中南大學

CENTRAL SOUTH UNIVERSITY

机器学习课设（报告）

ML Course Design (Report)

题　　目:	基于 Matlab 的 ID3 算法实现与改进
学生姓名:	支博文
指导老师:	高琰
学　　院:	自动化学院
专业班级:	智能 2002 班

本科生院制

2022 年 12 月

基于 Matlab 的 ID3 算法实现与改进

摘要

决策树是一种适用于解决分类和回归问题的监督机器学习算法，通过在每个节点应用分支条件递归构建，旨在以尽量使数据集划分后最“纯”的方式来对数据进行分类迭代[1]。ID3 决策树算法是一种对路径的搜索的贪心算法，其核心在于根据信息增益最大化来选择进行特征划分，递归地构建决策树[2]。本次课程设计使用 Matlab 实现 ID3 算法，并在此基础上通过改变特征选取方式为基尼指数、完善模型生成过程中的节点添加策略以及添加后剪枝操作等对 ID3 算法进行改进。

通过选取三个合适的 UCI 数据集 BreastCancer, tic-tac-toe 和 CMC 进行 10 次 10 折交叉验证①，获取验证过程中产生的平均精度、平均 AUC 值以及总耗时等指标判别比较各模型效率。

最终获取模型 10 次 10 折交叉验证[3]指标统计汇总表5-1，其中改进算法后生成的模型经过 10 次 10 折交叉验证，其指标无论是在平均精度、总花费时长，还是平均 AUC 值均有较大幅度的提升，改进效果明显。

为美观简便，同时设计 MatlabAPP 实现可视化操作，课程设计完整程序包见 GitHub 仓库ML-Course-Design，附录放置部分代码辅助正文引用。

关键字： ID3 算法改进　基尼指数　决策树节点处理　后剪枝　Matlab

① 总计 100 次模型验证

目录

第 1 章　ID3 决策树算法

1.1　决策树算法

决策树算法旨在以尽量使数据集划分后最“纯”的方式来对数据进行分类迭代，来形成一棵树结构的分类器，其中非叶子节点为特征，叶子节点为分类结果。最早的决策树学习策略 CLS（Concept Learning System）最早由厄尔 · 亨特（Earl B. Hunt）于 1962 年提出，其思想在于分而治之[4]。这之后在上世纪 80 年代初，罗斯 · 昆兰（J.Ross Quinlan）提出 ID3 决策树算法①，本质上 ID3 决策树算法是一种对路径的搜索的贪心算法，即这种决策树算法是非回溯的。

1.2　信息增益

ID3 算法度量分类“纯度”的标准在于比较分类前后信息熵的增益的大小，即信息增益，下面介绍计算信息增益所必需的公式，之后程序实现在算法实现阶段的核心便是实现并组织这些公式[5]。

首先是信息熵②的计算公式：

$$\text{Entropy} = -\sum_{i=1}^{n} p(x_i) * \log_2 p(x_i),$$

其中 $p(x_i)$ 为每个可选项 x_i 所占的比例。

其次是在 X 的条件下 Y 的条件熵计算公式：

$$\text{Entropy}(Y \mid X) = \sum_{i=1}^{n} p(x_i)\,\text{Entropy}(Y \mid x_i).$$

最后便是信息增益的计算公式：

$$\text{infoGain}(D \mid A) = \text{Entropy}(D) \text{ - } \text{Entropy}\ (D \mid A),$$

信息增益越大，意味着集合使用属性 A 来进行划分所得到的的“纯度提升”越大。在获取信息增益最大化的分类特征之后，ID3 算法便会根据选择的特征进一步递归地构建决策树。

① 主要贡献是在 CLS 算法的基础上引入了信息增益准则等改进方法

② 用于衡量信息量的多少

1.3 算法的可改进之处

ID3 算法的优点很明显，易于实现，并且对于非数值数据的处理也能有很好的效果。然而，在理解 ID3 算法的基础上，横向比较其他算法[6]，总结 ID3 算法有以下可改进之处：

- 算法倾向于选择子类别多的特征①；
- 算法不适合有连续属性特征的数据集，容易产生不可靠的分支；
- 算法仅适用于二分类问题；
- 算法只能处理离散型属性，对于连续型属性，在分类前需要进行离散化处理。

①由于信息增益本身的计算方法，子类别多的特征分类的结果会更纯，熵会更小

第 2 章　UCI 数据集选取

为验证模型效果，此次课程设计选取了三个 UCI 数据集，分别是 BreastCancer, tic-tac-toe 和 CMC。

2.1　Breast cancer data

Breast cancer data[①]具体信息如图2-1所示，该数据集描述乳腺癌是否复发，存在两个类，一个类有 201 个实例，另一类有 85 个实例。这些实例都由 9 个特征属性描述，这些特征中有连续的，也有离散的。数据集存在缺失，除去缺失实例后剩余 277 个实例。

Data Set Characteristics:	Multivariate	**Number of Instances:**	286	**Area:**	Life
Attribute Characteristics:	Categorical	**Number of Attributes:**	9	**Date Donated**	1988-07-11
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	677847

图 2-1　Breast cancer data 描述信息

2.2　Tic-Tac-Toe Endgame database

Tic-Tac-Toe Endgame database[②]具体信息如图2-2所示，该数据集对井字游戏结束时可能的棋盘配置进行了编码，假设 “x” 先下，目标是 “x 获胜”，即当 “x” 能够达成三连时结果为真，其余情况结果为假。数据集包含 958 个实例，每个实例有 9 个特征属性，各对应一个井字格，没有缺失属性值，实例中大约有 65.3% 是真。

Data Set Characteristics:	Multivariate	**Number of Instances:**	958	**Area:**	Game
Attribute Characteristics:	Categorical	**Number of Attributes:**	9	**Date Donated**	1991-08-19
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	303033

图 2-2　Tic-Tac-Toe Endgame database 描述信息

2.3　Contraceptive Method Choice

Contraceptive Method Choice[③]具体信息如图2-3所示，该数据集是 1987 年全国印度尼西亚避孕普及率调查的一个子集。根据女性的人口统计和社会经济特征来预测她当

[①]Donors: Ming Tan and Jeff Schlimmer (Jeffrey.Schlimmer@a.gp.cs.cmu.edu)

[②]Donor: David W. Aha (aha@cs.jhu.edu)

[③]Donor: Tjen-Sien Lim (limt@stat.wisc.edu)

前的避孕方法，分为三个类：no use, long-term methods, short-term methods。数据集包含 1473 个实例，每个实例有 9 个特征属性，没有缺失属性值。

Data Set Characteristics:	Multivariate	**Number of Instances:**	1473	**Area:**	Life
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	9	**Date Donated**	1997-07-07
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	246728

图 2-3　Contraceptive Method Choice 描述信息

ID3 算法一般仅适用于二分类问题，此次选取一个三分类数据集目的在于多维度测试之后在第4章中改进算法的效果。

第 3 章　Matlab 程序实现

3.1　ID3 算法核心实现

ID3 算法的核心部分主要需要实现以下步骤：

- 从根节点开始，计算所有可能的特征的信息增益，选择信息增益最大的特征作为节点的划分特征；
- 由该特征的不同取值建立子节点；
- 对子节点递归，构建决策树；
- 直到没有特征可以选择或类别完全相同为止，得到最终的决策树。

3.2　ID3 算法具体实现

详细程序代码见附录，下面介绍主要函数的实现功能[①]：

- 主函数 Tree.m

  在处理好 UCI 数据集中的缺失属性实例并将.data 格式转化为.mat 并导入主函数后，采用 10 次 10 折交叉验证划分训练集与测试集，调用 ID3 子函数进行决策树的构建，之后计算并输出 10 次 10 折交叉验证过程中模型表现的平均精度、平均 AUC 值和总耗时，详细代码实现见附录A。

- ID3.m

  基于 ID3 算法构造决策树，根据 chooseFeature 选择的特征递归构建决策树，详细代码实现见附录B。

- chooseFeature.m

  选择信息增益最大的属性特征，首先对数据的标签进行特殊处理，对于二分类问题，将标签调整的为 logical 数组，与第一个相同的为 1，不同的为 0，方便后续的

[①]避免代码冗长，本节不直接贴代码而引用附录

信息熵的计算。其次利用 tabulate 获取每个属性下的不同取值以及其数量，利用这些信息再对每一个属性取值调用 calShannonEnt 函数来计算信息熵。最后选取最大的信息熵，为了应对当有多个最大的属性特征时的情况，设置最大值随机选取，详细代码实现见附录B。

- calShannonEnt.m

  根据传入的逻辑数组计算信息熵，并添加当 $p = 0$ 时，信息熵 $p * log_2 p$ 为 0，详细代码实现见附录D

- predict.m

  对测试样本进行分类，若测试集中出现了数据集中没有的标签时，将无法进行预判而返回空数组，遇到这种情况将随机选取一个属性值进行预判，接下去进行预测，详细代码实现见附录E。

3.3 MatlabAPP 程序可视化

最终实现的算法程序包较大，链接关系较复杂，在改进算法重新运行的过程中操作并不美观简便，故设计 MatlabApp 实现操作可视化[7]，其详细代码实现见附录L，实现效果如图3-1左所示，选择模型改进方式后点击“开始预测”，待 10 次 10 折交叉验证完成后可得如图3-1右所示结果，实现程序操作美观简便。

图 3-1 MatlabAPP 程序可视化效果图

第 4 章　ID3 算法改进

4.1　基尼指数与节点处理

针对 ID3 算法的可改进之处，首先使用基尼指数来作为选择指标，基尼指数更偏向于连续属性有利于对含有连续属性的数据集进行决策树划分。其实现过程与之前的计算信息增益基本相同，只是最后选择的指标改为了最小的而非最大。只改变选择指标为基尼指数之后，10 次 10 折交叉验证指标基本与之前相同。

选择指标改基尼指数详细实现代码 chooseFeatureGini.m 见附录G，以及计算基尼指数的代码 getGini.m 见附录H。

其次，在生成叶子节点时，改变直接选取第一个的策略，转而找最多的标签值。在没有对应属性值的情况下，输出当前节点在训练时的最多标签值。这一部分是为了应对训练集中没有测试集的对应属性值，或者后续将该模型用于预测时没有对应属性值的情况。不使用之前的随机选择节点策略，避免带来新的误差。

最后，当属性只有一种值时，会产生没有划分能力的节点，故而将该属性将剔除避免加入训练中，使树的模型更加简单并且在判断属性是否为节点时也更加容易。在完成基尼指数与节点处理的改进后 10 次 10 折交叉验证指标相较原始模型有较大幅度的提升，之后在第5章中会详细分析。

节点处理详细改进代码 ID3_2.m 和 predict_2.m 见附录I和附录J。

4.2　后剪枝

此次后剪枝参考错误率降低剪枝（REP）[8]，将数据分为训练集和测试集，用训练集去生成一颗完整的决策树，用测试集去剪枝，程序需要实现步骤如下：

- 删除以某节点为根节点的树，使其成为叶子结点；
- 赋予该节点最常见的分类；
- 对比删除前和删除后的性能是否有所提升；
- 如果有则进行删除，没有则保留。

具体改进过程中，在 leaf 中加入原本在训练是当前数据集在这个节点的最多标签值之后，选用了 8 个数据集用于训练，其余 1 个用于验证，1 个用于测试。在完成基尼指数、节点处理与后剪枝的改进后 10 次 10 折交叉验证指标相较原始模型有较大幅度的提升，但相较于不使用后剪枝却有小幅降低[9]，之后在第5章中会详细分析。后剪枝详细实现代码见附录K。

第 5 章　指标结果分析

ID3 算法实现与改进运行结果汇总如图5-1所示，第一排选用 breastcancer 数据集，第二排选用 tic_tac_toe 数据集，第三排选用 cmc 数据集；第一列为原始模型 (ID3 算法)，第二列为采用基尼指数与节点处理的改进，第三列为采用采用基尼指数、节点处理与后剪枝的改进。

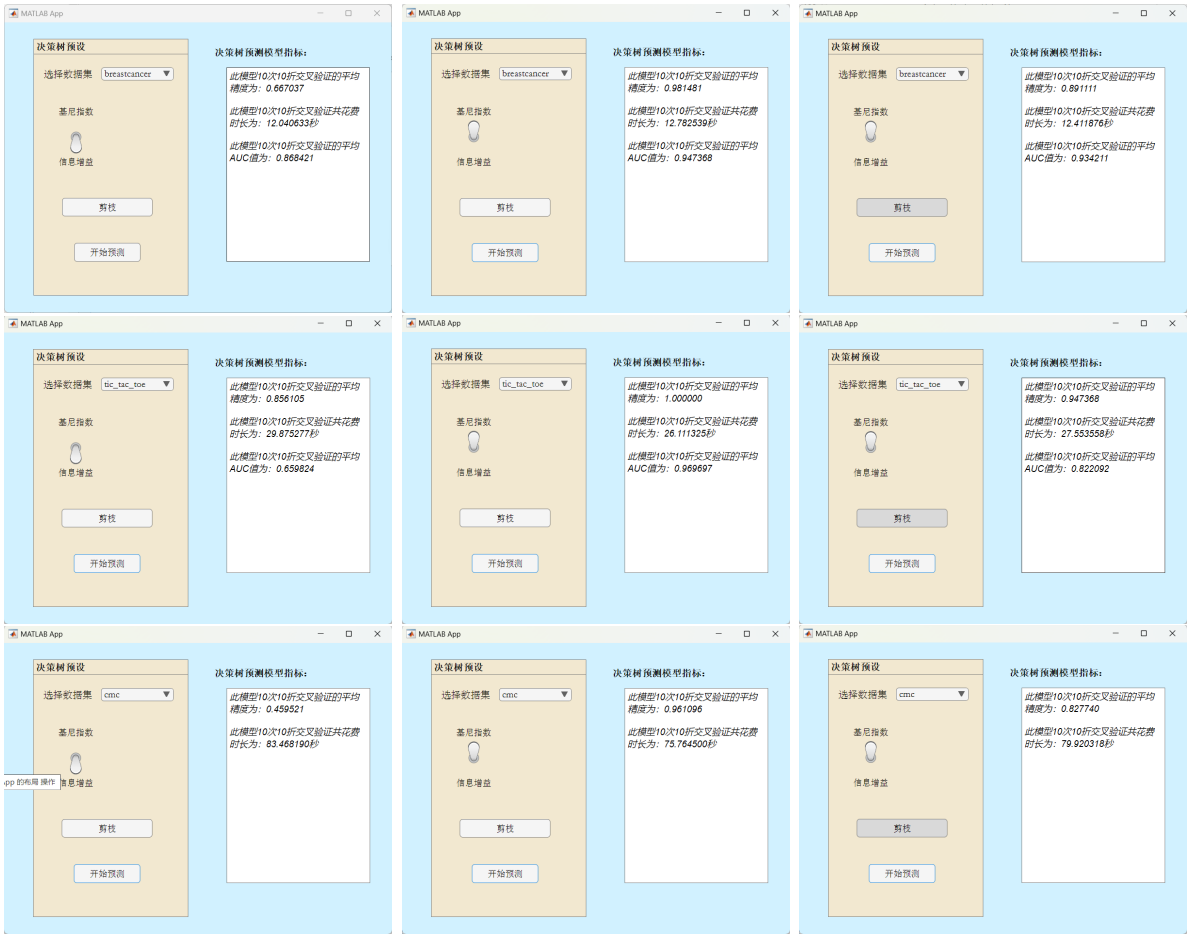

图 5-1　ID3 算法实现与改进运行结果汇总

汇总图5-1中的运行结果可得统计表5-1，分析表中数据可得以下信息：

- 改进后的 10 次 10 折交叉验证指标无论是在平均精度、总花费时长，还是平均 AUC 值均有较大幅度的提升，改进效果明显。

- 原始模型 ID3 算法对于含有连续属性的 cmc 三分类数据集的分类效果确实较差，10 次 10 折交叉验证的平均精度只有 0.460。改进后由于基尼指数的引入，对连续属性的分类更加友好，同时节点处理也提升了多分类的效果。

- 在三个模型中，tic_tac_toe 数据集的分类效果最好，原因在于该数据集描述的时井字棋的胜利情况，属性全部时离散的，并且规则性较强，十分适合决策树算法的分类，故而在使用基尼指数与节点处理的 ID3 算法改进后，在该数据集上甚至达到了 100% 的平均精度。

- 在使用基尼指数与节点处理的 ID3 算法改进后，进一步添加后剪枝发现算法的分类指标有小幅下降，原因在于前一步的改进效果已经十分出色，且此次后剪枝选用错误率降低剪枝，针对特定训练集的分类效果有所上升，但在原本改进模型效果较好的情况下，10 次 10 折交叉验证通过 100 次的指标平均稀释了上升的效果且此次剪枝一定程度上改变了原本较好的分类结构，故而指标有所降低。

- 注意到在同一模型下部分数据集 10 次 10 折交叉验证产生的平均 AUC 值与平均精度的比较结果并不同步，这是由于精度只是某个随机样本的一个属性指标，而 AUC 并不关注某个截断值的表现如何，是综合所有截断值的预测性能，所以精度高，AUC 不一定大，反之亦然[10]。

表 5-1 模型 10 次 10 折交叉验证指标统计汇总表

	数据集	**平均精度**	**花费时长**	**平均 AUC 值**	**实例数**
原始模型 (ID3 算法)	breastcancer	0.667	12.04s	0.868	277
	tic_tac_toe	0.856	29.88s	0.66	958
	cmc	0.460	83.47s	\	1473
改进 (基尼指数与节点处理 -无后剪枝)	breastcancer	0.981	12.78s	0.947	277
	tic_tac_toe	1.000	26.11s	0.97	958
	cmc	0.961	75.76s	\	1473
改进 (基尼指数与节点处理 -有后剪枝)	breastcancer	0.891	12.41s	0.934	277
	tic_tac_toe	0.947	27.55s	0.822	958
	cmc	0.828	79.92s	\	1473

参考文献

[1] 周志华. 机器学习[M]. 北京: 清华大学出版社, 2016. I

[2] 杜威铭, 冉羽. 决策树 ID3 算法研究[J]. 科技视界, 2018(11): 145-146. I

[3] BERRAR D. Cross-validation.[Z]. 2019. I

[4] QUINLAN J R. Induction of decision trees[J]. Machine learning, 1986, 1(1): 81-106. 1

[5] AZHAGUSUNDARI B, THANAMANI A S, et al. Feature selection based on information gain[J]. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 2013, 2(2): 18-21. 1

[6] TANGIRALA S. Evaluating the impact of gini index and information gain on classification using decision tree classifier algorithm[J]. International Journal of Advanced Computer Science and Applications, 2020, 11(2): 612-619. 2

[7] HARUN N H, HAMBALI H, HASSAN M G, et al. Matlab app designer: Learn by example (uum press)[M]. UUM Press, 2017. 6

[8] MOHAMED W N H W, SALLEH M N M, OMAR A H. A comparative study of reduced error pruning method in decision tree algorithms[C]//2012 IEEE International conference on control system, computing and engineering. IEEE, 2012: 392-397. 7

[9] 郑伟, 马楠. 一种改进的决策树后剪枝算法[J]. 计算机与数字工程, 2015, 43(6): 960-966. 8

[10] HUANG J, LING C X. Using auc and accuracy in evaluating learning algorithms[J]. IEEE Transactions on knowledge and Data Engineering, 2005, 17(3): 299-310. 10

附　录

附录 A　原始模型主函数 Tree.m

```
clear;
clc;
addpath('Dataset\');

%% 读取数据
load('breastcancer.mat')
% load('tic_tac_toe.mat')
% load('cmc.mat')
dataset_choose = breastcancer;
size_data = size(dataset_choose);

%% 10次10折交叉验证
k_t = 10;
cross_time = 10;
y_lable = dataset_choose(2:size_data(1),size_data(2));
T_P=zeros(k_t,cross_time);
tic;
for i = 1:cross_time
% 分为训练集和测试集(10折),
y_1 = find(strcmp(y_lable(:),y_lable(1)));
y_2 = find(~strcmp(y_lable(:),y_lable(1)));
y_1_length = length(y_1);
y_2_length = length(y_2);
y_1_perNum = floor(y_1_length/k_t);
```

```
y_2_perNum = floor(y_2_length/k_t);
y_1_randIndex = randperm(y_1_length);
y_2_randIndex = randperm(y_2_length);
% D中存放了10组数据索引
D_index = zeros(y_1_perNum+y_2_perNum,k_t);
for j = 1:k_t
D_index(:,j)=[...
y_1(y_1_randIndex(y_1_perNum*(j-1)+1:y_1_perNum*j));...
y_2(y_2_randIndex(y_2_perNum*(j-1)+1:y_2_perNum*j))];
end
D_index = D_index+1;
perNum_D = y_1_perNum+y_2_perNum;
% 训练10折交叉验证
for k=1:k_t
% 获取此时的数据集以及测试集
x_train = dataset_choose(...
[1; reshape(D_index(:,1:k-1),[],1);...
reshape(D_index(:,k+1:k_t),[],1)],:) ;
x_test = dataset_choose(D_index(:,k),:);
% 训练
size_data = size(x_train);
dataset = x_train(2:size_data(1),:); % 纯数据集
labels = x_train(1,1:size_data(2)-1); % 属性标签
% 生成决策树
mytree = ID3(dataset,labels);
% 预测测试集标签并计算精度
y_test = predict(x_test(:,1:end-1),mytree,labels);
T_P(i,k) = sum(strcmp(y_test',x_test(:,end)))/perNum_D;
```

```
end
end
toc;
% 绘制ROC曲线并计算AUC
y_all = predict(breastcancer(2:size_data(1),1:end-1),mytree,
    labels);
T_All = sum(strcmp(y_all',breastcancer(2:size_data(1),end)))/(
    size_data(1)-1);
auc = plot_roc(y_all',breastcancer(2:size_data(1),end));

%% 结果输出
fprintf('10次10折交叉验证的精度结果为:\n');
for i=1:10
fprintf('第%d次:%f\n',i,mean(T_P(i,:)));
fprintf('\t%f\t%f\t%f\t%f\t%f\n',T_P(i,1:5));
fprintf('\t%f\t%f\t%f\t%f\t%f\n',T_P(i,6:10));
end
fprintf('平均精度为: %f\n',mean(mean(T_P)));
% 决策树的可视化
[nodeids_,nodevalue_,branchvalue_] = print_tree(mytree);
tree_plot(nodeids_,nodevalue_,branchvalue_);
```

附录 B　ID3 算法构建 ID3.m

```
function myTree = ID3(dataset,labels)
% ID3算法构建决策树
% 输入参数:
% dataset: 数据集
% labels: 属性标签
```

```
% 输出参数:
% tree: 构建的决策树
size_data = size(dataset);
classList = dataset(:,size_data(2));    %得到标签

% 全为同一类, 熵为0
if length(unique(classList))==1
myTree =  char(classList(1));
return
end

% 去除完全相同的属性, 避免产生没有分类结果的节点
choose=ones(1,size_data(2));
for i=1:(size_data(2)-1)
featValues = dataset(:,i);
uniqueVals = unique(featValues);
if(length(uniqueVals)<=1)
choose(i)=0;
end
end
labels=labels((choose(1:size_data(2)-1))==1);
dataset=dataset(:,choose==1);

size_data = size(dataset);
classList = dataset(:,size_data(2));

% 属性集为空, 用找最多数
if size_data(2) == 1
```

```
34 temp=tabulate(classList);
35 value=temp(:,1);                    % 属性值
36 count=cell2mat(temp(:,2));          % 不同属性值的各自数量
37 index=find(max(count)==count);
38 choose=index(randi(length(index)));
39 myTree =  char(value(choose));
40 return
41 end
42
43 bestFeature = chooseFeature(dataset);              % 找到信息增益
       最大的特征
44 bestFeatureLabel = char(labels(bestFeature));      % 得到信息增
       益最大的特征的名字，即为接下来要删除的特征
45 myTree = containers.Map;
46 leaf = containers.Map;
47 featValues = dataset(:,bestFeature);
48 uniqueVals = unique(featValues);
49
50 labels=[labels(1:bestFeature-1) labels(bestFeature+1:length(
       labels))]; % 删除该特征
51
52 % 形成递归，一个特征的按每个类别再往下分
53 for i=1:length(uniqueVals)
54 subLabels = labels(:)';
55 value = char(uniqueVals(i));
56 subdata = splitDataset(dataset,bestFeature,value);     % 取出该
       特征值为value的所有样本，并去除该属性
57 leaf(value) = ID3(subdata,subLabels);
```

```
myTree(char(bestFeatureLabel)) = leaf;
end

end
```

附录 C　基于信息增益特征选取 chooseFeature.m

```
function bestFeature=chooseFeature(dataset,~)
% 选择信息增益最大的属性特征
% 数据预处理
[N,M]=size(dataset);                    %样本数量N
M=M-1;                                  %特征个数M
y=strcmp(dataset(:,M+1),dataset(1,M+1)); %标签y(以第一个标签为1)
x=dataset(:,1:M);                       %数据x
gain = (1:M);                           %创建一个数组，用于储存每个特征的信息增益
%bestFeature;                           %最大信息增益的特征
Ent_D=calShannonEnt(y);                 %当前信息熵
%计算信息增益
for i=1:M
% 计算第i种属性的增益
temp=tabulate(x(:,i));
value=temp(:,1);                %属性值
count=cell2mat(temp(:,2));      %不同属性值的各自数量
Kind_Num=length(value);         %取值数目
Ent=zeros(Kind_Num,1);
% i属性下 j取值的信息熵
for j=1:Kind_Num
```

```
% 在第j种取值下正例的数目
Ent(j)= calShannonEnt( y(strcmp(x(:,i),value(j))) );
end
gain(i)=Ent_D-count'/N*Ent;
end
%随机挑选一个最大值
max_gain=find(gain==max(gain));
choose=randi(length(max_gain));
bestFeature=max_gain(choose);
%%%%==========================================
end
```

附录 D　信息熵的计算 calShannonEnt.m

```
function shannonEnt = calShannonEnt(y)
% 计算信息熵
% y对应的标签,为1或0, 对应正例与反例
N=length(y);                %标签长度
P_T=sum(y)/N;               %正例概率
P_F=(N-sum(y))/N;             %反例概率
if(P_T==0||P_F==0)
% 使得p*log2p为0
shannonEnt = 0;
return
end
shannonEnt=-(P_T*log2(P_T)+P_F*log2(P_F));   %信息熵
end
```

附录 E　对测试样本进行分类 predict.m

```
function y_test=predict(x_test,mytree,feature_list)
% 对测试样本进行分类

y_test = {};
row = size(x_test);

for j= 1:row(1)
% 按队列顺序一个一个进
queue = {mytree};
feature_name = 0;
feature = 0;

while ~isempty(queue)
node = queue{1};
% 除去节点
queue(1) = [];
tag = 2;
% 直到找到叶节点
if string(class(node))~="containers.Map"
y_test{j} = node;
continue
elseif length(node.keys)==1
% 得到mytree节点的名字
feature_name = char(node.keys);
% mytree该特征所在的坐标
id = ismember(feature_list,feature_name);
```

```
x = x_test(j,:);
% 得到测试数据的特征属性
feature = x(id);
tag = 1;
end


%tag==2 即要走入下个节点
if tag==2
if string(class(node))=="containers.Map"
hasKeys=0;
keys = node.keys();
for i = 1:length(keys)
key = keys{i};
c = char(feature);
if strcmp(key,c)
% 队列变成该节点下面的节点
queue=[queue,{node(key)}];
hasKeys=1;
end
end
if(~hasKeys)
key = keys{randi(length(keys))};
% 队列变成该节点下面的节点
queue=[queue,{node(key)}];
end
end
end
```

```
55
56 %tag==1 即要选则符合测试数据的特征属性，这样就不用历遍整个
       mytree
57 if tag==1
58 if string(class(node))=="containers.Map"
59 keys = node.keys();
60 for i = 1:length(keys)
61 key = keys{i};
62 queue=[queue,{node(key)}];
63 end
64 end
65 end
66 end
67 if length(y_test)<j
68 test=1;
69 end
70 end
71
72 end
```

附录 F　改进-模型主函数 Tree_improve.m

```
1     clear;
2     clc;
3     addpath('Dataset\');
4     addpath('Improve\');
5
6     %% 读取数据
7     load('breastcancer.mat')
```

```
8   % load('tic_tac_toe.mat')
9   % load('cmc.mat')
10  dataset_choose = breastcancer;
11  size_data = size(dataset_choose);
12  p = 'recurrence-events';
13  n = 'no-recurrence-events';
14
15  %% 10次10折交叉验证
16  k_t=10;
17  cross_time=10;
18  y_lable=dataset_choose(2:size_data(1),size_data(2));
19  T_P=zeros(k_t,cross_time);
20  tic;
21  for i=1:cross_time
22  %分为训练集和测试集(10折),
23  y_1=find(strcmp(y_lable(:),y_lable(1)));
24  y_2=find(~strcmp(y_lable(:),y_lable(1)));
25  y_1_length=length(y_1);
26  y_2_length=length(y_2);
27  y_1_perNum=floor(y_1_length/k_t);
28  y_2_perNum=floor(y_2_length/k_t);
29  y_1_randIndex=randperm(y_1_length);
30  y_2_randIndex=randperm(y_2_length);
31  D_index=zeros(y_1_perNum+y_2_perNum,k_t);
32  for j=1:k_t
33  D_index(:,j)=[...
34  y_1(y_1_randIndex(y_1_perNum*(j-1)+1:y_1_perNum*j));...
35  y_2(y_2_randIndex(y_2_perNum*(j-1)+1:y_2_perNum*j))];
```

```
36      end
37      D_index=D_index+1;
38      perNum_D=y_1_perNum+y_2_perNum;
39      for k=1:k_t
40      if k~=k_t
41      x_train = dataset_choose(...
42      [1; reshape(D_index(:,1:k-1),[],1);...
43      reshape(D_index(:,k+2:k_t),[],1)],:) ;
44      % 这里加上了属性标签行
45      else
46      x_train = dataset_choose([1; reshape(D_index(:,2:k-1),[],1)
            ],:) ;
47      end
48      x_valid = dataset_choose(D_index(:,k),:);
49      x_test = dataset_choose(D_index(:,mod(k+1,k_t)+1),:);
50      % 训练
51      size_data = size(x_train);
52      dataset = x_train(2:size_data(1),:); % 纯数据集
53      labels = x_train(1,1:size_data(2)-1); % 属性标签
54      % 生成决策树
55      mytree = ID3_2(dataset,labels);
56      % 剪枝
57      [correct,tree_pruning] = pruning(x_valid,mytree,labels);
58      % 预测测试集标签并计算精度
59      y_test=predict_2(x_test(:,1:end-1),tree_pruning,labels);
60      T_P(i,k)=sum(strcmp(y_test',x_test(:,end)))/perNum_D;
61      %          auc(i,k) = plot_roc(y_test',x_test(:,end),p,n);
62      end
```

```
63      end
64      toc;
65      auc_1 = mean(mean(auc(i,k)));
66      % y_all = predict(breastcancer(2:size_data(1),1:end-1),
           mytree,labels);
67      % T_All = sum(strcmp(y_all',breastcancer(2:size_data(1),end
           )))/(size_data(1)-1);
68      % auc = plot_roc(y_all',breastcancer(2:size_data(1),end));
69
70      %% 结果输出
71      fprintf('10次10折交叉验证的精度结果为:\n');
72      for i=1:10
73      fprintf('第%d次:%f\n',i,mean(T_P(i,:)));
74      fprintf('\t%f\t%f\t%f\t%f\t%f\n',T_P(i,1:5));
75      fprintf('\t%f\t%f\t%f\t%f\t%f\n',T_P(i,6:10));
76      end
77      fprintf('平均精度为: %f\n',mean(mean(T_P)));
78      % 决策树的可视化
79      [nodeids_,nodevalue_,branchvalue_] = print_tree(mytree);
80      tree_plot(nodeids_,nodevalue_,branchvalue_);
```

附录 G　改进-基于基尼指数的特征选取 chooseFeatureGini.m

```
1  function bestFeature=chooseFeatureGini(dataset,~)
2  % 选择基尼指数最小的属性特征
3
4  %数据预处理
5  [N,M]=size(dataset);                    %样本数量N
6  M=M-1;                                  %特征个数M
```

```
y=strcmp(dataset(:,M+1),dataset(1,M+1)); %标签y(以第一个标签为1)
x=dataset(:,1:M);                        %数据x
Gini_index = zeros(1,M);                 %创建一个数组，用于储存每个特征的基尼指数
%bestFeature;                            %最小基尼系数的特征

%计算基尼指数
for i=1:M
% 计算第i种属性的基尼指数
temp=tabulate(x(:,i));
value=temp(:,1);                 %属性值
count=cell2mat(temp(:,2));       %不同属性值的各自数量
Kind_Num=length(value);          %取值数目
Gini=zeros(Kind_Num,1);
% i属性下 j取值的基尼指数
for j=1:Kind_Num
% 在第j种取值下正例的数目
Gini(j)= getGini( y(strcmp(x(:,i),value(j))) );
end
Gini_index(i)=count'/N*Gini;
end
%随机挑选一个最小值
min_GiniIndex=find(Gini_index==min(Gini_index));
choose=randi(length(min_GiniIndex));
bestFeature=min_GiniIndex(choose);
end
```

附录 H　改进-计算基尼指数 getGini.m

```
function Gini = getGini(y)
% 计算基尼指数
% y对应的标签,为1或0, 对应正例与反例
N=length(y);                %标签长度
P_T=sum(y)/N;               %正例概率
P_F=1-P_T;                  %正例概率
Gini=1-P_T*P_T-P_F*P_F;   %基尼指数
end
```

附录 I　改进-节点处理 1 ID3_2.m

```
function [myTree] = ID3_2(dataset,labels)
% ID3算法构建决策树
% 输入参数:
% dataset: 数据集
% labels: 属性标签
% 输出参数:
% tree: 构建的决策树
size_data = size(dataset);
classList = dataset(:,size_data(2));    %得到标签

%全为同一类, 熵为0
if length(unique(classList))==1
myTree =  char(classList(1));
return
end
```

```
16
17 %去除完全相同的属性，避免产生没有分类结果的节点
18 choose=ones(1,size_data(2));
19 for i=1:(size_data(2)-1)
20 featValues = dataset(:,i);
21 uniqueVals = unique(featValues);
22 if(length(uniqueVals)<=1)
23 choose(i)=0;
24 end
25 end
26 labels=labels((choose(1:size_data(2)-1))==1);
27 dataset=dataset(:,choose==1);
28
29 size_data = size(dataset);
30 classList = dataset(:,size_data(2));
31 % 属性集为空，找最多数
32 temp=tabulate(classList);
33 value=temp(:,1);                % 属性值
34 count=cell2mat(temp(:,2));      % 不同属性值的各自数量
35 index=find(max(count)==count);
36 choose=index(randi(length(index)));
37 nodeLable =  char(value(choose));
38 if size_data(2) == 1
39 myTree =  nodeLable;
40 return
41 end
42
43 % 找到基尼指数最小的特征
```

```
bestFeature = chooseFeatureGini(dataset);
% 得到基尼指数最小的特征的名字, 即为接下来要删除的特征
bestFeatureLabel = char(labels(bestFeature));
myTree = containers.Map;
leaf = containers.Map;
featValues = dataset(:,bestFeature);
uniqueVals = unique(featValues);

labels=[labels(1:bestFeature-1) labels(bestFeature+1:length(
    labels))]; %删除该特征

% 形成递归, 一个特征的按每个类别再往下分
for i=1:length(uniqueVals)
subLabels = labels(:)';
value = char(uniqueVals(i));
subdata = splitDataset(dataset,bestFeature,value);      %取出该特
    征值为value的所有样本,并去除该属性
leaf(value) = ID3_2(subdata,subLabels);
end
leaf('nodeLabel')= nodeLable;
myTree(char(bestFeatureLabel)) = leaf;
end
```

附录 J　改进-节点处理 2 predict_2.m

```
function y_test=predict_2(x_test,mytree,feature_list)
% 对测试样本进行分类

y_test = {};
```

```
5  row = size(x_test);
6
7
8  for j= 1:row(1)
9  % 按队列顺序一个一个进
10 queue = {mytree};
11 feature_name = 0;
12 feature = 0;
13
14 while ~isempty(queue)
15 node = queue{1};
16 % 除去节点
17 queue(1) = [];
18 tag = 2;
19 % 直到找到叶节点
20 if string(class(node))~="containers.Map"
21 y_test{j} = node;
22 continue
23 else
24 % 除去nodelable标签 (不影响检测)
25 keys = node.keys();
26 index=find(strcmp(keys,'nodeLabel'));
27 if(~isempty(index))
28 keys=[keys(1:(index-1)),keys((index+1):end)];
29 end
30
31 if length(keys)==1
32 % 得到mytree节点的名字
```

```
feature_name = char(keys);
% mytree该特征所在的坐标
id = ismember(feature_list,feature_name);
x = x_test(j,:);
% 得到测试数据的特征属性
feature = x(id);
tag = 1;
end
% tag==2 即要走入下个节点
if tag==2
hasKeys=0;
for i = 1:length(keys)
key = keys{i};
c = char(feature);
if strcmp(key,c)
% 队列变成该节点下面的节点
queue=[queue,{node(key)}];
hasKeys=1;
end
end
if(~hasKeys)
% 选取该标签的最大值
y_test{j}=node('nodeLabel');
end
end

% tag==1 即要选则符合测试数据的特征属性，这样就不用历遍整个
   mytree
```

```
60 if tag==1
61 for i = 1:length(keys)
62 key = keys{i};
63 % 队列变成该节点下面的节点
64 queue=[queue,{node(key)}];
65 end
66 end
67 end
68 end
69 end
70 end
```

附录 K　改进-后剪枝 pruning.m

```
1  function [correct,tree_pruning] = pruning(x_V,tree,feature_list
       )
2  % 剪枝
3  % correct: 返回的数据集的预测值正确程度数组, 1为预测正确
4  % tree_pruning: 剪枝后的数组
5  % x_V: 训练集
6  % tree: 剪枝前的树
7  % feature_list: 训练集的标签
8  if(string(class(tree))~="containers.Map")
9  %达到叶节点, 计算标签与当前数据的真实标签的异同
10 %将结果保存在correct数组中
11 correct=strcmp(x_V(:,end),tree)';
12 tree_pruning=tree;%返回原本的节点
13 return;
14 else
```

```
15 size_data = size(x_V);
16 labels=feature_list;                %数据的属性
17 Feature=char(tree.keys);            %当前节点的属性
18 FeatureIndex=strcmp(labels,Feature);%节点属性在所有属性中的索引
19 FeatureValue=x_V(:,FeatureIndex);    %所有属性
20 x_V=x_V(:,logical([~FeatureIndex,1])); %删除该特征
21 feature_list=feature_list(~FeatureIndex);
22 theTree = containers.Map;%新的节点以及边
23 theLeaf = containers.Map;
24 leaf=tree(Feature);%原本的叶子节点
25 keys=leaf.keys;      %获取属性的取值
26 %除去nodelable标签 (不影响检测)
27 index=find(strcmp(keys,'nodeLabel'));
28 if(~isempty(index))
29 keys=[keys(1:(index-1)),keys((index+1):end)];
30 end
31
32 correct=[];   %数据将包含目前数据预测的正确与否, 为0-1数组
33 for i=1:length(keys)
34 value=keys{i};
35 x_V_value=x_V(strcmp(FeatureValue,value),:); %删除拥有特征的数
      量
36 if(~isempty(x_V_value))
37 %数据集里有该取值, 计算预测结果正确与否
38 [correct_per,theLeaf(value)] = pruning(x_V_value,leaf(value),
      feature_list);
39 correct=[correct,correct_per];
40 else
```

```
%数据集里没有该取值，保留原本的节点
theLeaf(value)=leaf(value);
end
end
theLeaf('nodeLabel')= char(leaf('nodeLabel'));%获取之前的节点
theTree(Feature) = theLeaf;

acc = sum(correct)/length(correct);%原本的精度
acc_pruning = strcmp(x_V(:,end),leaf('nodeLabel'))/size_data(1)
    ;%不划分的精度
if(acc<=acc_pruning)
%假如不划分的精度更高，那么选取原本训练时最多的标签
tree_pruning= char(leaf('nodeLabel'));
else
%保留树
tree_pruning=theTree;
end
end

end
```

附录 L　MatlabAPP 可视化 UI 设计代码

```
classdef TreeUI < matlab.apps.AppBase

% Properties that correspond to app components
properties (Access = public)
UIFigure        matlab.ui.Figure
zhibiao         matlab.ui.control.TextArea
```

```
Label_2          matlab.ui.control.Label
yushe            matlab.ui.container.Panel
startpri         matlab.ui.control.Button
pruning          matlab.ui.control.StateButton
featurechoose    matlab.ui.control.ToggleSwitch
datachoose       matlab.ui.control.DropDown
Label            matlab.ui.control.Label
end

% Callbacks that handle component events
methods (Access = private)

% Button pushed function: startpri
function startpriButtonPushed(app, event)
addpath('Dataset\');
addpath('Improve\');
disp("start!");
if app.featurechoose.Value == "信息增益"
% 读取数据
if app.datachoose.Value == "breastcancer"
load('breastcancer.mat')
dataset_choose = breastcancer;
p = 'recurrence-events';
n = 'no-recurrence-events';
end
if app.datachoose.Value == "tic_tac_toe"
load('tic_tac_toe.mat')
dataset_choose = tic_tac_toe;
```

```
35 p = 'positive';
36 n = 'negative';
37 end
38 if app.datachoose.Value == "cmc"
39 load('cmc.mat')
40 dataset_choose = cmc;
41 end
42 size_data = size(dataset_choose);
43
44 %% 10次10折交叉验证
45 k_t = 10;
46 cross_time = 10;
47 y_lable = dataset_choose(2:size_data(1),size_data(2));
48 T_P=zeros(k_t,cross_time);
49 tic;
50 for i = 1:cross_time
51 %分为训练集和测试集(10折),
52 y_1 = find(strcmp(y_lable(:),y_lable(1)));%与第一个标签相同的为
    一层次
53 y_2 = find(~strcmp(y_lable(:),y_lable(1)));%其余为另一个层次
54 y_1_length = length(y_1);
55 y_2_length = length(y_2);
56 y_1_perNum = floor(y_1_length/k_t);
57 y_2_perNum = floor(y_2_length/k_t);
58 y_1_randIndex = randperm(y_1_length);
59 y_2_randIndex = randperm(y_2_length);
60 D_index = zeros(y_1_perNum+y_2_perNum,k_t); %D中存放了10组数据
    索引
```

```
for j = 1:k_t                                    %有数据被丢弃
D_index(:,j)=[...
y_1(y_1_randIndex(y_1_perNum*(j-1)+1:y_1_perNum*j));...
y_2(y_2_randIndex(y_2_perNum*(j-1)+1:y_2_perNum*j))];
end
D_index = D_index+1;
perNum_D = y_1_perNum+y_2_perNum;
% 训练10折交叉验证
for k=1:k_t
% 获取此时的数据集以及测试集
x_train = dataset_choose(...
[1; reshape(D_index(:,1:k-1),[],1);...
reshape(D_index(:,k+1:k_t),[],1)],:) ;      % 这里加上了属性标签行
x_test = dataset_choose(D_index(:,k),:);   % 选择最后两个当测试集
% 训练
size_data = size(x_train);
dataset = x_train(2:size_data(1),:); % 纯数据集
labels = x_train(1,1:size_data(2)-1); % 属性标签
% 生成决策树
mytree = ID3(dataset,labels);
% 预测测试集标签并计算精度
y_test = predict(x_test(:,1:end-1),mytree,labels);
T_P(i,k) = sum(strcmp(y_test',x_test(:,end)))/perNum_D;
if app.datachoose.Value ~= "cmc"
auc(i,k) = plot_roc(y_test',x_test(:,end),p,n);
end
```

```
87  end
88  end
89  [nodeids_,nodevalue_,branchvalue_] = print_tree(mytree);
90  tree_plot(nodeids_,nodevalue_,branchvalue_);
91  toc;
92  cellArrayText{1} = sprintf('此模型10次10折交叉验证的平均精度
        为: %f\n\n',mean(mean(T_P)));
93  cellArrayText{2} = sprintf('此模型10次10折交叉验证共花费时长
        为: %f秒\n\n',toc);
94  if app.datachoose.Value ~= "cmc"
95  cellArrayText{3} = sprintf('此模型10次10折交叉验证的平均AUC值
        为: %f\n',mean(mean(auc(i,k))));
96  end
97  app.zhibiao.Value = cellArrayText;
98  end
99  if app.featurechoose.Value == "基尼指数"
100 if app.pruning.Value == 0
101 % 读取数据
102 if app.datachoose.Value == "breastcancer"
103 load('breastcancer.mat')
104 dataset_choose = breastcancer;
105 p = 'recurrence-events';
106 n = 'no-recurrence-events';
107 end
108 if app.datachoose.Value == "tic_tac_toe"
109 load('tic_tac_toe.mat')
110 dataset_choose = tic_tac_toe;
111 p = 'positive';
```

```
112 n = 'negative';
113 end
114 if app.datachoose.Value == "cmc"
115 load('cmc.mat')
116 dataset_choose = cmc;
117 end
118 size_data = size(dataset_choose);
119
120 %% 10次10折交叉验证
121 k_t=10;
122 cross_time=10;
123 y_lable=dataset_choose(2:size_data(1),size_data(2));
124 T_P=zeros(k_t,cross_time);
125 tic;
126 for i=1:cross_time
127 %分为训练集和测试集(10折),
128 y_1=find(strcmp(y_lable(:),y_lable(1)));%与第一个标签相同的为一
      层次
129 y_2=find(~strcmp(y_lable(:),y_lable(1)));%其余为另一个层次
130 y_1_length=length(y_1);
131 y_2_length=length(y_2);
132 y_1_perNum=floor(y_1_length/k_t);
133 y_2_perNum=floor(y_2_length/k_t);
134 y_1_randIndex=randperm(y_1_length);
135 y_2_randIndex=randperm(y_2_length);
136 D_index=zeros(y_1_perNum+y_2_perNum,k_t);
137 for j=1:k_t                                    %有数据被丢弃
138 D_index(:,j)=[...
```

```
y_1(y_1_randIndex(y_1_perNum*(j-1)+1:y_1_perNum*j));...
y_2(y_2_randIndex(y_2_perNum*(j-1)+1:y_2_perNum*j))];
end
D_index=D_index+1;
perNum_D=y_1_perNum+y_2_perNum;
for k=1:k_t
if k~=k_t
x_train = dataset_choose(...
[1; reshape(D_index(:,1:k-1),[],1);...
reshape(D_index(:,k+2:k_t),[],1)],:) ;      % 这里加上了属性标签行
else
x_train = dataset_choose([1; reshape(D_index(:,2:k-1),[],1)],:) ;      % 这里加上了属性标签行
end
x_valid = dataset_choose(D_index(:,k),:);                    % 选择验证集
x_test = dataset_choose(D_index(:,mod(k+1,k_t)+1),:);   % 选择测试集
% 训练
size_data = size(x_train);
dataset = x_train(2:size_data(1),:); % 纯数据集
labels = x_train(1,1:size_data(2)-1); % 属性标签
% 生成决策树
mytree = ID3_2(dataset,labels);
% 预测测试集标签并计算精度
y_test=predict_2(x_test(:,1:end-1),mytree,labels);
T_P(i,k)=sum(strcmp(y_test',x_test(:,end)))/perNum_D;
```

```
163 if app.datachoose.Value ~= "cmc"
164 auc(i,k) = plot_roc(y_test',x_test(:,end),p,n);
165 end
166 end
167 end
168 [nodeids_,nodevalue_,branchvalue_] = print_tree(mytree);
169 tree_plot(nodeids_,nodevalue_,branchvalue_);
170 toc;
171 cellArrayText{1} = sprintf('此模型10次10折交叉验证的平均精度
    为: %f\n\n',mean(mean(T_P)));
172 cellArrayText{2} = sprintf('此模型10次10折交叉验证共花费时长
    为: %f秒\n\n',toc);
173 if app.datachoose.Value ~= "cmc"
174 cellArrayText{3} = sprintf('此模型10次10折交叉验证的平均AUC值
    为: %f\n',mean(mean(auc(i,k))));
175 end
176 app.zhibiao.Value = cellArrayText;
177 end
178 if app.pruning.Value == 1
179 % 读取数据
180 if app.datachoose.Value == "breastcancer"
181 load('breastcancer.mat')
182 dataset_choose = breastcancer;
183 p = 'recurrence-events';
184 n = 'no-recurrence-events';
185 end
186 if app.datachoose.Value == "tic_tac_toe"
187 load('tic_tac_toe.mat')
```

```
188 dataset_choose = tic_tac_toe;
189 p = 'positive';
190 n = 'negative';
191 end
192 if app.datachoose.Value == "cmc"
193 load('cmc.mat')
194 dataset_choose = cmc;
195 end
196 size_data = size(dataset_choose);
197
198 %% 10次10折交叉验证
199 k_t=10;
200 cross_time=10;
201 y_lable=dataset_choose(2:size_data(1),size_data(2));
202 T_P=zeros(k_t,cross_time);
203 tic;
204 for i=1:cross_time
205 %分为训练集和测试集(10折),
206 y_1=find(strcmp(y_lable(:),y_lable(1)));%与第一个标签相同的为一
      层次
207 y_2=find(~strcmp(y_lable(:),y_lable(1)));%其余为另一个层次
208 y_1_length=length(y_1);
209 y_2_length=length(y_2);
210 y_1_perNum=floor(y_1_length/k_t);
211 y_2_perNum=floor(y_2_length/k_t);
212 y_1_randIndex=randperm(y_1_length);
213 y_2_randIndex=randperm(y_2_length);
214 D_index=zeros(y_1_perNum+y_2_perNum,k_t);
```

```
215 for j=1:k_t                                    %有数据被丢弃
216 D_index(:,j)=[...
217 y_1(y_1_randIndex(y_1_perNum*(j-1)+1:y_1_perNum*j));...
218 y_2(y_2_randIndex(y_2_perNum*(j-1)+1:y_2_perNum*j))];
219 end
220 D_index=D_index+1;
221 perNum_D=y_1_perNum+y_2_perNum;
222 for k=1:k_t
223 if k~=k_t
224 x_train = dataset_choose(...
225 [1; reshape(D_index(:,1:k-1),[],1);...
226 reshape(D_index(:,k+2:k_t),[],1)],:) ;      % 这里加上了属性标签
    行
227 else
228 x_train = dataset_choose([1; reshape(D_index(:,2:k-1),[],1)],:)
    ;      % 这里加上了属性标签行
229 end
230 x_valid = dataset_choose(D_index(:,k),:);                % 选择验
    证集
231 x_test = dataset_choose(D_index(:,mod(k+1,k_t)+1),:);   % 选择测
    试集
232 % 训练
233 size_data = size(x_train);
234 dataset = x_train(2:size_data(1),:); % 纯数据集
235 labels = x_train(1,1:size_data(2)-1); % 属性标签
236 % 生成决策树
237 mytree = ID3_2(dataset,labels);
238 % 剪枝
```

```
239 [correct ,mytree] = pruning(x_valid ,mytree ,labels);
240 % 预测测试集标签并计算精度
241 y_test=predict_2(x_test(:,1:end-1),mytree ,labels);
242 T_P(i,k)=sum(strcmp(y_test',x_test(:,end)))/perNum_D;
243 if app.datachoose.Value ~= "cmc"
244 auc(i,k) = plot_roc(y_test',x_test(:,end),p,n);
245 end
246 end
247 end
248 [nodeids_,nodevalue_,branchvalue_] = print_tree(mytree);
249 tree_plot(nodeids_,nodevalue_,branchvalue_);
250 toc;
251 cellArrayText{1} = sprintf('此模型10次10折交叉验证的平均精度
      为: %f\n\n',mean(mean(T_P)));
252 cellArrayText{2} = sprintf('此模型10次10折交叉验证共花费时长
      为: %f秒\n\n',toc);
253 if app.datachoose.Value ~= "cmc"
254 cellArrayText{3} = sprintf('此模型10次10折交叉验证的平均AUC值
      为: %f\n',mean(mean(auc(i,k))));
255 end
256 app.zhibiao.Value = cellArrayText;
257 end
258 end
259 end
260
261 % Value changed function: pruning
262 function pruningValueChanged(app, event)
263 value = app.pruning.Value;
```

```
264
265 end
266 end
267
268 % Component initialization
269 methods (Access = private)
270
271 % Create UIFigure and components
272 function createComponents(app)
273
274 % Create UIFigure and hide until all components are created
275 app.UIFigure = uifigure('Visible', 'off');
276 app.UIFigure.Color = [0.8196 0.9451 1];
277 app.UIFigure.Position = [100 100 640 480];
278 app.UIFigure.Name = 'MATLAB App';
279
280 % Create yushe
281 app.yushe = uipanel(app.UIFigure);
282 app.yushe.Title = '决策树预设';
283 app.yushe.BackgroundColor = [0.949 0.9098 0.8157];
284 app.yushe.FontName = '宋体';
285 app.yushe.FontWeight = 'bold';
286 app.yushe.FontSize = 16;
287 app.yushe.Position = [49 29 256 425];
288
289 % Create Label
290 app.Label = uilabel(app.yushe);
291 app.Label.HorizontalAlignment = 'right';
```

```
292 app.Label.FontName = '华文宋体';
293 app.Label.FontSize = 16;
294 app.Label.Position = [12 356 85 22];
295 app.Label.Text = '选择数据集';
296
297 % Create datachoose
298 app.datachoose = uidropdown(app.yushe);
299 app.datachoose.Items = {'breastcancer', 'tic_tac_toe', 'cmc'};
300 app.datachoose.FontName = '华文宋体';
301 app.datachoose.FontSize = 16;
302 app.datachoose.Position = [112 356 120 22];
303 app.datachoose.Value = 'breastcancer';
304
305 % Create featurechoose
306 app.featurechoose = uiswitch(app.yushe, 'toggle');
307 app.featurechoose.Items = {'信息增益', '基尼指数'};
308 app.featurechoose.FontName = '宋体';
309 app.featurechoose.FontSize = 14;
310 app.featurechoose.Position = [58 235 24 55];
311 app.featurechoose.Value = '信息增益';
312
313 % Create pruning
314 app.pruning = uibutton(app.yushe, 'state');
315 app.pruning.ValueChangedFcn = createCallbackFcn(app,
      @pruningValueChanged, true);
316 app.pruning.Text = '剪枝';
317 app.pruning.FontName = '宋体';
318 app.pruning.FontSize = 14;
```

```
319 app.pruning.Position = [47 131 150 31];
320
321 % Create startpri
322 app.startpri = uibutton(app.yushe, 'push');
323 app.startpri.ButtonPushedFcn = createCallbackFcn(app,
        @startpriButtonPushed, true);
324 app.startpri.FontName = '宋体';
325 app.startpri.FontSize = 14;
326 app.startpri.Position = [67 56 110 31];
327 app.startpri.Text = '开始预测';
328
329 % Create Label_2
330 app.Label_2 = uilabel(app.UIFigure);
331 app.Label_2.HorizontalAlignment = 'right';
332 app.Label_2.FontName = '宋体';
333 app.Label_2.FontSize = 16;
334 app.Label_2.FontWeight = 'bold';
335 app.Label_2.Position = [343 421 168 22];
336 app.Label_2.Text = '决策树预测模型指标: ';
337
338 % Create zhibiao
339 app.zhibiao = uitextarea(app.UIFigure);
340 app.zhibiao.FontSize = 15;
341 app.zhibiao.FontAngle = 'italic';
342 app.zhibiao.Position = [368 85 237 322];
343
344 % Show the figure after all components are created
345 app.UIFigure.Visible = 'on';
```

```
346 end
347 end
348
349 % App creation and deletion
350 methods (Access = public)
351
352 % Construct app
353 function app = TreeUI
354
355 % Create UIFigure and components
356 createComponents(app)
357
358 % Register the app with App Designer
359 registerApp(app, app.UIFigure)
360
361 if nargout == 0
362 clear app
363 end
364 end
365
366 % Code that executes before app deletion
367 function delete(app)
368
369 % Delete UIFigure when app is deleted
370 delete(app.UIFigure)
371 end
372 end
373 end
```