index
/home/marbille/UP/CS 180/mp2/DecisionTreeJuntadoMarbille.py

DecisionTreeJuntadoMarbille

```
@author: Marbille Juntado
Copyright: 2017

This program performs Decision Tree learning on dataset provided by tic-tac-toe.data.
It is based on the ID3 algorithm. Two experiments have been performed that outputs several
files consisting of the node trace, decision tree, confusion matrix, and accuracy results.
```

Modules

math random sys

Classes

TreeNode

class **TreeNode**

Methods defined here:

__init__(self, name)
```
Construct a new 'TreeNode' object.

:param name: The name of node
```

predictResults(self, cases, a)
```
Returns a list containing the predicted outcomes
```

predictResultsRecurse(self, case, a)
```
Recursively, method returns the predicted classification of the leaf nodes (bottom-
most)
```

visualizeTree(self)
```
Visualizes the tree
```

visualizeTreeRecurse(self, level)
```
Includes a log/trace of each node as the tree builds itself recursively
```

Functions

buildDTree(examples, targetAttribute, attributes)
```
Returns the root node of the decision tree

:param examples: Each line of the training set
```

constructTreeFromFile(filepath)
```
Builds a decision tree from the training data set file
```

entropy(p, e)
```
Calculates entropy

:param p: list of probabilities for each value
:param e: list of information gain for each value
```

gather_data(filename)

```
    This function is used in reading the data from the original data set file
```

getAttributesFromFile(filepath)
```
    The first line of the test file contains the attributes (categories)
```

getMostCommonLabel(nodes)
```
    Returns the most dominant classification in the list of nodes
```

getMostCommonValue(attr, examples, values)
```
    Returns the value with the highest frequency of the given attribute
```

header(data)
```
    Useful for data sets without any attribute names. Generically labels
    each attribute as 'Attribute + <number>' to accommodate different datasets.
    The last attribute is named 'Classification' (+/-).
```

infoGain(count1, count2)
```
    Returns the information gain at any particular level of tree construction

    :param count1: Contains the number of positively-classified training examples
    :param count2: Contains the number of negatively-classified training exampels
```

isNegative(word)
```
    Boolean function that determines whether a word is negative.
    Used in the classification of training example.
```

isPositive(word)
```
    Boolean function that determines whether a word is positive.
    Used in the classification of training example.
    :param word: any string
```

parseTestCases(filepath)
```
    Parses the test cases from the test data set file
```

returnAttributeHighestInfoGain(attributes, examples)
```
    Returns the attribute with the highest information gain and the corresponding value

    :param attributes: The attributes (categories) of the data
    :param examples: The training examples from the data set
```

split_data(data)
```
    Randomly divides the original data set into two equal sets:
    Training and Testing data sets

    :param: The original data sets
```

uniqueValues(attrIndex, examples)
```
    Returns list of the distinct values of the current attribute
```