《数据挖掘》大作业一报告

姓名： 王晨奕 学号：515030910362

姓名： 翟拙存 学号：515030910141

姓名： 李玮腾 学号：5140219262

一、实验目的

1. 给定多个候选实验数据集，选定某个数据集作为大作业的实验数据，建立适合该数据集的分类模型；

2. 了解数据挖掘中分类的各阶段，并能实现建立分类模型的算法。

二、数据分析和探索

本次实验选取HTRU观测数据为实验数据集，建立适合该数据集的分类模型，根据观测数据集分类判断所观测的恒星是否为脉冲星。

1. 数据集分析

HTRU2 (High Time Resolution Universe)观测数据集中包含了HTRU观测到的候选脉冲星的各项属性数据。

脉冲星是一种罕见的中子星，其辐射的无线电波能够在地球上被观测到。对脉冲星的观测研究有助于我们更深入地认识时空、星际物质媒介、物质的状态等方面的问题。

当脉冲星旋转时，其发射的光束扫过天空。当这些光束到达地球时，会产生可被侦测到的宽带无线电。当脉冲星迅速旋转时，这一过程周期性地重复。因此，对脉冲星的搜索需要用大型无线电望远镜寻找周期性无线电信号。

每个脉冲星会产生略微不同的发射图样，每次旋转都会发生轻微变化。因此，每一份“候选脉冲星”的信号检测数据是取一定时长内脉冲星多次转动的观测数据平均值。在没有额外信息的情况下，每个候选脉冲星都可能是一颗真正的脉冲星。然而，实际上几乎所有的检测都是由射频干扰（RFI）和噪声引起的，从而很难找到有效的信号。

为了提高脉冲星分析的效率，引入了机器学习算法，用于自动标记候选脉冲星。其中广泛用到分类模型，其将候选数据集视为二元分类问题。在候选脉冲星中，真实的脉冲星实例是占少数的正类，而由干扰噪声引起的虚假实例是占多数的负类。为了便于解释相关数据的意义，这里只提供少数几个标志性的数据作为分类属性。

实验数据集包含由射频干扰和噪声引起的16,259个虚假实例，以及1,639个真实脉冲星实例。以上数据实例均已经过人工检查。

实验数据集以两种格式呈现：CSV和ARFF。每个候选项的数据以两个文件的格式存储在不同的行中。每行首先列出各属性变量，最后一项为其类别，使用的类别标签为0（负类）和1（正类）。数据中不包含位置信息或其他天文细节，仅为使用PulsarFeatureLab工具从候选项文件中提取的特征数据。

实验数据集共包含17,898个候选项实例，其中1,639个为正类，16,259个为负类

2. 数据属性分析

每个候选项数据由8个连续变量和1个类别变量组成。前四个是从脉冲星完整轮廓（IntegratedProfile）中获得的简单统计数据。这是一组连续的变量，是在时间和频率上进行平均的信号数据。其余的四个变量则是从色散值-信噪比（DM-SNR）曲线中获得的简单统计数据。相关属性如下：

1) 完整轮廓（IntegratedProfile）均值；

2) 完整轮廓（IntegratedProfile）标准差；

3) 完整轮廓（IntegratedProfile）偏度；

4) 完整轮廓（IntegratedProfile）超峰度；

5) 色散值-信噪比（DM-SNR）曲线均值；

6) 色散值-信噪比（DM-SNR）曲线标准差；

7) 色散值-信噪比（DM-SNR）曲线偏度；

8) 色散值-信噪比（DM-SNR）曲线超峰度；

9) 类别。

其中，均值：mean =E[X]，标准差：standard deviation = D[X] = sqrt(E[X-E[X]]^2)，偏度：skewness = E[ ((X-E[X])/D[X])^3 ]，超峰度：excess kurtosis=E[(X-E[X])^4]/D[X]^4 -3。

三、分类算法的选择

实验分类算法选用决策树C4.5算法。

1. 分类算法选择理由

决策树算法具有一下优点：

1) 计算复杂度不高，易于理解和解释，可以理解决策树所表达的意义；

2) 数据预处理阶段较为简单；

3) 能够处理连续数据型的属性，且可对多属性的数据集构造决策树；

4) 在相对短的时间内能够对大数据集合作出可行且效果良好的分类结果。

具体到实验数据集中，实验数据的分类属性共有8个连续性的数据型变量。对于连续性的数据变量，需要将其离散化。通过观察各个属性数据与类别的关系，发现同一类别的数据均仅在一个区域内较为集中，因此可考虑在不同类别的两个集中区域间找到一个合适的分割点作为数据离散化的判断标准。从而可将连续性数据离散化，依据其数值划分为两段，适合于决策树算法的划分决策。此外，数据类别仅有正类、负类两种类别，且数据集有一定规模，这些情况均适合于使用决策树算法进行分类。

尽管决策树算法有忽略属性之间相关性、容易出现过拟合等缺点，但在实际运用中可通过数据预处理、剪枝等方法解决这些问题。

2. 决策树构建过程

1) 确定模型目标；

2) 数据预处理，确定训练集数据、验证集数据、目标变量、自变量；

3) 使用训练集进行决策树构建，从根节点开始进行不断的划分，通过划分后 节点的纯度确定最佳划分，并进行剪枝；

4) 使用测试集进行模型的验证；

5) 确定模型。

3. C4.5算法

输入：数据集D

C4.5( D ){

Tree = {};

if （D 纯度足够高or满足其他结束条件）

then结束;

end if;

for (all attribute a∈ D) do

for(对属性a进行所有规定的划分) do

计算每次划分的分割增益比GAINRATIO;

end for;

取GAINRAIO最高的一次划分作为属性a的最优划分;

end for;

取所有属性中最优划分下GAINRATIO的最大的属性作为a(best);

Tree = 在root上创建测试a(best)的决策节点;

D(v) = 依据属性a(best)划分产生的子数据集;

for (all D(v)) do

Tree(v) = C4.5(D(v));

将相应的树枝连接到Tree(v)上;

end for;

return Tree;

}

4. 划分判断标准

1) 节点熵：Entropy，衡量一个节点的同质性（纯度）。式中p(j|t)是指类别 j在节点t中的频度。

***Entropy(t) = -***

2) 信息增益：GAIN(split)，衡量一次划分所达到的熵减。父节点p作一次k划 分，i为划分出的子节点，n为节点中的数据数量。 

3) 划分信息：SplitINFO，衡量一次划分的信息量。



4) 划分增益比：GainRATIO(split)，对信息增益GAIN(split)进行调整，对高 熵值的划分进行惩罚。



5. 剪枝判断标准

1) 消极一般化错误：Pessimistic generalization errors，衡量是否出现过 拟合。式中e(T)为各子节点的总误判率，N为子节点数。



2) 后剪枝：Post-prune，最优划分完成后，通过比较父节点与子节点的消极 一般化错误决定是否进行剪枝处理。由于实验数据集进行二元划分，因此 N=2。当两个子节点总的消极一般化错误e(t1)+e(t2)+2\*0.5大于父节点的 消极一般化错误e(p) + 0.5时，判断为产生了过拟合，需要进行剪枝。

四、数据预处理

1. 数据预处理方法

1) 划分训练集和测试集

初始数据集共含有17,898项数据，其中负类16,259项，正类1,639项。为执行决策树分类算法，需要将原始数据集分为训练集和测试集。按7:3的比例分别从原始数据的正类和负类中随机划分成两组数据，归入训练集和测试集中。训练集包含有12528项数据，测试集含有5370项数据。

2) 数据预处理

超峰度excess kurtosis为峰度值kurtosis减3的修正值，首先将数据中的超峰度integrated profile excess kurtosis和DM-SNR curve excess kurtosis分别加3，改为峰度值kurtosis。观察各项属性数据的分布，发现部分属性数据如integratdprofile mean的分布接近于线性，而有的属性数据如DM-SNR mean的分布接近于指数增长。为了更好地寻找划分点，需要将各项属性数据处理为接近线性。因此将近似指数增长的几项属性数据（DM-SNR mean, DM-SNR kurtosis, integrated profile kurtosis）作取对数处理，使得修正后的属性数据分布近似为线性。

2. 数据预处理效果

训练集包含有12528项数据，其中负类11377项，正类1151项；测试集包含有5370项数据，其中负类4882项，正类488项。预处理后的各项属性数据的分布均近似线性，易于寻找划分点。

五、模型生成过程

1. 编译环境

Spyder(Python3.6)

2. 模型生成过程详解

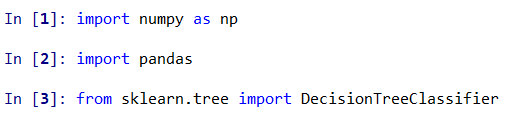
Step1：

导入指定库

• numpy (as np)

• pandas

• DecisionTreeClassifier from sklearn.tree



Step2：

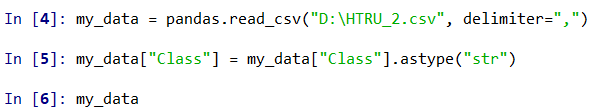
用my\_data读取来自skulls.csv的数据。声明以下变量

• X 作为 特征矩阵 (就是my\_data的数据)

• Y 作为 响应向量 (目标)

• targetNames 作为 响应向量名称列表 (目标名称)

• featureNames 作为 特征矩阵列名称列表



Out[6]:

Motip Sdotip Ekotip ... EkotDc SotDc Class

0 140.562500 55.683782 -0.234571 ... 7.975532 74.242225 0

1 102.507812 58.882430 0.465318 ... 10.576487 127.393580 0

2 103.015625 39.341649 0.323328 ... 7.735822 63.171909 0

3 136.750000 57.178449 -0.068415 ... 6.896499 53.593661 0

4 88.726562 40.672225 0.600866 ... 14.269573 252.567306 0

5 93.570312 46.698114 0.531905 ... 10.621748 131.394004 0

6 119.484375 48.765059 0.031460 ... 19.206230 479.756567 0

7 130.382812 39.844056 -0.158323 ... 13.539456 198.236457 0

8 107.250000 52.627078 0.452688 ... 9.001004 107.972506 0

9 107.257812 39.496488 0.465882 ... 7.397080 57.784738 0

10 142.078125 45.288073 -0.320328 ... 6.076266 37.831393 0

11 133.257812 44.058244 -0.081060 ... 11.972067 195.543448 0

12 134.960938 49.554327 -0.135304 ... 3.893934 14.131206 0

13 117.945312 45.506577 0.325438 ... 8.943212 82.475592 0

14 138.179688 51.524484 -0.031852 ... 5.155940 26.143310 0

15 114.367188 51.945716 -0.094499 ... 9.050612 96.611903 0

16 109.640625 49.017652 0.137636 ... 13.367926 223.438419 0

17 100.851562 51.743522 0.393837 ... 8.302242 71.584369 0

18 136.093750 51.691005 -0.045909 ... 4.345438 18.673649 0

19 99.367188 41.572202 1.547197 ... 2.208808 3.662680 1

20 100.890625 51.890394 0.627487 ... 6.953168 52.279440 0

21 105.445312 41.139969 0.142654 ... 7.739552 68.519771 0

22 95.867188 42.059922 0.326387 ... 11.249331 177.230771 0

23 117.367188 53.908614 0.257953 ... 4.807783 25.522616 0

24 106.648438 56.367182 0.378355 ... 9.378660 96.860225 0

25 112.718750 50.301270 0.279391 ... 4.691827 21.276210 0

26 130.851562 52.432857 0.142597 ... 9.464164 115.673159 0

27 119.437500 52.874815 -0.002549 ... 9.008352 94.755657 0

28 123.210938 51.078012 0.179377 ... 10.080333 112.558591 0

29 102.617188 49.692354 0.230439 ... 12.646535 171.832902 0

... ... ... ... ... ... ...

17868 96.859375 49.822769 0.276811 ... 11.743961 160.619545 0

17869 137.375000 50.762775 0.085075 ... 4.782510 23.379982 0

17870 110.765625 55.342186 0.066100 ... 3.120461 8.463600 0

17871 119.007812 52.091041 0.215294 ... 10.339768 111.760522 0

17872 139.898438 44.281034 -0.044618 ... 15.156427 280.553590 0

17873 89.453125 35.848827 0.731656 ... 11.203558 142.473878 0

17874 126.554688 54.275888 0.207825 ... 10.469100 122.767628 0

17875 133.140625 43.706181 -0.012686 ... 9.821958 131.521033 0

17876 111.765625 52.943585 0.671724 ... -0.696153 -1.260403 1

17877 118.296875 49.932824 0.144401 ... 16.739483 355.416666 0

17878 90.148438 40.786044 0.457139 ... 6.102231 37.979520 0

17879 107.664062 44.462463 0.296046 ... 5.536014 30.245117 0

17880 105.492188 43.500718 0.154180 ... 9.377229 95.200946 0

17881 130.500000 46.149553 0.084445 ... 12.533831 211.327081 0

17882 131.000000 55.935789 -0.023598 ... 15.019974 289.362790 0

17883 84.421875 41.775555 0.757467 ... 14.502863 258.119819 0

17884 106.875000 47.571328 0.199440 ... 8.427475 78.259366 0

17885 133.820312 43.478161 0.136691 ... 18.528395 444.411748 0

17886 127.070312 47.663564 0.006552 ... 16.122702 296.507738 0

17887 121.375000 53.245158 0.103772 ... 16.258042 303.880023 0

17888 98.726562 50.407823 0.565124 ... 22.018589 561.833787 0

17889 126.625000 55.721826 0.002946 ... 23.913761 660.197035 0

17890 143.671875 45.302647 -0.045769 ... 5.706651 33.802613 0

17891 118.484375 50.608483 -0.029059 ... 27.446113 830.638550 0

17892 96.000000 44.193113 0.388674 ... 9.634927 104.821623 0

17893 136.429688 59.847421 -0.187846 ... 15.450260 285.931022 0

17894 122.554688 49.485605 0.127978 ... 2.945244 8.297092 0

17895 119.335938 59.935939 0.159363 ... 2.499517 4.595173 0

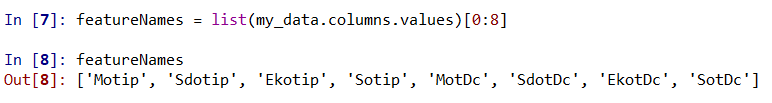
17896 114.507812 53.902400 0.201161 ... 10.007967 134.238910 0

17897 57.062500 85.797340 1.406391 ... -1.597527 1.429475 0

[17898 rows x 9 columns]

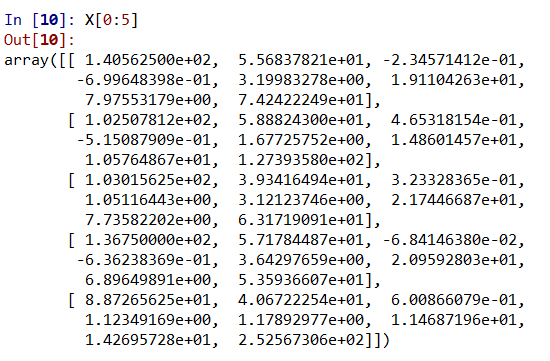
Step3：

获得数据集的属性名列表



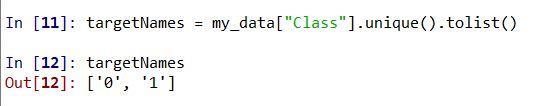
Step4：

清洗数据集，将数据集中不包含数值的目标列和表示序号的列删除



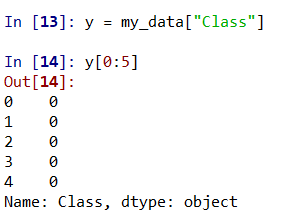
Step5：

列出不同的分类结果值



Step6：

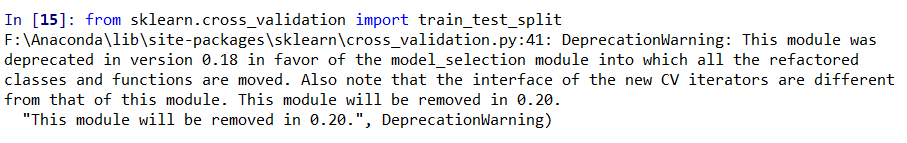
将分类结果值放入目标向量Y中



Step7：

设置决策树

我们将从原始数据中分离出训练和测试数据。为此，我们需要借助sklearn.cross\_validation的train\_test\_split方法



Step8：

train\_test\_split 会返回四个参数，我们将它们命名为:

X\_trainset, X\_testset, y\_trainset, y\_testset

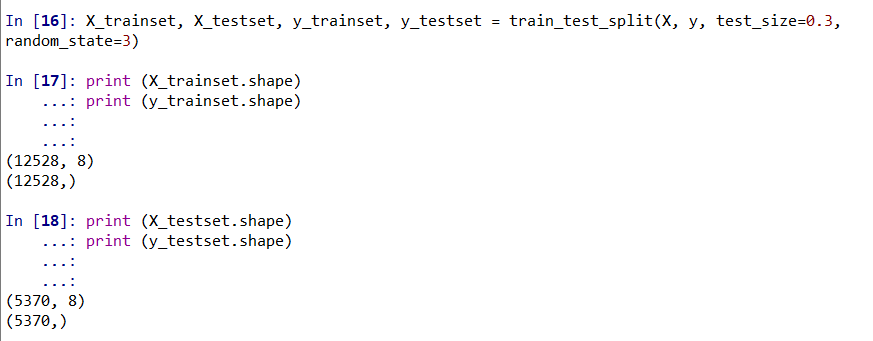
train\_test\_split 需要如下参数:

X, y, test\_size=0.3, and random\_state=3.

X 和 y表示分离之前的数据集，test\_size表示测试数据的占比，random\_state保证我们每次调用得到相同的分离结果

输出X\_trainset和 y\_trainset的结构，确保维度相匹配

输出X\_testset 和 y\_testset的结构，确保维度相匹配

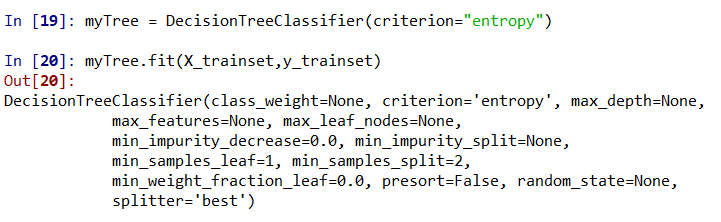


Step9：

我们将先创建DecisionTreeClassifier（决策树分类器）的实例，赋值给myTree

我们设置 criterion="entropy" ，这样能够得到每个节点的信息

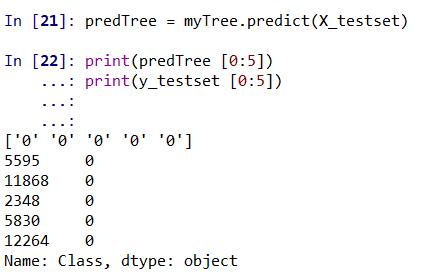
现在我们用训练数据集 X\_trainset 和 y\_trainset 来进行训练



Step10：

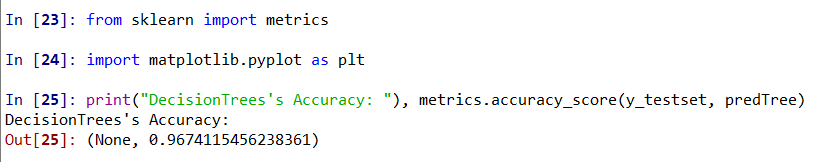
训练完成的模型针对测试数据集X\_testset进行预测

如果想要对预测值和实际值做比较，那么可以打印出 predTree 和 y\_testset



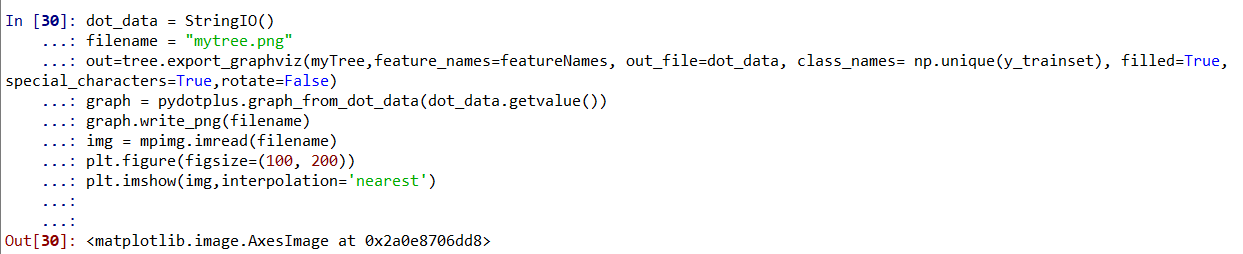
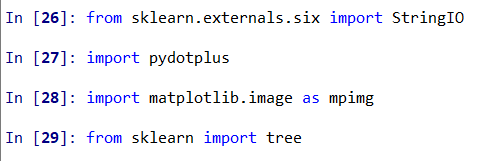
Step11：

接下来，从sklearn中导入metrics，检查我们的模型的准确性

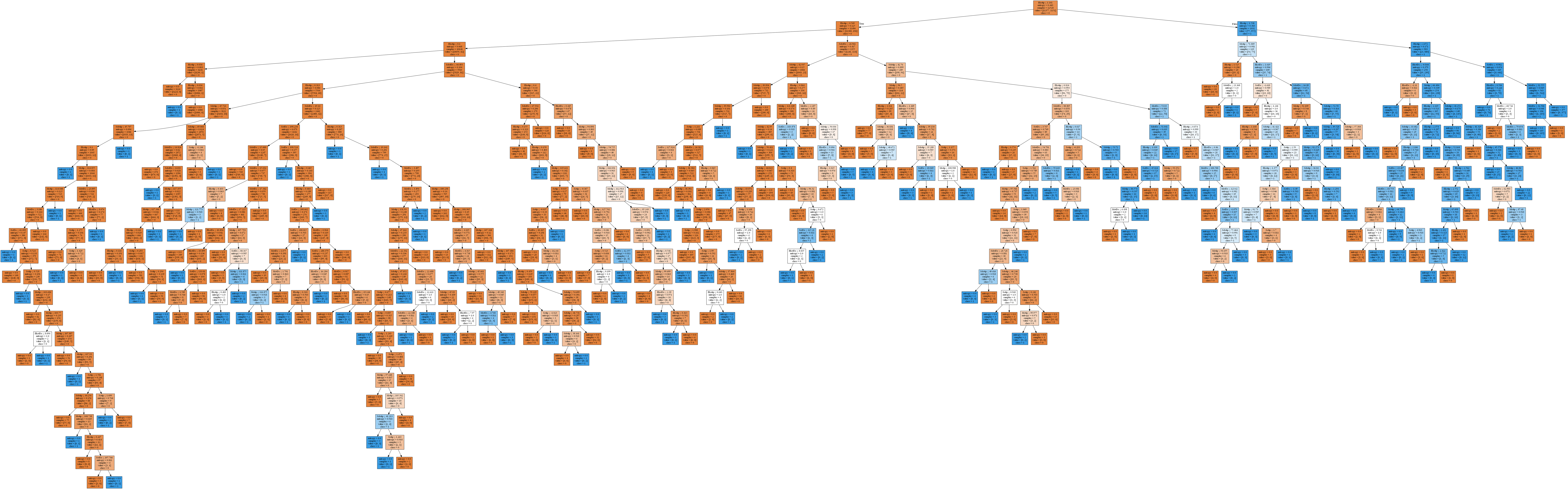


Step12：

最后，我们可以可视化生成的决策树



可视化决策树(mytree.png)



六、模型评估

1，模型评估方案

选取测试集中的数据为样本代入该决策树分类模型进行分类验证，并对分类的结果进行计算处理，使用以下标准对该模型进行评估。

1) 模型分类评估矩阵

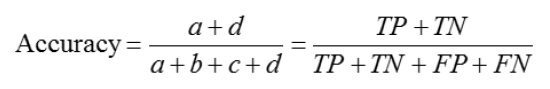
a) 真正类：True Positive, TP，被模型预测为正类的正样本；

b) 伪正类：False Positive, FP，被模型预测为正类的负样本；

c) 伪负类：False Negative, FN，被模型预测为负类的正样本；

d) 真负类：True Negative, TN，被模型预测为负类的负样本。

2) 准确率：Accuracy，衡量模型预测的总正确率。



3) 查全率（真正率）：RecallRate，TruePositiveRate，衡量实际正类 能够被正确判断为正类的比率。



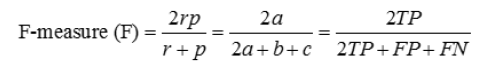
4) 伪正率：False Negative Rate，衡量模型判断为负类的实例中实为正 类的比率。

fp rate = FN/FN+TN

5) 精确度：Precision，衡量模型判断为正类的实例中实为正类的比率。



6) F量度：F-measure，衡量模型分类的整体正确率，结果偏向于真正 类、伪正类以及伪负类。



2. 主要函数和数据结构

（1）混淆矩阵

metrics.confusion\_matrix(y\_testset, predTree, labels=None, sample\_weight=None)

（2）准确率计算函数

metrics.accuracy\_score(y\_testset, predTree)

3. 模型评估结果

模型评估结果如下表，由表中数据可知，该决策树分类模型对测试集数据的准确率为96.7%、查全率为80.9%、伪正率为1.9%、精确度为82.3%、F量度为81.6%。

|  |  |  |  |
| --- | --- | --- | --- |
| 测试集数据评估结果 | | 模型预测类型 | |
| 负类 | 正类 |
| 实际类型 | 负类 | 4797 | 85 |
| 正类 | 93 | 395 |

TP = 395

FP = 85

FN = 93

TN = 4797

|  |  |
| --- | --- |
| （百分比%） | 测试集数据 |
| 准确率 | 96.7% |
| 查全率 | 80.9% |
| 伪正率 | 1.9% |
| 精确度 | 82.3% |
| F量度 | 81.6% |

七、源代码以及生成的决策树图像

根据以上12个步骤，整理得到分类算法的代码（中间变量打印语句被注释掉）

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jun 12 22:54:58 2018

@author: Administrator

"""

import numpy as np

import pandas

from sklearn.tree import DecisionTreeClassifier

my\_data = pandas.read\_csv("D:\HTRU\_2.csv", delimiter=",")

my\_data["Class"] = my\_data["Class"].astype("str")

#my\_data

featureNames = list(my\_data.columns.values)[0:8]

#featureNames

X = my\_data.drop(my\_data.columns[[8]], axis=1).values

#X[0:5]

targetNames = my\_data["Class"].unique().tolist()

#targetNames

y = my\_data["Class"]

#y[0:5]

from sklearn.cross\_validation import train\_test\_split

X\_trainset, X\_testset, y\_trainset, y\_testset = train\_test\_split(X, y, test\_size=0.3, random\_state=3)

#print (X\_trainset.shape)

#print (y\_trainset.shape)

#print (X\_testset.shape)

#print (y\_testset.shape)

myTree = DecisionTreeClassifier(criterion="entropy")

myTree.fit(X\_trainset,y\_trainset)

predTree = myTree.predict(X\_testset)

#print(predTree [0:5])

#print(y\_testset [0:5])

from sklearn import metrics

import matplotlib.pyplot as plt

print("DecisionTrees's Accuracy: "), metrics.accuracy\_score(y\_testset, predTree)

from sklearn.externals.six import StringIO

import pydotplus

import matplotlib.image as mpimg

from sklearn import tree

'''%matplotlib inline'''

dot\_data = StringIO()

filename = "mytree.png"

out=tree.export\_graphviz(myTree,feature\_names=featureNames, out\_file=dot\_data, class\_names= np.unique(y\_trainset), filled=True, special\_characters=True,rotate=False)

graph = pydotplus.graph\_from\_dot\_data(dot\_data.getvalue())

graph.write\_png(filename)

img = mpimg.imread(filename)

plt.figure(figsize=(100, 200))

plt.imshow(img,interpolation='nearest')

可视化决策树(mytree.png)

