机器学习决策树实验报告

汤深文 1120212493

1. 实验思路：

本次实验使用python进行编写，主要采用了python中的sklearn库进行决策树的构建，采用了python中的pandas库对tsv文件数据进行读入和预处理，编写成为最后的py脚本，第一问中传入两个参数，第二问中传入六个参数。

1. 具体实现：

（1）第一问inspection.py：

使用pandas库中的pd.read\_csv进行tsv文件的读入，然后第一问的关键就在于计算两个目标输出值在训练集中的数量。首先采用.shape[0]直接得到训练集一共有多少行，然后取第一行作为一个输出值，遍历tsv文件中的最后一列，得到文件中跟第一行输出值一样的数量，然后再用总行数减去第一个输出值的数量，得到第二个输出值的数量。

然后通过公式计算训练集上的熵值和采用多数投票的错误率，将结果输出到相应的txt文件中。如果参数不够，将会输出"请输入: python inspection.py input\_file output\_file"，如果成功写入，将会输出"成功写入!"

1. 第二问decisionTree.py

同样先使用pandas库中的pd.read\_csv对tsv文件进行读入，然后第二问中为了使用sklearn进行决策树的快速生成，我们需要对于tsv中字符串类的数据进行预处理，将特征值和目标值的字符串用数字进行替换。在我的具体实现中，我采用了暴力比较得到两个不一样的值，作为特征值的两种输入和目标值的两种输出，将他们分别用0，1进行代替。最后输出到txt文件和label文件的时候，再从0和1转换成原有形式进行写入文件。

然后替换后，将数据分类成特征列和目标列，前n-1列都作为特征列，即data.iloc[:, :-1]；最后一列作为目标列，即data.iloc[:, -1] 。然后再调用sklearn库中的函数，根据题目要求，传入相对应的参数，最后生成决策树：tree.DecisionTreeClassifier(criterion="entropy",max\_depth=max\_depth1)。

其中采用的方法是entropy，即通过熵值进行决策树生成，然后max\_depth根据传入的参数max\_depth1即可生成题目要求的决策树。

对于决策树的可视化问题，采用：

feature\_names = X.columns

tree.export\_text(clf,feature\_names=feature\_names)

print(text\_representation)

的方法，保留原有的特征列名和输出列名，不过对于输出的值，在决策树可视化中仍然还是0和1的样子，因此我又在输出决策树下方输出了特征值0和1分别代表的意思。

然后将决策树预测结果写入两个label文件中，然后通过：

predictions = clf.predict(Xtest)  
 accuracy = accuracy\_score(Ytest, predictions)  
 error\_store = 1 - accuracy

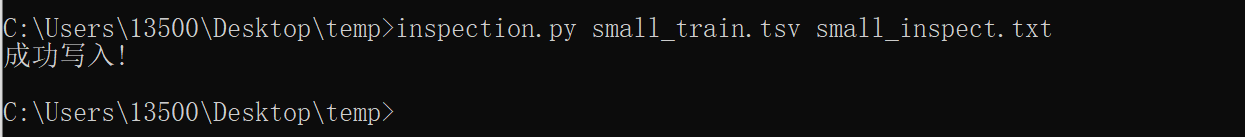
获得决策树的预测错误率，分别得到其在训练集和测试集的错误率，然后写入相对应的文件中。最后，如果成功写入，将会输出："成功写入!"，如果参数数量不对，将会输出："请输入正确的参数的形式"

1. 实验结果以及分析：

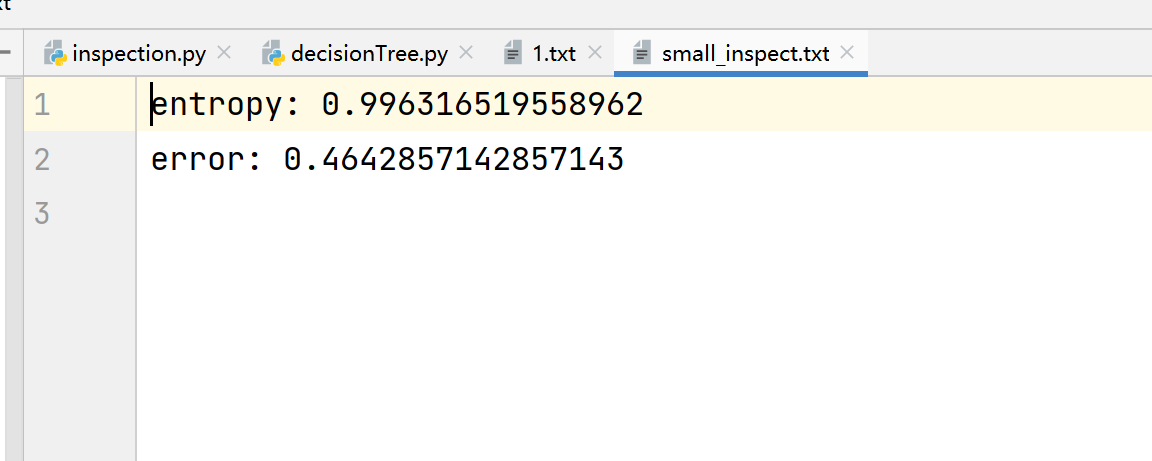
首先通过命令行进入到py文件所在的文件夹，在此处展示中为：C:\Users\13500\Desktop\temp，然后进行脚本命令运行py文件

第一问：

* 1. 对于inspection.py文件的脚本写入格式：



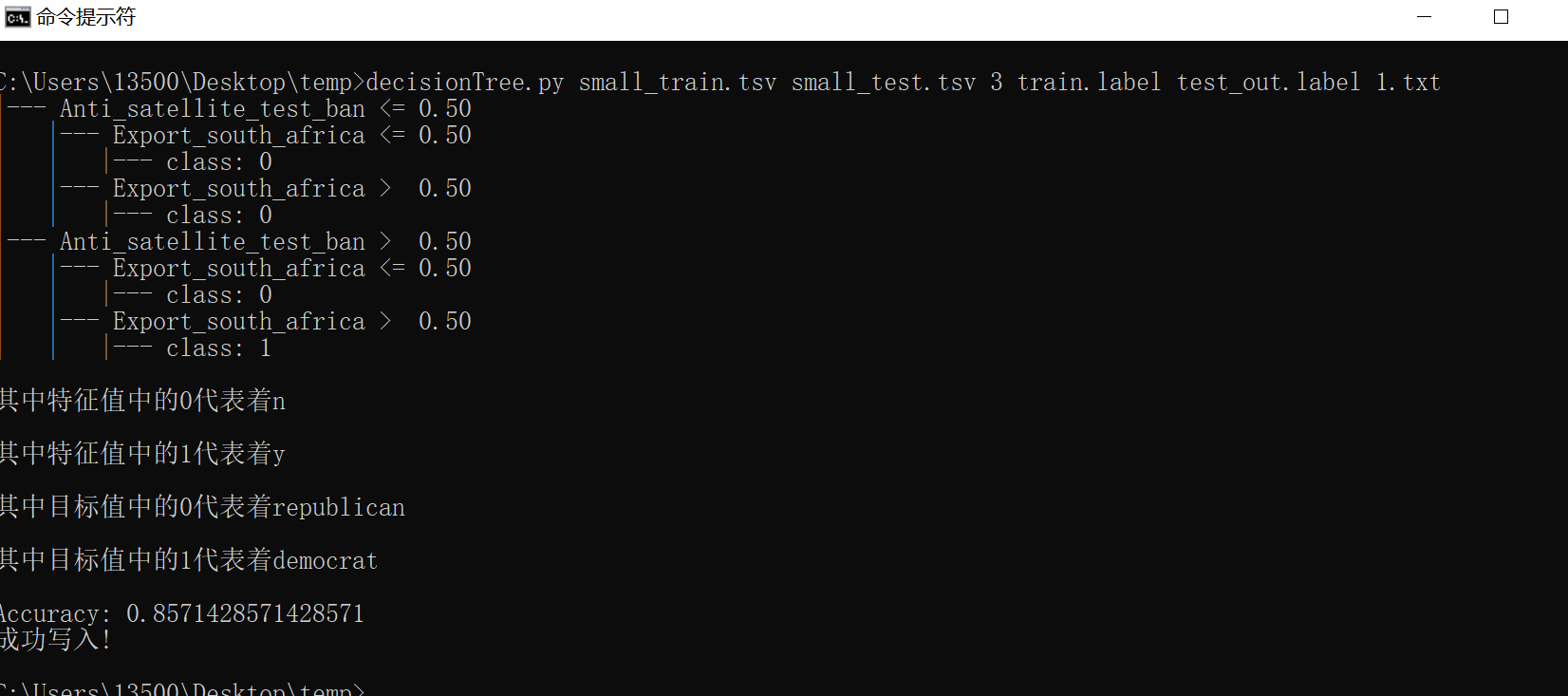
* 1. 运行结果：



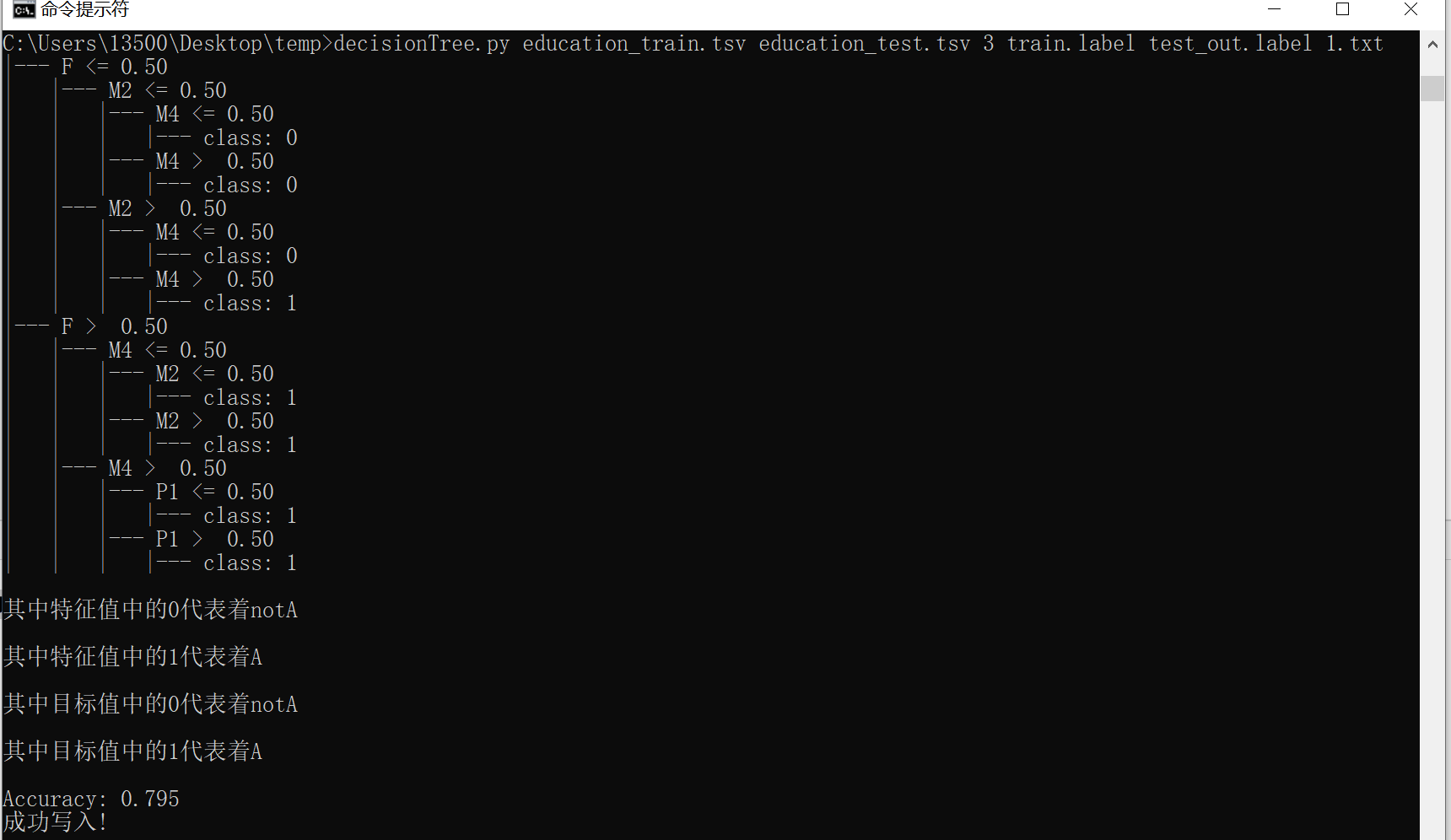
分析：结果与题目所给出的示例结果一样，满足题目要求

第二问：

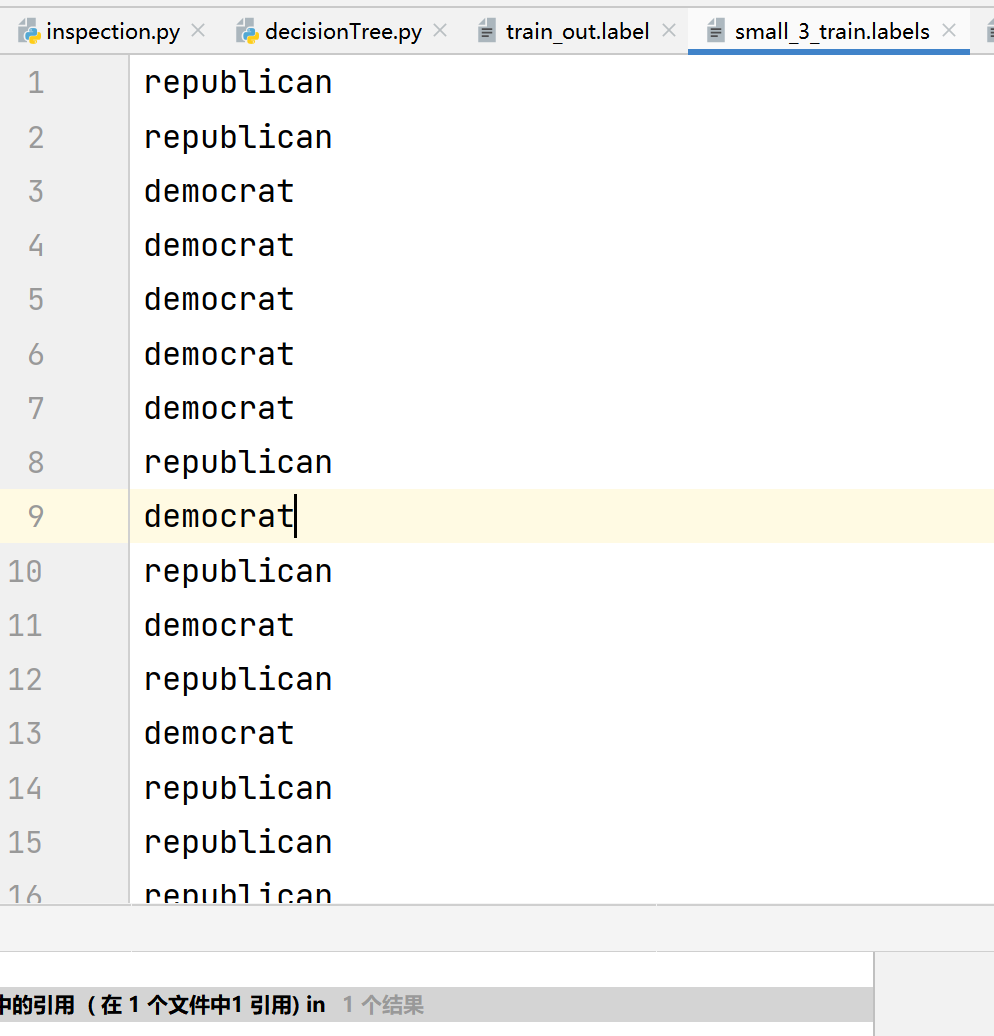
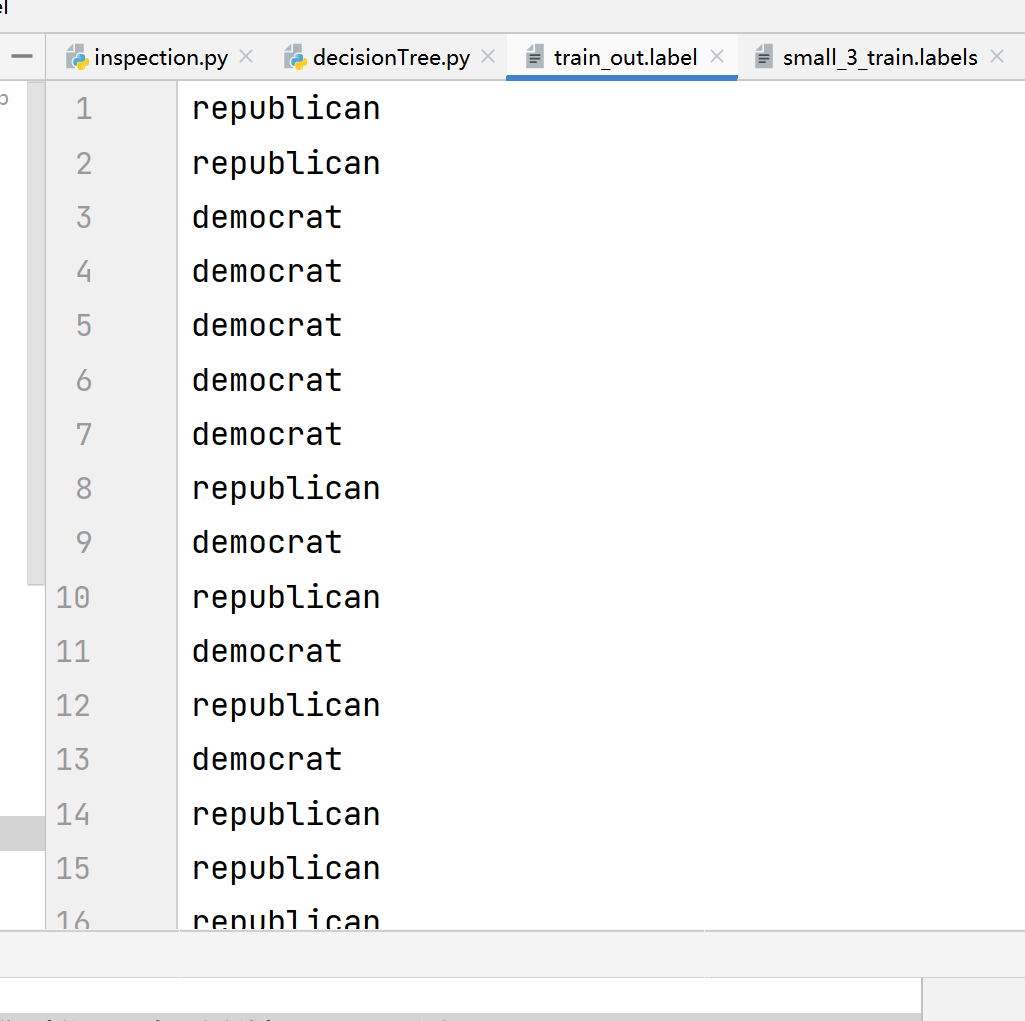
1. 对于decisionTree.py文件的脚本写入格式：



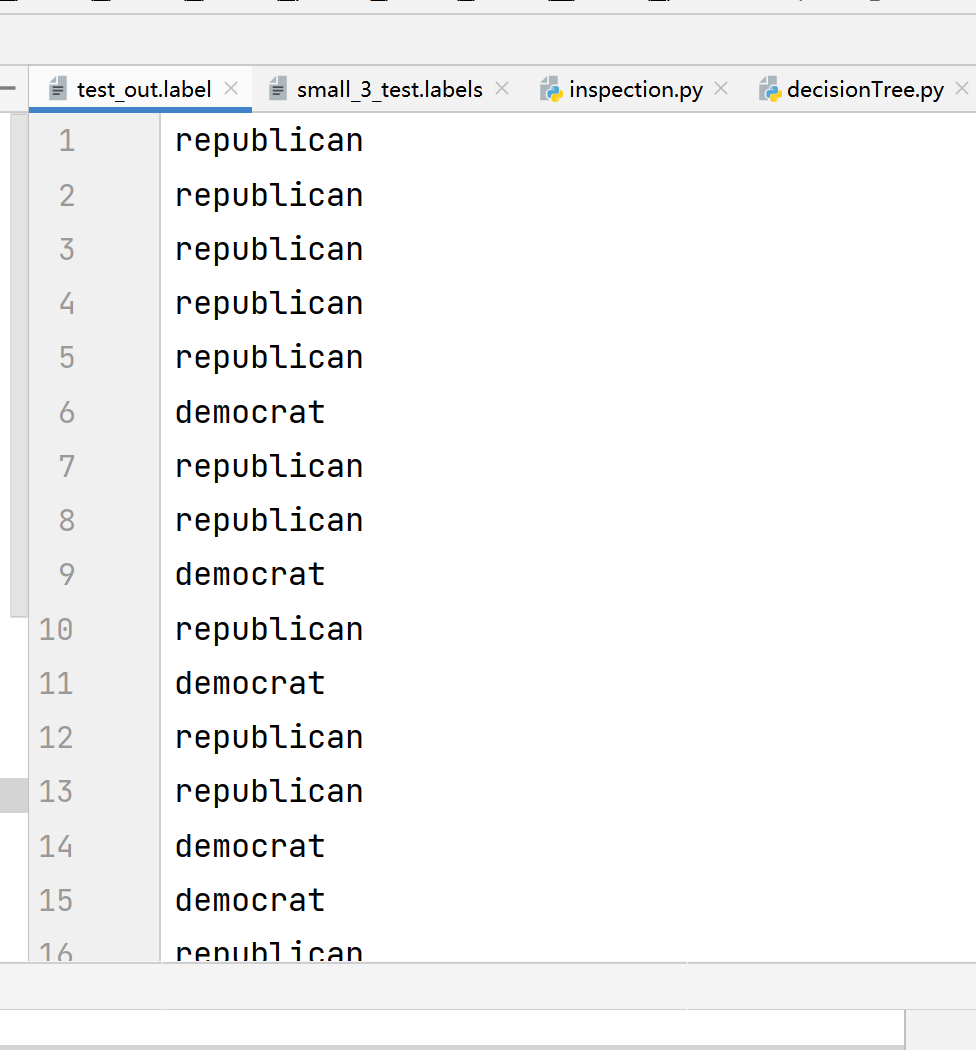
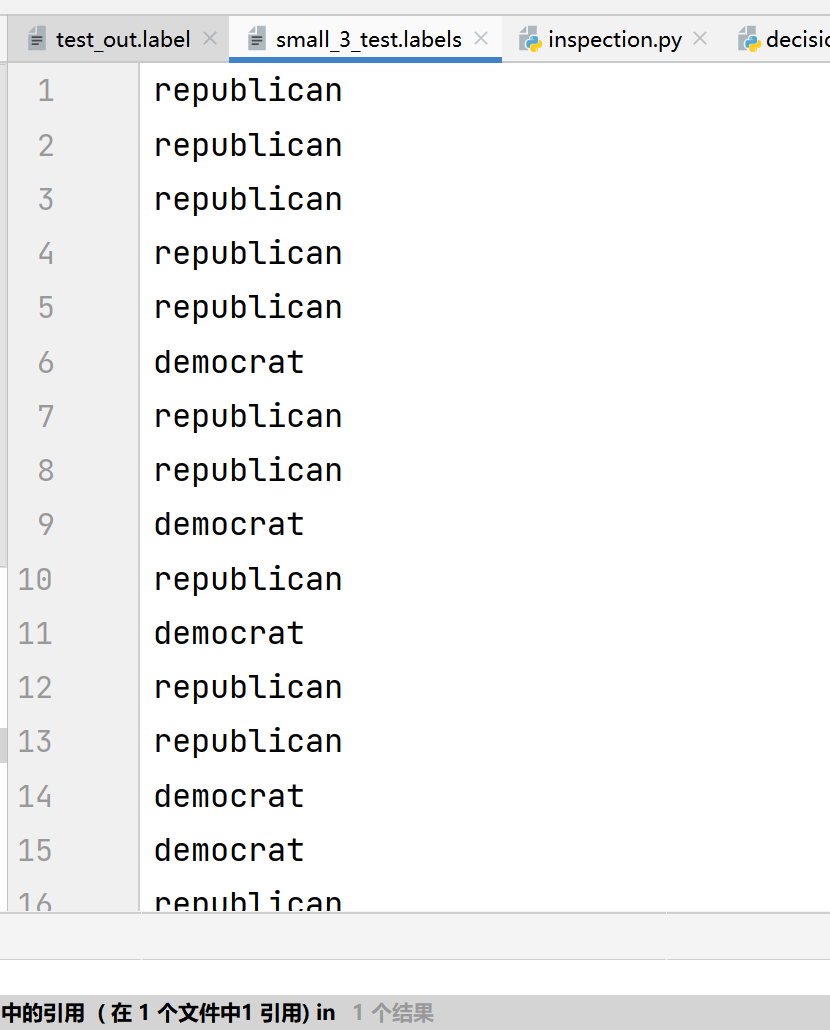
其中输出了决策树的样子，对于其中的0和1难以直接替代，因此将其代表的意思输出在下文中，不同的数据集同样适用。例如对于education的：



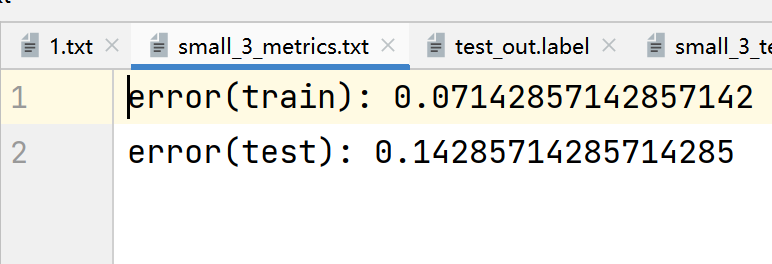
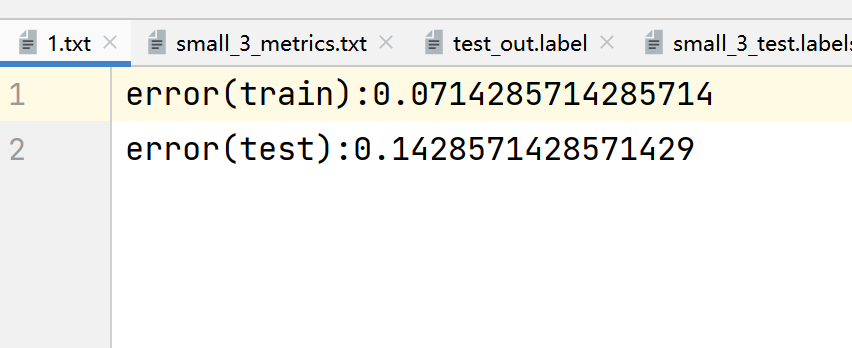
1. 结果分析：



对于train——label的预测结果跟题目所给出来的示例结果一样。



对于test\_label的预测结果也跟题目所给出来的示例结果一样。



正确率也跟题目所给示例结果一样（只是位数更多）

综上，结果符合题目要求。