**实验**

**实验要求：**

用C4.5算法对鸢尾花数据集进行分类。

C4.5算法具体可网上查询具体算法进行学习。

数据：iris数据集包含在sklearn库当中，具体在sklearn\datasets\data文件夹下，文件名为iris.csv。也可自行网上下载。

**环境：python 3**

**验收方式**：实验报告

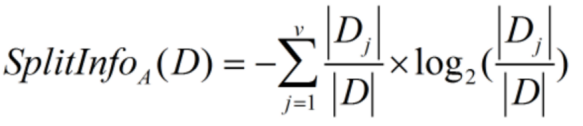
**报告书写要求**：

1. **C4.5算法**

C4.5是一系列用在机器学习和数据挖掘的分类问题中的算法。它的目标是**[监督学习](https://baike.baidu.com/item/%E7%9B%91%E7%9D%A3%E5%AD%A6%E4%B9%A0?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/C4.5%E7%AE%97%E6%B3%95/_blank)**：给定一个数据集，其中的每一个元组都能用一组属性值来描述，每一个元组属于一个互斥的类别中的某一类。C4.5的目标是通过学习，找到一个从属性值到类别的映射关系，并且这个映射能用于对新的类别未知的实体进行分类。

C4.5由J.Ross Quinlan在ID3的基础上提出的。[ID3算法](https://baike.baidu.com/item/ID3%E7%AE%97%E6%B3%95/5522381?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/C4.5%E7%AE%97%E6%B3%95/_blank)用来构造决策树。决策树是一种类似流程图的树结构，其中每个内部节点（非树叶节点）表示在一个属性上的测试，每个分枝代表一个测试输出，而每个树叶节点存放一个类标号。一旦建立好了决策树，对于一个未给定类标号的元组，跟踪一条有根节点到叶节点的路径，该叶节点就存放着该元组的预测。决策树的优势在于不需要任何领域知识或参数设置，适合于探测性的知识发现。

C4.5算法用**[信息增益率](https://baike.baidu.com/item/%E4%BF%A1%E6%81%AF%E5%A2%9E%E7%9B%8A%E7%8E%87/7693446?fromModule=lemma_inlink" \t "https://baike.baidu.com/item/C4.5%E7%AE%97%E6%B3%95/_blank)**来选择属性。ID3选择属性用的是子树的信息增益，这里可以用很多方法来定义信息，ID3使用的是熵（entropy，熵是一种不纯度度量准则），也就是熵的变化值，而C4.5用的是信息增益率，公式：



在决策树构造过程中进行剪枝，因为某些具有很少元素的结点可能会使构造的决策树过适应（Overfitting），如果不考虑这些结点可能会更好。

**算法优点**：产生的分类规则易于理解，准确率较高。

**缺点**：在构造树的过程中，需要对数据集进行多次的顺序扫描和排序，因而导致算法的低效。此外，C4.5只适合于能够驻留于内存的数据集，当训练集大得无法在内存容纳时程序无法运行。

1. **程序实现**
2. **数据处理**

首先得到全部数据：

|  |
| --- |
| def load\_preprocess\_data():  iris\_data = []  cnt = 0  # 得到全部数据  with open("iris.csv") as f:  lines = f.readlines()  for line in lines:  if cnt == 0:  cnt += 1  continue  tmp = line.strip().split(',')  for i, e in enumerate(tmp):  if i < 4:  tmp[i] = float(e)  iris\_data.append(tmp)  print("数据集：\n", iris\_data)  iris\_labels = ["花萼长度", "花萼宽度", "花瓣长度", "花瓣宽度"] |

由于四种特征全为连续型数值，将其转换为离散型特征，找到分界点的阈值。Petal\_width的分界情况如图所示

|  |
| --- |
| categ\_sepal\_len = pd.cut(iris['sepal\_len'],3)  catag\_sepal\_wid = pd.cut(iris['sepal\_width'], 3)  catag\_petal\_len = pd.cut(iris['petal\_len'], 3)  catag\_petal\_wid = pd.cut(iris['petal\_width'], 3) |

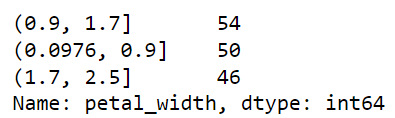


图1 数据集分界点

划分数据集，使得测试集为20%，其余80%为训练集

|  |
| --- |
| def get\_train\_test\_dadaset(iris\_data, split = 0.8):  num = len(iris\_data)  train\_data = []  test\_data = []  for i in range(num):  if i % 5 == 0:  test\_data.append(iris\_data[i])  else:  train\_data.append(iris\_data[i])  return train\_data, test\_data |

1. **公式代码**

因C4.5算法中使用到了较多如信息熵、信息增益率等公式，因此对于这些公式进行代码定义

|  |
| --- |
| # 计算数据集香农信息熵  def cal\_shannon\_entropy(dataset):  num = len(dataset)  label\_dic = {}  for row in dataset:  if row[-1] not in label\_dic.keys():  label\_dic[row[-1]] = 0  label\_dic[row[-1]] += 1  entropy = 0  for key in label\_dic:  p = label\_dic[key] / float(num)  entropy -= p \* np.log2(p)  return entropy |

1. **选择最优特征**

|  |
| --- |
| # c4.5算法选择最优特征  def choose\_feature\_to\_split(dataset, c45 = False):  length = len(dataset)  num\_feature = len(dataset[0]) - 1 #特征个数  base\_entropy = cal\_shannon\_entropy(dataset)  best\_info\_gain = 0  best\_info\_gain\_ratio = 0  best\_feature\_index = -1  for feature\_index in range(num\_feature):  feat\_list = [row[feature\_index] for row in dataset]  feat\_dic = Counter(feat\_list) #统计特征种类及出现次数  cur\_entropy = 0  cur\_feat\_entropy = 0 # 数据集关于当前特征的信息熵, 用于c4.5算法计算  for feat in feat\_dic:  sub\_dataset = split\_dataset(dataset, feature\_index, feat)  sub\_entropy = cal\_shannon\_entropy(sub\_dataset)  cur\_entropy += feat\_dic[feat] / float(length) \* sub\_entropy  cur\_feat\_entropy -= feat\_dic[feat] / float(length) \* np.log2(feat\_dic[feat] / float(length))  # C4.5算法计算  gain\_ration = (base\_entropy - cur\_entropy) / cur\_feat\_entropy  if gain\_ration > best\_info\_gain\_ratio:  best\_info\_gain\_ratio = gain\_ration  best\_feature\_index = feature\_index  return best\_feature\_index |

1. **构建决策树**

|  |
| --- |
| # 深度优先构建决策树  def create\_decision\_tree(dataset, labels, c45):  class\_list = [row[-1] for row in dataset]  # 如果数据集全部是同一个类别，直接返回该类别  if len(class\_list) == class\_list.count(class\_list[0]):  return class\_list[0]  # 如果没有特征可以继续划分或者特征区分度为0，返回出现次数最多的类别  if len(dataset[0]) == 1 or choose\_feature\_to\_split(dataset, c45) == -1:  class\_dic = Counter(class\_list)  return class\_dic.most\_common()[0][0]  # 根据c4.5算法选择信息增益最高的特征作为当前分类特征  best\_feature\_index = choose\_feature\_to\_split(dataset, c45)  best\_feature\_label = labels[best\_feature\_index]  labels.remove(best\_feature\_label)  # 以字典的形式构建决策树  decision\_tree = {best\_feature\_label : {}}  feat\_list = [row[best\_feature\_index] for row in dataset]  feat\_dic = Counter(feat\_list)  # 根据当前最优特征划分子数据集  for feat in feat\_dic:  sub\_dataset = split\_dataset(dataset, best\_feature\_index, feat)  sub\_labels = labels[:]  # 递归构建子数据集决策树  decision\_tree[best\_feature\_label][feat] = create\_decision\_tree(sub\_dataset, sub\_labels)  return decision\_tree |

1. **分类结果**

分类结果如图所示，可以看到，在选取150组数据中20%（即30组）数据作为测试集的情况下，有六组错误分类，准确率为80%。

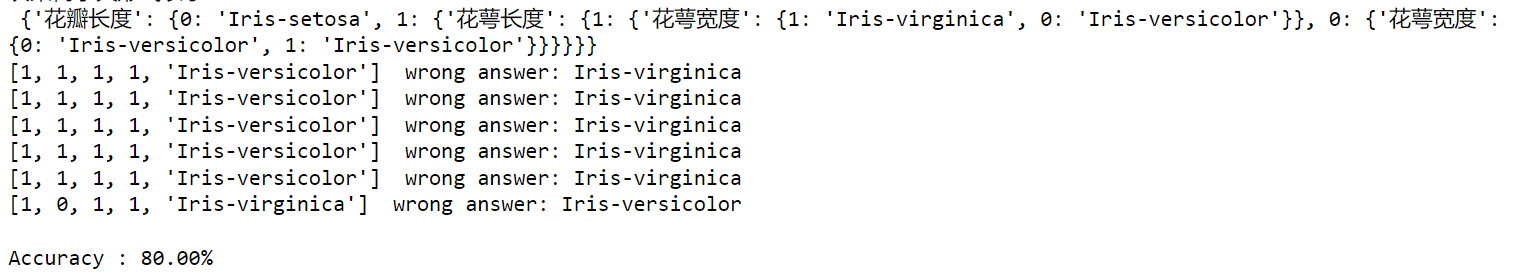


图2 分类结果

但通过分析发现，构建的决策树是略有问题的，属于同一种的数据，却有分支。这里分析是因为鸢尾花数据集本来是连续性数据，这里**强行离散化**处理，并不能很好的进行区分，所以在这些分支，每一个分支下对应的数据均没有把数据完全分开，即未达到递归的第一个终止条件，达到了第二个条件，而且在每个分支中最大的都是同一种标签，就出现了这种情况。后期需要针对这种情况，进一步改进优化。

1. **收获与体会**

本次实验中，学习了使用C4.5算法对鸢尾花进行分类，C4.5算法是基于机器学习中较常使用的决策树算法，进行特征点的选择。算法中涉及到信息熵、信息增益、信息增益率等信息论中的内容，算法解释性较好，对于分类的结果准确度也较好。

相较于上一次k-nn算法，此次实验对于代码编写、数学理解都有了一定的培养，也从两次对同一任务使用不同算法中，体会到了机器学习的乐趣。也从分类结果中，分析处理造成准确度不太理想的主要原因，也为后续的改进工作和日后的学习，打下基础。