**机器学习项目报告**

**决策树分类器的设计与实现**

---

**1. 任务概述**

本次作业要求实现两个程序：

1. inspection.py：计算数据集的标签熵和多数投票分类的错误率。

2. decisionTree.py：构建决策树分类器，支持最大深度控制，并在多个数据集上进行测试。

**2. 设计思路**

2.1 数据预处理

输入格式：所有数据集为 `.tsv` 文件，首行为属性名，最后一列为标签。

处理逻辑：

- 使用 `load\_data` 函数读取文件，分离特征和标签。

- 所有特征已二值化（取值为 `'y'` 或 `'n'`），无需额外处理。

2.2 决策树核心算法

基于 ID3算法（信息增益最大化）实现，关键设计如下：

1. 节点分裂标准：

- 选择信息增益（互信息）最大的属性进行分裂。

- 信息增益计算：

IG(Y, X) = H(Y) - H(Y|X)

2.终止条件：

- 当前节点所有标签相同（纯度100%）。

- 达到预设的 `max\_depth`。

- 无有效属性可分裂（所有属性取值相同）。

3. \*\*预测规则\*\*：

- 在叶节点采用多数投票法，平局时选择字典序靠后的标签。

2.3 输出要求

- 预测结果：生成训练集和测试集的 `.labels` 文件。

- 评估指标：计算并输出训练集和测试集的错误率。

- 树结构打印：按深度优先顺序打印节点信息，包含属性、分裂值和标签统计。

**3. 实现细节**

3.1 关键数据结构（决策树节点）

```python

class TreeNode:

def \_\_init\_\_(self, attribute=None, value=None, left=None, right=None, label\_counts=None):

self.attribute = attribute # 分裂属性名

self.value = value # 分裂值（'y'或'n'）

self.left = left # 左子树

self.right = right # 右子树

self.label\_counts = label\_counts # 标签统计（Counter对象）

```

3.2 核心函数

1. calculate\_entropy(labels)：

- 计算标签分布的熵（以2为底）。

2. calculate\_mutual\_information(features, labels, attribute\_index)：

- 计算某属性的信息增益。

3. build\_tree()：

- 递归构建决策树，动态跳过无效分裂属性。

4. predict()：

- 根据树结构对单个样本分类。

5. print\_tree()：

- 格式化打印树结构，确保显示所有标签（包括计数为0的）。

3.3 处理边界情况

- 单一取值属性：跳过无法分裂的属性。

- 空数据集：直接返回叶节点。

- 平局投票：按字典序选择标签。

**4. 代码优化点**

1. 效率优化：

- 使用 `Counter` 快速统计标签频次。

- 预计算所有属性的信息增益，避免重复计算。

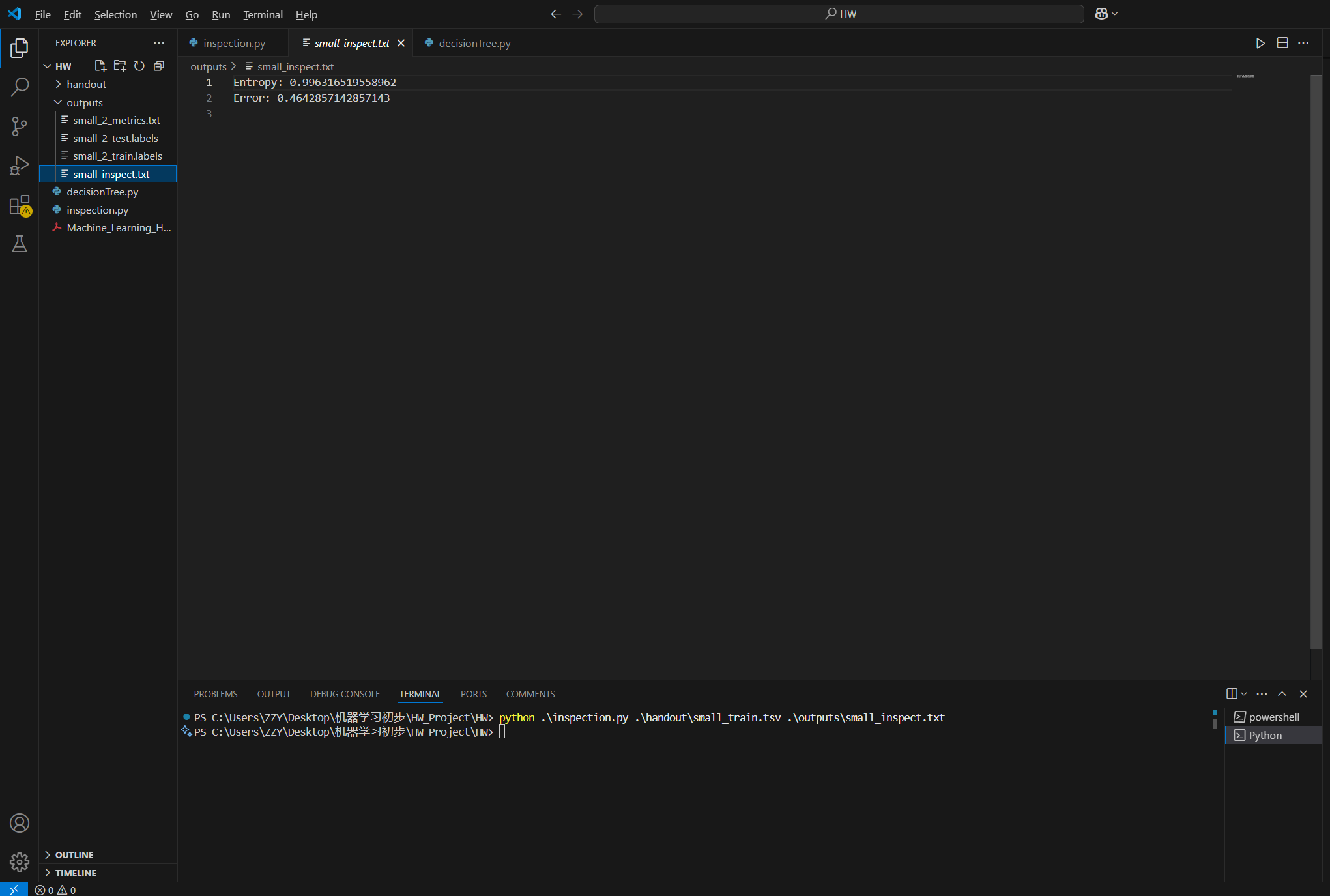
2. Bobustness：

- 检查特征取值的多样性，防止 `IndexError`。

- 统一处理标签格式（如 `[X democrat/Y republican]`）。

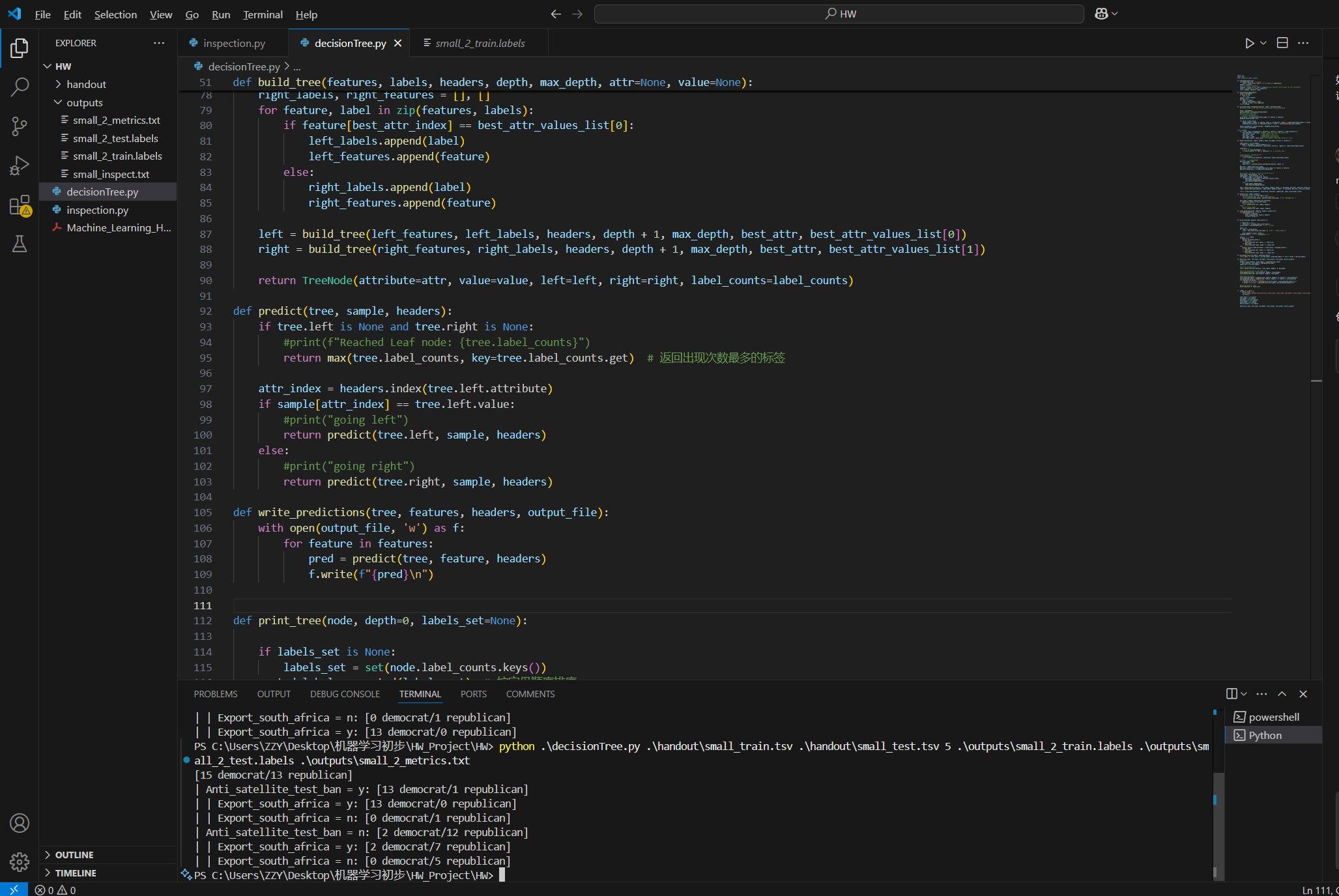
**5. 实验结果**

1、在small测试集上运行inspection程序

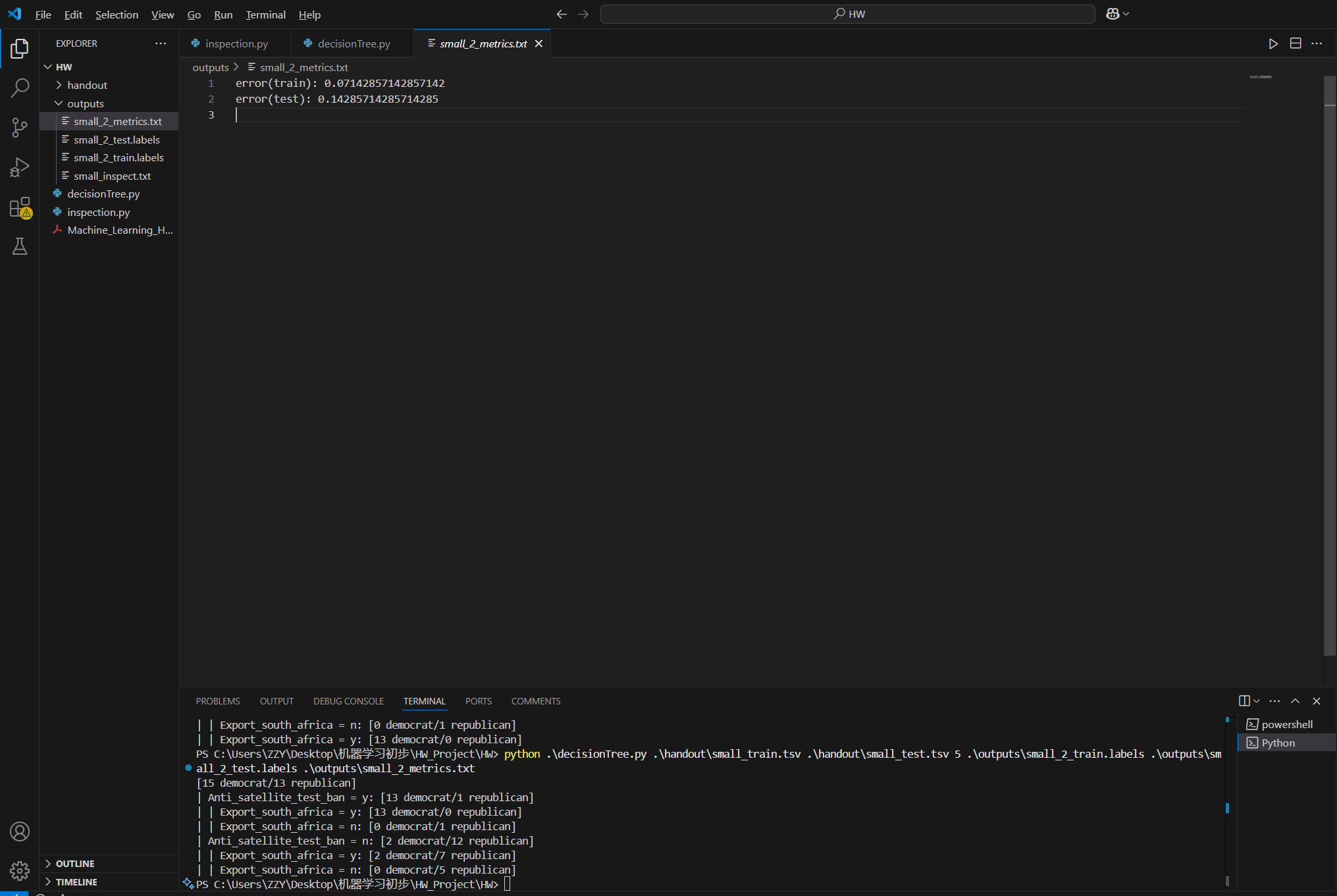
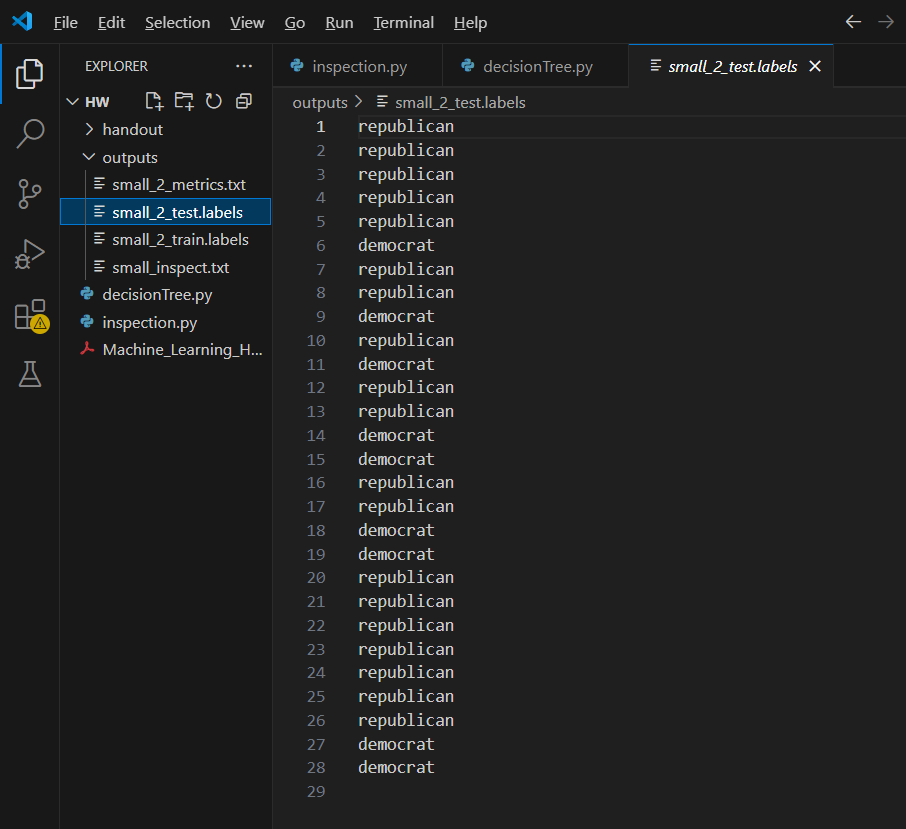


得到预期输出结果

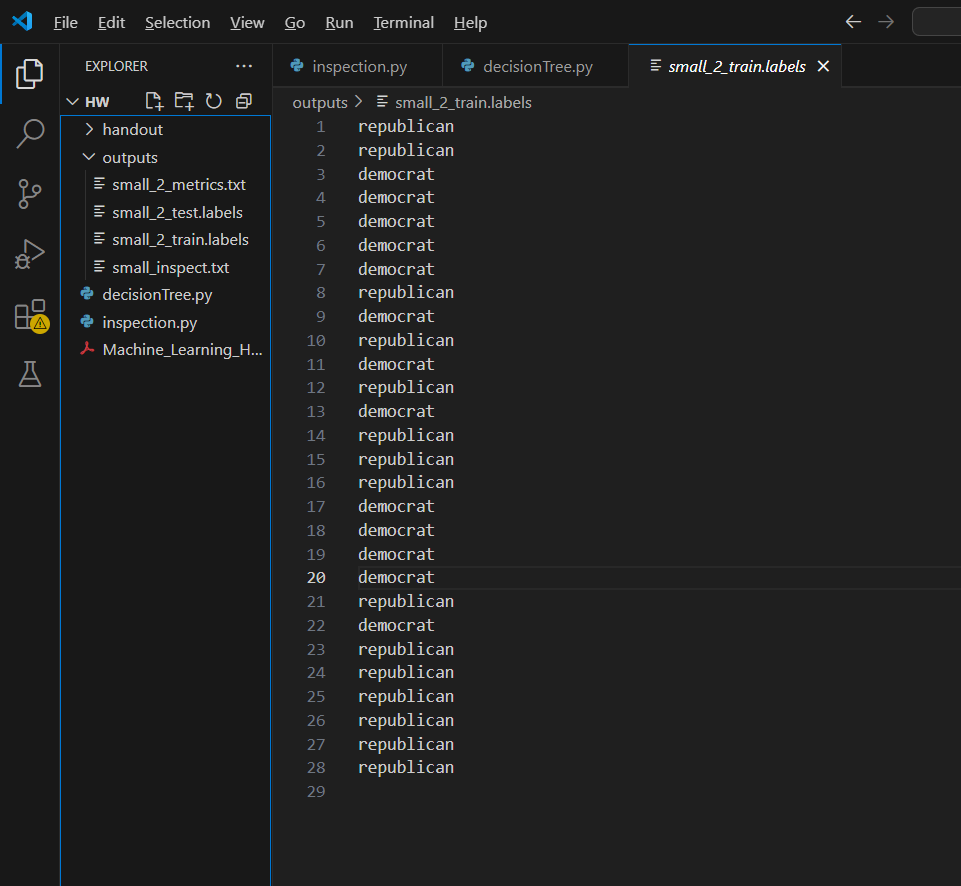
2、在small测试集上运行决策树程序



可以看到程序产生预期输出，print出了训练中构成的决策树结构

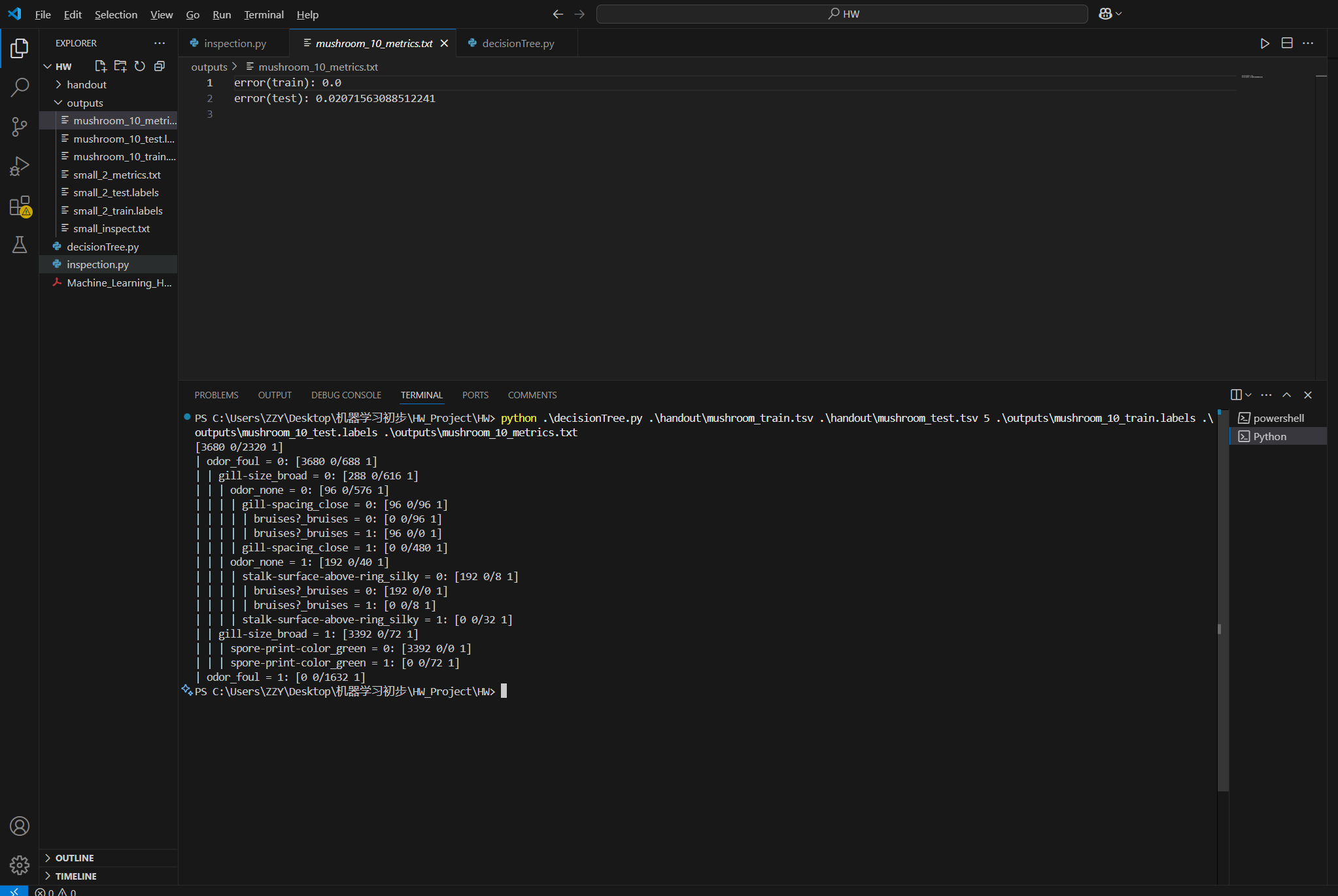


使用决策树预测训练集和验证集的输出结果

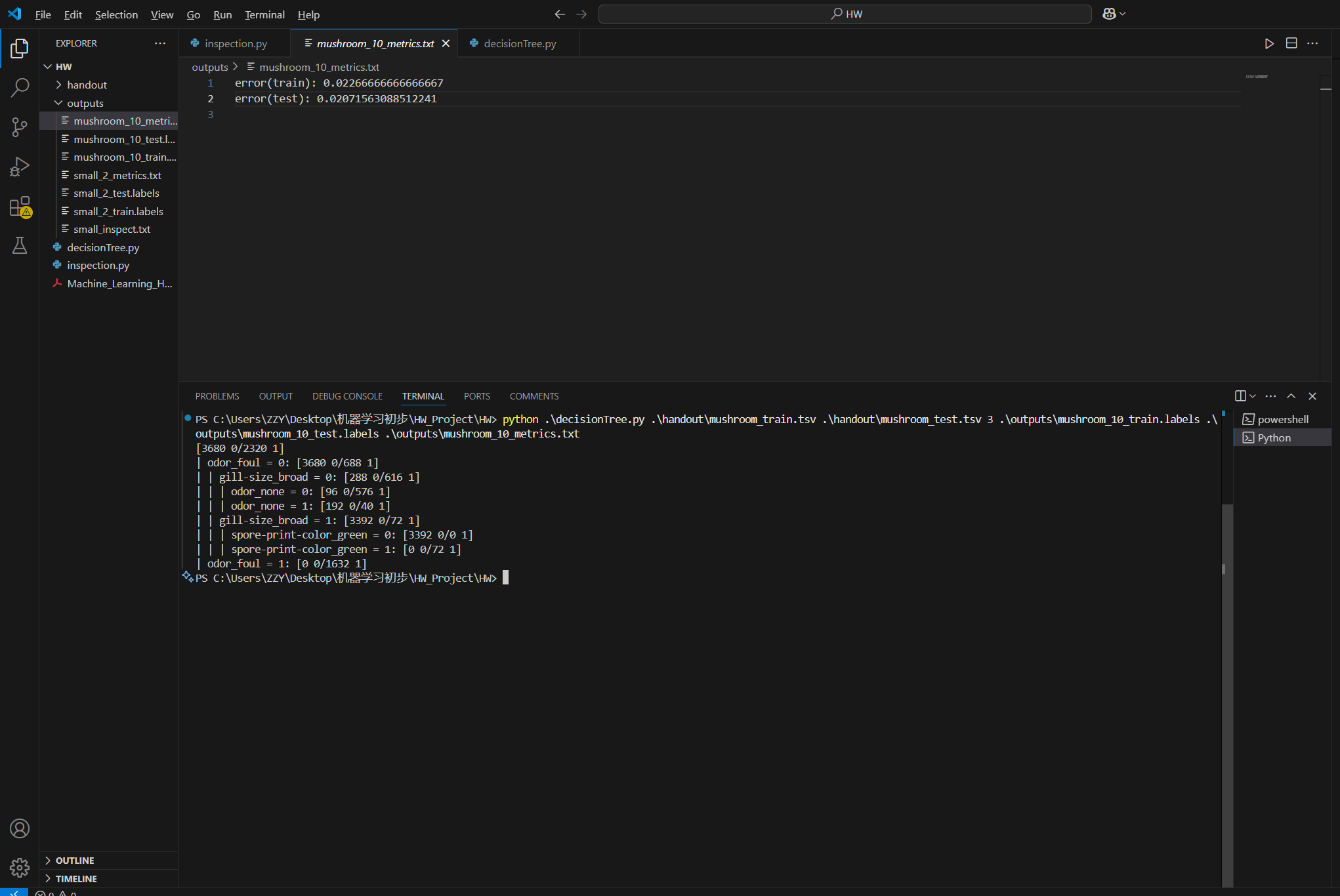


输出metrics文件中的训练和验证误差

3、在其他数据集上运行



在mushroom数据集上设置maxdepth为5的预测结果



在mushroom数据集上设置maxdepth为3的预测结果