CISC 440 - Artificial Intelligence & Robotics
Spring 2020

Assignment 08: Due Tuesday, 5/12 by 11:59pm

For this assignment, you will implement a decision tree that identifies what I'm having for dinner. You will create the tree using the "most important attribute" algorithm described in the book/slides/lecture/etc.

Begin by forking and cloning the provided assignment as described in the "Assignment Setup and Submission" document in Canvas. You will need to finish the implementation of the DecisionTree class.

WhatsForDinner

The possible decisions are specified in the dinner.model.Decision enum. They include restaurant, pizza, leftovers, a big recipe, a small recipe, scraps, and fast food.

To create the tree, your program will read in a list of examples, which are modeled using the dinner.model.Example class. Each example instance contains values for each of the attributes and the corresponding decision that was made.

Attributes are modeled using the dinner.model.Attribute enum. They include the day of the week, if we have leftovers, if we're sick of leftovers, if we're really hungry, if we have pizza, if we already ate pizza this week, and if we're tired.

Examples are analogous to rows in the table on slide 15. Attributes are analogous to the inner columns in that table. Decisions are analogous to the final column in that table.

DecisionTree (dinner.learn)

getMostImportantAttribute(examples : List<Example>, attributes : List<Attribute>) : Attribute

This static method identifies and returns the attribute (present in the **attributes** list) that decides the most examples in **examples**.

It should use the provided **generateDecisionMap** method, which, given a list of examples and an attribute, returns a map where:

- each key is a different value of the given attribute (or branch of the tree), and
- each key's corresponding value is a map, where
    - each key is a different decision, and
    - each key's corresponding value is the number of examples with the matching attribute value and decision

This method should also use the provided **getNumOutcomesDecided** method, which takes in the map returned by **generateDecisionMap**, and returns the number of examples that are decided. For an example, see the tree on the right in slide 15. If this method were given a map with the "Patrons" attribute, it would return 6, since 6 of the 12 examples are decided (those following the "None" and "Some" branches).

createDecisionTree(examples : List<Example>, attributes : List<Attribute>) : void

My implementation of this method calls a recursive helper method named **createDecisionTreeHelper**, that includes two additional parameters: the name of the parent node and the name of the incoming branch. This recursive helper method identifies the nodes in the tree in a depth-first order by printing each node, incoming branch, and parent, in the format seen below. The following output corresponds with the tree on slide 17.

Node: Patrons?

Parent: Patrons?
Branch: None
Decision (2): No

Parent: Patrons?
Branch: Some
Decision (4): Yes

Parent: Patrons?
Branch: Full
Node: Hungry?

Parent: Hungry?
Branch: No
Decision (2): No

Parent: Hungry?
Branch: Yes
Node: Type?

Parent: Type?
Branch: French
Decision (0): Yes

Parent: Type?
Branch: Italian
Decision (1): No

Parent: Type?
Branch: Thai
Node: Fri/Sat?

Parent: Fri/Sat?
Branch: No
Decision (1): No

Parent: Fri/Sat?
Branch: No
Decision (1): Yes

Parent: Type?
Branch: Burger
Decision (1): Yes

Odds and Ends

Your DecisionTree.java file will be used along with the other provided files when grading. Your grade be determined largely by the number of tests passed in DecisionTreeTest.java. While these tests are a good indicator of correctness, they do not guarantee a fully correct implementation. In other words, passing all tests may not translate to a 100% on the assignment.

Submit your assignment by pushing your code to your gitlab repository fork by the deadline.