https://zhuanlan.zhihu.com/p/27905967

1.决策树算法的概念：（entropy熵）

决策树算法主要是指决策树进行创建中进行树分裂(划分数据集)的时候选取最优特征的算法，他的主要目的就是要**选取一个特征**能够将分开的数据集尽量的规整，也就是尽可能的纯. 最大的原则就是: 将无序的数据变得更加有序。

这里总结下三个常用的方法:

1. 信息增益(information gain)
2. 增益比率(gain ratio)
3. 基尼不纯度(Gini impurity)

2.数学期望概念:

在概率论和统计学中，一个离散性随机变量的**期望值**（或**数学期望**、或均值，亦简称**期望**，物理学中称为期待值）是试验中每次可能的结果乘以其结果概率的总和。

$$E(X) = 1 \cdot \frac{1}{6} + 2 \cdot \frac{1}{6} + 3 \cdot \frac{1}{6} + 4 \cdot \frac{1}{6} + 5 \cdot \frac{1}{6} + 6 \cdot \frac{1}{6}$$
$$= \frac{1+2+3+4+5+6}{6} = 3.5$$

3.信息增益 (Information gain)

这里涉及到了信息论中的一些概念：某个事件的信息量，信息熵，信息增益等，关于事件信息的通俗解释可以看知乎上的一个回答

- 某个事件ii的信息量: 这个事件发生的概率的负对数

$$TI = -log(P(x_i))$$

TI=-log(P(x_{i}))

- 信息熵就是平均而言一个事件发生得到的信息量大小，也就是**信息量的期望值**

$$H = \sum_{i=1}^{n} H(x_i) = -\sum_{i=1}^{n} P(x_i) log(P(x_i))$$

H = \sum_{i=1}^{n}H(x_{i}) = -\sum_{i=1}^{n}P(x_{i})log(P(x_{i}))

任何一个序列都可以获取这个序列的信息熵，也就是将此序列分类后统计每个类型的概率，再用上述公式计算，使用Python实现如下:

```
def get_shanno_entropy(self, values):
    ''' 根据给定列表中的值计算其Shanno Entropy
    '''
    uniq_vals = set(values)
    val_nums = {key: values.count(key) for key in uniq_vals}
    probs = [v/len(values) for k, v in val_nums.items()]
    entropy = sum([-prob*log2(prob) for prob in probs])
    return entropy
```

信息增益

我们将一组数据集进行划分后，数据的信息熵会发生改变，我们可以通过使用信息熵的计算公式分别计算被划分的子数据集的信息熵并计算他们的平均值(期望值)来作为分割后的数据集的信息熵。新的信息熵的相比未划分数据的信息熵的减小值便是信息增益了. 这里我在最初就理解错了，于是写出的代码并不能创建正确的决策树。

假设我们将数据集D划分成k份D1,D2,...,Dk，则划分后的信息熵为:

$$H_{splited} = \sum_{j=1}^{k} P(D_j)H(D_j) = \sum_{j=1}^{k} \frac{len(D_j)}{len(D)} H(D_j)$$

H_{splited} = \sum_{j=1}^{k}P(D_{j})H(D_{j}) = \sum_{j=1}^{k} \frac{len(D_{j})}{len(D)} H(D_{j})

信息增益便是两个信息熵的差值

$$Gain_{splited} = H - H_{splited}$$

Gain_{splited} = H - H_{splited}

在这里我主要使用信息增益来进行属性选择，具体的实现代码如下:

```
def choose_best_split_feature(self, dataset, classes):
    ''' 根据信息增益确定最好的划分数据的特征
    :param dataset: 待划分的数据集
    :param classes: 数据集对应的类型
    :return: 划分数据的增益最大的属性索引
    '''
    base_entropy = self.get_shanno_entropy(classes)
    feat_num = len(dataset[0])
    entropy_gains = []
    for i in range(feat_num):
        splited_dict = self.split_dataset(dataset, classes, i)
```

```
            new_entropy = sum([
                len(sub_classes)/len(classes)*self.get_shanno_entropy(sub_classes)
                for _, (_, sub_classes) in splited_dict.items()
            ])
            entropy_gains.append(base_entropy - new_entropy)
        return entropy_gains.index(max(entropy_gains))
```

增益比率

增益比率是信息增益方法的一种扩展，是为了克服信息增益带来的弱泛化的缺陷。因为按照信息增益选择，总是会倾向于选择分支多的属性，这样会是的每个子集的信息熵最小。例如给每个数据添加一个第一无二的id值特征，则按照这个id值进行分类是获得信息增益最大的，这样每个子集中的信息熵都为0，但是这样的分类便没有任何意义，没有任何泛化能力，类似过拟合。

因此我们可以通过引入一个分裂信息来找到一个更合适的衡量数据划分的标准，即增益比率。

分裂信息的公式表示为:

$$SplitInfo(D) = \sum_{j=1}^{k} \frac{len(D_j)}{len(D)} log(\frac{len(D_j)}{len(D)})$$

SplitInfo(D) = \sum_{j=1}^{k} \frac{len(D_{j})}{len(D)} log(\frac{len(D_{j})}{len(D)})

可见如果数据分的越多，分裂信息的值就会越大

这时候把分裂信息的值放到分母上便会中和信息增益带来的弊端。

$$GainRatio = \frac{Gain}{SplitInfo}$$

GainRatio = \frac{Gain}{SplitInfo}

当然SplitInfo有可能趋近于0，这个时候增益比率就会变得非常大而不可信，因此有时还需在分母上添加一个平滑函数，具体的可以参考参考部分列出的文章

基尼不纯度(Gini impurity)

基尼不纯度的定义:

$$I_G(D) = 1 - \sum_{i=1}^{m} p_i^2$$

I_{G}(D) = 1 - \sum_{i=1}^{m}p_{i}^{2}

其中m表示数据集D中类别的个数, pi表示某种类型出现的概率。可见当只有一种类型的时候基尼不纯度的值为0，此时不纯度最低。

针对划分成k个子数据集的数据集的基尼不纯度可以通过如下式子计算:

$$I_G^{splited}(D) = \sum_{j=1}^{k} \frac{len(D_j)}{len(D)} I_G(D))$$

I_{G}^{splited}(D) = \sum_{j=1}^{k} \frac{len(D_{j})}{len(D)}I_{G}(D))

由此我们可以根据不纯度的变化来选取最有的树分裂属性

$$\Delta I_G = I_G - I_G^{splited}$$

\Delta I_{G} = I_{G} - I_{G}^{splited}