

Question 2

Incorrect

Mark 0.00 out of 1.00

Update your DTNode to have a new method `leaves` that returns the number of leaves in the decision tree.

This method is not needed for the rest of the quiz (e.g. for tree training) but it is a useful function for debugging and studying the size of the learnt trees in various conditions.

For example:

Test	Result
<code>n = DTNode(True)</code> <code>print(n.leaves())</code>	1
<code>t = DTNode(True)</code> <code>f = DTNode(False)</code> <code>n = DTNode(lambda v: 0 if not v else 1)</code> <code>n.children = [t, f]</code> <code>print(n.leaves())</code>	2

Answer: (penalty regime: 0, 10, 20, ... %)

```

1 class DTNode:
2     """
3     Node for a decision tree used as both
4     1. decision, and
5     2. leaf nodes.
6     """
7
8     def __init__(self, decision, children=[]):
9         """
10        A DTNode object must be initialisable with a decision,
11        :param decision:
12            1. either a function that takes an object (typically a feature vector) and
13               indicates which child should be followed (when the object is node is a decision node);
14            2. or a value which represents the classification or regression result (when the object is a leaf node).
15        :param children:
16            set to a data structure that maps the output of the decision function to a specific child.
17            We assume the output of the decision function is an index into a list.
18        """
19        self.decision = decision
20        self.children = children
21
22    def leaves(self):
23        if not callable(self.decision):
24            return 1
25
26        else:
27            return sum([child_node.leaves() for child_node in self.children])
28
29    def predict(self, input_object):
30        """
31        recursive method
32        :param input_object:
33            (e.g. a feature vector).
34            If it's a leaf node, the input can be anything. It's simply ignored
35        :return:
36            result of the decision tree for that input.
37        """
38
39        if not callable(self.decision):
40            return self.decision
41
42        else:
43            i = self.decision(input_object) # function that indicates index of which child to followed
44            child_node = self.children[i] # maps the output of the decision function to a specific child.
45            return child_node.predict(input_object)

```

Precheck

Check

	Test	Expected	Got	
✓	<pre>n = DTNode(True) print(n.leaves())</pre>	1	1	✓
✓	<pre>t = DTNode(True) f = DTNode(False) n = DTNode(lambda v: 0 if not v else 1) n.children = [t, f] print(n.leaves())</pre>	2	2	✓
✓	<pre>tt = DTNode(False) tf = DTNode(True) ft = DTNode(True) ff = DTNode(False) t = DTNode(lambda v: 0 if v[1] else 1) f = DTNode(lambda v: 0 if v[1] else 1) t.children = [tt, tf] f.children = [ft, ff] n = DTNode(lambda v: 0 if v[0] else 1) n.children = [t, f] print(n.leaves())</pre>	4	4	✓

Testing was aborted due to runtime errors.

One or more hidden tests failed.

Show differences

Incorrect

Marks for this submission: 0.00/1.00. This submission attracted a penalty of 0.10.