人工智能实验报告 LAB2

(2021学年秋季学期)

课程名称：Artificial Intelligence

教学班级	计科2班	专业（方向）	计算机科学与技术
学号	19335174	姓名	施天予

一、 实验题目

决策树

二、 实验内容

1、算法原理

决策树的通用算法

决策树的生成算法分为以下几个**步骤**：

- 初始化：创建根节点，拥有全部的数据集和特征
- 选择特征：遍历当前节点的数据集和特征，依据某种原则，选择一个特征
- 划分数据：依据所选特征的不同取值，将当前数据集划分为若干个子集
- 创建节点：为每个子数据集创建一个子节点，并删去刚刚选中的特征
- 递归建树：对每个子节点，回到第二步进行递归调用，直到达到边界条件，则回溯
- 完成建树：叶子节点采用多数投票的方式判定自身的类别

若当前节点的数据集为D，特征集为A，则递归的**边界条件**的判断方式如下（满足其一即可）：

- 若D中的样本属于同一类别C，则将当前的节点标记为C类叶节点
- A为空集，或D中所有样本在A中所有特征上取值相同，则无法划分。当前节点标记为叶节点，类别为D中出现最多的类
- D为空集，则将当前节点标记为叶节点，类别为父节点中出现最多的类

(以上通用算法参考ppt)

ID3决策树

ID3决策树是采用**信息增益**来决定通过哪个特征作为决策点的。信息增益越大，说明该特征对得到结果的帮助越大，则优先选用信息增益最大的特征作为决策点。信息增益的算法如下：

假设训练数据集为D，$|D|$表示样本容量，样本有K个类，记为$C_k, k=1,2,\ldots,K$，其中$|C_k|$表示该类的样本个数。依据特征A的n个不同取值$\{a_1,a_2,\ldots,a_n\}$，将D划分为n个子集$\{D_1,\ldots,D_n\}$，记子集D_i中属于类C_k的样本集合为D_{ik}。

计算数据集D的经验熵：

$$H(D)=-\sum_{k=1}^{K}\frac{|C_k|}{|D|}\log\frac{|C_k|}{|D|}$$

计算特征A对数据集D的条件熵：

$$H(D|A)=\sum_{i=1}^{n}\frac{|D_i|}{|D|}H(D_i)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}\sum_{k=1}^{K}\frac{|D_{ik}|}{|D_i|}\log\frac{|D_{ik}|}{|D_i|}$$

计算信息增益：

$$g(D,A)=H(D)-H(D|A)$$

对所有的特征，都计算出相应的信息增益。比较各个特征的信息增益，最终选择使得信息增益最大的特征作为决策点。

C4.5决策树

C4.5决策树是ID3决策树的升级版。ID3决策树以信息增益作为划分训练数据集的特征，存在偏向于选择取值较多的特征的问题，即该属性划分出的子类很多的话，信息增益就会更大。使用信息增益率可以对这一问题进行校正。所以C4.5决策树使用**信息增益率**作为选取属性作为决策点的判断依据。首先能够通过ID3决策树上述提到的算法计算出数据集的信息增益，得到信息增益后再除以特征的信息熵即可计算信息增益率。

n为特征A的每种取值的个数，则数据集D关于特征A的值的熵为：

$$SplitInfo(D,A)=-\sum_{i=1}^{n}\frac{|D_i|}{|D|}log_2\frac{|D_i|}{|D|}$$

特征A对数据集D的信息增益率，定义为信息增益与数据集D关于特征A的值的熵的比，即：

$$gRatio(D,A)=\frac{g(D,A)}{SplitInfo(D,A)}$$

信息增益率，考虑到了特征的多种取值对信息增益的影响。每次选择信息增益率最大的属性作为决策点生成分支节点。

CART决策树

CART决策树使用了**基尼系数**来作为划分数据集的指标，基尼系数代表了模型的不纯度，基尼系数越小，代表不确定性越小，不纯度越低，特征越好。选择基尼系数最小的特征作为决策点

假设有n个类，样本点属于第i个类的概率是p_i，则概率分布的基尼指数定义为

$$gini(D_j|A=A_j)=\sum_{i=1}^{n}p_i(1-p_i)=1-\sum_{i=1}^{n}p_i^2$$

计算特征A条件下数据集D的基尼系数

$$gini(D,A)=\sum_{j=1}^{v}p(A_j)*gini(D_j|A=A_j)$$

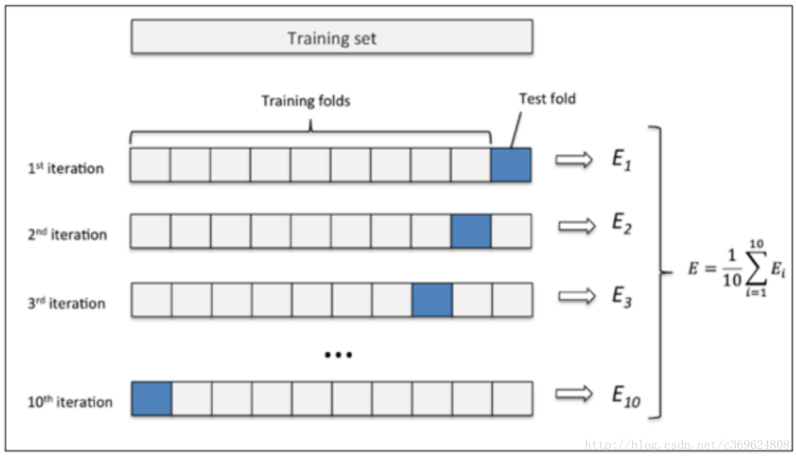
通过计算得到所有特征的基尼系数后，选择基尼系数最小的特征划分数据集。

数据集的划分

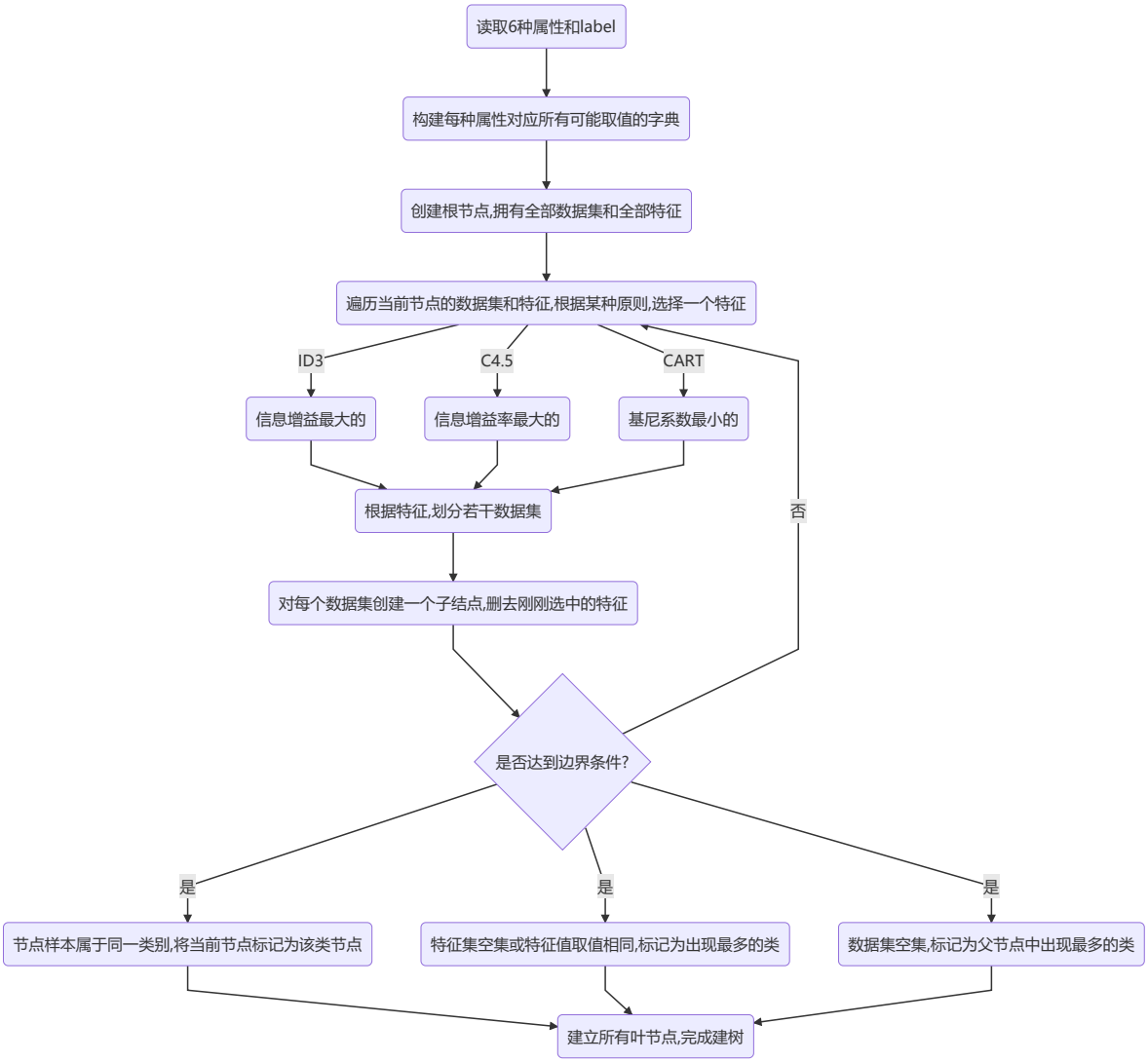
开始我考虑使用的是直接选取一定比例的数据集作为验证集，但按照“5：5”，“6：4”，“7：3”，“8：2”这样的方式不好掌控，导致误差比较大

后来我想起来老师讲过的“**k折交叉验证法**”， 先将数据集D划分为k个大小相似的子集，然后遍历k个子集，每次用k-1个子集加起来作为训练集，剩下的1个子集作为验证集；这样就可以获得k组训练集/验证集，从而可以进行k次训练和验证，最终返回的是k个验证结果的均值，作为这1个k值对应的准确率。

当 K = 10 时，示意图如下：



2、流程图


读取6种属性和label
构建每种属性对应所有可能取值的字典
创建根节点,拥有全部数据集和全部特征
遍历当前节点的数据集和特征,根据某种原则,选择一个特征
ID3
C4.5
CART
信息增益最大的
信息增益率最大的
基尼系数最小的
根据特征,划分若干数据集
否
对每个数据集创建一个子结点,删去刚刚选中的特征
是否达到边界条件?
是
是
是
节点样本属于同一类别,将当前节点标记为该类节点
特征集空集或特征值取值相同,标记为出现最多的类
数据集空集,标记为父节点中出现最多的类
建立所有叶节点,完成建树

3、关键代码展示

计算经验熵或者某个特征的信息熵

```
def cal_entropy(data_set, attribute_index):
    # data_set：数据集
    # attribute_index：所选属性的下标
    cnt = {}  # 存储属性的取值及其数量
    for line in data_set:
        value = line[attribute_index]
        cnt[value] = cnt.get(value, 0) + 1  # 计算属性的每种取值的个数
    entropy = 0.0
    for count in cnt.values():
        entropy -= float(count) / len(data_set) * math.log2(p)     #
用经验熵公式计算
    return entropy
```

计算信息增益

```
def cal_gain(data_set, dic, attribute_index, empirical_entropy):
    # data_set：数据集
    # dic：属性字典，对应每种属性的取值
    # attribute_index：所选属性的下标
    # empirical_entropy：经验熵
    total = len(data_set)
    conditional_entropy = 0.0
    for value in dic[attribute_index]:  # 遍历这个属性对应的取值
        sub_set = split_dataset(data_set, attribute_index, value)   #
获得对应属性的子数据集
        p = len(sub_set) / total
        conditional_entropy += p * cal_entropy(sub_set, -1)  # 计算条件
熵
    return empirical_entropy - conditional_entropy
```

计算基尼系数

```
def cal_gini(data_set, attribute_index):
    # data_set：数据集
    # attribute_index：所选属性的下标
    attribute_count = {}    # 所选属性的每种取值的数量
    attribute_label = {}    # 所选属性的每种取值对应的label数量
```

```
    total = len(data_set)
    for line in data_set:
        value = line[attribute_index]
        attribute_count[value] = attribute_count.get(value, 0) + 1
        attribute_label[value] = attribute_label.get(value, {})     #
get默认返回空
        if line[-1] not in attribute_label[value]:
            attribute_label[value][line[-1]] = 0    # 将对应的label的次
数赋为0
        attribute_label[value][line[-1]] += 1
    gini = 0.0
    for value in attribute_count.keys():
        size = attribute_count[value]   # 当前属性的某种取值的数量
        temp = 1
        for x in attribute_label[value].values():
            temp -= np.square(x / size)
        gini += size / total * temp     # 计算基尼系数
    return gini
```

根据不同的方法：ID3、C4.5、CART

选出要划分的最优属性

```
def choose_attribute(data_set, dic, left_attribute, strategy):
    # data_set: 数据集
    # dic: 属性字典，对应每种属性的取值
    # left_attribute: 当前剩下的的属性集合，存的是属性在dic中对应的下标
    # strategy: 特征选择的方法
    if strategy == "ID3":
        gain_list = []
        for attribute_index in left_attribute:   # 遍历每种属性的下标
            empirical_entropy = cal_entropy(data_set, -1)  # 经验熵
            gain = cal_gain(data_set, dic, attribute_index,
empirical_entropy)
            gain_list.append(gain)
        max_index = np.argmax(gain_list)  # 信息增益最大的下标
        return left_attribute[max_index]
    if strategy == "C4.5":
        gainRatio_list = []
        for attribute_index in left_attribute:   # 遍历每种属性的下标
            empirical_entropy = cal_entropy(data_set, -1)  # 经验熵
            gain = cal_gain(data_set, dic, attribute_index,
empirical_entropy)
            SplitInfo = cal_entropy(data_set, attribute_index)    # 计
算特征的信息熵
            if SplitInfo == 0:     # 说明这个属性对决策没贡献
                continue
            gainRatio_list.append(gain/SplitInfo)
        max_index = np.argmax(gainRatio_list)  # 信息增益率最大的下标
```

```
        return left_attribute[max_index]
    if strategy == "CART":
        gini_list = []
        for i in left_attribute:
            gini = cal_gini(data_set, i)    # 计算基尼系数
            gini_list.append(gini)
        min_index = np.argmin(gini_list)    # 基尼系数最小的下标
        return left_attribute[min_index]
        for attribute_index in left_attribute:   # 遍历每种属性的下标
            empirical_entropy = cal_entropy(data_set, -1)  # 经验熵
            gain = cal_gain(data_set, dic, attribute_index,
empirical_entropy)
            gain_list.append(gain)
        max_index = np.argmax(gain_list)  # 信息增益最大的下标
        return left_attribute[max_index]
```

构建决策树

当遇到3种边界条件，停止建树，返回叶子节点（用类来当作节点的思想来源网络）

通过3种不同方法：ID3、C4.5、CART选择最优属性进行划分

```
def create_tree(data_set, dic, left_attribute, parent_label, method):
    # data_set: 数据集
    # dic: 属性字典，对应每种属性的取值
    # left_attribute: 当前剩下的的属性集合，存的是属性在dic中对应的下标
    # parent_label: 父节点的label
    # method: 特征选择的方法
    if len(data_set) == 0:  # data_set为空集，取父节点的属性
        return TreeNode(label=parent_label)
    if label_list.count(label_list[0]) == len(label_list):  # data_set
里的样本都取同一label
        return TreeNode(label=label_list[0])
    if len(left_attribute) == 0:    # 没有属性可选时，即left_atttribute
为空时，找出众数标签
        label = max(label_list, key=label_list.count)
        return TreeNode(label=label)
    left_attribute.remove(choose_attribute(data_set, dic,
left_attribute, method))   # 去除最好的属性
    for value in dic[best_attribute]:     # 用最好的属性划分data_set，构
建子树
        sub_set = split_dataset(data_set, best_attribute, value)
        branch[value] = create_tree(sub_set, dic, left_attribute[:],
parent_label, method)
    return TreeNode(label=parent_label, attribute=best_attribute,
branch=branch)
```

计算验证集上的准确率

```
def cal_accuracy(valid_set, root):
    cnt = 0
    for line in valid_set:
        cur = root
        while cur.branch is not None:
            cur = cur.branch[line[cur.attribute]]   # 直到叶子节点
        if cur.label == line[-1]:
            cnt += 1
    return cnt/len(valid_set)
```

验证集验证（k从2到9）

对每个k值输出它的平均准确率

用matplotlib画出折线图

```
def validation(data_set, attribute_set, dic, method):
    x = []
    y = []
    k_best = 0
    accuracy_best = 0
    print(method, '方法')
    for k in range(2, 10):
        temp = 0
        for i in range(k):  # 对不同的验证集取均值作为一个k的结果
            train_set, valid_set = k_fold(data_set, k, i)
            left_attribute = list(range(0, len(attribute_set) - 1))
            root = create_tree(train_set, dic, left_attribute, -1,
method)
            temp += cal_accuracy(valid_set, root)
        accuracy = temp / k
        print('用', k,'折交叉验证时，准确率为', accuracy)
        if accuracy > accuracy_best:
            k_best = k
            accuracy_best = accuracy
        x.append(k)
        y.append(accuracy)
    print('用', k_best,'折交叉验证时最佳，最高准确率为', accuracy_best)
    print()
    plt.plot(x, y, label=method)
    plt.grid(True)
    plt.xlabel('k')
    plt.ylabel('accuracy')
    plt.title(method)
    plt.legend()
    plt.show()
```

计算树的节点数（便于检测建树是否正确，对于实验结果无影响）

```
def cal(root):
    queue = []
    queue.append(root)
    res = 0
    while queue:            # 层序遍历
        size = len(queue)
        res += size
        for i in range(0, size):
            node = queue[0]
            queue.pop(0)
            if node.branch is not None:
                for value in dic[node.attribute]:
                    queue.append(node.branch[value])
    return res
```

4、创新点

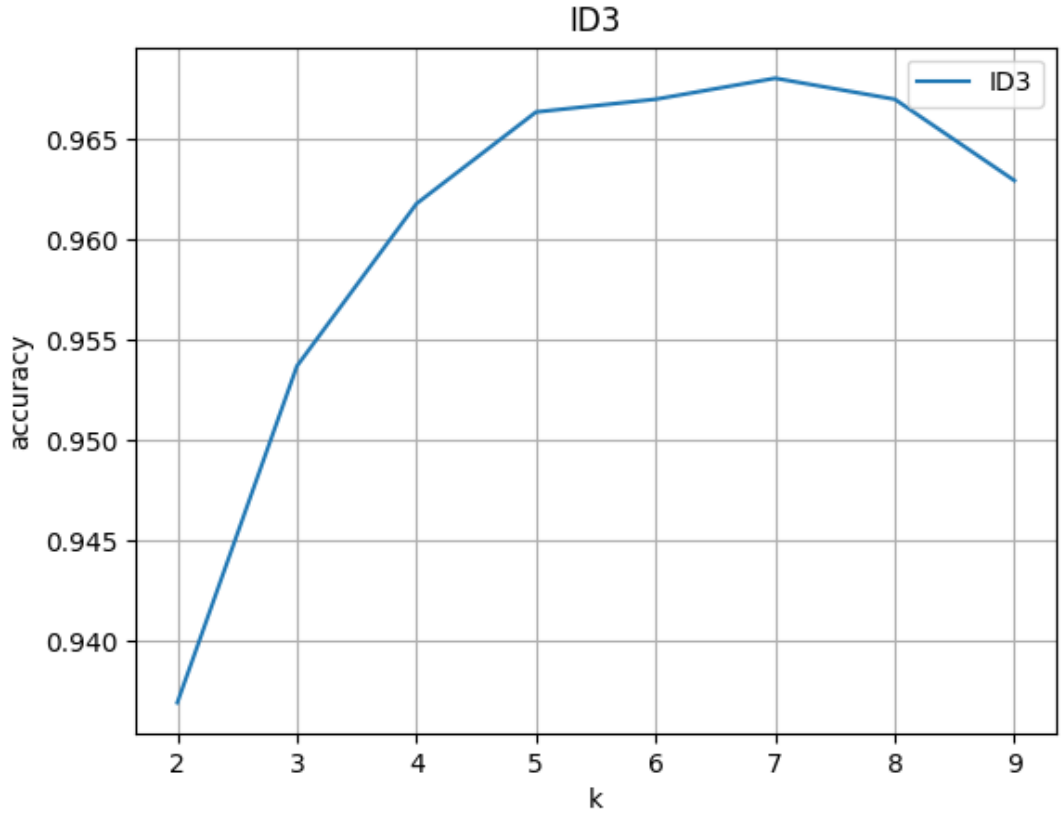
- 采用k折交叉验证法，对于三种特征选择方法都进行了k从2到9的验证，计算出最高准确率
- 写了一个计算树的节点的简单函数，方便检测建树是否正确

三、 实验结果及分析

1、实验结果展示示例

ID3算法

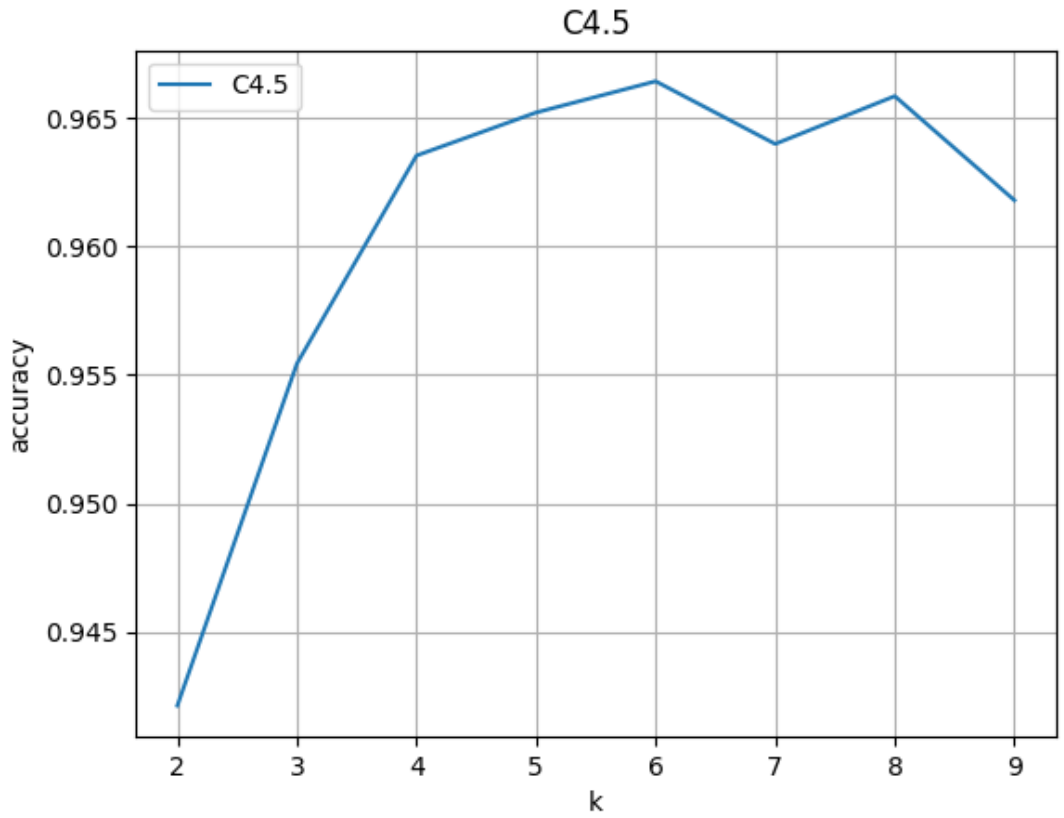
```
ID3 方法
用 2 折交叉验证时，准确率为 0.9369212962962963
用 3 折交叉验证时，准确率为 0.9537037037037037
用 4 折交叉验证时，准确率为 0.9618055555555556
用 5 折交叉验证时，准确率为 0.9663768115942029
用 6 折交叉验证时，准确率为 0.967013888888889
用 7 折交叉验证时，准确率为 0.9680603948896632
用 8 折交叉验证时，准确率为 0.9670138888888888
用 9 折交叉验证时，准确率为 0.9629629629629629
用 7 折交叉验证时最佳，最高准确率为 0.9680603948896632
```



根据输出结果和折线图可知，利用ID3算法的信息增益指标进行特征选择时，准确率整体随着k的增大先上升后下降，上升的原因是一开始k太小训练样本过少，下降的原因可能是k过大验证集数据量较小，随机性较大。因为准确率整体还是很高的，误差也不是很大，所以ID3方法十分可靠。

C4.5算法

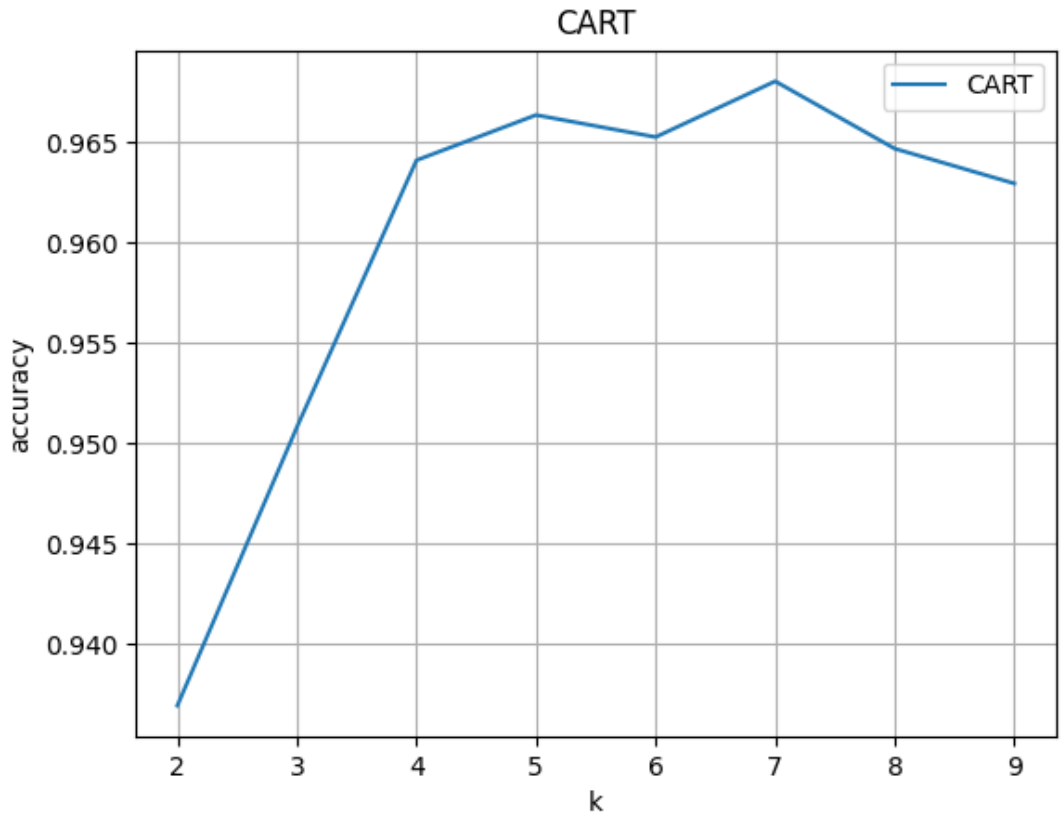
```
C4.5 方法
用 2 折交叉验证时，准确率为 0.9421296296296297
用 3 折交叉验证时，准确率为 0.9554398148148149
用 4 折交叉验证时，准确率为 0.9635416666666667
用 5 折交叉验证时，准确率为 0.9652173913043478
用 6 折交叉验证时，准确率为 0.9664351851851852
用 7 折交叉验证时，准确率为 0.9639953542392566
用 8 折交叉验证时，准确率为 0.9658564814814814
用 9 折交叉验证时，准确率为 0.9618055555555556
用 6 折交叉验证时最佳，最高准确率为 0.9664351851851852
```



根据输出结果和折线图可知，利用C4.5算法的信息增益率指标进行特征选择时，准确率整体随着k的增大在一定范围内上下波动，但其实波动范围也不是很大。因为每个属性的取值不是很多，所以用信息增益率对比信息增益的效果不是特别明显。所以总体来说C4.5准确率也很高，但比ID3略微差一点。

CART算法

```
CART 方法
用 2 折交叉验证时，准确率为 0.9369212962962963
用 3 折交叉验证时，准确率为 0.9508101851851851
用 4 折交叉验证时，准确率为 0.9641203703703703
用 5 折交叉验证时，准确率为 0.9663768115942029
用 6 折交叉验证时，准确率为 0.9652777777777778
用 7 折交叉验证时，准确率为 0.9680603948896632
用 8 折交叉验证时，准确率为 0.9646990740740741
用 9 折交叉验证时，准确率为 0.9629629629629629
用 7 折交叉验证时最佳，最高准确率为 0.9680603948896632
```



根据输出结果和折线图可知，利用CART算法的GINI指数进行特征选择时，准确率整体随着k的增大先上升，后在一定范围内波动。整体准确率也十分高，误差范围比较小，且整体准确率结果与ID3算法十分相近，可能是由于数据量比较小的原因。

2、评测指标展示及分析

	ID3算法	C4.5算法	CART算法	准确率	最优k值
初始	1	0	0	96.8060%	7
优化1	0	1	0	96.6435%	6
优化2	0	0	1	96.8060%	7
最优结果	1	0	0	96.8060%	7

在本次的决策树实验中，我使用ID3方法和CART方法在 k = 7 时得到的最高准确率一样高，可能是由于数据量不是特别大，都能达到96.8060%。总而言之，这次实验决策树的三种特征选择方法都能取得不错的结果（有可能是过拟合了）。

四、思考题

1、决策树有哪些避免过拟合的方法?

可以通过**剪枝**的方法避免决策树的过拟合

预剪枝：在生成子节点时，如果对当前节点，划分后决策树在验证集上的准确率没有提高，则不继续划分，而是直接将当前节点设置为叶子节点

后剪枝：先生成完整决策树，后序遍历，对每个非叶节点，如果将该节点变成叶节点，在验证集上的准确率不会降低，则设置其为叶节点

2、C4.5相比于ID3的优点是什么，C4.5又可能有什么缺点?

优点：ID3只考虑信息增益作为选择特征作为决策树划分的依据，在某个特征可选的特征值很多时，往往会使得信息增益更高。也就是说，信息增益偏向取值较多的特征，从而使决策树产生大量分支。C4.5考虑到了特定特征下的数据集信息熵，使用信息增益率的指标，从而解决了ID3的缺陷。

缺点：C4.5需要对数据进行多次的扫描和计算，效率较低，只适合小规模数据集。并且，ID3和C4.5都只能处理离散数据而不能处理连续性数据。

3、如何用决策树来进行特征选择（判断特征的重要性）？

一般有三种方法构建决策树：**ID3、C4.5、CART**。ID3采用信息增益，C4.5采用信息增益比，选取信息增益或信息增益比最大的特征选择；CART采用基尼系数，越小代表不纯度越低，选取基尼系数最小的特征选择。