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

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

```python
class DTNode:
    """
    Node for a decision tree used as both
        1. decision, and
        2. leaf nodes.
    """

    def __init__(self, decision, children=[]):
        """
        A DTNode object must be initialisable with a decision,
        :param decision:
            1.  either a function that takes an object (typically a feature vector) and
                indicates which child should be followed (when the object is node is a decision node);
            2.  or a value which represents the classification or regression result (when the object is a leaf node).
        :param children:
            set to a data structure that maps the output of the decision function to a specific child.
            We assume the output of the decision function is an index into a list.
        """
        self.decision = decision
        self.children = children

    def leaves(self):
        if len(self.children) == 0:
            return 1

        else:
            return sum([child_node.leaves() for child_node in self.children])

    def predict(self, input_object):
        """
        recursive method
        :param input_object:
            (e.g. a feature vector).
            If it's a leaf node, the input can be anything. It's simply ignored
        :return:
            result of the decision tree for that input.
        """

        if not callable(self.decision):
            return self.decision

        else:
            i = self.decision(input_object)  # function that indicates index of which child to followed
            child_node = self.children[i]  # maps the output of the decision function to a specific child.
            return child_node.predict(input_object)
```

Precheck   Check

	Test	Expected	Got	
✔	`n = DTNode(True)` `print(n.leaves())`	1	1	✔

✔	`t = DTNode(True)`
	`f = DTNode(False)`
	`n = DTNode(lambda v: 0 if not v else 1)`
	`n.children = [t, f]`
	`print(n.leaves())`
		2	2	✔

✔	`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())`
		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 0.20.