# 5 决策树与随机森林

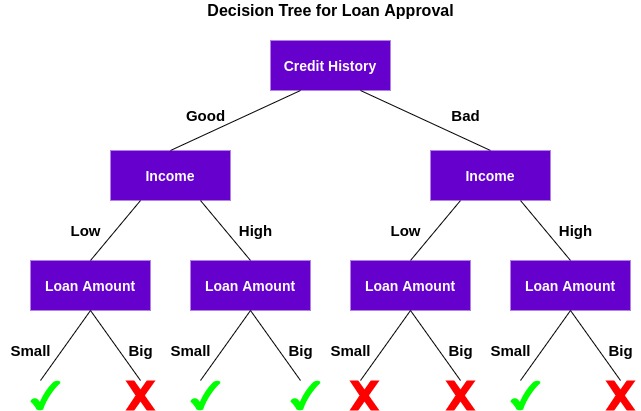
假设银行必须为客户批准一笔小额贷款，而银行需要迅速做出决定。银行检查此人的信用记录和财务状况，发现他们还没有偿还旧贷款。因此，银行拒绝了申请。但问题是，对于银行庞大的金库来说，贷款数额非常小，他们本可以在非常低风险的情况下批准贷款。因此，银行失去了赚钱的机会。

现在，又一个贷款申请将在几天内完成，但这一次银行提出了一个不同的策略——多个决策过程。有时它先检查信用记录，有时它先检查客户的财务状况和贷款金额。然后，银行结合这些多个决策过程的结果，决定向客户发放贷款。

即使这一过程比前一个过程花费更多的时间，银行也可以利用这一方法获利。这是一个典型的例子，集体决策优于单一决策过程。现在，你知道这两个过程代表了什么吧？

### **决策树简介**

决策树是一种有监督的机器学习算法，可用于分类和回归问题。决策树仅仅是为了达到特定结果而做出的一系列顺序决策。下面是一个正在运行的决策树的示例（使用上面的示例）



让我们了解这棵树是如何工作的。

首先，它检查客户是否有良好的信用记录。在此基础上，将客户分为信用记录良好的客户和信用记录不良的客户两类。然后，它检查客户的收入，并再次将他/她分为两组。最后，它检查客户要求的贷款金额。根据检查这三个特征的结果，决策树决定是否应该批准客户的贷款。

特征/属性和条件可以根据数据和问题的复杂性而改变，但总体思路保持不变。因此，决策树根据数据中的一组特征/属性（在本例中为信用历史、收入和贷款金额）做出一系列决策。

思考：为什么决策树首先检查信用评分而不是收入？

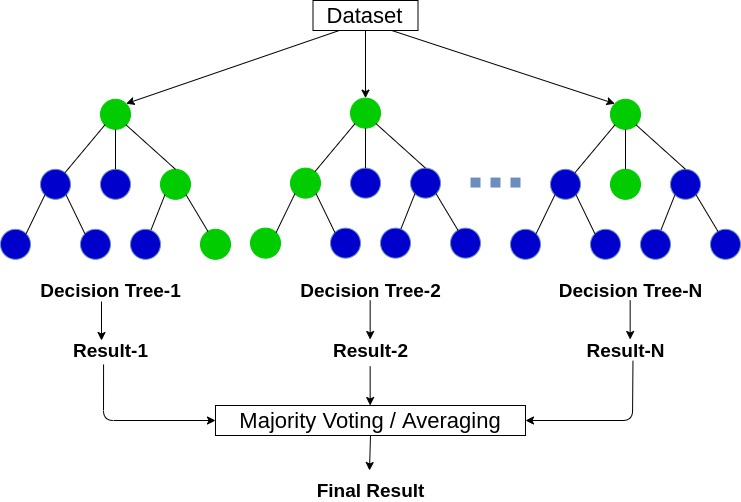
这被称为特征的重要性，要检查的属性序列是根据基尼系数或信息增益等标准确定的。

### **随机森林**

决策树算法易于理解和解释。但通常，一棵树不足以产生有效的结果。这就引入随机森林的概念。随机森林是一种基于树的机器学习算法，它利用多个决策树的能力进行决策。顾名思义，这是一片由树组成的“森林”！

但为什么我们称之为“随机”森林？那是因为它是一个由随机创建的决策树组成的森林。决策树中的每个节点对随机的特征子集进行运算以计算输出。然后，随机森林将各个决策树的输出组合起来生成最终的输出。

简单地说：**随机森林算法将多个（随机创建的）决策树的输出组合起来生成最终输出**。



编码实现贷款预测分析，贷款预测数据集。这是一个二元分类问题，在这个问题上，我们必须根据一组特定的特征来确定一个人是否应该获得贷款。

#### **步骤1：加载库和数据集**

首先导入所需的Python库和数据集

|  |
| --- |
| import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.metrics import f1\_score  from sklearn.model\_selection import train\_test\_split  # 导入数据集  df=pd.read\_csv('./data\_format1/bank\_data.csv')  df.head() |

#### **步骤2：数据预处理**

现在，是任何数据科学项目中最关键的部分——数据预处理和特征工程。在这一节中，将处理数据中的分类变量，并输入缺失的值。用模式来估算分类变量中的缺失值，对于连续变量，用平均值（对于各个列）来估算。另外，将对数据中的分类值进行标签编码。

|  |
| --- |
| df['sex']=df['sex'].map({'MALE':1,'FEMALE':0})  df['region']=df['region'].map({'INNER\_CITY':0,'TOWN':1,'RURAL':2})  df['married']=df['married'].map({'NO':1,'YES':0})  df['car']=df['car'].map({'NO':1,'YES':0})  df['save\_act']=df['save\_act'].map({'NO':1,'YES':0})  df['current\_act']=df['current\_act'].map({'NO':1,'YES':0})  df['mortgage']=df['mortgage'].map({'NO':1,'YES':0})  df['pep']=df['pep'].map({'NO':1,'YES':0})  df.head()  #mode()：方法用于计算DataFrame或Series中某一列的众数，众数指的是出现频率最高的值或值的集合  rev\_null=['sex','region','married','car','save\_act','current\_act','mortgage']  df[rev\_null]=df[rev\_null].replace({np.nan:df['sex'].mode(),  np.nan:df['region'].mode(),  np.nan:df['married'].mode(),  np.nan:df['car'].mode(),  np.nan:df['save\_act'].mode(),  np.nan:df['current\_act'].mean(),  np.nan:df['mortgage'].mean()}) |

#### **步骤3：创建训练和测试集**

现在，我们将数据集按80:20的比例分别拆分为训练集和测试集：

|  |
| --- |
| df=df.drop(columns=['ID'])  X=df[['age','sex','region','married','car','save\_act','current\_act','mortgage']]  y=df['pep']  X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size = 0.2, random\_state = 42) |

查看分割后的数据

|  |
| --- |
| print('Shape of X\_train=>',X\_train.shape)  print('Shape of X\_test=>',X\_test.shape)  print('Shape of Y\_train=>',Y\_train.shape)  print('Shape of Y\_test=>',Y\_test.shape) |

#### **步骤4：建立和评估模型**

由于我们有训练集和测试集，现在是时候训练我们的模型并对贷款申请进行分类了。首先，我们将在此数据集上训练决策树：

|  |
| --- |
| # 建立决策树  from sklearn.tree import DecisionTreeClassifier  dt = DecisionTreeClassifier(criterion = 'entropy', random\_state = 30)  dt.fit(X\_train, Y\_train)  dt\_pred\_train = dt.predict(X\_train)  #训练集评估  dt\_pred\_train = dt.predict(X\_train)  print('Training Set Evaluation F1-Score=>',f1\_score(Y\_train,dt\_pred\_train)) |

#### **步骤5：建立和评估模型**

由于我们有训练集和测试集，现在是时候训练我们的模型并对贷款申请进行分类了。首先，我们将在此数据集上训练决策树：

|  |
| --- |
| # 建立随机森林分类器  from sklearn.ensemble import RandomForestClassifier  rfc = RandomForestClassifier(criterion = 'entropy', random\_state =30)  rfc.fit(X\_train, Y\_train)  #训练集评估  rfc\_pred\_train = rfc.predict(X\_train)  print('Training Set Evaluation F1-Score=>',f1\_score(Y\_train,rfc\_pred\_train)) |