山东大学 计算机科学与技术 学院

机器学习 课程实验报告

学号：	姓名：	班级：
实验题目： Decision Tree		
实验学时： 5	**实验日期：** 2018/11/20-25	

实验目的：

对于 Wine 数据集，使用 python 或 matlab 建立决策树对红酒的质量进行预测分类，熟悉掌握决策树的实际运用。

硬件环境：

操作系统

Windows 10 家庭中文版 64-bit

CPU

Intel Core i5 7200U @ 2.50GHz 41 ° C

Kaby Lake-U/Y 14nm 工艺

RAM

8.00GB 单个的-通道 未知 (15-15-15-35)

主板

HP 81D1 (U3E1)

图像

Generic PnP Monitor (1920x1080@60Hz)

Intel HD Graphics 620 (HP)

存储器

476GB NVMe THNSN5512GPUK TO (未知)

40GB Microsoft 虚拟磁盘 (File-backed Virtual)

光盘驱动器

没有检测到光纤磁盘驱动

音频

Conexant ISST Audio

软件环境：

Win10 + python2.7

实验步骤与内容：

一、 构造决策树类

1、以具有最佳熵和信息增益为标准来进行属性的分裂，执行或二路或多路分割。

2、树的生长停止条件为要么全部在同一类（叶子）中，要么没有属性待分割选择或者没有剩下数据，除此之外，还使用修剪、正则化技术来停止分裂。

3、对连续数据采用均匀分布的朴素假设，并根据范围使用 10 个均匀区间进行划分。

4、根据 1：9 的比例划分测试集和训练集，可根据情况构造根据顺序的十种划分构造十颗树，选择有最好结果的树。

5、在训练数据集采用 10 折的交叉验证选择树的结构模型（对不同的剪枝和属性选择），采用后剪枝方案。

Tree 类的构造函数如下：

```
def __init__(self):
        self.attribs = []
        self.attribType = {}
        self.nodeList = {}
        self.data = []
        self.kTile = 15
        self.useGini = False
        self.useBinEntropy=False
        self.branchForEntropyGainOnly = False
        self.returnMajorityNotDefault = False
        self.pruneTree = True
        self.target = 'quality'
        self.ccf = 0
```

其中 attribs 为属性值列表，attribType 为属性的类别，可以为数值型和字符串型，在本例 wine 数据集中只有数值型，nodeList 存储节点列表，data 存储数据样本信息，ktile 用来动态调整选择出在训练集上有着最好结果的树结构，useGini 可作为可选项来比较使用基尼系数是否有更好的结果，其它的作为可选的拓展功能，target 属性是标签，ccf 用于衡量剪枝的复杂度，帮助选择怎样的减枝方案。

二、 程序执行流程说明

1、首先调用 **main** 主函数成簇入口：

```
if __name__ == "__main__":
    if len(sys.argv) > 1:
        if len(sys.argv) > 2 :
            run_decision_tree(True,sys.argv[2])
        else:
            run_decision_tree(True,None)
    else:
        run_decision_tree(False,None)
```

如果之前已经生成过了训练好的决策树，可以调用 python my_tree.py useSavedTree <json_file_name_to_read> 调用指定训练好的 json 文件决策树，执行 python my_tree.py useSavedTree 将调用默认的 my_decision_tree.json 文件作为决策树来对输入数据进行测试分类。

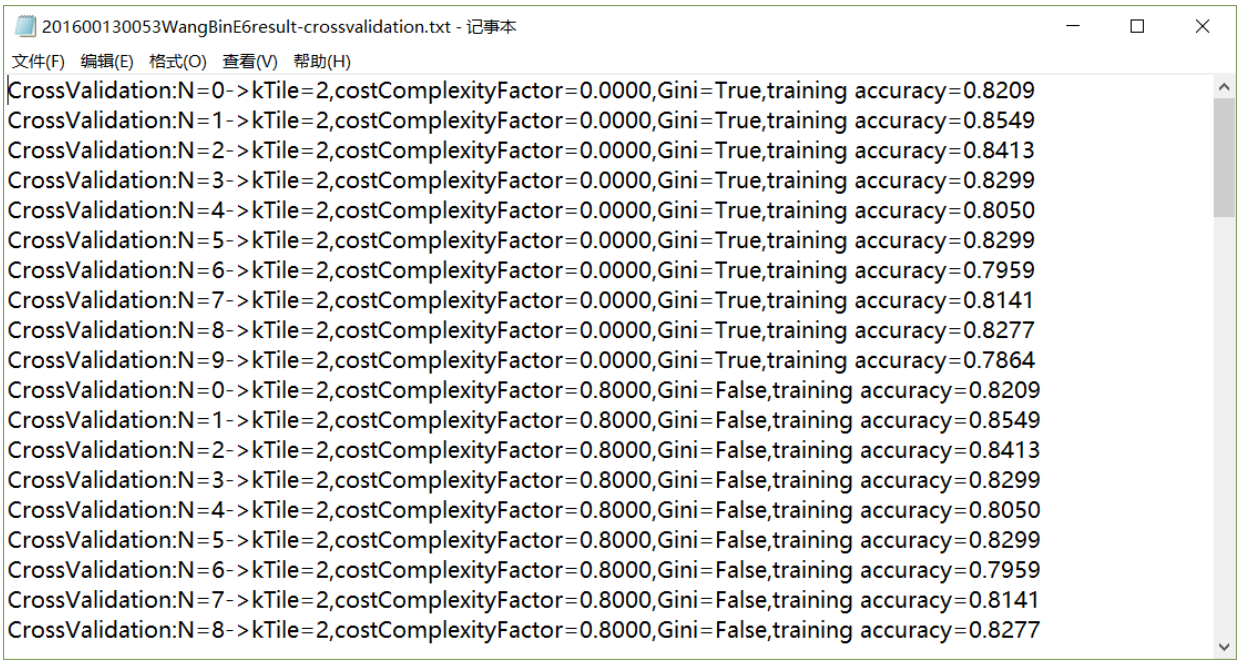
2、调用执行 `def run_decision_tree(useSavedDecisonTree,fileName)`

首先根据输入选择是否创建训练新的决策树还是调用已有的决策树，若是前者则会创建 tree 类对象，然后调用 `tree.learn( training_set, title)`，训练新的决策树，主要函数 learn()思路如下所示：

```
# 对训练集进行 10 次交叉验证
# 学习决策树和调整参数
# 选择 K 的大小 for Ktile
# 选择修剪复杂成本来帮助修剪枝
# 从交叉验证结果返回最佳树
# 用于测量已被保留的测试数据的精度。
```

```
for k in [2,4,5,8]:
    for useGini in [True,False]:
        self.useGini = useGini
        cv_avgAcc=0
        self.kTile = k
        pbuf = "kTile=%d,costComplexityFac
        count=0.0
        for N in xrange(0, 10):
```

使用三个循环来选出最好的决策树，中间过程打印显示如下：



最后将有最好表现的降序打印显示：

```
Best Training Results:kTile=4,costComplexityFactor=0.0000,Gini=False,Average Training accuracy: 0.8435
Best Training Results:kTile=5,costComplexityFactor=0.2000,Gini=False,Average Training accuracy: 0.8292
Best Training Results:kTile=8,costComplexityFactor=0.4000,Gini=False,Average Training accuracy: 0.8242
Best Training Results:kTile=2,costComplexityFactor=0.8000,Gini=False,Average Training accuracy: 0.8206
```

3、用测试集来进行测试得到最终数据

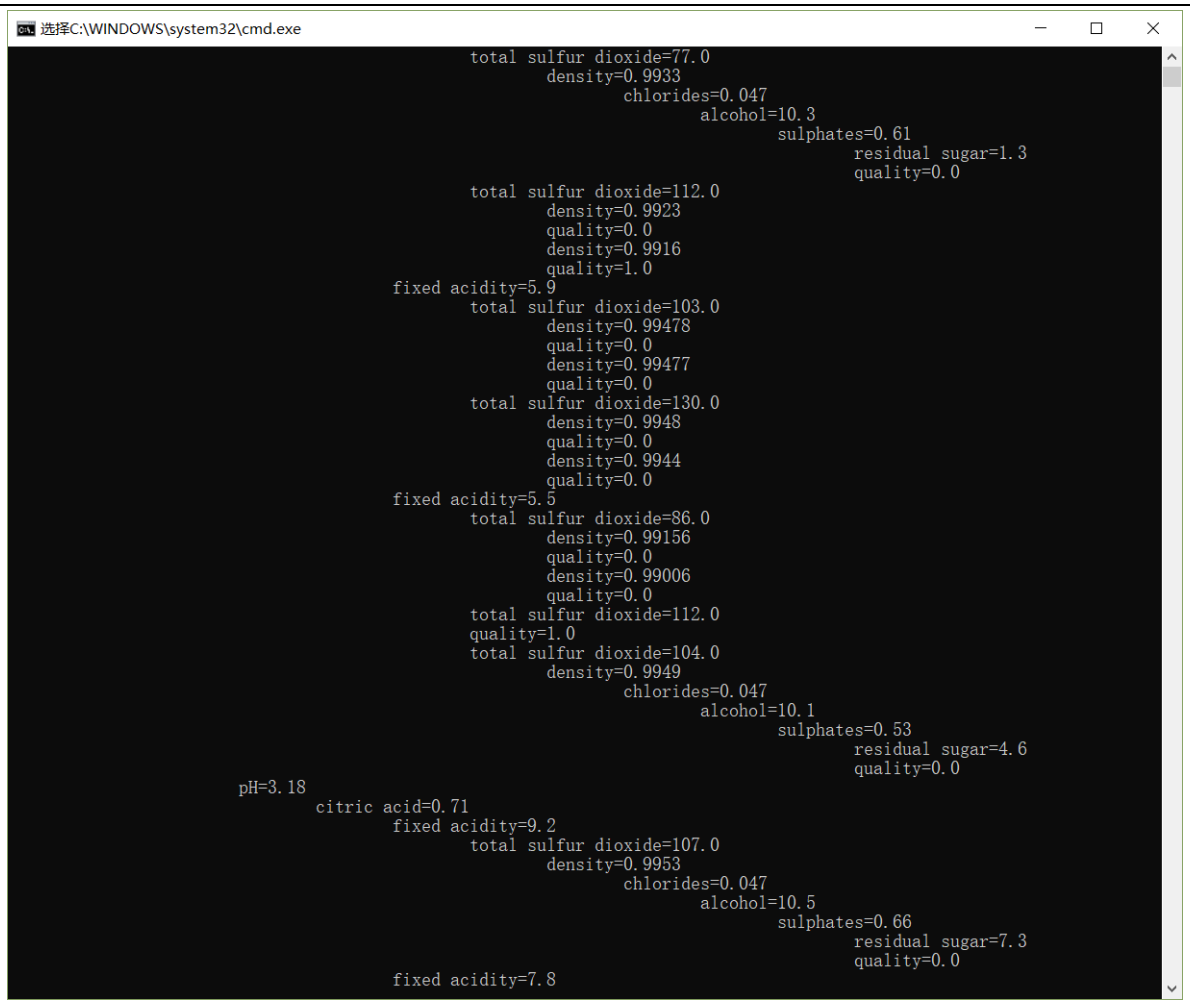
注意使用的基线版本是全预测为 0，有大概 0.78 的准确率，最终得到的树是不使用基尼系数，取 ktile=4，的树结构，在测试集上准确率如下：

accuracy: 0.8344

```
CrossValidation:N=6->kTile=5,costComplexityFactor=0.2000,Gini=False,training accuracy=0.8254
CrossValidation:N=7->kTile=5,costComplexityFactor=0.2000,Gini=False,training accuracy=0.8186
CrossValidation:N=8->kTile=5,costComplexityFactor=0.2000,Gini=False,training accuracy=0.8005
CrossValidation:N=9->kTile=5,costComplexityFactor=0.2000,Gini=False,training accuracy=0.8205
CrossValidation:N=0->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8186
CrossValidation:N=1->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8277
CrossValidation:N=2->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8413
CrossValidation:N=3->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8209
CrossValidation:N=4->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8277
CrossValidation:N=5->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8299
CrossValidation:N=6->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8413
CrossValidation:N=7->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8231
CrossValidation:N=8->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8027
CrossValidation:N=9->kTile=8,costComplexityFactor=0.2000,Gini=True,training accuracy=0.8091
CrossValidation:N=0->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8186
CrossValidation:N=1->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8277
CrossValidation:N=2->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8413
CrossValidation:N=3->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8209
CrossValidation:N=4->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8277
CrossValidation:N=5->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8299
CrossValidation:N=6->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8413
CrossValidation:N=7->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8231
CrossValidation:N=8->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8027
CrossValidation:N=9->kTile=8,costComplexityFactor=0.4000,Gini=False,training accuracy=0.8091
Best Tree from Cross Validation:kTile=4,costComplexityFactor=0.0000,Gini=False,training accuracy=0.8435
Best Training Results:kTile=4,costComplexityFactor=0.0000,Gini=False,Average Training accuracy: 0.8435
Best Training Results:kTile=5,costComplexityFactor=0.2000,Gini=False,Average Training accuracy: 0.8292
Best Training Results:kTile=8,costComplexityFactor=0.4000,Gini=False,Average Training accuracy: 0.8242
Best Training Results:kTile=2,costComplexityFactor=0.8000,Gini=False,Average Training accuracy: 0.8206
Saving the decision tree to disk as json
accuracy: 0.8344
```

三、 关于数据可视化

将节点列表进行分析处理打印后如下：



对于决策树结果打印显示比较乱，效果显示不是很好。

附录：程序源代码（请见附件中的 my_tree.py 文件）