**作业4.3**

1. **实验要求**

试编程实现基于信息熵进行划分选择的决策树算法，并为表4.3中数据生成一棵决策树。

1. **实验原理**

决策树在对类进行划分时，主要采用了信息增益为依据进行划分，计算各个属性的信息熵，并计算其与划分之前的信息熵作比较得到信息增益。取其中信息增益最大的一个属性划分该层，再递归调用上述过程继续划分属性，直到最后的样本中只有一个类，或者各属性的值相同（此时取其中类最多的作为划分的类）。

1. **实验过程**

这里使用了treePloter中的plottree模块用来画出决策树，决策树的计算采用递归调用的方式，并将计算的结果保存在字典里打印出来。

1. 定义信息熵的计算。
2. 定义信息增益的计算。
3. 返回信息增益最大的那个属性。
4. 建树，并递归调用决策树。
5. 画图。

代码如下：

**import** math  
**import** pandas **as** pd  
**import** numpy **as** np  
  
**from** treePlotter **import** createPlot  
**def** entropy(data):  
 label\_values = data[data.columns[-1]]  
 *#Returns object containing counts of unique values.* counts = label\_values.value\_counts()  
 s = 0  
 **for** c **in** label\_values.unique():  
 freq = float(counts[c])/len(label\_values)  
 s -= freq\*math.log(freq,2)  
 **return** s  
  
**def** is\_continuous(data,attr):  
 *"""Check if attr is a continuous attribute"""* **return** data[attr].dtype == **'float64'  
  
def** split\_points(data,attr):  
 *"""Returns Ta,Equation(4.7),p.84"""* values = np.sort(data[attr].values)  
 **return** [(x+y)/2 **for** x,y **in** zip(values[:-1],values[1:])]  
  
**def** discrete\_gain(data,attr):  
 V = data[attr].unique()  
 s = 0  
 **for** v **in** V:  
 data\_v = data[data[attr]== v]  
 s += float(len(data\_v))/len(data)\*entropy(data\_v)  
 **return** (entropy(data) - s,**None**)  
  
**def** continuous\_gain(data,attr,points):  
 *"""Equation(4.8),p.84,returns the max gain along with its splitting point"""* entD = entropy(data)  
 *#gains is a list of pairs of the form (gain,t)* gains = []  
 **for** t **in** points:  
 d\_plus = data[data[attr] > t]  
 d\_minus = data[data[attr] <= t]  
 gain = entD - (float(len(d\_plus))/len(data)\*entropy(d\_plus)+float(len(d\_minus))/len(data)\*entropy(d\_minus))  
 gains.append((gain,t))  
 **return** max(gains)  
  
**def** gain(data,attr):  
 **if** is\_continuous(data,attr):  
 points = split\_points(data,attr)  
 **return** continuous\_gain(data,attr,points)  
 **else**:  
 **return** discrete\_gain(data,attr)  
  
**def** majority(label\_values):  
 counts = label\_values.value\_counts()  
 **return** counts.index[0]  
  
**def** id3(data):  
 attrs = data.columns[:-1]  
 *#attrWithGain is of the form [(attr,(gain,t))], t is None if attr is discrete* attrWithGain = [(a,gain(data,a)) **for** a **in** attrs]  
 attrWithGain.sort(key = **lambda** tup:tup[1][0],reverse = **True**)  
 **return** attrWithGain[0]  
  
**def** createTree(data,split\_function):  
 label = data.columns[-1]  
 label\_values = data[label]  
 *#Stop when all classes are equal* **if** len(label\_values.unique()) == 1:  
 **return** label\_values.values[0]  
 *#When no more features, or only one feature with same values, return majority* **if** data.shape[1] == 1 **or** (data.shape[1]==2 **and** len(data.T.ix[0].unique())==1):  
 **return** majority(label\_values)  
 bestAttr,(g,t) = split\_function(data)  
 *#If bestAttr is discrete* **if** t **is None**:  
 *#In this tree,a key is a node, the value is a list of trees,also a dictionary* myTree = {bestAttr:{}}  
 values = data[bestAttr].unique()  
 **for** v **in** values:  
 data\_v = data[data[bestAttr]== v]  
 attrsAndLabel = data.columns.tolist()  
 attrsAndLabel.remove(bestAttr)  
 data\_v = data\_v[attrsAndLabel]  
 myTree[bestAttr][v] = createTree(data\_v,split\_function)  
 **return** myTree  
 *#If bestAttr is continuous* **else**:  
 t = round(t,3)  
 node = bestAttr+**'<='**+str(t)  
 myTree = {node:{}}  
 values = [**'yes'**,**'no'**]  
 **for** v **in** values:  
 data\_v = data[data[bestAttr] <= t] **if** v == **'yes' else** data[data[bestAttr] > t]  
 myTree[node][v] = createTree(data\_v,split\_function)  
 **return** myTree  
  
**if** \_\_name\_\_ == **"\_\_main\_\_"**:  
 f = pd.read\_csv(filepath\_or\_buffer = **'w3.csv'**, sep = **','**)  
 data = f[f.columns[1:]]  
 tree = createTree(data,id3)  
 print(tree)  
 createPlot(tree)

由于在调试过程中原来的参考代码基于的是python2.7,而本人的是python3， 其改变了dict.keys,返回的是dict\_keys对象,支持iterable 但不支持indexable，因此报错，本人将treeplot函数中的dict\_keys对象转换成了List对象解决了该问题。

1. **实验结果**

决策树的字典形式和图像形式分别如下：

{'texture': {'fuzzy': 'no', 'clear': {'density<=0.381': {'yes': 'no', 'no': 'yes'}}, 'slightly paste': {'touch': {'hard slip': 'no', 'soft sticky': 'yes'}}}}

