第二章 决策树

一、分类

1. 分类的目的
   分类的目的是**对数据进行有序整理和分类**
2. 分类模型三步法

- **模型的构建**：每个元组被假定属于预定义的类别，这个类别是由其中一个的名为类标号的属性决定的，用于构建模型的所有元组叫做训练集
- **模型评估**：**基于测试集对模型的准确率进行估算**，将测试样本的标签与分类得到的结果进行比较。
  其中准确率是指模型对测试集样本进行正确分类的样本所占总样本数的百分比，**测试集如果和训练集混合在一起就会引发过拟合**
- **预测**：**预测新数据的类标号**

二、决策树

1. 决策树的概念
   由数据的不同属性逐次划分数据集，直至得到的数据子集只包含同一类数据（或达到某种限制条件）为止，这样形成的一种树型结构，称为决策树。
2. 决策树的构建思想
   对于同样的数据，我们可能有不同的决策树构建方法，哪个是最好的呢，这个如何进行评估
   基于已有分类结果的数据，对新数据进行分类
   我们采用分治的思想：
   将数据分类为子集，判断子集是否是纯的，如果是就停止，否则重复这一过程，观察新数据划分到哪个新子集。我们将完全划分到一个类别的叫做纯的子集
3. **决策树的结构**
   - 根节点：**代表整个数据集最初做出的决策**
   - 内部节点：**象征着对输入特征进行选择的节点，每个内部节点都有若干分支**
   - 分支：**代表一个决策的结果，连接到另外的节点**
   - 叶节点：**代表最终决策或者预测**，不会再进一步划分
4. 纯度的划分
   如果区分不同的子集是否纯呢，一般来说完全分类到一个类别的是纯的子集，完全均匀分布的是最不纯的子集
5. 决策树特征选择标准：
   - 信息增益（information gain——Entropy) ID3

- 信息增益率（information gain radio）C4.5
- 基尼指数（Gini Impurity）CART

三、信息增益(Information Gain)

1. 信息的价值与不确定性
   - 信息用于消除随机不确定性
   - $P(U|V)$：在**接收方收到消息v的条件下，发送方发送消息u的概率**。信息传输存在随机误差。
   - 先验不确定性$P(U)$：在传输之前，**接收方无法判断发送方是什么状态，以及什么信息将会被传输**
   - 后验不确定性$P(U|V)$：在传输之后，**信息接收方从发送方收到了信息，先验不确定性一定程度上被消除**
   - $P(U)=P(U|V)$：接收方并没有收到任何信息
   - $P(U|V)=0$：信息被接受方全完接受，先验不确定性被彻底消除
   - 一般来说，干扰会破坏信息的传递，所以先验不确定性无法被完全消除
   - 信息的价值：$I(u_i)=log_2\frac{1}{P(u_i)}=-log_2P(u_i)$
2. **信息熵**
   - 熵是一个用来衡量数据混乱程度的指标
   - 信息熵：信息质量的数学期望。一种与随机变量相关的不确定度量
   - 数学期望：所有离散随机变量的可能取值与其概率乘积之和

     $$E(X)=Sum(p(x)*x)$$

     考虑到发送方发送消息的所有随机变量的可能取值，即所有可能事件所带来的期望信息量：

     $$Ent(U)=\sum_{i=1}^{C}P(u_i)log_2\frac{1}{P(u_i)}=-\sum_{i=1}^{C}P(u_i)log_2P(u_i)$$

   - 后验熵：当接受方接受到信息v_i，发送方发送信息U的概率是$P(U|v_i)$，平均不确定性如下所示：

     $$Ent(U|v_i)=\sum_{i}P(u_i|v_i)log_2\frac{1}{P(u_i|v_i)}=-\sum_{i}P(u_i|v_i)log_2P(u_i|v_i)$$

   - 后验熵的数学期望：在接受方接受到信息V后，来自发送方U的信息不确定性仍然存在：

     $$Ent(U|V)=\sum_{j}P(v_i)\sum_{i}P(u_i|v_i)log_2\frac{1}{P(u_i|v_i)}=-\sum_{j}P(v_i)\sum_{i}P(u_i|v_i)log_2P(u_i|v_i)$$

- 信息增益

$$Gain(U|V) = Ent(U) - Ent(U|V)$$

上述公式反映的是信息V消除不确定性的程度
信息熵描述的是随机变量的不确定性，信息熵越小，信息越纯，也代表里面的信息更少
低熵值代表数据分布非常规则，高熵值代表标签混乱

在选择根节点之前我们计算分类结果的熵值
随后针对特定属性的分类结果，分别计算每一个的熵值，最后计算该属性的熵值
信息增益=父节点的熵值-子节点的熵值
划分子集的纯度都高于父节点的纯度

2. **决策树划分过程**

- 计算根节点之前首先计算现在示例的信息熵，计算根节点信息熵，$Ent(U)$。
例如最后分类结果是yes/no，那么就可以计算

$$Ent(U) = -\frac{N_{yes}}{N}log_2\frac{N_{yes}}{N} - \frac{N_{no}}{N}log_2\frac{N_{no}}{N}$$

- 计算如果将其中一个特征作为根节点进行分类，通过计算这个特征的不同取值计算用这个特征作为根节点时候的后验熵的数学期望，$Ent(U|V)$。
例如针对年龄划分为青年，中年，老年，对年龄这一属性计算条件熵：

$$Ent(U|\text{年龄}) = Ent(U|\text{年龄} = \text{青年}) + Ent(U|\text{年龄} = \text{中年}) + Ent(U|\text{年龄} = \text{老年})$$

- 接着计算信息增益$Gain(U|V) = Ent(U) - Ent(U|V)$。
例如这里计算年龄对U的信息增益：

$$Gain(U|\text{年龄}) = Ent(U) - Ent(U|\text{年龄})$$

- 根据上述流程，依次计算如果将某个特征作为根节点的信息增益，选取信息增益最大的属性作为根节点进行分枝
- 随后根据根节点的取值划分各个子集，例如说年龄作为根节点，年龄有三个取值分别为(青年，中年，老年)，我们需要根据年龄的类别将数据集进行划分，每个子集中都是同样的年龄类别，计算划分后每个子集的熵值，这时如果这个子集的取值完全一样，这时候子集是纯的，计算得到的熵值是0，那就直接变为叶节点；
- 如果这个子集的取值不完全一样，即这个子集不是纯的，就需要继续对其进行划分，我们需要计算这个子集的结果的信息熵$Ent(D_{\text{青年}})$，分别计算剩余的其他属性对应的熵值，计算对应的信息增益，对信息增益进行比较，选择作为节点继续往下分。

3. ID3算法的缺点

- 无法处理连续变量
- 无法处理缺失值
- 生成较深的树，容易过拟合
- 有多分类属性往往有更高的information gain，导致这种属性更加容易被选为分裂节点

四、信息增益率（Gain Radio)

1. 信息增益率对信息增益的改进:
   - 信息增益率旨在减少在高度分支化的预测器的信息增益的偏差
   - 信息增益率通过考虑分裂时的分支数量（即内在信息）来调整信息增益，有助于减轻对具有众多不同取值的特征的偏向。
   - 它是通过将信息增益除以特征的固有信息来计算得出的，这一过程反映了该特征在产生分裂时所具有的潜力。
   - 解决这个问题的办法是设法对那些会导致分支数量极多的属性进行惩罚。 C4.5 使用一种称为“增益比”的分割标准。
2. **信息增益率的计算**
   信息增益率被用来根据该属性的熵值将属性所带来的信息增益标准化处理
   使用信息增益率来调整信息增益

$$GainRadio(D,S)=\frac{InfoGain(D,S)}{SplitInfo(D,S)}$$

   这里的SplitInfo是特征S本身的熵，例如特征S的取值是yes和no，这里计算的是：

$$SplitInfo(S)=-\frac{N_{yes}}{N}log_2\frac{N_{yes}}{N}-\frac{N_{no}}{N}log_2\frac{N_{no}}{N}$$

3. 如何根据增益率来划分属性：
   从候选划分中找到有更高信息增益率的属性
4. 离散化：
   离散化的特征对异常数据非常有抗性，经过离散化处理后，有信息损失，在另一方面会简化模型减少模型过拟合的风险
   特征被离散化之后，模型会变得更加稳定
   针对数值型数据我们可以根据其间隔进行离散化，找到若干分割点，随后我们可以分别计算分割点的信息增益率，最后选择有最大的信息增益率对应的分割点作为最佳分割点。
5. 缺失值处理
   - 信息增益的计算
     当属性中出现缺失值的时候，计算熵值可以直接忽略，例如20条数据中该属性出现了5个缺失值，则计算的时候自动忽略，按照总数15计算熵值，但是最后计算Gain的时候需要加权处理

$$Gain=(1-\frac{D_{miss}}{D_{all}})(Ent(U)-Ent(U|V)$$

   - 缺失值的分配
     已知非缺失数据经过划分后在不同类别的数量，我们可以计算其比例$\frac{D_i}{D_{exist}}$，例如分类红，白，黑，其中数据划分到红色的非缺失值有5个，总的非缺失值的数据有15条，则可以计算得到比例为$\frac{1}{3}$可以得到我们将缺失值的变量分配到这个种类的权重为$\frac{1}{3}$，后面根据这个权重重新计算各个样本比例，并且计算熵值

五、Gini指数（Gini Index)

1. Gini指数介绍
   基尼指数或基尼不纯度用于衡量在随机选取某一变量时，该变量被错误分类的程度或可能性。
   基尼指数（基尼不纯度）是 CART 算法的分割标准，确定用于节点分裂的最合适的特征
2. **Gini指数计算**
   考虑一个数据集D，包含k个类别的样本，在某个节点处样本属于类别i的概率可以表示为p_i，然后D的Gini不纯度就可以被定义为：

$$Gini = 1 - \sum_{i=1}^{k} p_i^2 = 1 - \sum_{i=1}^{k} (\frac{|C_i|}{|D|})^2$$

   其中，k是类别的总数，p_i是从类别i中选取数据点的概率（类别i在数据集中的比例）
   数据集越不纯，Gini系数越高，相反Gini系数越低，数据集越纯，熵值越低
   Gini系数位于0到1之间
   0代表所有元素属于一个类别或者说只存在一个类，表示完全纯净
   1代表这些元素在各种类别中随机分布
   0.5表示元素均匀分布在不同类别中
   例如针对颜色，有两个颜色红色和蓝色，红色有4个，蓝色有6个

$$Gini(\text{颜色}) = 1 - p_{\text{红色}}^2 - p_{\text{蓝色}}^2 = 1 - 0.4^2 - 0.6^2 = 0.48$$

3. **计算每个决策树的Gini不纯度步骤**
   - 计算每个节点的Gini不纯度

$$Gini(D) = 1 - \sum_{i=1}^{k} p_i^2$$

   - 对每个分裂点计算Gini不纯度，假设一个数据集D被特征A划分为两个子集D_1和D_2大小分别是n_1和n_2，Gini不纯度可以定义为：

$$Gini_A(D) = \frac{n_1}{n} Gini(D_1) + \frac{n_2}{n} Gini(D_2)$$

     这里是针对D根据特征A划分计算Gini不纯度，里面的$Gini(D_i)$用来表示在A类别为i的时候划分得到的数据子集D_i的Gini不纯度(一般来说计算到这里就可以)
   - 选择具有最低增益的Gini指数的分割方式：
     为了获取某一属性的Gini增益，需要从原始的不纯度中减去加权不纯度，最好的分裂方法是最大Gini增益，Gini增益这么计算：

$$\Delta Gini(A) = Gini(D) - Gini_A(D)$$

     选取最小的Gini指数对应最大的Gini增益来进行划分
     Gini不纯度可以被理解为一个最小化分类错误可能得标准
4. Gini指数与熵Entropy的比较

基尼指数	熵
它是随机选择的集合中被错误分类的概率	熵测量集合中的不确定性或者随机性的量
基尼系数是一个线性度量	熵是一个对数度量
它可以被解释为分类器中预期错误率	它可以被解释为指定实例类所需的平均信息量
基尼指数的范围是[0,1]，其中0表示完美纯度，1表示最大不纯度	熵的范围是[0,log(c)]其中c是类的数量
基尼系数通常用于CART(分类和回归树)算法	熵通常用于ID3和C4.5

最大熵：当所有类别出现的概率相等时候，此时熵达到最大值，对于个别类别，每个类别发生的概率是$\frac{1}{c}$，根据熵的定义公式：$Ent = log(C)$
最小熵：当只有一个类别发生而其他类别都不发生时，熵达到最小值

六、过拟合

1. **什么是过拟合**
   当机器学习或者深度学习模型在测试集预测不准确的时候我们可以说这是过拟合
   过拟合是一个在训练数据和没见过的测试数据的评估问题
2. 模型复杂度
   决策树变得过于复杂，可以完美地拟合训练数据，但是对新数据的泛化却很难
3. 记忆误差
   模型过于关注特定的数据点或者噪声，从而阻碍泛化能力
   过于详尽的规则：可能创建规则对于训练过于严苛，导致了对新数据的表现不好
4. 防止过拟合的思路：
   移除决策树中对预测能力没有显著贡献的部分有助于简化模型，防止其在训练时候记住噪声。剪枝可以依次移除对模型表现有最低影响的点
5. **常见的防止过拟合方法：**
   - 限制树的深度：为决策树设定了一个最大深度，限制了其拥有的层级和分支的数量
   - 最小化叶节点的样本数量：规定创建叶节点所需的最小样本数量，可确保每个叶节点都包含足够的数据以进行有意义的预测。这有助于防止模型生成过于特定的规则，这些规则仅适用于训练数据中的少数实例，从而减少过拟合现象。
   - 特征选择和工程化：特征选择是指挑选出那些最具信息量、能增强预测能力的特征，同时舍弃冗余或无用的特征。特征工程则涉及对特征进行转换或组合，以创建新的有意义的变量，从而提高模型的性能。
   - 集成方法：诸如随机森林这样的集成方法会将多个决策树组合起来，以减少过拟合现象。

七、剪枝

1. 剪枝的类别：
   预剪枝（提前停止）：可以提前终止决策树的生长过程，以免其变得过于复杂防止训练数据出现过度拟合的情况，因为这种过度拟合会导致在面对新数据时表现不佳
   后剪枝（减少节点）在树完全生长完毕后，后剪枝操作包括移除分支或节点，以增强模型的泛化能力。
2. 常见的预剪枝技巧
   - 最大深度：限制决策树中深度的最大值。
   - 每叶节点的最小样本数：设定每个叶节点中样本数量的最低阈值。
   - 每个分割的最小样本数：指定拆分节点所需的最小样本数量。
   - 最大特征数量：限制用于划分的特征数量。
     通过早期修剪，我们能得到一棵结构更简单、不太容易在训练数据上过度拟合的树。
3. 常见的后剪枝技巧：
   - **最小代价复杂度剪枝（Minimal Cost-Complexity Pruning, MCCP）：** 为每个子树分配一个基于其**准确率和复杂度**的“代价”，然后选择**代价最低**的子树作为最终模型
   - **减少误差剪枝（Reduced Error Pruning）：** 删除那些**对整体准确率影响不显著**的分支。
   - **最小不纯度减少剪枝（Minimum Impurity Decrease）：** 如果一个节点在划分后带来的**不纯度减少（如基尼系数或熵的下降）低于某个阈值**，则对其进行剪枝。
   - **最小叶节点样本数剪枝（Minimum Leaf Size）：** 删除那些**样本数量少于指定阈值**的叶节点。
4. 最低成本：
   对任何子树$T < T_{max}$，成本复杂度$R_\alpha(T)$：

$$R_\alpha(T) = R(T) + \alpha|T|$$

   其中$R(T)$是T的错误，α是复杂度参数，$|T|$是树T的叶节点个数
   最终选择哪一个子树则取决于α。如果α的值为0：则会选择最大的那棵树。
   当α趋近于无穷大时：这棵树只有一个根节点。
   一般来说，已给定一个预先选定的值α，则需找出使$R_\alpha(T)$达到最小值的子树$T(\alpha)$。

八、交叉验证

1. **什么是交叉验证**
   交叉验证是机器学习中的一种技术，用于评估模型在未见过的数据上的表现。这包括将可用的数据分成多个部分或子集，然后将其中一个部分用作验证集，而将其余部分用于训练模型。
2. 交叉验证是用来做什么的?
   交叉验证的主要目的是防止过拟合现象的出现。过拟合是指模型在训练数据上训练得过于完善，但在未见过的数据（即测试数据）上的表现却很糟糕。通过在多个验证集上对模型

进行评估，交叉验证能够更准确地估计模型的泛化性能，即其在未见过的数据上表现良好的能力。

3. **交叉验证的种类**
   - **k折交叉验证k-fold**
     将数据集划分为k个子集（折），对所有子集进行训练，但保留（k-1个）子集用于训练模型的评估。重复k次，每次使用一个不同的子集作为测试集。所有样本均用于训练和测试。真正的误差用平均错误率估计
   - **留一法leave one out**
     在整个数据集上进行训练，但只保留其中的一个数据点，然后对每个数据点进行迭代操作。
     该模型通过使用样本进行训练，并针对遗漏的一个样本进行测试，然后对数据集中的每个数据点重复这一过程。
     缺点:
     这会导致测试模型出现更大的差异，因为我们是基于一个数据点来进行测试的。如果这个数据点是个异常值，就可能会导致更大的差异。
     由于它会反复执行“数据点的数量”次，所以会耗费大量的执行时间。
   - **留出法Hold-out**
     对给定数据集的一部分进行训练，而其余部分则用于测试目的。这是一种简单且快速的评估模型的方法。
     主要的缺点是：
     我们对数据集的50%部分进行训练，有可能剩余的50%的数据中包含了一些重要的信息，而我们在训练模型时却忽略了这些信息，从而导致了较高的偏差（高偏差）。
4. 交叉验证实现
   神经网络中确定隐藏层神经元数量的方法可以用交叉验证来实现
   步骤1：针对不同的参数值，进行10倍交叉验证。
   步骤2：选择那个能获得最佳交叉验证分数的模型参数。注意：若采用交叉验证法进行参数调整，您还需要另外一组独立样本，以便准确评估最终模型的性能。

九、决策树的评价

1. 优点
   - 易于理解和解释：能够被人类轻松理解并解释，即便是那些没有机器学习背景的人也能做到。
   - 无需进行特征缩放：无需进行标准化处理或者是对数据的缩放。
   - 处理非线性关系：能够捕捉特征与目标变量之间的非线性关系。
2. 缺点
   - 过拟合：很容易在训练数据上过度拟合，尤其是当这些数据结构复杂、节点数量众多时。
   - 对具有更多层级的特征的偏好：倾向于具有更多层级的特征。

- 不稳定因素：数据中的微小变化就可能导致生成出完全不同的树形结构