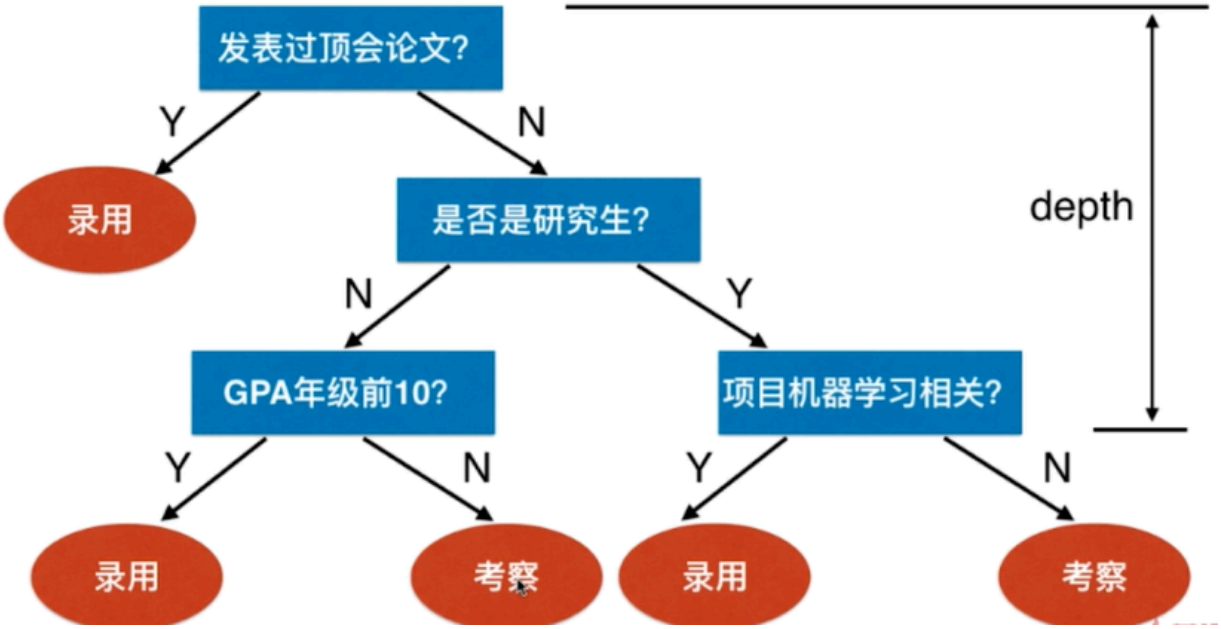
决策树

董峦 新疆农业大学

引言

决策树（Decision Tree）是一类常用的机器学习方法。它符合人类判断事物的方式，是可解释性十分突出的模型。假如有下图所示的人才录用标准，则人事部门拿到简历后只要从树根（顶部）开始，根据应聘者的相关属性值沿着箭头方向移动，到达叶子节点就能判定是否录用了。



上图中，蓝色矩形是节点（根节点及中间节点），红色椭圆是叶子节点对应决策结果。每个节点对应于一个属性测试，该树根据测试结果只有两个分支（如黑色箭头所示），因此这是一个二叉树。该树的深度是三。上图中的叶子结点只有两种取值，因此该决策树是一个二分类模型。

决策树与之前介绍的分类模型最大的一个不同点是对分类数据（categorical data）适应性好，比如性别、教育背景、婚姻状况、飞机舱位等等，这些数据的取值没有严格的大小或先后关系，只是给事物一个数字代码而已。决策树反而在处理数值型数据（numerical data），特别是数据有连续的取值时，缺少生成分支的可靠依据。

本文用决策树模型分析泰坦尼克号旅客和幸存者数据，试图发现生还者的一些特征。

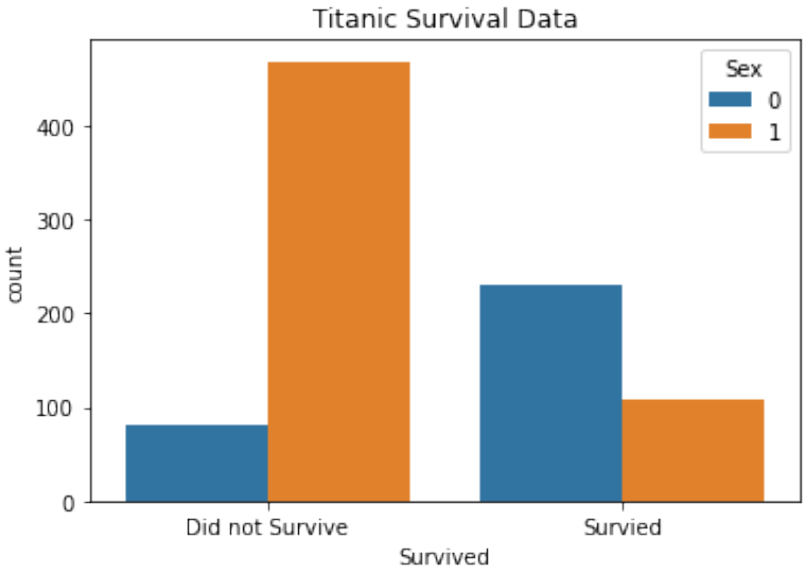
数据集

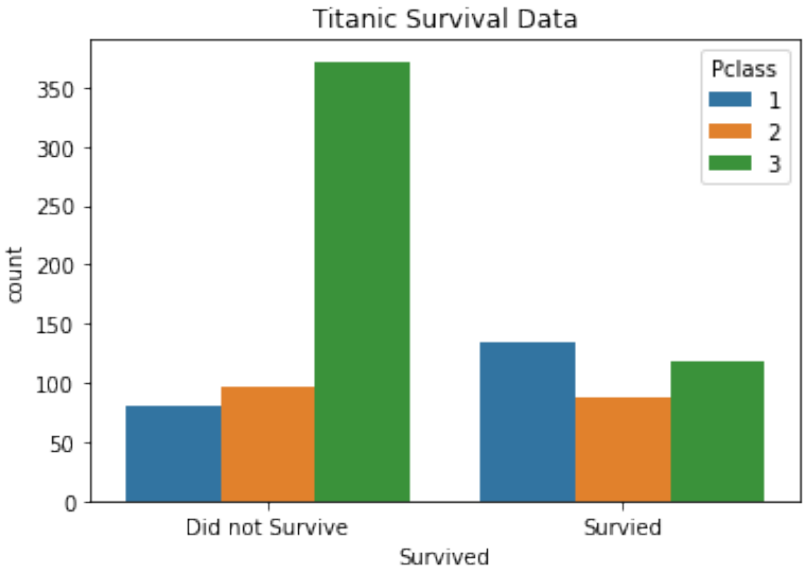
泰坦尼克号旅客和幸存者数据来自 Kaggle网站（https://www.kaggle.com/competitions/titanic/data）。该数据集的特征及其含义如下表所示

特征	含义	取值
PassengerId	旅客编号	
Survival	是否生还。是标签列	0 = 未生还, 1 = 生还
Pclass	船票等级（代表社会经济地位）	1 = 一等, 2 = 二等, 3 = 三等
Sex	性别	male 和 female
Age	年龄	实数，0~80岁
Sibsp	登船的同辈或配偶数量	整数 0~8
Parch	登船的父母或子女数量	整数 0~6
Ticket	票号	字符串
Fare	票价	实数，连续取值
Cabin	船舱号	字符串
Embarked	登船地点	C 、Q、或 S

该数据集中训练集有891个样本，均带有 Survived 标签，测试集有418个样本，没有标签。本文为了评估模型性能未采用原始的测试集而是从训练集中分出部分数据作为测试集。

以下两幅图可以帮助你对幸存者特征有个大致了解





可以看出幸存者大致特征是：一等舱的女性，而死亡的人主要是三等舱男性。

载入数据并查看数据基本信息

```
import pandas as pd

D = pd.read_csv('titanic/train.csv', sep=',')
D.info()
```

```
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
```

可见 Age、Cabin 和 Embarked 特征中有空值，本文用 Age 列的均值填补空值，去除 PassengerId、Name、Ticket、Cabin、Embarked 这些特征。去除这些特征带有较强的主观意识，是否合理不同的人有不同看法，比如登船地点与是否获救有多大程度关系呢，这个问题交给读者去检验。

```
# 用均值填补年龄列的空值
D['Age'].fillna(D['Age'].mean(skipna=True).round(), inplace=True)
```

```
# 重命名标签列
D = D.rename(columns={'Survived': 'Label'})
# 去除 PassengerId , Name, Ticket, Cabin, Embarked 列
del D['PassengerId']
del D['Name']
del D['Ticket']
del D['Cabin']
del D['Embarked']

# 分割数据集
D.sample(frac=1).reset_index(drop=True) # 置乱
trainset = D.loc[:700, :]
testset = D.loc[701:, :] # 190个测试样本
testset = testset.reset_index(drop=True)
```

模型

树是一种递归结构，每一个节点下都是一棵子树，因此决策树的生成是一个递归过程。从数据的角度看，根节点包含样本全集，从根节点向下每个节点包含的样本集合根据属性测试结果被划分到不同的子节点中，从根节点到每个叶节点的路径对应了一个判定测试序列。决策树的生成过程如下图所示（该图来自周志华所著的《机器学习》第四章）

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
　　属性集 $A = \{a_1, a_2, \ldots, a_d\}$.

过程： 函数 TreeGenerate(D, A)

1: 生成结点 node;
2: **if** D 中样本全属于同一类别 C **then**
3: 　将 node 标记为 C 类叶结点; **return**
4: **end if**
5: **if** $A = \varnothing$ **OR** D 中样本在 A 上取值相同 **then**
6: 　将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; **return**
7: **end if**
8: 从 A 中选择最优划分属性 a_*;
9: **for** a_* 的每一个值 a_*^v **do**
10: 　为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
11: 　**if** D_v 为空 **then**
12: 　　将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; **return**
13: 　**else**
14: 　　以 TreeGenerate(D_v, $A \setminus \{a_*\}$)为分支结点
15: 　**end if**
16: **end for**

输出： 以 node 为根结点的一棵决策树

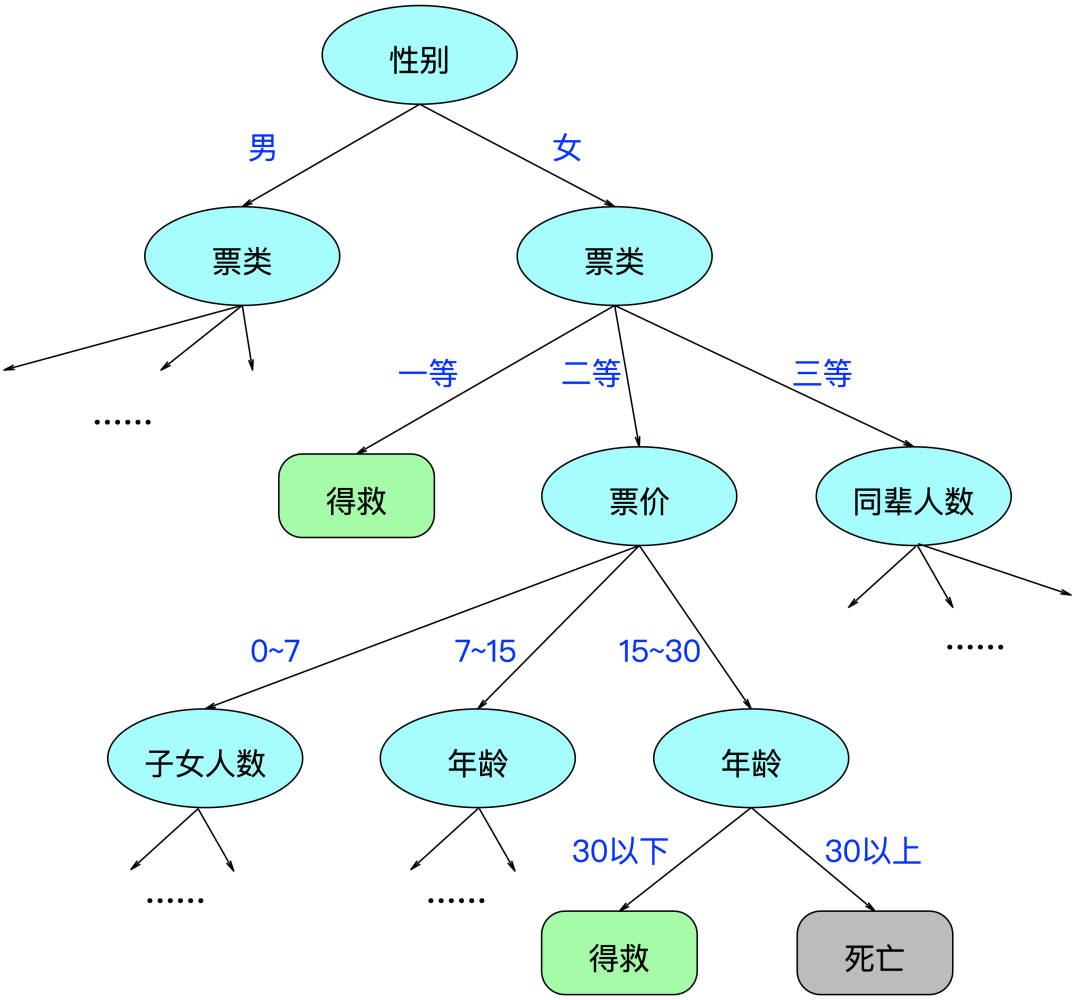
本文中 D 的数据结构是 Pandas 的 DataFrame，A 组织成如下数据结构，其中 Age 和 Fare 特征是连续的，本文依据统计结果的四分位数将两个特征各划分为四个区间。

	Label	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.758889	0.523008	0.381594	32.204208
std	0.486592	0.836071	13.002570	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	30.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
A = {
    'Pclass': {'type':'descrete', 'value':[1, 2, 3]},
    'Sex':    {'type':'descrete', 'value':['female', 'male']},
    'Age':    {'type':'continue', 'value':[[-np.inf, 22], [22, 30], [30, 35], [35,
np.inf]]},
    'SibSp':  {'type':'descrete', 'value':[0, 1, 2, 3, 4, 5, 8]},
    'Parch':  {'type':'descrete', 'value':[0, 1, 2, 3, 4, 5, 6]},
    'Fare':   {'type':'continue', 'value':[[-np.inf, 7.9], [7.9, 14.5], [14.5, 31],
[31, np.inf]]},
}
```

算法中的2~7行描述了递归的返回条件，第12行的 `return` 在实现中并未采用，因为这样实现后第9行对应的循环将被迫结束，根据周志华所著《机器学习》的上下文判断，该处的 return 确实未执行。

本文将要构建的决策树如下图所示（示意图）



决策树里**学习**的过程体现在算法的第8行，即如何选择最优划分属性。对上图来说，为什么根节点是年龄而不是其它属性（特征）呢？一般而言，随着划分过程不同细化，人们希望决策树的分支节点所包含的样本尽可能属于同一类别，即节点的“纯度”越来越高。怎么定义这个“纯度”便引出了不同的算法。

ID3算法

ID3决策树学习算法以信息增益为准则来选择最优划分属性。首先定义信息熵（information entropy），假定当前样本集合D中第k类样本所占比例为$p_k(k=1,2,\dots,\kappa)$，则信息熵是

$$Ent(D)=-\sum_{k=1}^{\kappa}p_k log_2 p_k$$

设某个属性a有V个可能的取值$\{a^1,a^2,\dots,a^V\}$，如果使用a来划分样本集将产生V个分支，其中第v个分支包含D中所有在a上取值为a^v的样本，记作D^v。计算出D^v上的信息熵，再考虑到不同分支节点所包含的样本数，给与权重$|D^v|/|D|$，其中绝对值运算表示求集合的大小。用属性a 对样本集D进行划分所获得的信息增益（information gain）是

$$Gain(D,a)=Ent(D)-\sum_{v=1}^{V}\frac{|D^v|}{|D|}Ent(D^v)$$

如果用信息增益作为选择最优划分属性的依据，则

$$a^*=\underset{a\in A}{\operatorname{argmax}}(Gain(D,a))$$

C4.5 算法

上述信息增益准则对可取值数目较多的属性有所偏好，为减少这种偏好可能带来的不利影响，C4.5算法使用增益率（gain ratio）来选择最优划分属性，增益率定义为

$$GainRatio(D,a)=\frac{Gain(D,a)}{IV(a)}$$

其中

$$IV(a)=-\sum_{v=1}^{V}\frac{|D^v|}{|D|}log_2\frac{|D^v|}{|D|}$$

IV称为a的“固有值”，a的可能取值数目越多 $IV(a)$ 的值通常越大，因此减弱了上述不利影响。但需要注意的是增益率准则对可取值数目较少的属性有所偏好，因此该算法采用一种启发式方法选择最佳划分属性：先从候选划分属性中选择增益率高于平均水平的属性，再从中选择增益率最大的。

CART

CART算法使用基尼指数（gini index）来选择划分属性，首先定义基尼值

$$Gini(D)=1-\sum_{k=1}^{\kappa}p_k^2$$

基尼值越小数据集的“纯度”越高。属性a的基尼指数定义为

$$GiniIndex(D,a)=\sum_{v=1}^{V}\frac{|D^v|}{|D|}Gini(D^v)$$

CART算法选择那个使得划分后基尼指数最小的属性作为最优划分属性

$$a^*=\underset{a\in A}{\operatorname{argmin}}(GiniIndex(D,a))$$

测试

在决策树上进行测试也是一个递归过程，从根节点开始根据样本在属性上的取值在树上移动，伪代码如下

```
输入：tree 是一个决策树，sample 是一个样本
evaluateTree(tree, sample):
if 节点是叶子结点：
    return 叶子结点取值
else:
    根据 sample 在该属性上的取值选择子树 subtree
    return evaluateTree(subtree, sample)
endif
```

在代码模板中，决策树用 Python 字典表示，单个样本是 Pandas 的 Series 数据结构。用C4.5算法在泰坦尼克数据集上生成的决策树如下所示（局部），其中根节点是 Sex，Sex 取值为 famale 的时候进入新的分支节点 Pclass，Pclass 取值为 1 时进入 Fare 节点，Fare 节点取值介于14.5和31之间时进入 Age 节点，根据 Age 取值最终到达叶子结点，根据叶子节点取值进行预测。叶子节点是 "Result": "....." 这样的键值对（没有定式，笔者这样设计而

已），“..... by parent node”是算法第12行对应的情况。

观察该决策树可以进一步得出结论：生还的主要是头等舱女性，但年龄大的生还的较少。


```
{
    "Sex": {
        "female": {
            "Pclass": {
                "1": {
                    "Fare": {
                        "-inf ~ 7.9": {
                            "Result": "Yes (68/71) by parent node"
                        },
                        "7.9 ~ 14.5": {
                            "Result": "Yes (68/71) by parent node"
                        },
                        "14.5 ~ 31": {
                            "Age": {
                                "-inf ~ 22": {
                                    "Result": "Yes (1/1)"
                                },
                                "22 ~ 30": {
                                    "Result": "Yes (3/4) by parent node"
                                },
                                "30 ~ 35": {
                                    "Result": "Yes (3/4) by parent node"
                                },
                                "35 ~ inf": {
                                    "Result": "Yes (2/3)"
                                }
                            }
                        },
                        "31 ~ inf": {
                            "Parch": {
                                "0": {
                                    "Result": "Yes (47/47)"
                                },
                                "1": {
                                    "Result": "Yes (10/10)"
                                },
                                "2": {
                                    "SibSp": {
                                        "0": {
                                            "Result": "Yes (4/4)"
                                        },
                                        "1": {
                                            "Age": {
                                                "-inf ~ 22": {
                                                    "Result": "No (1/2)"
```

```
                                        },
                                        "22 ~ 30": {
                                            "Result": "No (1/1)"
                                        },
                                        "30 ~ 35": {
                                            "Result": "No (2/3) by parent node"
                                        },
                                        "35 ~ inf": {
                                            "Result": "No (2/3) by parent node"
                                        }
                                    }
                                },
                                "2": {
                                    "Result": "Yes (1/1)"
                                },
                                "3": {
                                    "Result": "Yes (2/2)"
                                },
                                "4": {
                                    "Result": "Yes (8/10) by parent node"
                                },
                                "5": {
                                    "Result": "Yes (8/10) by parent node"
                                },
                                "8": {
                                    "Result": "Yes (8/10) by parent node"
                                }
                            }
                        },
                        "3": {
                            "Result": "Yes (65/67) by parent node"
                        },
                        "4": {
                            "Result": "Yes (65/67) by parent node"
                        },
                        "5": {
                            "Result": "Yes (65/67) by parent node"
                        },
                        "6": {
                            "Result": "Yes (65/67) by parent node"
                        }
                    }
                }
            },
          ......
        }
    }
}
```

```
}
```

经测试集上测试，决策树的分类结果如下：

```
accuracy: 0.84, precision: 0.84, recall: 0.73, F1: 0.77
```

而在训练集上的表现如下，对比可见过拟合未发生。

```
accuracy: 0.87, precision: 0.87, recall: 0.73, F1: 0.81
```

小结

对照 Kaggle 上的排名情况，本文用单一的决策树模型在泰坦尼克生还者数据集上取得了普通的成绩。Kaggle 上成绩较好的模型是随机森林（random forest）模型，该模型是利用大量决策树的预测做 voting 形成最终结果。

决策树模型符合人类面临决策问题时的处理方式，划分属性的选择是各算法的关键。本文生成的决策树过宽过深，尽管从测试结果上看未发生过拟合，但这是不是最好的结果还有待检验。剪枝（pruning）是决策树克服过拟合的主要手段。

C4.5算法采用二分法对连续属性进行处理，本文处理 Age 和 Fare 两个连续属性时利用了四分位数。还需要注意的是，与离散属性不同，若当前节点划分属性为连续属性，该属性还可作为其后代节点的划分属性。

思考

- 填补年龄字段的空值还有没有其它合理手段?
- 从数据集中划分出一个验证集，检验预剪枝技术能否提高模型泛化性能