《模式识别与机器学习》

实验：决策树分类

院　　系: 自动化系

班　　级: 自 31 班

学生姓名: 苟左

学　　号: 2023012274

1 实验目的

1. 理解决策树的基本原理及其变种算法（ID3、C4.5、CART）的特点与差异。
2. 掌握信息熵、信息增益、信息增益率、基尼指数等特征选择准则的计算方法。
3. 实现预剪枝和后剪枝算法，理解剪枝在决策树中的作用与意义。
4. 在不同规模数据集上比较决策树算法的性能，分析过拟合现象及其缓解方法。

2 实验任务与方法

2.1 数据集

本实验使用两个数据集：

1. **Melon 数据集**：迷你西瓜数据集，包含 17 个训练样本和 7 个测试样本，4 个特征（纹理、根蒂、色泽、脐部），2 个类别（好瓜/坏瓜）。该数据集用于算法验证和可视化。
2. **CelebA 子集**：人脸属性数据集，包含 250 个训练样本和 50 个测试样本，40 个二值属性特征（如性别、头发颜色、五官特征等），10 个类别（不同人物 ID）。该数据集用于评估算法在实际问题上的性能。

2.2 决策树算法

2.2.1 ID3 算法

ID3（Iterative Dichotomiser 3）使用**信息增益**作为特征选择准则：

$$\mathrm{Gain}(D,a)=\mathrm{Ent}(D)-\sum_{v=1}^{V}\frac{|D^v|}{|D|}\mathrm{Ent}(D^v) \tag{1}$$

其中信息熵定义为：

$$\mathrm{Ent}(D)=-\sum_{k=1}^{|\mathcal{Y}|}p_k\log_2 p_k \tag{2}$$

ID3 选择信息增益最大的特征进行划分。

2.2.2 C4.5 算法

C4.5 算法使用**信息增益率**作为特征选择准则，解决了 ID3 偏好取值多的特征的问题：

$$\text{Gain\_ratio}(D,a) = \frac{\text{Gain}(D,a)}{\text{IV}(a)} \tag{3}$$

其中固有值（Intrinsic Value）定义为：

$$\text{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|} \tag{4}$$

2.2.3 CART 算法

CART（Classification And Regression Tree）使用**基尼指数**作为特征选择准则：

$$\text{Gini}(D) = 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2 \tag{5}$$

$$\text{Gini\_index}(D,a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \text{Gini}(D^v) \tag{6}$$

CART 选择基尼指数最小的特征进行划分。

2.3 剪枝算法

2.3.1 预剪枝（Pre-pruning）

预剪枝在决策树构建过程中，每次划分前先评估划分是否能提升泛化性能：

1. 计算划分前在验证集上的准确率 $\text{Acc}_{\text{before}}$
2. 计算划分后在验证集上的准确率 $\text{Acc}_{\text{after}}$
3. 若 $\text{Acc}_{\text{after}} \leq \text{Acc}_{\text{before}}$，则禁止划分，将当前节点标记为叶节点

优点：降低过拟合风险，减少训练和测试时间。

缺点：可能欠拟合，因为某些划分虽然当前无法提升性能，但后续划分可能有益。

2.3.2 后剪枝（Post-pruning）

后剪枝在决策树完全生长后，自底向上地考察每个非叶节点：

1. 计算子树在验证集上的准确率 Acc_{tree}
2. 将子树替换为叶节点，计算叶节点在验证集上的准确率 Acc_{leaf}
3. 若 $\text{Acc}_{\text{leaf}} > \text{Acc}_{\text{tree}}$，则执行剪枝，用叶节点替换子树

优点：泛化性能通常优于预剪枝，因为基于完整的树结构做决策。

缺点：训练时间较长，需要先生成完整决策树。

3 实验实现与代码片段

3.1 信息熵计算（tree.py）

Listing 1: 信息熵计算

```
def cal_entropy(dataset):
    numEntries = len(dataset)
    labelCounts = {}

    # 统计每个类别的样本数量
    for featVec in dataset:
        currentLabel = featVec[-1]
        if currentLabel not in labelCounts:
            labelCounts[currentLabel] = 0
        labelCounts[currentLabel] += 1

    entropy = 0.0
    # 计算信息熵：Ent(D) = -sum(p_k * log2(p_k))
    for count in labelCounts.values():
        prob = count / numEntries
        entropy += -prob * log(prob, 2)

    return entropy
```

3.2 ID3 特征选择（tree.py）

Listing 2: ID3 算法的特征选择

```
def ID3_chooseBestFeatureToSplit(dataset):
    numFeatures = len(dataset[0]) - 1
    baseEnt = cal_entropy(dataset)
    bestInfoGain = 0.0
    bestFeature = -1

    for i in range(numFeatures):
        featList = [example[i] for example in dataset]
        uniqueVals = set(featList)

        newEnt = 0.0
        for value in uniqueVals:
```

```
            subdataset = splitdataset(dataset, i, value)
            prob = len(subdataset) / len(dataset)
            newEnt += prob * cal_entropy(subdataset)

        infoGain = baseEnt - newEnt

        if infoGain >= bestInfoGain:
            bestInfoGain = infoGain
            bestFeature = i

    return bestFeature
```

3.3 C4.5 特征选择（tree.py）

Listing 3: C4.5 算法的特征选择

```
def C45_chooseBestFeatureToSplit(dataset):
    numFeatures = len(dataset[0]) - 1
    baseEnt = cal_entropy(dataset)
    bestInfoGainRatio = 0.0
    bestFeature = -1

    for i in range(numFeatures):
        featList = [example[i] for example in dataset]
        uniqueVals = set(featList)

        newEnt = 0.0
        IV = 0.0

        for value in uniqueVals:
            subdataset = splitdataset(dataset, i, value)
            prob = len(subdataset) / len(dataset)
            newEnt += prob * cal_entropy(subdataset)
            IV += -prob * log(prob, 2) if prob != 0 else 0

        infoGain = baseEnt - newEnt
        # 避免除零错误
        if IV == 0:
            infoGainRatio = 0.0
        else:
            infoGainRatio = infoGain / IV  # 信息增益率

        if infoGainRatio >= bestInfoGainRatio:
            bestInfoGainRatio = infoGainRatio
            bestFeature = i
```

```
    return bestFeature
```

3.4 CART 特征选择（tree.py）

Listing 4: CART 算法的特征选择

```
def CART_chooseBestFeatureToSplit(dataset):
    numFeatures = len(dataset[0]) - 1
    bestGini = float('inf')
    bestFeature = -1

    for i in range(numFeatures):
        featList = [example[i] for example in dataset]
        uniqueVals = set(featList)
        gini = 0.0

        for value in uniqueVals:
            subdataset = splitdataset(dataset, i, value)
            classCounts = Counter([example[-1] for example in subdataset])

            # 计算子数据集的基尼值
            prob = len(subdataset) / len(dataset)
            sub_gini = 1.0
            for count in classCounts.values():
                sub_prob = count / len(subdataset)
                sub_gini -= sub_prob ** 2

            gini += prob * sub_gini

        if gini <= bestGini:
            bestGini = gini
            bestFeature = i

    return bestFeature
```

3.5 预剪枝实现（tree.py）

Listing 5: 预剪枝关键代码

```
if pre_pruning:
    # 计算划分前在测试集上的分类准确率
    ans = [example[-1] for example in test_dataset]
    result_counter = Counter([example[-1] for example in dataset])
    pre_split_output = result_counter.most_common(1)[0][0]
    pre_split_acc = cal_acc([pre_split_output] * len(test_dataset), ans)
```

```
    # 计算划分后准确率
    outputs = []
    ans = []
    for value in sorted(uniqueVals):
        cut_testset = splitdataset(test_dataset, bestFeat, value)
        cut_dataset = splitdataset(dataset, bestFeat, value)

        for vec in cut_testset:
            ans.append(vec[-1])

        if len(cut_dataset) == 0:
            leaf_output = majorityCnt(classList)
        else:
            leaf_output = majorityCnt([example[-1] for example in cut_dataset])

        outputs += [leaf_output] * len(cut_testset)

    post_split_acc = cal_acc(outputs, ans)

    # 如果划分后准确率没有提升，则禁止划分
    if post_split_acc <= pre_split_acc:
        return pre_split_output
```

3.6 后剪枝实现（tree.py）

Listing 6: 后剪枝关键代码

```
if post_pruning and len(test_dataset) != 0:
    # 计算后剪枝前的准确率（保留子树）
    tree_output = classifytest(DecisionTree, featLabels, test_dataset)
    ans = [example[-1] for example in test_dataset]
    tree_acc = cal_acc(tree_output, ans)

    # 计算后剪枝后的准确率（替换为叶节点）
    result_counter = Counter([example[-1] for example in dataset])
    post_prune_output = result_counter.most_common(1)[0][0]
    post_prune_acc = cal_acc([post_prune_output] * len(test_dataset), ans)

    # 如果剪枝后准确率更高，则执行剪枝
    if post_prune_acc > tree_acc:
        return post_prune_output
```

4 实验结果

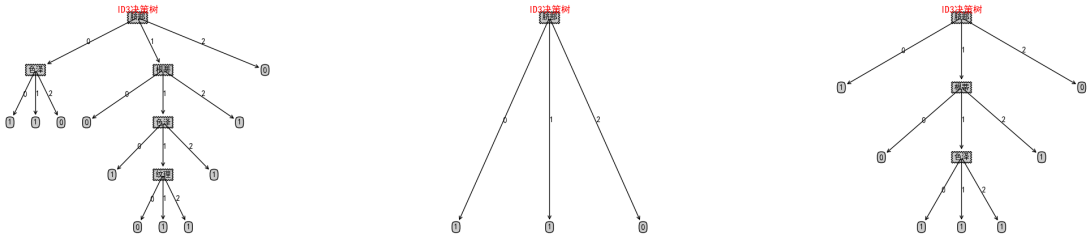
4.1 Melon 数据集实验

Melon 数据集初始信息熵 $\text{Ent}(D) = 1.0$（正负样本平衡）。表 1 展示了 ID3 算法在不同剪枝策略下的性能对比。

表 1: Melon 数据集上 ID3 算法的实验结果

剪枝策略	树深度	测试准确率	正确/总数	树结构复杂度
不剪枝	5	42.86%	3/7	高（过拟合）
预剪枝	1	**71.43%**	5/7	低（最简单）
后剪枝	3	**71.43%**	5/7	中等

结论：不剪枝导致严重过拟合（42.86%），剪枝将准确率提升至 71.43%（提升 67%）。预剪枝树最简单（深度 1），后剪枝保留更多结构（深度 3）但准确率相同。首选特征为**脐部**和**色泽**（信息增益均为 0.275）。图 1 展示了三种策略的决策树结构对比。



(a) 不剪枝（过拟合） (b) 预剪枝（最简） (c) 后剪枝（平衡）

图 1: Melon 数据集 ID3 决策树结构对比

4.2 CelebA 数据集实验

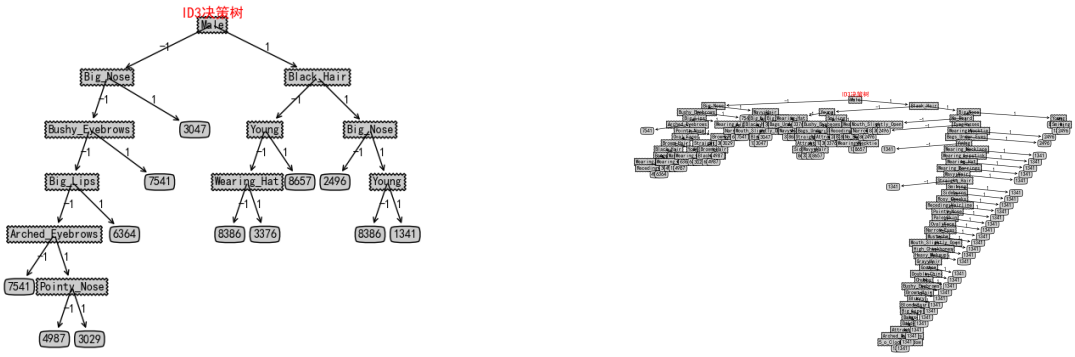
CelebA 子集的初始信息熵为 $\text{Ent}(D) = 3.322$，显著高于 Melon 数据集，反映了 10 分类问题的高复杂度。表 2 总结了在 CelebA 数据集上的实验结果。

表 2: CelebA 数据集上不同算法的实验结果

算法	剪枝策略	测试准确率	正确/总数	首选特征
ID3	预剪枝	76%	38/50	Male（信息增益 0.896）
ID3	后剪枝	**86%**	43/50	Male（信息增益 0.896）
C4.5	预剪枝	68%	34/50	Big_Nose（增益率 0.589）
C4.5	后剪枝	84%	42/50	Big_Nose（增益率 0.589）
CART	预剪枝	76%	38/50	Male（基尼指数 0.838）
CART	后剪枝	84%	42/50	Male（基尼指数 0.838）

4.2.1 ID3 算法在 CelebA 上的表现

ID3 算法首选特征为 **Male**（性别），信息增益高达 **0.896**，远超其他特征。**预剪枝**达到 76% 准确率，**后剪枝**达到 **86%** 准确率（最佳性能）。图 2 展示了 ID3 在不同剪枝策略下的决策树结构。



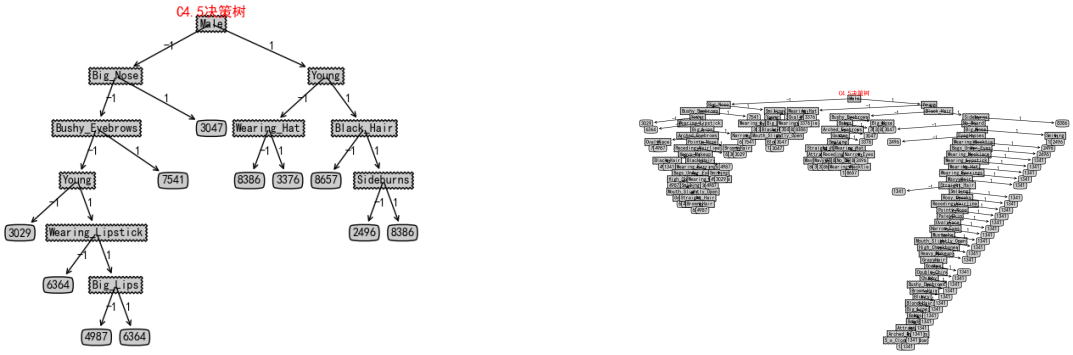
(a) ID3 预剪枝 (b) ID3 后剪枝

图 2: CelebA 数据集上 ID3 决策树结构对比

特征重要性（信息增益前 5）：Male (0.896) > Wearing_Lipstick (0.764) > Big_Nose (0.582) > Heavy_Makeup (0.530) > Arched_Eyebrows (0.503)。

4.2.2 C4.5 算法在 CelebA 上的表现

C4.5 使用信息增益率作为准则，首选特征为 **Big_Nose**（增益率 0.589），而非 ID3 选择的 Male 特征（增益率约 0.9）。**预剪枝**达到 68% 准确率，**后剪枝**达到 84% 准确率。图 3 展示了 C4.5 在不同剪枝策略下的决策树结构。



(a) C4.5 预剪枝 (b) C4.5 后剪枝

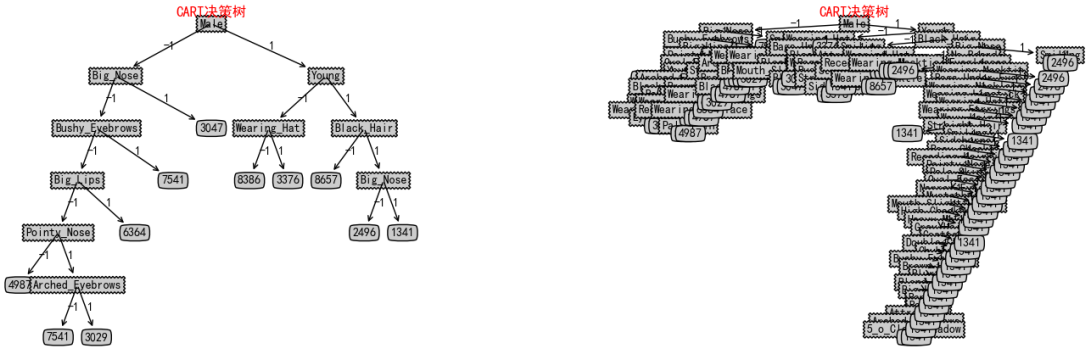
图 3: CelebA 数据集上 C4.5 决策树结构对比

特征重要性（信息增益率前 5）：Big_Nose (0.589) > Arched_Eyebrows (0.581) > Goatee (0.511) > Black_Hair (0.475) > Chubby (0.462)。

C4.5 的 IV 惩罚了取值分布不均衡的特征，使特征选择更平衡，但在该任务中高判别力的不平衡特征（Male）恰恰最有价值，因此性能略逊于 ID3。

4.2.3 CART 算法在 CelebA 上的表现

CART 使用基尼指数作为划分准则，计算效率更高（无需对数运算）。基尼指数公式为 $\text{Gini}(D) = 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2$，与信息熵类似但更简洁。CART 首选特征同样为 **Male**（基尼指数 0.838），与 ID3 一致。**预剪枝**达到 76% 准确率，**后剪枝**达到 84% 准确率。图 4 展示了 CART 在不同剪枝策略下的决策树结构。



(a) CART 预剪枝 (b) CART 后剪枝

图 4: CelebA 数据集上 CART 决策树结构对比

特征重要性（基尼指数前 5）：Male (0.838) > Big_Nose (0.853) > Wearing_Lipstick (0.864) >

Arched_Eyebrows (0.869) > Heavy_Makeup (0.896，数值越小越重要)。

CART 的一个重要特点是生成**二叉树**结构，而 ID3 和 C4.5 生成多叉树。CART 与 ID3 在特征选择上高度一致（均选 Male），性能相当（预剪枝均 76%，后剪枝分别 84% vs 86%），证明了基尼指数与信息增益在实际问题上的等价性。

5 算法对比与分析

5.1 三种算法对比

表 3: ID3、C4.5、CART 算法对比

特性	ID3	C4.5	CART
划分准则	信息增益	信息增益率	基尼指数
计算复杂度	中等	高（需计算 IV）	低（无对数）
特征偏好	偏好多值特征	修正 ID3 偏好	无明显偏好
树结构	多叉树	多叉树	二叉树
CelebA 准确率	**86%**（后剪枝）	84%（后剪枝）	84%（后剪枝）
首选特征	Male	Big_Nose	Male

算法性能分析：

- **ID3 vs CART**：两者在特征选择上高度一致（均选 Male 为首选特征），性能相近（86% vs 84%），这验证了信息增益与基尼指数在分类任务中的**理论等价性**。基尼指数本质上是信息熵的二阶近似，两者都衡量数据集的不纯度，因此在实际应用中往往得到相似的划分结果。
- **C4.5 的特殊性**：C4.5 通过固有值（IV）惩罚取值数目多的特征，选择了 Big_Nose 而非 Male。虽然 Male 的信息增益最高（0.896），但其取值分布不均衡导致 IV 较大，使信息增益率降低。这种修正在某些场景下能避免过拟合，但在本数据集上，高判别力的不平衡特征（Male）恰恰最有价值，因此 C4.5 性能略逊（84% vs 86%）。
- **计算效率**：CART 的基尼指数计算最快（仅需平方运算），ID3 次之（需对数），C4.5 最慢（需同时计算信息增益和 IV）。在大规模数据集上，CART 的计算优势更为明显。

5.2 两种剪枝策略对比

表 4: 预剪枝与后剪枝对比

特性	预剪枝	后剪枝
剪枝时机	构建中（贪心）	构建后（全局）
训练速度	快	慢
泛化能力	较好	**更好**
Melon	71.43%	71.43%
CelebA (ID3)	76%	**86%** ↑10%
CelebA (C4.5/CART)	68%/76%	**84%** ↑16%

剪枝策略分析：

- **后剪枝的优势：**在大规模复杂数据集（CelebA）上，后剪枝显著优于预剪枝（ID3 提升 10 个百分点，C4.5/CART 提升 16 个百分点）。这是因为后剪枝基于**完整树结构进行全局优化**，能够识别那些单独看无效但组合后有协同效应的特征划分。例如，某个特征在根节点划分时可能无法提升验证集性能，但与其子节点的特征组合后可能产生强判别能力。
- **预剪枝的局限：**预剪枝采用贪心策略，在构建过程中一旦某次划分无法提升验证集性能就停止生长，可能过早终止，导致欠拟合。在 CelebA 上，预剪枝的 C4.5 仅达 68%，显著低于后剪枝的 84%，证明了这一点。
- **小数据集的特殊性：**在 Melon 数据集上，预剪枝和后剪枝性能相同（均 71.43%），这是因为样本量小（17 个训练样本），决策树深度有限，两种策略的差异不明显。但剪枝仍然关键——不剪枝仅 42.86%，剪枝后提升 67%。

6 实验结论

本实验系统实现并比较了 ID3、C4.5、CART 决策树算法及预剪枝、后剪枝策略，主要结论：

1. **算法性能：**CelebA 数据集上，ID3+ 后剪枝最优（86%），C4.5/CART+ 后剪枝均达 84%。ID3 简单高效，但偏好多值特征；C4.5 通过 IV 平衡特征选择，但计算较复杂；CART 使用基尼指数作为划分准则，与 ID3 性能接近。
2. **剪枝效果：**后剪枝在大数据集上显著优于预剪枝（CelebA：ID3 提升 10%，C4.5/CART 提升 16%）；剪枝有效缓解过拟合（Melon：从 42.86% 提升至 71.43%）。
3. **特征重要性：**CelebA 数据集中，Male（性别）是最强判别特征（信息增益 0.896，基尼指数 0.838），其次是 Wearing_Lipstick（0.764）和 Big_Nose（0.582）。