CPSC429/529: Machine Learning

Program 2: Decision Tree

Program Description

In this programming assignment, you can **form a group of two** to complete the ID3 algorithm provided to you (The pseudocode is given below).

Algorithm 4.1 Pseudocode description of the ID3 algorithm.

Require: set of descriptive features d

Require: set of training instances $\mathcal{D}$

1: if all the instances in $\mathcal{D}$ have the same target level C then
2: return a decision tree consisting of a leaf node with label C
3: else if d is empty then
4: return a decision tree consisting of a leaf node with the label of the majority target level in $\mathcal{D}$
5: else if $\mathcal{D}$ is empty then
6: return a decision tree consisting of a leaf node with the label of the majority target level of the dataset of the immediate parent node
7: else
8: d [*best*] $\leftarrow \arg\max_{d \in \mathrm{d}} IG(d, \mathcal{D})$
9: make a new node, $Node_{\mathrm{d}[best]}$, and label it with d [*best*]
10: partition $\mathcal{D}$ using d [*best*]
11: remove d [*best*] from d
12: for each partition $\mathcal{D}_i$ of $\mathcal{D}$ do
13: grow a branch from $Node_{\mathrm{d}[best]}$ to the decision tree created by rerunning *ID3* with $\mathcal{D} = \mathcal{D}_i$

Figure 1: The pseudocode for ID3 Algorithm

Particularly, you need to perform the following tasks:

1. Complete the four functions in the `dtree.py` file, including:

```
def entropy(self,classData):
""" calculate the entropy based on classData"""

def info_gain(self,data,classData,featureIndex):
```

```
""" Calculate informatin information"""

def majority_class (self, classData):
""" find the majority of class"""

def predictionAccuracy(self, predicted, actual):
""" compute prediction accuracy on predicted data"""
```

2. Run your program on "`play.data`", "`spam.data`", and "`vegetation.data`", and do screenshots for these three decision trees (Like Figures 1-3), and save them in an output file.

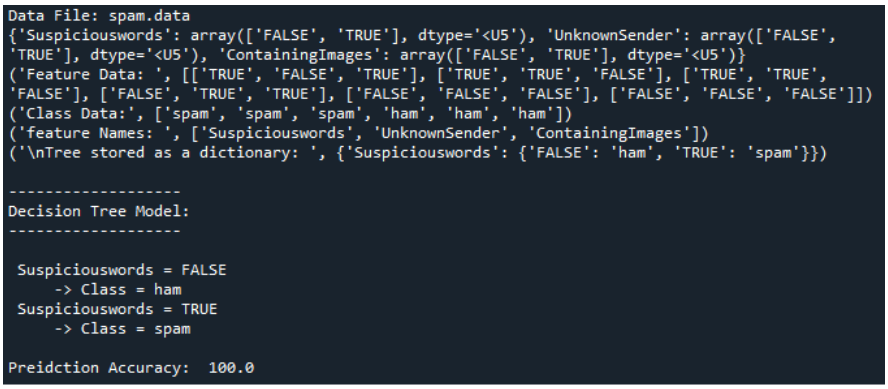

Figure 2: A sample decision tree output on "`spam.data`"

Submission

Upload the following items on D2L dropbox, including:

1. The source code (`dtree.py` needed only).
2. Program output file (should contain three decision trees).

Note: If you programmed with another group member, only one submission is sufficient. Make a note (your partner name) in your submission comment.

```
Data File: play.data
{'Outlook': array(['Overcast', 'Rain', 'Sunny'], dtype='<U8'), 'Temp': array(['Cool', 'Hot',
'Mild'], dtype='<U8'), 'Humid': array(['High', 'Normal'], dtype='<U8'), 'Wind': array(['Strong',
'Weak'], dtype='<U8')}
('Feature Data: ', [['Sunny', 'Hot', 'High', 'Weak'], ['Sunny', 'Hot', 'High', 'Strong'],
['Overcast', 'Hot', 'High', 'Weak'], ['Rain', 'Mild', 'High', 'Weak'], ['Rain', 'Cool', 'Normal',
'Weak'], ['Rain', 'Cool', 'Normal', 'Strong'], ['Overcast', 'Cool', 'Normal', 'Strong'], ['Sunny',
'Mild', 'High', 'Weak'], ['Sunny', 'Cool', 'Normal', 'Weak'], ['Rain', 'Mild', 'Normal', 'Weak'],
['Sunny', 'Mild', 'Normal', 'Strong'], ['Overcast', 'Mild', 'High', 'Strong']])
('Class Data:', ['No', 'No', 'Yes', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'Yes', 'Yes', 'Yes'])
('feature Names: ', ['Outlook', 'Temp', 'Humid', 'Wind'])
('\nTree stored as a dictionary: ', {'Outlook': {'Overcast': 'Yes', 'Rain': {'Wind': {'Strong':
'No', 'Weak': 'Yes'}}, 'Sunny': {'Humid': {'High': 'No', 'Normal': 'Yes'}}}})

-------------------
Decision Tree Model:
-------------------

 Outlook = Overcast
     -> Play = Yes
 Outlook = Rain
     Wind = Strong
         -> Play = No
     Wind = Weak
         -> Play = Yes
 Outlook = Sunny
     Humid = High
         -> Play = No
     Humid = Normal
         -> Play = Yes

Preidction Accuracy:  100.0
```

Figure 3: A sample decision tree output on "`play.data`"

```
Data File: vegetation.data
{'STREAM': array(['FALSE', 'TRUE'], dtype='<U8'), 'SLOPE': array(['flat', 'moderate', 'steep'],
dtype='<U8'), 'ELEVATION': array(['high', 'highest', 'low', 'medium'], dtype='<U8')}
('Feature Data: ', [['FALSE', 'steep', 'high'], ['TRUE', 'moderate', 'low'], ['TRUE', 'steep',
'medium'], ['FALSE', 'steep', 'medium'], ['FALSE', 'flat', 'high'], ['TRUE', 'steep', 'highest'],
['TRUE', 'steep', 'high']])
('Class Data:', ['chaparral', 'riparian', 'riparian', 'chaparral', 'conifer', 'conifer',
'chaparral'])
('feature Names: ', ['STREAM', 'SLOPE', 'ELEVATION'])
('\nTree stored as a dictionary: ', {'ELEVATION': {'high': {'SLOPE': {'flat': 'conifer',
'moderate': 'chaparral', 'steep': 'chaparral'}}, 'highest': 'conifer', 'low': 'riparian',
'medium': {'STREAM': {'FALSE': 'chaparral', 'TRUE': 'riparian'}}}})

-------------------
Decision Tree Model:
-------------------

 ELEVATION = high
     SLOPE = flat
         -> VEGETATION = conifer
     SLOPE = moderate
         -> VEGETATION = chaparral
     SLOPE = steep
         -> VEGETATION = chaparral
 ELEVATION = highest
     -> VEGETATION = conifer
 ELEVATION = low
     -> VEGETATION = riparian
 ELEVATION = medium
     STREAM = FALSE
         -> VEGETATION = chaparral
     STREAM = TRUE
         -> VEGETATION = riparian

Preidction Accuracy:  100.0
```

Figure 4: A sample decision tree output on "`vegetation.data`"