实验二 决策树的实现

18308133 刘显彬

1.实验原理

1.1 决策树

决策树也是一种监督的预测算法，但与KNN不一样的是，决策树算法将**在训练数据中提取信息**，然后构建一个针对该训练集的**规则集**，此后的测试集/验证集，是运行在此规则集上进行预测的，而非KNN是实时对原生输入训练样例进行相似度比对。决策树的规则集是由一系列**IF-THEN-ELSE规则**（意味着分枝）组合而成，因此可以由**树形的结构**表示，该树的每一个非叶子节点都是一个IF-THEN-ELSE规则连接到下一个子节点，叶子结点存储这一支分支（从根到该叶子）的预测值。

决策树的建立，即是构建上述IF-THEN结构，每一个节点都会根据**某种规则**选择出一个特征属性，根据**该属性的不同取值**来进行IF-THEN搭建子分支。这种规则必须能反映出特征属性和分类结果之间的联系紧密度，**选出的特征**需和结果的联系度**最为紧密**，这里我们选取的规则是**基于信息论**的：包括了ID3、C4.5、Cart等模型。之所以要选择最为紧密的一个属性值进行分支，是因为这样做可能可以使得整体的不确定度下降（也就是更能预测到结果）；因为我们没有各个类/属性取值的先验概率分布知识，因此，训练集的作用就在于：给予一个**概率的近似评估**。同时要注意的是，一旦一个属性用过之后，后续不能在选择这个属性进行评估，因为一个数据的某个属性只有一个值。

决策树的预测，给定一个新的测试数据，只需要根据它的各项属性值从根（根据规则）搜索到叶子即可获取它的预测值（运行在规则集）。因此对于任意的测试数据，其预测的时间复杂度为O(|V|)，|V|为特征维数，这是远优于KNN预测算法的。

1.2 ID3模型

通过信息论的角度来看分类结果的紧密度：当信息量越大的时候，说明不确定性越大（此时各个类的出现概率就越均匀）；反之，当某一个类j的出现概率接近1时，此时的信息量很小（接近0），这种情况下，随机给出一个数据，它属于类j的概率接近1，也就是说，这种情况的不确定度是非常小的。

信息熵就是一种信息量大小的考量：它满足上述的性质，其定义如下：

$$H(X) = -\sum_{i=1}^{v} p_i * log(p_i),\ p_i\text{为类}i\text{的出现概率},v\text{为类数目，这里的}p_i\text{是用}D_i/D\text{近似的。} \tag{1}$$

在此基础之上，定义条件熵如下：

$$\begin{gathered} H(X|a) = -\sum_{i=1}^{v} p(x_i|a) * log(p(x_i|a)),\ p(x_i|a)\text{为在情况}a\text{下，类}i\text{的出现概率。} \\ H(X|A) = -\sum_{a\in A} p(a)\sum_{i=1}^{v} p(x_i|a) * log(p(x_i|a)) \end{gathered} \tag{2}$$

熵模型下，两种信息的联系度可以通过下式表达出来：

$$Gain_A(D) = H(D) - H(D|A),\ D\text{为数据集，}A\text{为某个属性的值域} \tag{3}$$

ID3正是基于这种信息联系度：Gain（信息增益），来筛选出与分类结果联系最紧密的特征（基于该特征的信息增益最大）；但是ID3算法有个缺陷：当某个属性A可取值的数目很多时，H(X|A)的值会倾向0（简单考虑，当A的取值为无限个的时候，每一种的概率就会趋向于0），但这种趋势并不是因为其联系度变紧密导致的熵减少，因此需要有手段来平衡这种因为属性取值数目变化导致的熵减。

1.3 C4.5模型

C4.5的基本思想与ID3类似，只是考量信息联系度的方法略有不同。为了克服ID3模型的缺陷，这里采用了正则化的手段以减少因为属性个数值的增加带来的影响：

$$[1]GainRatio_A(D) = Gain_A(D)/SplitInfo_A(D),\\ where\ SplitInfo_A(D) = -\sum_{a\in A}\frac{D_a}{D} * log(\frac{D_a}{D}) \tag{4}$$

因子SplitInfo相对于特征取值个数的增长与Gain相对特征取值个数的增长是类似的，因此能够平衡ID3模型的缺陷，C4.5算法正是采用GainRatio（信息增益率）作为信息联系程度的考量；

1.4 Cart模型

Cart模型采用了一种叫Gini指数的信息联系度考量方法，gini指数的平凡定义如下：

$$Gini(D) = \sum_{a\in A} p_a(1-p_a) = 1 - \sum_{a\in A} p_a^2 \tag{5}$$

当需要划分数据集的时候，Gini指数应该对每部分子数据集进行计算，同时乘以子数据集在总数据集的比例作为权重，并累加；当要从A属性划分数据集时，其Gini指数计算如下：

$$Gini_{split=A}(D) = \sum_{a\in A}\frac{D_a}{D} * Gini(D_a) \tag{6}$$

像前两个模型的分析角度分析Gini指数：当分类的不确定性较大时，每个类别的出现概率应该均匀分布（否则就有倾向性），反之，则会向某些类别进行聚集；如此，当不确定性大的时候，分类概率的累加和小，gini指数会很大，当不确定性小的时候，gini指数就会趋向0；因此，gini指数越小，不确定性越小，该属性和分类之间的相关性越强。

$$\lim_{n\to\infty} n * (\frac{1}{n})^2 = 0, 假设n是类别的个数，\frac{1}{n}是各个类别的概率假设服从均匀分布 \tag{7}$$

因此Cart模型在对决策树节点裂分的时候，要考虑的是使得Gini_split小的属性，这是与ID3、C4.5不一样的地方。

1.5 [1] 决策树的形成过程

1. 初始化：建立根节点，其数据集为训练集全集，所有特征属性均可用于评估。
2. 选择特征&数据划分：根据以上三种模型（的任意一种），对所有可评估的属性进行评估，选出确定性最大（信息熵最小）的一个属性A，根据该属性的取值进行子节点分裂，利用IF-THEN结构进入子节点，并将属于该子节点的数据分配给该子节点作为训练集，特征集减去A。
3. 重复2，直至停止条件。
4. 停止条件为：

   4.1 无特征可供继续划分，此时返回训练集中数目最多的label作为结果。

4.2 训练集label全部一致，返回该label作为结果。

2.伪代码

2.1 **给出数据集的熵计算已经信息增益的计算**：

Algorithm 1 cal-HD(l)

Input: $l : label\ of\ dataset(D)$
Output: $H(D)$
  $labels \leftarrow label \in l$
  $freq \leftarrow frequency\ of\ label$
  $n \leftarrow num\ of\ label$
  HD $= 0$
  for $i \leftarrow 0\ to\ n$ **do**
    HD $+= freq[i] * log(freq[i])$;
  end for
  reurn HD

Algorithm 2 Gain(d, attr, HD)

Input: d: dataset, $attr$: split attr, HD: entropy of d
Output: $gain$
  // 分裂数据
  $values \leftarrow value \in d[attr]$;
  $N \leftarrow len(d)$;
  set $spData =$ empty dict;
  for v such that $v \in values$ **do**
    $spData[v] = subD, wheresubD[attr] ==' v' andsubD \in d$
  end for
  $H(D)_A$=0
  for sub such that $sub \in spData$ **do**
    // 计算各个子数据集的熵，并求和
    $H(D)_A \mathrel{+}= \frac{len(sub)}{N} * cal\_HD(sub)$;
  end for
  return $HD - H(D)_A$

2.2 **计算信息增益率以及基尼指数**

Algorithm 3 GainRatio(d, attr, HD)

Input: d: dataset, $attr$: split attr, HD: entropy of d
Output: $gainRatio$
  // 像前一个算法一样分裂数据集
  $spData \leftarrow$ split d with val $\in attr$
  // 计算$SplitInfo$
  $splitInfo \leftarrow 0$
  $N \leftarrow len(d)$;
  for sub such that $sub \in spData$ **do**
    $SplitInfo \mathrel{+}= \frac{len(sub)}{N} * log(\frac{len(sub)}{N})$
  end for
  return $\frac{Gain(d,attr,HD)}{SplitInfo}$

Algorithm 4 Gini(d)

Input: d: dataset
Output: $gini$
  // 分成不同的类
  $spData \leftarrow$ split d with different label
  $N \leftarrow len(d)$
  计算这些类的比重
  $freqs \leftarrow \frac{len(dset)}{N}\ \forall dset \in spData$

  return $1 - \sum_{freq \in freqs} freq^2$

Algorithm 5 Gini(d, attr)

Input: d: dataset, $attr$: split attr
Output: $gini$
  // 像前一个算法一样分裂数据集
  $spData \leftarrow$ split d with val $\in attr$
  $N \leftarrow len(d)$
  $freqs \leftarrow \frac{len(dset)}{N}\ \forall dset \in spData$

  $gini \leftarrow 0$
  for $i \leftarrow 0\ to\ len(freqs)$ **do**
    $gini \mathrel{+}= freqs[i] * Gini(spData[i])$;
  end for
  return $gini$

2.3 建树：

Algorithm 6 buildTree(root, d, alg)

Input: d: dataset, $root$:决策树根, alg:计算信息熵的算法

```
if d.attr == null or only one label in d.labels then
  这是一片叶子
  root['attr']='leaf', root['val']=vote_max(d.labels)
  return root
else
  best ← −inf;
  bestattr ←'';
  //根据给定的算法找出最优的属性
  for all attr ∈ d.attr do
    best ← alg(d, attr);
    bestattr ← argmax(best, attr);
  end for

  //然后对最优属性进行分裂，对子节点进行迭代
  root['attr'] ← bestattr
  spData ← d splitted by bestattr;
  for sub ∈ spData do
    root['val'] ← buildTree(root['val'], sub, alg);
  end for
end if
```

Output: $root$

2.4 预测：

Algorithm 7 predict(root, data)

Input: root: 决策树树根, data: 待预测的数据

Output: predVal: 预测值

```
cur ← root
//当前不是叶子时，进行搜索
attr ← cur['attr']
while attr != 'leaf' do
  //进入data[attr]对应的一支分枝
  cur ← cur['val'][data[attr]]
  attr ← cur['attr']
end while
//返回叶子的预测label
return cur['val']
```

3.关键代码分析

3.1 数据集D的熵：

公式见1.4节的公式（1）：

```python
def calHD(dataset, labelDict=None):
    # HD = - \sum_{j=1}^{l}{Dj/D * log(Dj/D)} where l is type of label
    if labelDict == None:
        labelDict = getlabel(dataset)   #get the label, and frequency of label
    N = len(dataset)
    HD = 0
    for label, freq in labelDict.items():
        HD += -freq/N * np.log(freq/N)
    return HD
```

3.2 ID3：Gain计算

数据集划分：传入参数为：数据集dataset以及划分的属性attr，先提取attr的所有值，再进行划分；数据划分用的是pandas库的$dataset[attr] == val$可以提取符合条件索引的功能。

```python
def splitDataset(dataset:pd.DataFrame, Attr:Union[str, int], val:Union[int, List[int], None])-> pd.DataFrame:
    if val == None:
        val = set()
        for i in dataset[Attr]:
            val.add(i)
        val = sorted(list(val))

    splitData = [pd.DataFrame(columns=dataset.columns)]*(len(val))
    for i in range(len(val)):
        splitData[i] = splitData[i].append(dataset[dataset[Attr] == val[i]])
```

ID3具体函数如下：

接受参数包括总数据集的熵HD，划分的属性attr，以及总数据集dataset；

流程为：先根据属性attr划分成子数据集，对所有子数据集进行熵计算，再乘以该子数据集的比重，加到相对熵HD_A中；返回：HD-HD_A，即是信息增益。

```python
def ID3(dataset, attr, HD):
    HD_A = 0
    N = len(dataset)
    spData, _ = splitDataset(dataset, attr, None)
    for each in spData:
        prob = len(each) / N
        # HD_A = - \sum_{j=1}^{v}{Dj/D * HD(Dj)}
        HD_A += prob * calHD(each)
    return HD - HD_A
```

3.3 C4.5：GainRatio计算

在ID3计算的基础之上加入了一个$SplitInfo$分裂信息的计算，该因子的计算在公式（4）中给出，实现也非常简单，先划分子数据集，再求出各子数据集的比重，计算其熵即可；返回的信息增益率$GainRatio$在ID3的$Gain$基础上除以$SplitInfo$即可：

```python
def GainRatio(dataset, attr, HD):
    HD_A = 0
    N = len(dataset)
    assert N != 0
    spData, _ = splitDataset(dataset, attr, None)
    Dj_Div_D = np.array([len(each) for each in spData]).reshape(1, -1) / N

    # splitInfo_with_a = - \sum_{j=1}^{v} { Dj/D * log(Dj/D) }, where v is the values of attribute a, D
    spInfo = max(- Dj_Div_D.dot(np.log(Dj_Div_D.T)), 1e-3)

    assert spInfo != 0

    for each in spData:
        prob = len(each) / N
        HD_A += prob * calHD(each)
    return (HD - HD_A) / spInfo
```

3.4 Cart：Gini计算

先实现无划分的gini指数的计算：根据label的值进行分类，统计不同label的数目，转换为概率，应用公式（5）进行gini指数计算即可；

总数据集的Gini 指数实现：先划分子数据集，再用子数据集的比重乘上子数据集的gini指数：

```python
def Cart(dataset, attr):
    # gini = \sum_{j=1}^{v} {Dj/D * gini(Dj)}

    def Gini(data):
        # gini = 1 - \sum_{j=1}^{l} {(Dj/D)^2}, where l is type of label
        GD = 0
        N = len(data)
        labelDict = getlabel(data)
        for key, val in labelDict.items():
            GD += np.square(val / N)
        # 1 - sum(prob^2)
        return 1-GD

    Cart = 0
    N = len(dataset)
    spData, _ = splitDataset(dataset, attr, None)

    for each in spData:
        prob = len(each) / N
        Cart += prob * Gini(each)
    return Cart
```

3.5 决策树的搭建：

决策树的规则，我是用字典的形式进行存储的，其中所有结点都有属性：（1）节点的划分属性'attr'，叶子节点的'attr'是'L'（label）；（2）节点的多数投票预测值'pred'，当节点的'attr'=='L'的时候，该属性就是预测的label值，非叶子节点的'pred'可为剪枝提供信息。（3）决策'dec'：存储着它的子节点，形式为{'val1':node1, 'val2':node2...}，根据不同的val进入不同的子树。

3.5.1 决策树的建树阶段：

选择特征：根据提供传入的参数alg决定采用的算法（ID3、C4.5、Cart），遍历所有属性，求出这些属性的信息熵，求出最优者作为划分属性（bestcol）

```python
# choose feature}
bestcol, bestg = '', -np.inf
HD = calHD(dataset, labelDict)
for col in dataset.columns[:-1]:
    if alg.lower() == 'cart':
        g = -Cart(dataset, col)
    elif alg.lower() == 'c4.5':
        g = GainRatio(dataset, col, HD)
    else:
        g = ID3(dataset, col, HD)

    if bestg < g:
        bestcol, bestg = col, g
```

划分数据集：

```python
# split dataset
spData, val = splitDataset(dataset, bestcol, None)
root['attr'] = bestcol
```

递归：现在我们得到了划分子数据集，可以根据属性bestcol的不同取值，开始形成当前节点的规则（IF val THEN node1）存入字典的key：'dec'中，并且开始递归处理子数据集，choose是我的递归函数名：

```python
# recursion
for i in range(len(spData)):
    del spData[i][bestcol]
    root['dec'][val[i]] = {'attr':'', 'dec':{}, 'pred':0}
    choose(root['dec'][val[i]], spData[i])
```

处理截止情况：当数据集label只有一种或者所有属性已经被划分，停止，已经是叶子了

```python
# leaf
labelDict = getlabel(dataset)
thisLeafPred = max(labelDict.items(), key=operator.itemgetter(1))[0]
root['pred'] = thisLeafPred
if len(labelDict) == 1:          # when the label of dataset is the same
    root['attr'] = 'L'
    root['dec'] = thisLeafPred
    root['pred'] = thisLeafPred
    return
elif len(dataset.columns) == 1: # or when all attribution has been ran out
    root['attr'] = 'L'
    root['dec'] = thisLeafPred
    root['pred'] = thisLeafPred
    return
```

预测：传入测试数据，根据规则一路递归到叶子，取出label即可（这里出现了一个小意外）

```python
def predictSingle(self, data):
    cur = self.root
    while cur['attr'] != 'L':
        col = cur['attr']
        if data[col] not in cur['dec']:
            return cur['pred']
        else:
            cur = cur['dec'][data[col]]
    return cur['dec']
```

3.6 数据集划分和结果介绍

这次实验采用K折交叉验证的去考察不同K以及采用的算法之间的结果

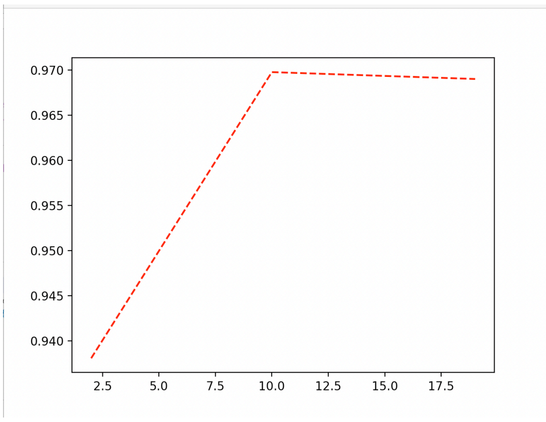
首先将数据进行打乱，将潜在的相似样本打散，避免聚集， 然后执行K折交叉验证：划分数据步骤如下：将打散的样本分成K块，依次取出第i=1，2，3...K块的数据作为validation-set，其余部分作为trainset：

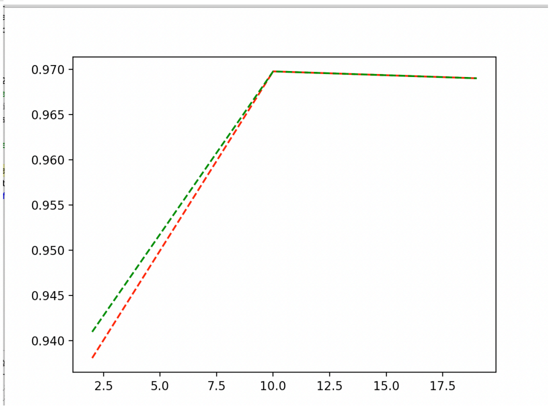
```
for i in range(1, k+1):
    ac = 0
    # choose the i-th part for validation
    train_set = dataset.iloc[:num*(i-1)].append(dataset.iloc[num*i:]).reset_index(drop=True)
    validation_set = dataset.iloc[num*(i-1):num*(i)].reset_index(drop=True)
    t = decisionTress(train_set)
```

取K次accuary作平均，作为该算法采用K折验证的结果

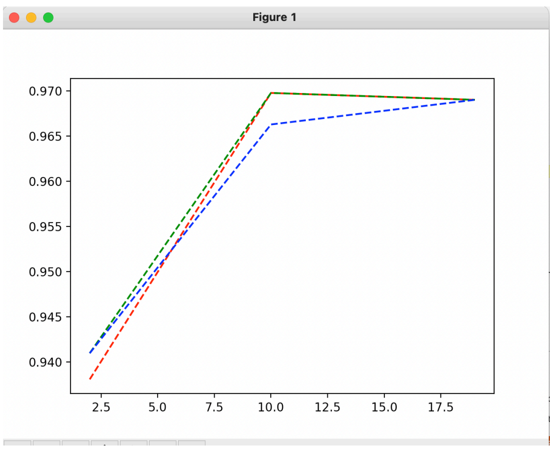
4.实验结果和分析

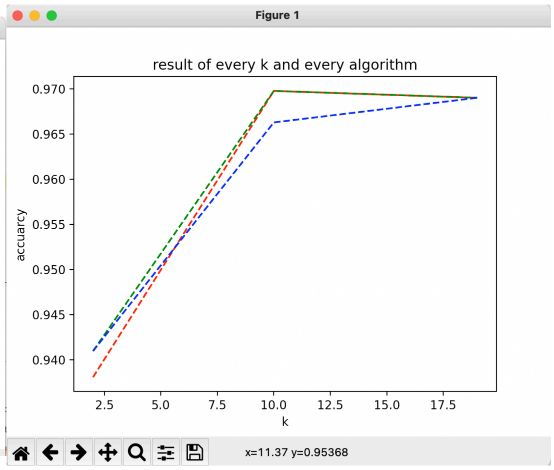
（下面四个图出自同一个结果，前三个图是最后一个图的中间状态，因为有时候会发现，id3的曲线被c4.5覆盖，所以设置断点，输出了中间的id3、c4.5准确率曲线）





red--id3　　　　green--c4.5





blue--cart

可见，在K折交叉验证下，这三种模型的准确率都很高，而且对于每一个算法，随着K值的提高，决策树的预测准确度变化都是先上升后下降，当K=10时，准确率达到最高：首先随着K的增大，训练数据采样率上升，换言之，训练数据越来越丰富，因此学习的特征越普遍，准确率上升。当这种上升到达阈值后，这时候或许会发生：验证集数目少，被噪声影响的概率加大，通过训练集学习的特征无法很好泛化到这里，也有可能只是方差的原因造成的（[2]因为当K很大的时候（留一法：K=n）时，也有一说法是能更稳定）。

对于ID3的算法表现，貌似并没有想象中的差，如上面的分析，C4.5对ID3进行了改进，但也没有因此变得表现特别出色。可能原因为：数据集的各个特征属性取值的数目较为均衡，无需特意平衡该权重，当属性取值分布较为均匀的时候，各属性$SplitInfo$也近乎一样，C4.5模型退化为ID3，因此C4.5的表现与ID3较为一致；当K较小的时候，训练数据属性取值分布不均匀，这时候C4.5的模型表现更好。

而Gini指数考虑的平方运算相对于ID3、C4.5的熵模型较为简单，是熵模型的一阶泰勒展开[3],获取的信息量更少，因此表现不如熵模型也应该是合理的。当

$$H(X) = -\sum_{i=1}^{v} p_i * log(p_i) \approx -\sum_{i=1}^{v} p_i * (1 - p_i) \tag{8}$$

5.思考题

5.1 决策树有哪些避免过拟合的方法?

过拟合的原因是，模型过度学习到了训练数据的属性，过度的意思是，将一批训练数据的独有特征表现，当成了训练数据上的**普遍**特征来学习，从而模型会非常“青睐”与训练数据相类似的测试数据，但是因为带有训练数据过多的影子，所以无法很好泛化到新数据上。

所以想要缓解过拟合，很直接的想法就是：避免模型在特定的训练数据上“学太多”。在模型角度出发，减少参数/降低模型复杂度就是一个很好的办法，模型复杂度下降 ，意味着模型的表达能力会下降，不能细致地表达某类特征，但这恰好是我们需要的：因为表达能力下降，这就会在训练中， 去push我们的模型去在训练数据中找到**更普遍**的模式、更一般的特征（因为它没有足够的能力去过度学习某些数据特有的东西了），也就缓解了过拟合。

非模型方面的过拟合：其次，避免在特定的训练数据学习，可以增加训练数据的数目，这样可以增加训练数据的多样性，从而不会引导模型去学习特定的特征（而是去学习普遍的特征）。噪音也是一种“特定的特征”，当训练数据少的时候，噪声会特别突出。

因此，对于决策树而言，剪枝或者限制树高，就是很好的能降低决策树复杂度的办法。

5.2 C4.5相比于ID3的优点是什么，C4.5又可能有什么缺点?

正如在原理分析一节中提到的，ID3会倾向于选择一些取值数目多的特征，而C4.5克服这一个缺陷，而且ID3在面对连续型的属性时，无法利用熵进行信息提取（因为每个取值都可能是唯一的，因此信息熵对于所有属性都是一致的），C4.5引入$SplitInfo$或许能平衡这种情况，从而处理连续型的数据。但C4.5运行的对数运算更多，建树阶段会花费更多的耗时；当划分的子数据集中，如果规模与总数据集相近，SplitInfo会趋向于0，此时会有“divided by zero”的风险（事实上在这次实验中也碰到了，采用的是一个粗糙的处理方法：加上一个小的偏置量）

5.3 如何用决策树来进行特征选择（判断特征的重要性）？

按照信息熵算法的分析那一节所述，决策树优先选择与分类结果最紧密的特征进行分裂，因此越靠近根的属性越重要。

6.遇到的问题和解决

一个问题是，在对某个数据（称为d）预测的时候，递归到某些非叶子节点时，d['attr']的值不在该叶子结点的值域中（也就是出现了缺失值），导致无法向下继续遍历。这里为了解决这个问题，给每一个非叶子节点也用多数投票的方法选出了一个预测label，当碰到这种情况的时候，就返回该节点的预测label作为结果。

7. 缺点

本次实验没有完成剪枝的工作，没能从实践出去去感受模型过拟合的影响；模型的构建也较为简陋，查看网络上的资料发现，C4.5和Cart模型的难度和创新不止信息熵的计算公式变更（C4.5的悲观剪枝、Cart的二叉性等都没有表达出来），Cart模型引入平方计算带来的时间优势也没有进行考量。

Reference

---

1. SYSU AI 实验课lab2课件 ↩
2. https://zhuanlan.zhihu.com/p/31924220 ↩
3. https://zhuanlan.zhihu.com/p/85731206 ↩