一个 confusion matrix
pd.crosstab(test['species'], preds, rownames=['Actual Species'],
colnames=['Predicted Species'])
```

预测结果：

```
Predicted Species   setosa   versicolor   virginica
Actual Species
setosa                13          0            0
versicolor             0          5            2
virginica              0          0           12
```

对角线上的数是预测正确的数量。我们可以看到随机森林的正确率很高。同时克服了决策树的 hard boundary 的问题。

#查看特征重要性比重

list(zip(train[features], clf.feature_importances_))

四种特征重要性比重：

('sepal length (cm)', 0.11185992930506346),

('sepal width (cm)', 0.016341813006098178),

('petal length (cm)', 0.36439533040889194),

('petal width (cm)', 0.5074029272799464)

我们可以看到 petal width（花瓣宽度）具有最大的重要性比重，对于结果的影响最大。

(7) Compare to the Naïve Bayes, what are the advantages and disadvantages of the decision tree learning?

Naïve Bayes **优点：**

- 朴素贝叶斯模型发源于古典数学理论，有着坚实的数学基础，以及稳定的分类效率。
- 对小规模的数据表现很好，能个处理多分类任务，适合增量式训练。
- 面对孤立的噪声点，朴素贝叶斯分类器是健壮的。通过在建模和分类时忽略样例，朴素贝叶斯对缺失数据不太敏感，算法也比较简单，常用于文本分类。

缺点：

- 需要计算先验概率。
- 分类决策存在错误率。
- 对输入数据的表达形式很敏感。
- 输入变量必须都是条件独立的。

决策树的优点：

- 计算简单，易于理解，可解释性强；
- 比较适合处理有缺失属性的样本；
- 能够处理不相关的特征；
- 在相对较短的时间内能够对大型数据源做出可行且效果良好的结果。

缺点：

- 容易发生过拟合（随机森林可以很大程度上减少过拟合）；
- 忽略了数据之间的相关性；
- 对于那些各类别样本数量不一致的数据，在决策树当中,信息增益的结果偏向于那些具有更多数值的特征。