**数据处理**

原始数据由Diagnosis、诊断、编号、虚部、实数部分、性别、年龄、吸烟等构成。其中Diagnosis、诊断两列为数据标签列，Diagnosis列为英文标签，诊断列为中文标签。本文中使用Diagnosis列作为标签列。编号列为唯一标识，虚部、实数部分、性别、年龄、吸烟等作为特征进行数据处理和分析。

**2.1 缺失值处理**

原始数据中有大量缺失数据，在处理缺失数据方面，本文尝试了以下四种方式：

（1）：填充该缺失数据所属列的数据的平均值

（2）：填充该缺失数据所属列的数据的中值

（3）：填充与该缺失数据最靠近的前一个数据

（4）：剔除缺失数据

四种处理缺失数据的方式借助pandas内置DataFrame类型完成。代码示例如下：

df\_all.dropna(axis=0,how='any',inplace=True) #删除表中含有任何NaN的行

#缺失值补全

#df\_all.fillna(df\_all.mean(),inplace = True) #平均值填充

#df\_all.fillna(df\_all.median(),inplace = True) #中值填充

#df\_all.fillna(method='ffill',inplace = True) #最靠近的前一个数据填充

**2.2 数据标准化**

处理完缺失数据后，对所有特征数据进行标准化处理。本文选取0-1标准化将所有特征数据标准化为0-1之间的一个数。0-1标准化也叫离差标准化，是对原始数据的线性变换，使结果落到[0,1]区间，转换函数如下：

其中max为样本数据的最大值，min为样本数据的最小值

Python代码实现如下：

def Normalization(x):  
    return [(float(i)-min(x))/float(max(x)-min(x)) for i in x]

**2.3 数据随机划分**

本文将处理后的数据按照70%为训练数据，30%为测试数据的方式进行多次随机划分，附录代码中为20次，可按要求更改次数，取多次数据划分后分类准确率的平均值为最终数据分类准确率。

**3 算法实现**

**3.1 决策树算法**

本文采用决策树算法处理分类问题。

决策树学习的算法通常是一个递归地选择最优特征，并根据该特征对训练数据进行分割，使得各个子数据集有一个最好的分类的过程。这一过程对应着对特征空间的划分，也对应着决策树的构建。

（1） 开始：构建根节点，将所有训练数据都放在根节点，选择一个最优特征，按着这一特征将训练数据集分割成子集，使得各个子集有一个在当前条件下最好的分类。

（2） 如果这些子集已经能够被基本正确分类，那么构建叶节点，并将这些子集分到所对应的叶节点去。

（3）如果还有子集不能够被正确的分类，那么就对这些子集选择新的最优特征，继续对其进行分割，构建相应的节点，如果递归进行，直至所有训练数据子集被基本正确的分类，或者没有合适的特征为止。

（4）每个子集都被分到叶节点上，即都有了明确的类，这样就生成了一棵决策树。

**3.2 连续数据离散化**

决策树算法需要统计每个特征特征值出现频次，原始数据中虚部、实数部分、年龄等列中大部分数据为连续数据，没有足够多的重合项，考虑将连续数据离散化。

本文考虑了两种连续数据离散化方式。

第一种是利用聚类方法将连续数据中具有相似特性的数据聚为一类，通过聚类算法将连续数据转化为离散数据。本文采用了k-means算法尝试将连续数据离散化。

算法步骤：

（1）（随机）选择K个聚类的初始中心；

（2）对任意一个样本点，求其到K个聚类中心的距离，将样本点归类到距离最小的中心的聚类，如此迭代n次；

（3）每次迭代过程中，利用均值等方法更新各个聚类的中心点(质心)；

（4）对K个聚类中心，利用2,3步迭代更新后，如果位置点变化很小(可以设置阈值)，则认为达到稳定状态，迭代结束，对不同的聚类块和聚类中心可选择不同的颜色标注。

关于k值选取，本文采用循环遍历的方式求取k=2到k=10的k-means聚类结果，依据聚类结果选择表现更好的k值作为最终的聚类个数。

Python代码如下：

#k-means聚类

def kmeansCluster(X):

X=np.array(X).reshape(-1, 1)

list\_score=[]

for n\_clu in range(2,10):

y\_pred = KMeans(n\_clusters=n\_clu).fit\_predict(X)

list\_score.append(metrics.calinski\_harabaz\_score(X, y\_pred))

n\_clu=list\_score.index(max(list\_score))+2

#n\_clu=5

y\_pred = KMeans(n\_clusters=n\_clu).fit\_predict(X)

return y\_pred

第二种方法是自行确定离散化方式，通过观察每个类别的数据和多次实验对比，本文选择了将0-1标准化的特征按照下表方式离散化。

|  |  |
| --- | --- |
| 原始数据 | 离散化数据 |
| [0,0.25) | 0 |
| [0.25,0.5) | 1 |
| [0.5,0.75) | 2 |
| [0.75,1] | 3 |

代码如下：

def discrete\_sort(data\_conse):

for i in data\_conse.index[0:7]:

if(data\_conse[i] >= 0 and data\_conse[i] < 0.25):

data\_conse[i] = 0

elif(data\_conse[i] >= 0.25 and data\_conse[i] < 0.5):

data\_conse[i] = 1

elif(data\_conse[i] >= 0.5 and data\_conse[i] < 0.75):

data\_conse[i] = 2

elif(data\_conse[i] >= 0.75 and data\_conse[i] <= 1):

data\_conse[i] = 3

return data\_conse

具体算法实现请参考附录代码。

**4 结论**

本文采用了剔除缺失值和自定义离散化处理的方式处理原始数据，并且通过数据随机划分多次训练预测取平均值得到预测结果。我们将处理后的数据按照70%为训练数据，30%为测试数据的方式进行多次随机划分，数据划分次数设定为20，得到下图所示每次划分的准确率和最终的平均准确率。

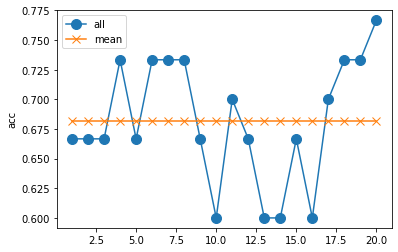


图3：平均准确率图像

由上图可以得知，数据集的划分对预测准确率有一定影响，我们通过多次划分取平均值来消除这些影响。最终，我们的模型在数据集上的预测准确率为0.681667。（由于随机划分的原因，每次运行代码预测准确率会有一定波动）

本文最终的预测准确率在0.65-0.7之间波动，预测准确率方面还有比较大的提升空间。

本文的决策树模型是基础的决策树模型，可以考虑使用集成学习的方式进行优化改进，如bagging算法，randomforest算法等。

在缺失数据集处理上，可以利用回归算法预测缺失值进行补全，针对数据聚类问题，基于密度的聚类方法，例如DNSCAN算法等，可以尝试使用提高聚类精度。