实验目标

以树为模型的分类器

编程语言：$Python3$

使用库：$Numpy$

分类任务：对个体的收入进行判断 $>50K$ or $\leq 50K$

数据集：$Adult(UCIMachineLearningRepository:AdultDataSet)$

算法设计

step1: 数据预处理

使用 `np.genfromtxt` 函数读入训练数据，缺失项用众数填补。

分出X, y

对y进行二值编码 `y = y == y[0]`

对X的非数值列进行数值编码。具体来说，使用 `np.unique` 函数去重后利用 `enumerate` 创建字典映射。

测试数据同样按照上述方式操作，编码使用到的映射与训练数据一致。

step2: 构造决策树

1. 找到最佳的划分标准进行划分
2. 把划分数据交给左右子树
3. 递归建树
4. 终止条件：y只剩下一类，或者达到最大递归深度

伪代码如下

```
def generate(X: numpy.ndarray, y: numpy.ndarray, depth: int) → Node:
    # 递归终止条件
    if |y| == 1: # y只有一类，直接预测为y
        node = Node(predict=y[0])
        return node

    if depth == max_depth: # 到达预设的最大递归深度，取y中更多的那一类作为预测结果
        node = Node(predict=np.sort(y)[y.shape[0] >> 1])
        return node
```

```
    feature, threshold = split_data(X, y) #找到划分结果最好的特征及其阈值
    node = Node(feature=feature, threshold=threshold) #创建结点

    left_X, left_y = X[i], y[i] where X[i, fearture] <= threshold #左子树的数据
    right_X, right_y = X[i], y[i] where X[i, fearture] > threshold #右子树的数据

    # 递归建树
    left_node = generate(left_X, left_y, depth + 1)
    left_node = generate(left_X, left_y, depth + 1)

    node.left = left_node
    node.right = letf_node

    return node
```

Step3: 最佳划分

本文采用 信息熵 作为划分标准，其计算公式为 $Ent(D) = -\sum_{k=1}^{|y|} p_k log_2 p_k.$

初始值：$best\_feature = None, best\_threshold = None, min\_entropy = +\infty$

1. 对于X中的每一个$feature$,求得该特征下的去重集合D.
2. 以D中的任意一个元素v作为阈值划分X,
   求得左右子树的加权熵$Ent(feature, v) = p_L * Ent(y[L]) + p_R * Ent(y[L])$,
   式中，$p_c = size(c)/size(X)$表示左右子树所得数据集的占比
3. 若$Ent(feature, v) < min_entropy$，更新$best_feature, best_threshold, min_entropy$

伪代码如下

```
def split_data(X, y):
    best_feature_index = None
    best_threshold = None
    best_entropy = inf

    for feature in X.features:
        D = set(X[feature])
        for v in D:
            left = i where X[i][feature] <= v
            right = i where X[i][feature] > v

            entropy = left  / y.shape[0] * entropy(y[left]) + right /
y.shape[0] * entropy(y[right])
```

```
                if entropy < best_entropy:
                    best_entropy = entropy
                    best_feature = feature
                    best_threshold = v

        return best_feature, best_threshold
```

step4: 决策树的预测

对输入X按照结点判断，如果 $X[feature] <= threshold$ 进入左子树，否则进入右子树。

搜索到叶子结点时，叶子结点的值即为输出。

step5: 决策森林

同时训练n棵决策树，每棵决策树的训练样本都是从总样本中随机抽取30%，训练完成后各自评估，按照准确率给予对应的权重。

预测时，每颗树给出自己的判断，加权求和后票数更多的一类即为输出。

step6: 评估指标

准确率($accuracy$)：分类正确的样本数占总样本数的比例

精度($precision$)：预测为正例的样本中实际为正例的比例

召回率($recall$)：预测为正例的样本占实际为正例的比例

$F1$值：$precision$与$recall$的调和平均数

实验结果

最大树高为10的决策树
准确率: 0.8546772311283091
精确率: 0.8711936887119369
召回率: 0.9502211499798955
$F1$ 值: 0.908992999461497

10棵树，最大树高为10的决策森林
准确率: 0.8431914501566243
精确率: 0.8380541871921182
召回率: 0.9850422195416164
$F1$ 值: 0.905622712653876