Documentation of dt_final.py

This script is an implementation of the ID3 algorithm that creates a decision tree for data classification.

Input & Output

Upon running of the script, it requires the input data file name. The user can just type the name of the file if it is in the same directory of the script or the absolute path of the file if it is in different location.

The user has also to supply a name for the desired xml output file.

Structure

The code is structured in 5 functions.

- **entropy(S)**

entropy is a function that takes a set/subset of the training examples S and returns a list contains the total count of the instances in the set, the entropy and the counts of different classes in the set.

```
In [2]: ent = entropy(formatted_data)
In [4]: ent
Out[4]: [1728, 0.6028704850060876, [1210, 384, 69, 65]]
```

- **partition(S, A):**

partition is a function that takes a set/subset of the training examples and attribute A and returns a dictionary its keys are the values of this attribute and the values of these keys are lists of the subsets of the data partitioned by the values of this attribute A.

```
In [10]: sets = partition(formatted_data, 'buying')
In [11]: type(sets)
Out[11]: dict
In [13]: sets.keys()
Out[13]: ['high', 'med', 'vhigh', 'low']
```

- **gain(S, A):**

'gain is a function that takes a set/subset of the training examples and attribute A and returns the information gain from using this attribute as a node.

```
In [14]: gain(formatted_data, 'buying')
Out[14]: 0.04822448458480699
In [15]: gain(formatted_data, 'safety')
Out[15]: 0.13109217827713193
```

- **best_attribute(S, attributes_list):**

best_attribute is a function that takes a set/subset of the training examples and an attribute list and returns the best of these attributes to partition the data upon.

```
In [16]: best_attribute(formatted_data,attributes_list)
Out[16]: 'safety'
```

- **id3(S, candidates, xml):**

id3 is the main function that takes set/subset of the training examples, a list of candidate attributes, XML node and returns new xml node in the XML tree as a child to the node it receives as an input. The new xml node should contain the new best attribute, one of its values, the entropy of the new set and the classes it has. If it is a leaf node so this node will contain in its text part the final class. It is a recursive function so it keep calling itself until it reaches a leaf node either because of perfect classification (the entropy = 0) or it doesn't have candidate attributes left so it returns a leaf node based on the majority vote.