熵与决策树

Minimal, System, Insight, Programming

本章，我们将从信息学的知识开始，学习决策树（一种基于划分的非线性模型），决策树算法基于不同的分裂准则（ID3、C4.5、CART）有多种变体，本章需要理解不同决策树分裂差异及优势劣势，并使用sklearn内置决策树模型进行建模，观察模型表现，调节模型参数提升模型性能。

学习目标：

- 学习信息增益、信息增益率等概念及决策树算法发展过程及优劣势，
- 在Kaggle 泰坦尼克比赛使用 `sklearn.tree.DecisionTreeClassifier` 进行建模，查看官方文档，了解 `max_depth` 、 `min_samples_split` 参数含义，调节参数，提交需大于0.765。使用决策树建模流程与之前学习的线性回归、逻辑回归并没有本质区别，模型可以通过 `from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor` 进行导入.

1. 从数学开始

信息量(Information Content / Shannon information)

有个人告诉你说，他有一个巨大的秘密。

你很好奇，问道：什么秘密?

他答：明天太阳将从东方升起。

相信你的第一反映是，这个人不是刚从精神病院跑出来的吧?

为什么你会有这样的反映呢？ 因为太阳从东方升起是一个确定事件，不需要别人告诉你，你也可以非常准确的预测，他所谓的大秘密，不包含任何信息量。

如果需要我们用非常量化的方式来定义信息怎么办？我们从几个显而易见的角度来看看，信息量我们用I表示：

1. 显然，一个事情如果必然发生，那么告知你的消息蕴含的信息量为0.
2. 当一个发生概率极低事情发生，那么告知你的消息蕴含的信息量将非大，比如有个人告诉你明天彩票中奖号码，而且还真的就是那个号码.
3. 信息量与事情发生的概率负相关.
4. 信息量是非负的，最低为0.
5. 两个独立事件发生的概率为$p(x_1) * p(x_2)$, 其消息的信息量应该满足 $I(x_1, x_2) = I(x_1) + I(x_2)$，可以看到，概率是乘法，但是信息量应该是加法，两个信息一起告诉你的信息量理所应当等于连个信息的信息量综合。

信息量是关于事件发生概率的函数，$I(x) = -log(p(x))$函数完美的符合上述论述：

1. $log(1) = 0$
2. 当概率p极低时候，$-log(p)$极大
3. -log(p) 与 p的确负相关
4. 由于0<= p <=1， 所以$-log(p)$一定是非负的

5. $-log(p1 * p2) = -log(p1) - log(p2)$

我们可以把不同概率对应的信息量画出来观察一下：

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline
```

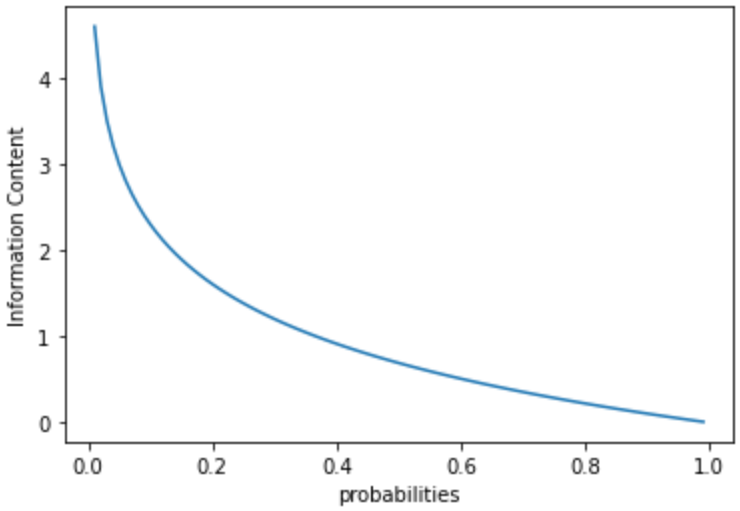
In [6]:
```
probabilities = np.arange(0, 1, 0.01)

I = -np.log(probabilities)

plt.plot(probabilities, I)
plt.xlabel("probabilities")
plt.ylabel("Information Content")
```

```
/tmp/ipykernel_1318164/636599697.py:3: RuntimeWarning: divide by zero encountered in log
  I = -np.log(probabilities)
```

Out[6]:
```
Text(0, 0.5, 'Information Content')
```



信息熵(Information Entropy)

理解了信息量，信息熵就很简单了，一个随机事件有多种发生的可能（太阳从东方升起或者从西方升起），所有可能情况的信息量的期望，就是信息熵：

$$H(x) = \sum_{i=1}^{m} p(x_i) I(x_i)$$

这里通过一个例子来理解信息熵,根据以往的天气,温度,湿度,有风的情况下统计了出去玩和不出去玩的情况如下:

weather	temperat	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes

weather	temperat	humidity	windy	play
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

由表格信息知道根据没有任何提示的情况下可以知道这一天玩的概率为9/14，不出去的概率为5/14,通过信息熵的公式得:

$$H(x) = -\frac{9}{14}log_2\frac{9}{14} - \frac{5}{14}log_2\frac{5}{14} = 0.940$$

但如果14天中13天都出去玩,只有一天不出去玩则对应的信息熵为:

$$H(x) = -\frac{13}{14}log_2\frac{13}{14} - \frac{1}{14}log_2\frac{1}{14} = 0.371$$

条件熵

信息熵是考虑该随机变量的所有可能取值，即所有可能发生事件所带来的信息量的期望。公式如下:

$$H(x) = \sum_{i=1}^{m} p(x_i)I(x_i)$$

我们的条件熵的定义是：定义为X给定条件下，Y的条件概率分布的熵对X的数学期望,这个还是比较抽象，下面我们解释一下：设有随机变量（X,Y），其联合概率分布为:

$$p(X = x_i, Y = y_i) = p_{ij}, i = 1, 2, \ldots, n; j = 1, 2, \ldots, m$$

条件熵H（Y|X）表示在已知随机变量X的条件下随机变量Y的不确定性,条件熵的公式如下:

$$H(Y|X) = \sum_{x\epsilon X} p(x)H(Y|X = x)$$

当我们通过天气划分时:当前统计结果中sunny、overcast、rainy的概率分别为：5/14、4/14、5/14
在sunny时玩得概率为2/5,对应的信息熵为:

$$H(play|weather = sunny) = -\frac{2}{5}log_2\frac{2}{5} - \frac{3}{5}log_2\frac{3}{5} = 0.971$$

overcast时玩的概率为1,对应的信息熵为:

$$H(play|weather = overcast) = 0$$

rainy时玩的概率为3/5,对应的信息熵为:

$$H(play|weather = rainy) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5} = 0.971$$

有了上面的铺垫之后，我们终于可以计算我们的条件熵了，我们现在需要求： H（Y|X = weather） 也就是说，我们想要求出当已知天气的条件下的条件熵。 根据公式我们可以知道，已知天气下的条件熵为：

$$H(play|weather) = 5/14 * 0.971 + 4/14 * 0 + 5/14 * 0.971 = 0.693$$

条件熵总结

其实条件熵意思是按一个新的变量的每个值对原变量进行分类，比如上面这个题把玩与不玩按天气分成了三类。
然后在每一个小类里面，都计算一个信息熵，然后每一个信息熵乘以各个类别的概率，然后求和。
我们用另一个变量对原变量分类后，因为新增了X的信息原变量的不确定性就会减小了

信息增益

我们前面说了，信息熵是代表随机变量的不确定度，条件熵代表在某一个条件下，随机变量的不确定度。 信息增益是：信息熵-条件熵。 换句话说，信息增益代表了在一个条件下，信息不确定性减少的程度。

通过信息熵和条件熵的例子,可以求得随机变量 `play` 的信息熵为：

$$H(play) = 0.940$$

$$H(play|weather) = 0.693$$

那么我们知道信息熵与条件熵相减就是我们的信息增益，为

$$Gain(weather) = 0.940 - 0.693 = 0.247$$

结论

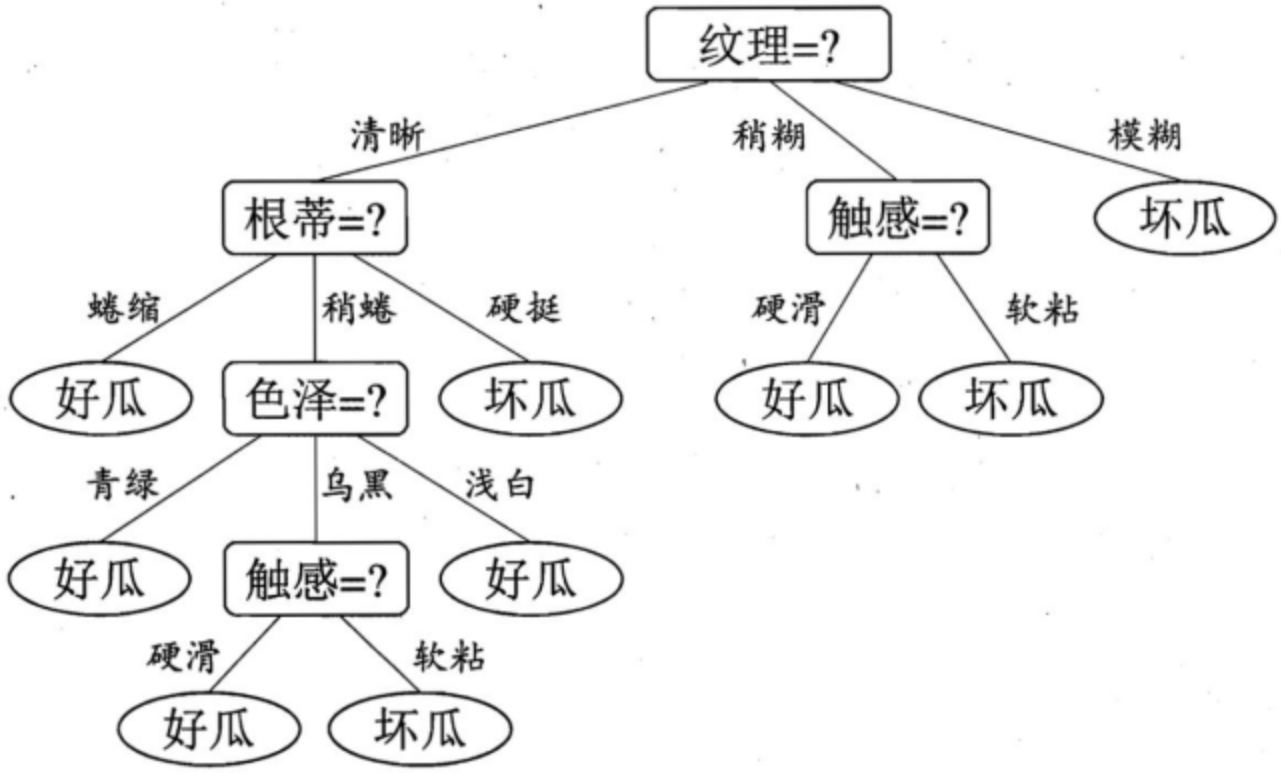
我们可以看出，如果什么都不知道的话，出去玩的不确定性有0.940。当我们知道天气的信息后，不确定度减少了0.247.也就是说，天气这个信息对于出去玩和不出去玩有一定的判断作用，这也就是决策树的原理，**通过树的分叉**创造各种条件，获得信息增益，降低目标的不确定度。

决策树认识

算法思想

决策树（decision tree）是一个树结构（可以是二叉树或非二叉树）,每一次分叉代表对某个特征属性的一次划分。预测结果为叶子节点（没有子节点的节点）所有样本标签的**平均值（回归问题）**或**样本最多的标签类别概率（分类问题）**。

下图展示了一个分类问题（好瓜、坏瓜分类）的决策树案例，在根节点处，根据纹理属于清晰、模糊、稍糊进行划分，然后再一次根据其他属性进行划分。最终在叶子节点统计属于该节点中的样本好瓜坏瓜的比例，将比例最高的类别作为预测结果，最高类别数量的占比作为预测概率值。



2. 基于信息增益的ID3决策树

决策树ID3算法的信息论基础

ID3是一种基于信息增益的分裂策略。在前面数学知识学习过程中，我们提到$H(Y)$度量了Y的不确定性，条件熵H(Y|X)度量了我们在知道X以后Y剩下的不确定性，已经条件X带来的信息增益$GAIN = H(Y) - H(Y|X)$。ID3算法就是用信息增益来判断，选择当前信息增益最大的特征及对应的分裂点。ID3是多叉树，且只能处理类别型（离散）特征，每次基于最大化信息增益GAIN选择分裂特征，特征有几个不同的类别，就会形成多少个分叉。

决策树ID3算法的思路

上面提到ID3算法就是用信息增益大小来判断当前节点应该用什么特征来构建决策树，用计算出的信息增益最大的特征来建立决策树的当前节点。这里我们举一个信息增益计算的具体的例子。比如我们有15个样本D，标签为0或者1。其中有9个输出为1， 6个输出为0。 样本中有个特征A，取值为A1，A2和A3。在取值为A1的样本的输出中，有3个输出为1， 2个输出为0，取值为A2的样本输出中,2个输出为1,3个输出为0， 在取值为A3的样本中，4个输出为1，1个输出为0.

样本D的熵为：

$$H(D) = -(\frac{9}{15}log_2\frac{9}{15} + \frac{6}{15}log_2\frac{6}{15}) = 0.971$$

样本D在特征下的条件熵为：

$$= -\frac{5}{15}(\frac{3}{5}log_2\frac{3}{5} + \frac{2}{5}log_2\frac{2}{5}) - \frac{5}{15}(\frac{2}{5}log_2\frac{2}{5} + \frac{3}{5}log_2\frac{3}{5}) - \frac{5}{15}(\frac{4}{5}log_2\frac{4}{5} + \frac{1}{5}log_2\frac{1}{5}) = 0.888$$

对应的信息增益为

$$GAIN(D,A) = H(D) - H(D|A) = 0.083$$

ID3 算法的核心思想就是以信息增益来度量特征选择，选择信息增益最大的特征进行分裂。算法采用自顶向下的贪婪搜索遍历可能的决策树空间。 其大致步骤为：

1. 初始化特征集合和数据集合；
2. 计算数据集合信息熵和所有特征的条件熵，选择信息增益最大的特征作为当前决策节点进行分裂；
3. 删除上一步使用的特征，在分裂后的每个节点上，执行第2步，直到该节点计算的信息增益小于某个阈值ϵ，则该节点停止分裂；
4. 所有节点停止分裂后，计算叶子节点（无子节点的节点）样本属于各个类别的概率。

在预测时，根据样本属性将样本归属到某个叶子节点，使用叶子节点各个类别的概率作为预测结果。

决策树ID3算法的不足

ID3算法虽然提出了新思路，但是还是有很多值得改进的地方。

1. ID3没有考虑连续特征，比如长度，密度都是连续值，无法在ID3运用。这大大限制了ID3的用途。
2. ID3采用信息增益大的特征优先建立决策树的节点。很快就被人发现，在相同条件下，取值比较多的特征比取值少的特征信息增益大。比如一个变量有2个值，各为1/2，另一个变量为3个值，各为1/3，其实他们都是完全不确定的变量，但是取3个值的比取2个值的信息增益大。
3. ID3算法对于缺失值的情况没有做考虑
4. 只通过阈值ϵ一个参数来控制过拟合

ID3 算法的作者昆兰基于上述不足，对ID3算法做了改进为C4.5算法.

3. C4.5决策树算法

上一节我们讲到ID3算法有四个主要的不足，

1. 不能处理连续特征，
2. 信息增益作为标准容易偏向于取值较多的特征，
3. 缺失值处理的问
4. 过拟合问题。 昆兰在C4.5算法中改进了上述4个问题。

- 对于第一个问题，不能处理连续特征， C4.5的思路是将连续的特征离散化。比如m个样本的连续特征A有m个，从小到大排列为$a1,a2,...,am$,则C4.5取相邻两样本值的平均数，一共取得m-1个划分点，其中第i个划分点Ti表示Ti表示为：$T_i = \frac{a_i + a_{i+1}}{2}$。对于这m-1个点，分别计算以该点作为二元分类点时的信息增益。选择信息增益最大的点作为该连续特征的二元离散分类点。比如取到的增益最大的点为at,则小于at的值为类别1，大于at的值为类别2，这样我们就做到了连续特征的离散化。要注意的是，与离散属性不同的是，如果当前节点为连续属性，则该属性后面还可以参与子节点的产生选择过程。

- 第二个问题，信息增益作为标准容易偏向于取值较多的特征的问题。作者引入一个信息增益比的变量$IR(X,Y)$，它是信息增益和特征熵的比值。表达式如下：

$$I_R(D,A) = \frac{I(D,A)}{H_A(D)}$$

  其中D为样本特征输出的集合，A为样本特征，对于特征熵$HA(D)$, 表达式如下：

$$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|}$$

其中n为特征A的类别数，Di为特征A的第i个取值对应的样本个数。|D|为样本个数。 特征数越多的特征对应的特征熵越大，它作为分母，可以校正信息增益容易偏向于取值较多的特征的问题。

- 第三个缺失值处理的问题，主要需要解决的是两个问题:
  一: 是在样本某些特征缺失的情况下选择划分的属性，
  二: 是选定了划分属性，对于在该属性上缺失特征的样本的处理。
  对于第一个子问题，对于某一个有缺失特征值的特征A。C4.5的思路是将数据分成两部分，对每个样本设置一个权重（初始可以都为1），然后划分数据，一部分是有特征值A的数据D1，另一部分是没有特征A的数据D2. 然后对于没有缺失特征A的数据集D1来和对应的A特征的各个特征值一起计算加权重后的信息增益比，最后乘上一个系数，这个系数是无特征A缺失的样本加权后所占加权总样本的比例。
  对于第二个子问题，可以将缺失特征的样本同时划分入所有的子节点，不过将该样本的权重按各个子节点样本的数量比例来分配。比如缺失特征A的样本a之前权重为1，特征A有3个特征值A1,A2,A3。 3个特征值对应的无缺失A特征的样本个数为2,3,4.则a同时划分入A1，A2，A3。对应权重调节为2/9,3/9, 4/9。

- 第4个问题，C4.5引入了正则化系数进行初步的剪枝。具体方法这里不讨论。

除了上面的4点，C4.5和ID3的思路区别不大。

决策树C4.5算法的不足与思考

C4.5虽然改进或者改善了ID3算法的几个主要的问题，仍然有优化的空间。
1). 依然非常容易过拟合
2). C4.5生成的是多叉树，即一个父节点可以有多个节点。很多时候，在计算机中二叉树模型会比多叉树运算效率高。如果采用二叉树，可以提高效率。
3). C4.5只能用于分类，如果能将决策树用于回归的话可以扩大它的使用范围。
4). C4.5由于使用了熵模型，里面有大量的耗时的对数运算,如果是连续值还有大量的排序运算。如果能够加以模型简化可以减少运算强度但又不牺牲太多准确性的话，那就更好了。

4. Cart 树

基尼指数（基尼不纯度）

绝大部分情况下熵(entropy) 和基尼指数(Gini Index)在决策树节点分裂时做出的决策都是等价的。那为啥存在两种常用算法呢?其实是因为一种使用了熵，而另一种使用了基尼指数，两个工作都很有开创性就都保留了下来。
先看一下如何定义节点分裂时的不纯度函数(impurity) 有三种(假设有k个类别) :

1. 误分率：把当前节点n下所有样本都划分为c类的误分率，也就是

$$1 - \max_{c \epsilon [1,k]} \frac{c}{n}$$

2. 熵：$H(x) = -\sum_{i=1}^{m} p(x_i) log p(x_i)$, p(x_i)代表了当前节点n中属于c类的比例
3. 基尼指数：$Gini(a) = \sum_{c=1}^{k} p(x_i) * (1 - p(x_i))$

不难看出，三个函数均为凸函数(convex function)，只不过误分率(函数1) 是分段线性函数(piece-wise linear)，有时候节点分裂会无法降低不纯度。所以函数2和3一般是常采用的手段，它们的优势如下:

1. 二者均为凸函数
2. 二者都可以微分所以便于数值计算
3. 二者都可以代表的函数1的误差上界(upper bound) 正因为它们都是光滑凸函数且为训练误分函数的错误.上界，所以不仅保证了每次节点分裂整体的不纯度函数会下降且更适合运算。在绝大部分情况下，二者都是等价的。如果非要说不同的话，就是熵的计算会需要求log，所以可能预算开销更大。但是求log是防止计算溢出的利器，特别适合用于处理极小概率的情况，所以并非只有缺点。 基尼指数是信息熵的1阶泰勒展开；

CART分类树算法的最优特征选择方法

我们知道，在ID3算法中我们使用了信息增益来选择特征，信息增益大的优先选择。在C4.5算法中，采用了信息增益比来选择特征，以减少信息增益容易选择特征值多的特征的问题。但是无论是ID3还是C4.5,都是基于信息论的熵模型的，这里面会涉及大量的对数运算。能不能简化模型同时也不至于完全丢失熵模型的优点? CART分类树算法使用基尼系数来代替信息增益比，基尼系数代表了模型的不纯度，基尼系数越小，则不纯度越低，特征越好。这和信息增益(比)是相反的。 具体的，在分类问题中，假设有K个类别，第k个类别的概率为pkpk,则基尼系数的表达式为：

$$Gini(p) = \sum_{k=1}^{K} p_k(1-p_k) = 1 - \sum_{k=1}^{K} p_k^2$$

如果是二类分类问题，如果属于第一个样本输出的概率是p，则基尼系数的表达式为：

$$Gini(p) = 2p(1-p)$$

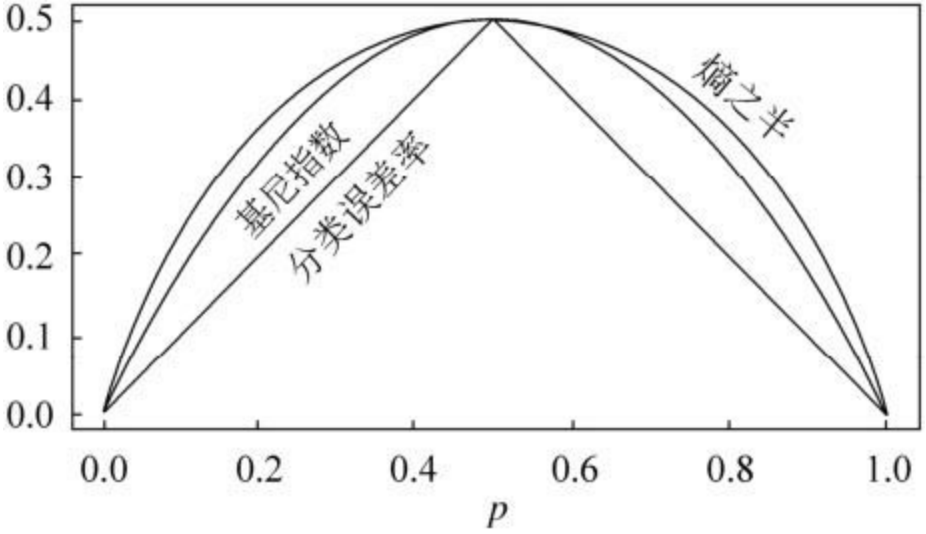
对于个给定的样本D,假设有K个类别, 第k个类别的数量为Ck,则样本D的基尼系数表达式为：

$$Gini(D) = 1 - \sum_{k=1}^{K}(\frac{|C_k|}{|D|})^2$$

特别的，对于样本D,如果根据特征A的某个值a,把D分成D1和D2两部分，则在特征A的条件下，D的基尼系数表达式为：

$$Gini(D,A) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2)$$

可以比较下基尼系数表达式和熵模型的表达式，二次运算比对数简单很多.尤其是二类分类的计算，更加简单。但是和熵模型的度量方式比，基尼系数对应的误差有多大呢？对于二类分类，基尼系数和熵之半的曲线如下：



从上图可以看出，基尼系数和熵之半的曲线非常接近，仅仅在45度角附近误差稍大。因此，基尼系数可以做为熵模型的一个近似替代。而CART分类树算法就是使用的基尼系数来选择决策树的特征。同时，为了进一步简化，CART分类树算法每次仅仅对某个特征的值进行二分，而不是多分，这样CART分类树算法建立起来的是二叉树，而不是多叉树。这样一可以进一步简化基尼系数的计算，二可以建立一个更加优雅的二叉树模型。

CART分类树算法对于连续特征和离散特征处理的改进

对于CART分类树连续值的处理问题，其思想和C4.5是相同的，都是将连续的特征离散化。唯一的区别在于在选择划分点时的度量方式不同，C4.5使用的是信息增益比，则CART分类树使用的是基尼系数。 具体的思路如下，比如m个样本的连续特征A有m个，从小到大排列为$a1,a2,...,am$，则CART算法取相邻两样本值的平均数，一共取得m-1个划分点，其中第i个划分点Ti表示Ti表示为：

$$T_i = \frac{a_i + a_{i+1}}{2}$$

对于这m-1个点，分别计算以该点作为二元分类点时的基尼系数。选择基尼系数最小的点作为该连续特征的二元离散分类点。比如取到的基尼系数最小的点为at,则小于at的值为类别1，大于at的值为类别2，这样我们就做到了连续特征的离散化。要注意的是，与ID3或者C4.5处理离散属性不同的是，如果当前节点为连续属性，则该属性后面还可以参与子节点的产生选择过程。 对于CART分类树离散值的处理问题，采用的思路是不停的二分离散特征。 对于ID3或者C4.5，如果某个特征A被选取建立决策树节点，如果它有A1,A2,A3三种类别，我们会在决策树上一下建立一个三叉的节点。这样导致决策树是多叉树。但是CART分类树使用的方法不同，他采用的是不停的二分.

例如:CART分类树会考虑把A分成$\{A1\}$和$\{A2,A3\}$，$\{A2\}$和$\{A1,A3\}$，$\{A3\}$和$\{A1,A2\}$三种情况，找到基尼系数最小的组合，比如$\{A2\}$和$\{A1,A3\}$，然后建立二叉树节点，一个节点是A2对应的样本，另一个节点是{A1,A3}对应的节点。同时，由于这次没有把特征A的取值完全分开，后面我们还有机会在子节点继续选择到特征A来划分A1和A3。这和ID3或者C4.5不同，在ID3或者C4.5的一棵子树中，离散特征只会参与一次节点的建立。

CART分类树算法的具体流程

上面介绍了CART算法的一些和C4.5不同之处， 算法输入是训练集D，基尼系数的阈值，样本个数阈值。 输出是决策树T。 我们的算法从根节点开始，用训练集递归的建立CART树。

1. 对于当前节点的数据集为D，如果样本个数小于阈值或者没有特征，则返回决策子树，当前节点停止递归。
2. 计算样本集D的基尼系数，如果基尼系数小于阈值，则返回决策树子树，当前节点停止递归。
3. 计算当前节点现有的各个特征的各个特征值对数据集D的基尼系数。缺失值的处理方法和C4.5算法里描述的相同。

4. 在计算出来的各个特征的各个特征值对数据集D的基尼系数中，选择基尼系数最小的特征A和对应的特征值a。根据这个最优特征和最优特征值，把数据集划分成两部分D1和D2，同时建立当前节点的左右节点，做节点的数据集D为D1，右节点的数据集D为D2。
5. 对左右的子节点递归的调用1-4步，生成决策树。 对于生成的决策树做预测的时候，假如测试集里的样本A落到了某个叶子节点，而节点里有多个训练样本。则对于A的类别预测采用的是这个叶子节点里概率最大的类别。

CART回归树算法

CART回归树和CART分类树的算法大部分是类似的，所以这里我们只介绍CART回归树和CART分类树的建立算法不同的地方。 首先，我们要知道什么是回归树，什么是分类树。两者的区别在于样本输出，如果样本输出是离散值，那么这是一颗分类树。如果果样本输出是连续值，那么那么这是一颗回归树。 除了概念的不同，CART回归树和CART分类树的建立和预测的区别主要有下面两点：

1. 连续值的处理方法不同
2. 决策树建立后做预测的方式不同。

对于连续值的处理，我们知道CART分类树采用的是用基尼系数的大小来度量特征的各个划分点的优劣情况。这比较适合分类模型，但是对于回归模型，我们使用了常见的和方差的度量方式，CART回归树的度量目标是，对于任意划分特征A，对应的任意划分点s两边划分成的数据集D1和D2，求出使D1和D2各自集合的均方差最小，同时D1和D2的均方差之和最小所对应的特征和特征值划分点。表达式为：

$$\underbrace{min}_{A,s}\left[\underbrace{min}_{c_1}\sum_{x_i\in D_1(A,s)}(y_i-c_1)^2+\underbrace{min}_{c_2}\sum_{x_i\in D_2(A,s)}(y_i-c_2)^2\right]$$

其中，c1为D1数据集的样本输出均值，c2为D2数据集的样本输出均值。 对于决策树建立后做预测的方式，上面讲到了CART分类树采用叶子节点里概率最大的类别作为当前节点的预测类别。而回归树输出不是类别，它采用的是用最终叶子的均值或者中位数来预测输出结果。 除了上面提到了以外，CART回归树和CART分类树的建立算法和预测没有什么区别。

CART算法小结

上面我们对CART算法做了一个详细的介绍，CART算法相比C4.5算法的分类方法，采用了简化的二叉树模型，同时特征选择采用了近似的基尼系数来简化计算。当然CART树最大的好处是还可以做回归模型，这个C4.5没有。下表给出了ID3，C4.5和CART的一个比较总结。

算法	支持模型	树结构	特征选择	连续值处理	缺失值处理	剪枝
ID3	分类	多叉树	信息增益	不支持	不支持	不支持
C4.5	分类	多叉树	信息增益比	支持	支持	支持
CART	分类，回归	二叉树	基尼系数，均方差	支持	支持	支持

CART算法还有什么提升空间?

1. 无论是ID3, C4.5还是CART，在做特征选择的时候都是选择最优的一个特征来做分类决策，但是大多数，分类决策不应该是由某一个特征决定的，而是应该由一组特征决定的。这样决策得到的决策树更加准确。这个决策树叫做多变量决策树(multi-variate decision tree)。在选择最优特征的时候，多变量决策树不是选择某一个最优特征，而是选择最优的一个特征线性组合来做决策。这个算法的代表是OC1，这里不多介绍。

2. 如果样本发生一点改动，就会导致树结构的剧烈改变。这可以通过集成学习里面的随机森林之类的方法解决。

决策树算法小结

决策树算法作为一个大类别的分类回归算法的优缺点如下,首先我们看看决策树算法的**优点**:

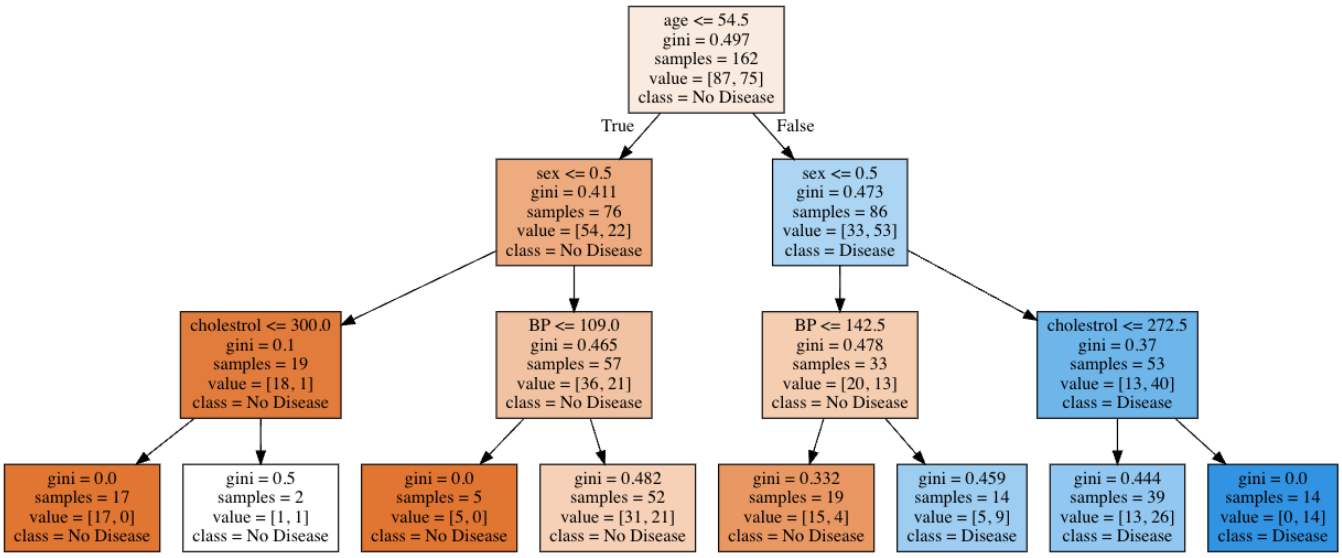
1. 简单直观，生成的决策树很直观。
2. 基本不需要预处理，不需要提前归一化，处理缺失值。
3. 使用决策树预测的代价是O(log2m)。 m为样本数。
4. 既可以处理离散值也可以处理连续值。很多算法只是专注于离散值或者连续值。
5. 可以处理多维度输出的分类问题。
6. 相比于神经网络之类的黑盒分类模型，决策树在逻辑上可以得到很好的解释
7. 可以交叉验证的剪枝来选择模型，从而提高泛化能力。
8. 对于异常点的容错能力好，健壮性高。

决策树算法的**缺点**:

1. 决策树算法非常容易过拟合，导致泛化能力不强。可以通过设置节点最少样本数量和限制决策树深度来改进。
2. 决策树会因为样本发生一点点的改动，就会导致树结构的剧烈改变。这个可以通过集成学习之类的方法解决。
3. 寻找最优的决策树是一个NP难的问题，我们一般是通过启发式方法，容易陷入局部最优。可以通过集成学习之类的方法来改善。
4. 有些比较复杂的关系，决策树很难学习，比如异或。这个就没有办法了，一般这种关系可以换神经网络分类方法来解决。
5. 如果某些特征的样本比例过大，生成决策树容易偏向于这些特征。这个可以通过调节样本权重来改善。

拓展阅读

1. 为什么C4.5决策树能处理连续特征，ID3树不能处理连续特征?
2. 为什么CART能做回归而ID3和C4.5不可以?
3. 决策树可视化 | 在Windows环境代码可能无法正常运行，最好在Kaggle Notebook环境尝试。



作业

1. 回答采用信息增益、信息增益率作为决策树分裂策略，有什么区别?
2. 在Titanic比赛数据集中，对年龄做对数处理(np.log(age))，是否会对决策树模型产生影响，为什么?
3. 如果发生过拟合情况，如果调节 `max_depth` 和 `min_samples_split` 参数?
4. 在Kaggle 泰坦尼克比赛使用 `sklearn.tree.DecisionTreeClassifier` 进行建模，查看官方文档，了解 `max_depth` 、 `min_samples_split` 参数含义，调节参数，提交需大于0.765。使用决策树建模流程与之前学习的线性回归、逻辑回归并没有本质区别，模型可以通过 `from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor` 进行导入.