**Programming Assignment #2 Report**

Computer Science and Engineering

2010003970

Namwoo Kim

1. **Summary of algorithm**

This program build decision tree by using information gain method as an attribute selection. The basic implemented structure is based on pseudo code from the textbook.

First, calculate information gain value from the current data set If we have a remaining attribute in this part of tree. The information gain value is based on some part of data set in some part of subtree. I’ve added another constraint for stop to build a tree deeper. This builder will stop when there’s no attribute to examine or all class label is same.

Second, find distinct values from selected attribute to split data set into several groups.

Third, keep building decision tree with split data set for each attribute.

Fourth, return root node of the decision tree If there’s no attribute for building tree or remaining data set.

1. **Detailed description of my codes**

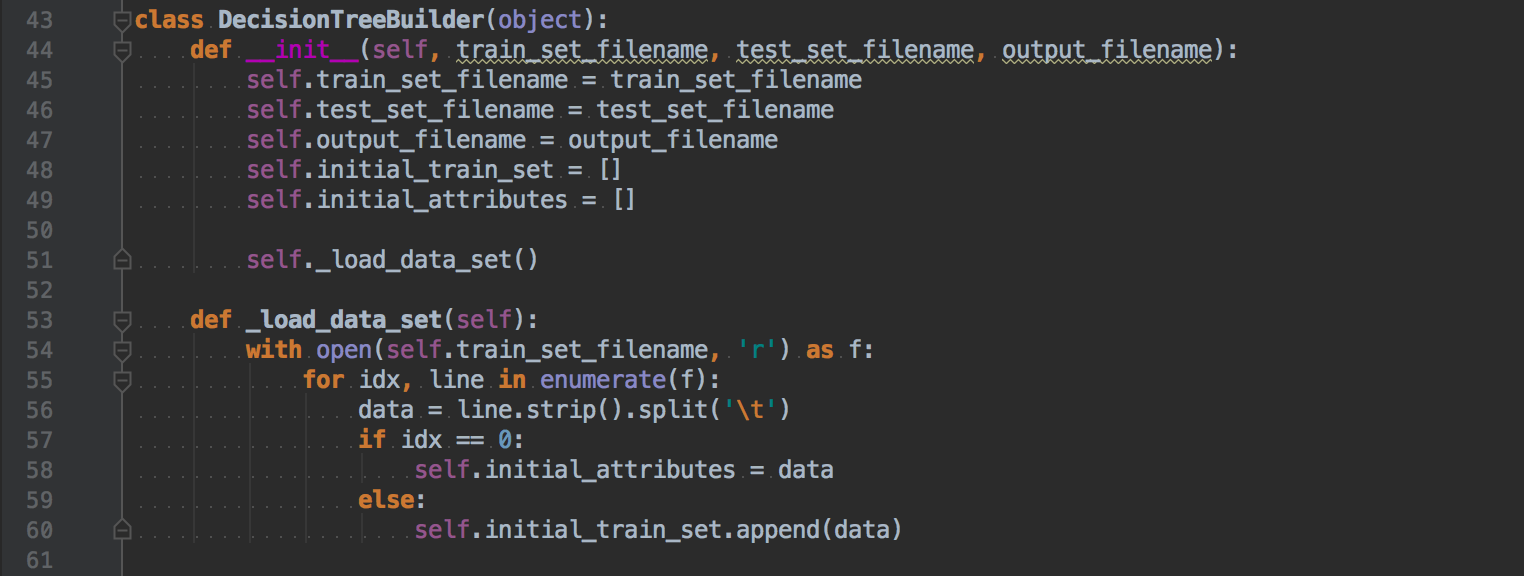
**<basic data structure>**

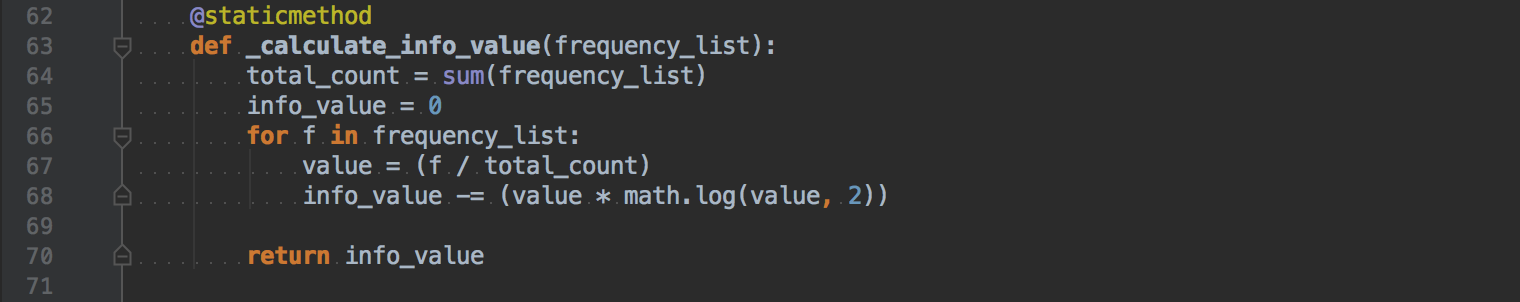
There are three types of node structure. Attribute node represents a single attribute which has several criteria for splitting training sets. Criteria node stores four types of data such as criteria name, remaining data set count, attribute node and class node. Attribute in criteria node is a selected major attribute which has high information gain value. Class node in criteria node is a result which has class label. Last, Class node is a single data structure who keeps label.

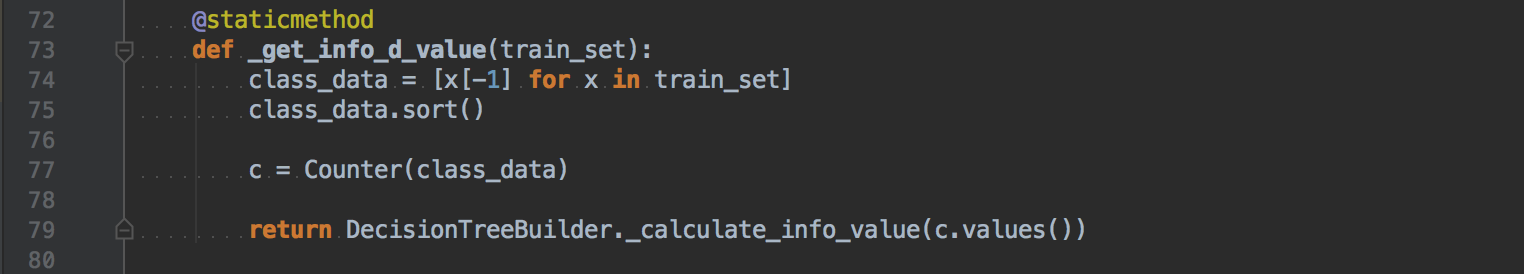
<screenshot below>

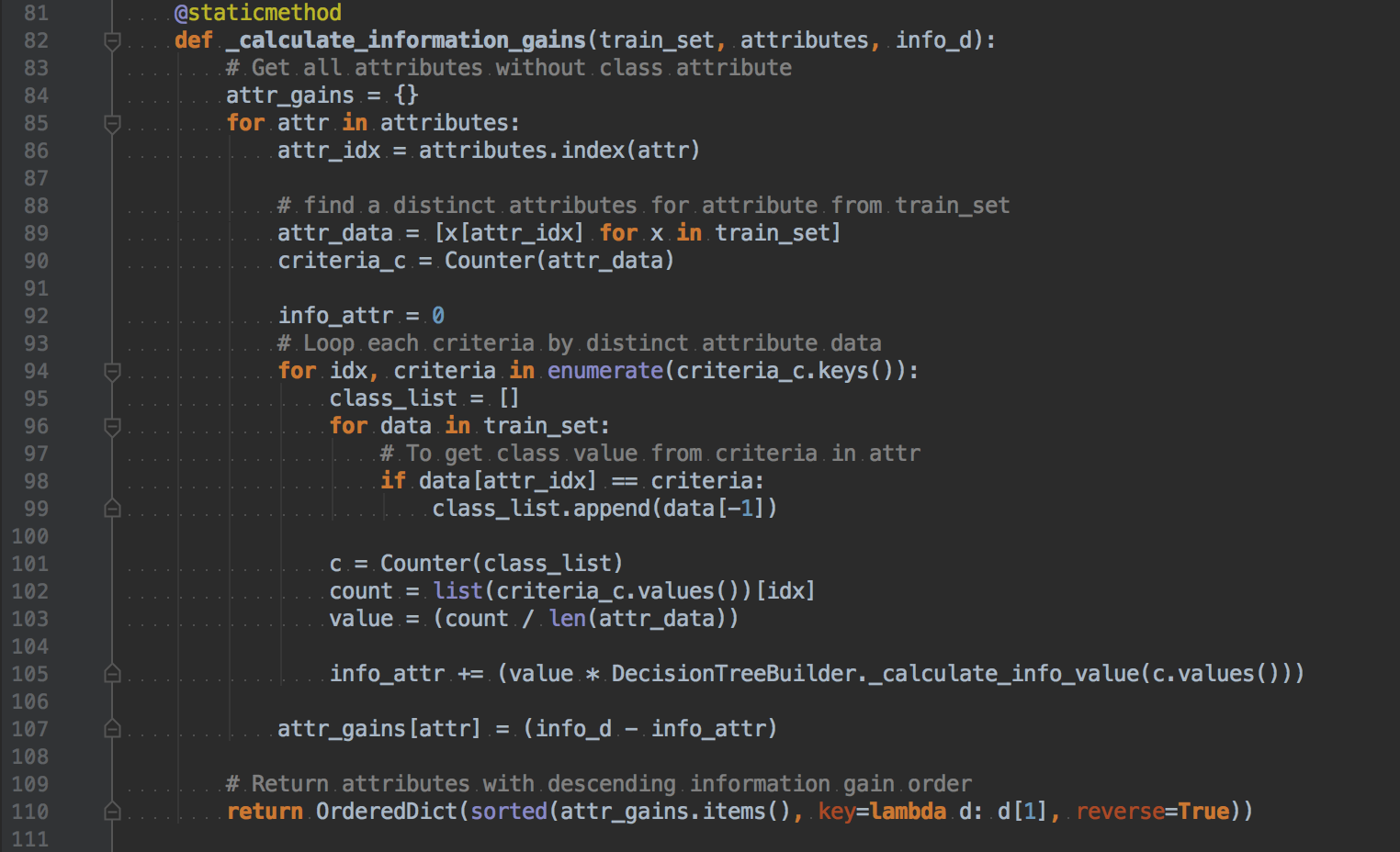


**<Initializer part>**



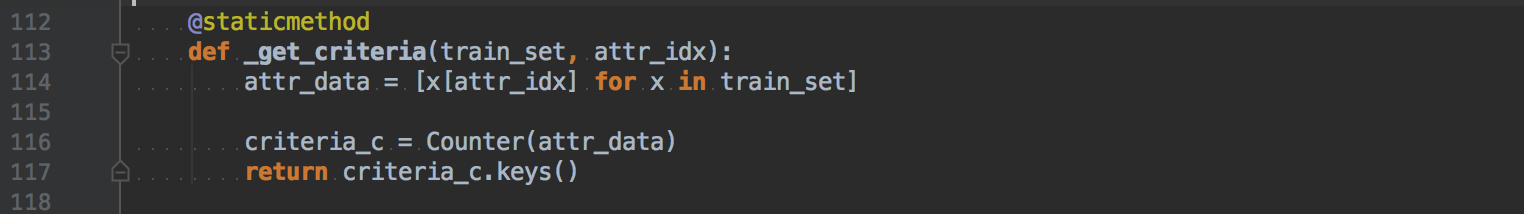
This is an initial function for DecisionTreeBuilder class. Function named \_load\_\_data\_set read train set files and add initial data into class variable (self.initlal\_train\_set / self.initial\_attributes).  
  
**<static function: \_calculate\_\_info\_value>**

This function calculates information “I(x,y)” value for given frequency\_list. (Line 63) Frequency list contains x Yes’s / y No’s and generate I(x,y) for calculating Info(attribute). (Line 65~70) To calculate Info(attribute), using this function for each criteria of attribute. Assume that you have two criteria, one consists of 3/10 which has I(1,2) and the other consists 7/10 which has I(6,1). Then looping for two criteria and call \_calculate\_\_info\_value([1, 2]) and \_calculate\_\_info\_value([6,1]) then using result to calculate Info(attribute) => 3/10 \* I(1,2) + 7/10 \* I(6,1).  
  
**<static function: \_get\_\_info\_d\_value>**  


This function calculates Info(D) for given train set. (Line 73) Class\_data list contains class values and sort to generate frequency list using Counter(class\_data) feature. (Line 74~77) C.value() contains frequency for each class variables. (Line 79) If we assume there’re three class with frequency as follows 7, 2, 6. C.values() represents [7,2,6] as a frequency values for each class. And then using function \_calculate\_\_info\_value which described above to calculate Info(D).  
  
**<static function: \_calculate\_\_information\_gains>**  


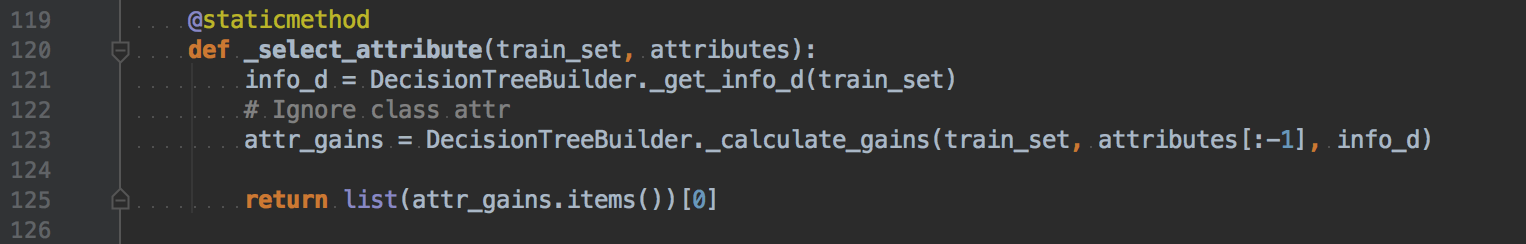
This function calculates information gain for all attributes as given in parameter “attributes” from given “train\_set” and given info(D) value. (Line 82) First iterating all attributes and get criteria from the train set. (Line 85~90) Then looping for criteria to calculate I(x,y) for each criteria. (Line 94~105) After that generate information gains with given Info(attribute) value. (Line 107)

**<static function: \_get\_\_criteria>**



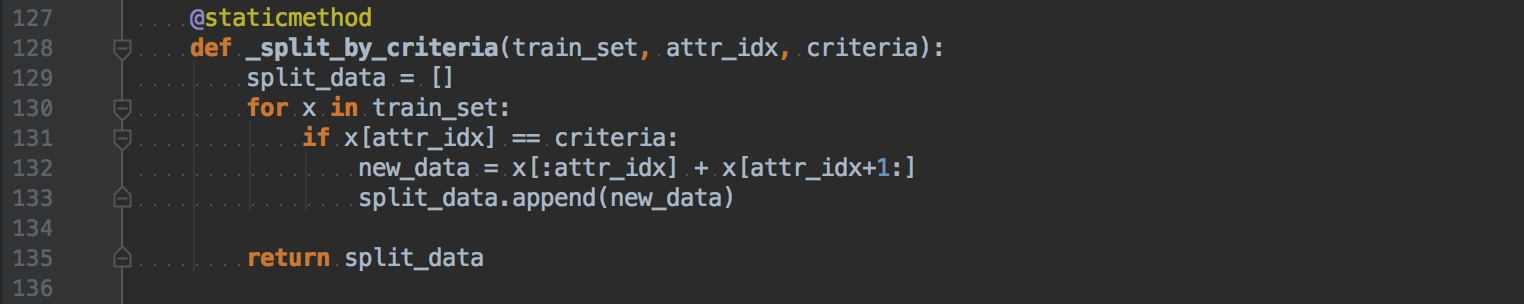
This function distinct values of specific attribute from given train set. Among train set, extract (attr\_idx)th attribute value from each row. (Line 114) Then, using Counter feature to generate distinct values. (Line 116~117) criteria\_c.keys() contains distinct values only.

**<static function: \_select\_\_attribute>**



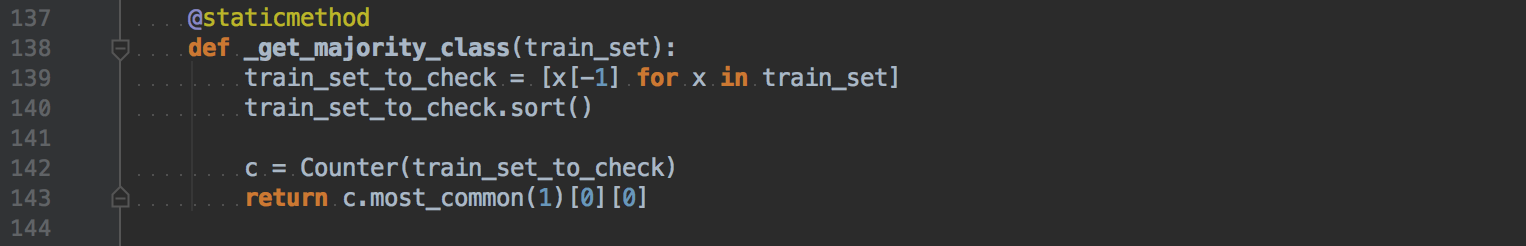
This function select attribute to build tree by choosing the highest information gain value from given train set and attributes. First, calculate Info(D) using given train set. (Line 121) Then, calculate all Info(attribute) using \_calculate\_\_gains function. (Line 123). The result will be (attribute name, information gain value) pairs for given attributes (Line 125)

**<static function: \_split\_\_by\_criteria>**



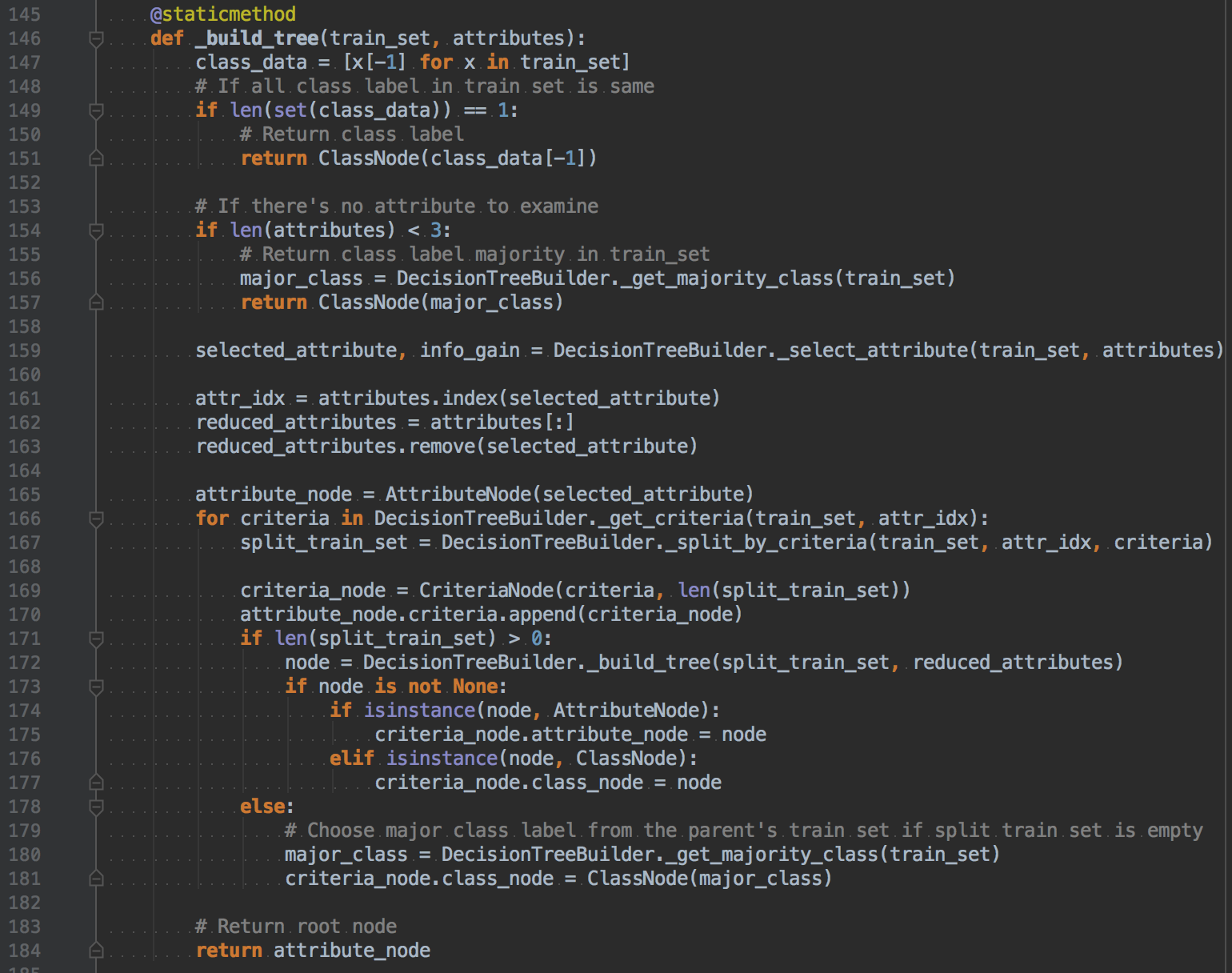
This function reduces train set by given train set, criteria and attribute index. (Line 128). First, retrieving all data from train set then check whether (attr\_idx)th attributes is same as criteria value.(Line 130~131) If so, we generate new data set excluding (attr\_idx)th attribute (which is same as criteria). (Line 132)

**<static function: \_get\_\_majority\_class>**



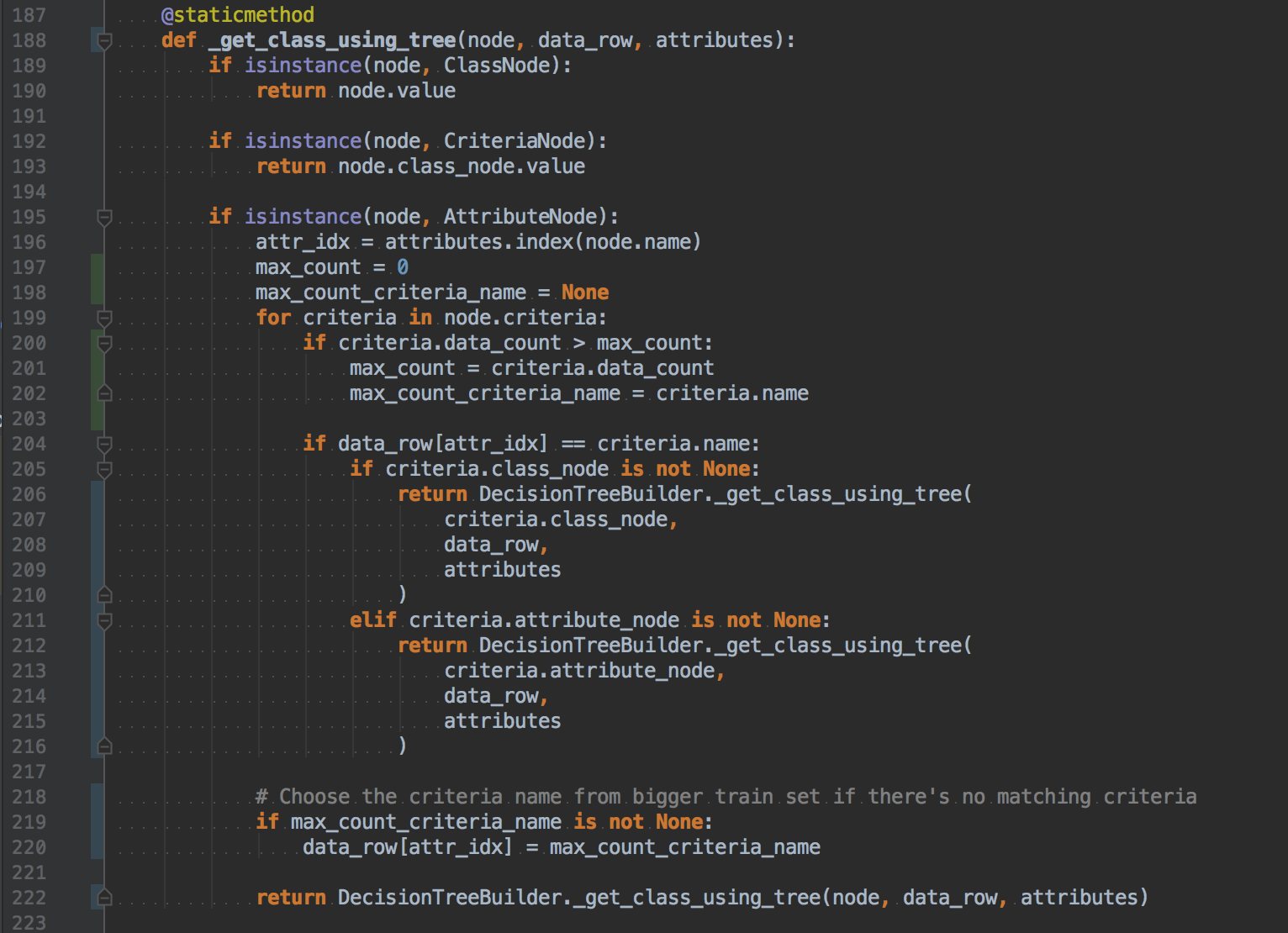
This function return most common class value among the train set If there’s no attribute to calculate information gain value. The list named train\_set\_to\_check contains all class values from given train set. (Line 139) Then, using Counter() function to get most common value(which is a class label). (Line 142~143)

**<static function: build\_tree>**

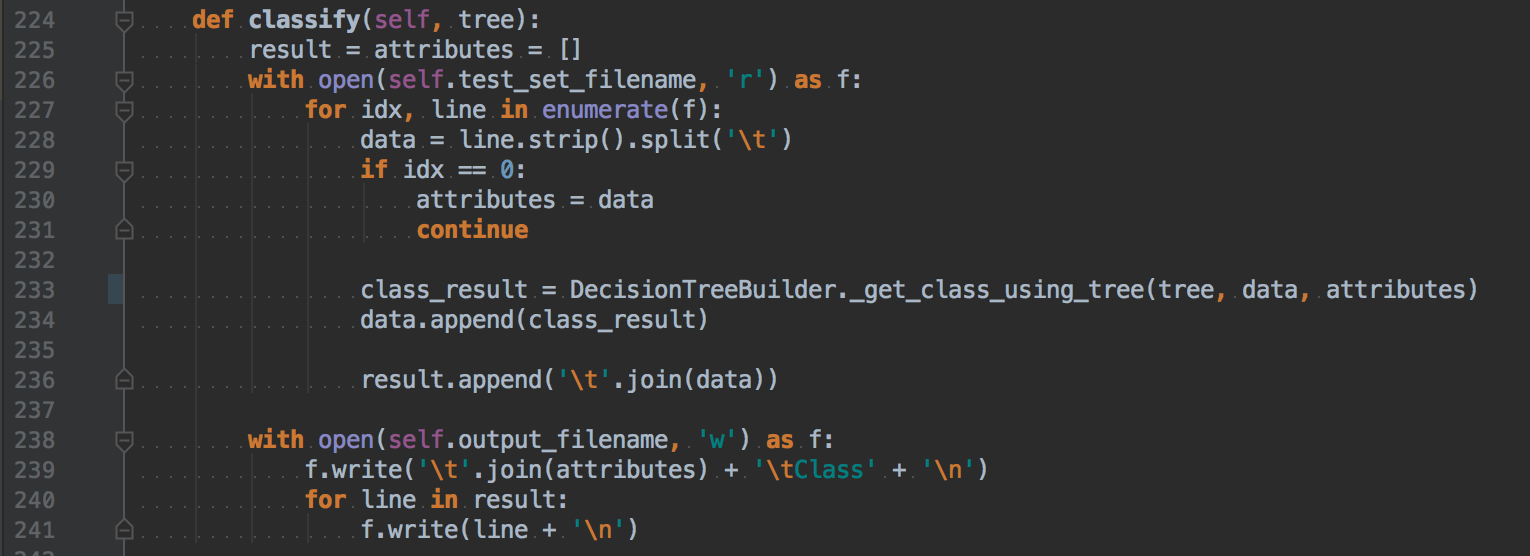


This function is a main part of the building decision tree. This function simply returns one of the nodes(Attribute node, Criteria node, Class node).  
First, we don’t have to run remain function call If all class data is same and then return final class label. (Line 147~151). Second, just return the most common class value from the train set if there’s no attribute to build another subtree. (Line 153~157). Third, we can calculate information gain value to select attribute among given attributes. (Line 159) Fourth, get criteria from selected attribute and then make a train set for each criteria value. (Line 166~170) Fifth, recursively call \_build\_\_tree function by using split train set and reduced attributes If there’re enough data to build subtrees. (Line 171~177) Finally, return an attribute node which is a root of the decision tree. (Line 184)

**<static method: \_get\_\_class\_using\_tree>**



This function generates class label for given data row. First, it will return class label if given node is a Class node or Criteria node which has labeled class. (Line 189~193) Second, compare with data row and the criteria from given decision tree and return class label value or getting deeper by calling test\_tree function with given attribute value in criteria node. (Line 195~216) Finally, if there’s no matching with criteria value and given data row. I just choose one of the criteria node from the biggest train set among current criteria to get class label value (Line 218~220)  
  
  
  
  
  
  
  
**<class function: classify>**

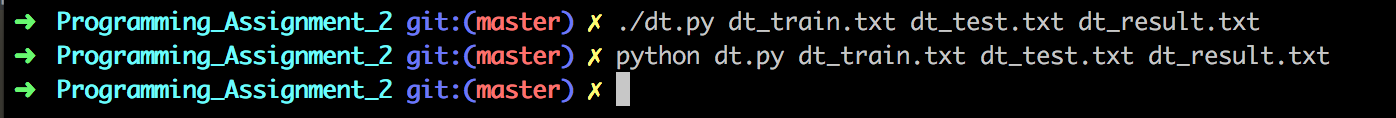


This function is simply get class label by giving built decision tree and data row. After that, write a result file with having class label values.

1. **Instructions for compiling my source code at TA’s computer**

This program tested with Python 2.7 and 3.6. You can easily run this program in any other environments which has python because python program doesn’t need to compile for run.

In mac OS, there is a preinstalled Python.

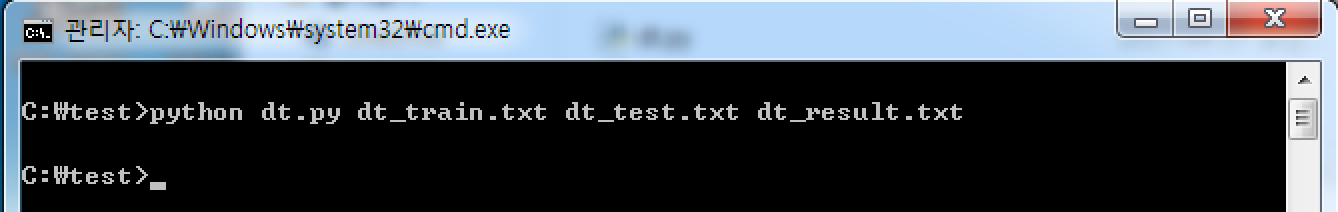


OK for both ways.

In windows, Install python 2.7 first and then execute this in your shell.

setx PATH "C:\Python27;C:\Python27\Scripts;C:\Python27\Lib\site-packages"

This is the most important part. Adding a PATH variable in the shell and quit and re-open terminal. If you enter python in terminal and It works, you can run my program.



1. **Any other specification of my implementation and testing**

I’ve tested both mac OS and Windows environments and successfully tested with the program that TA given to me. I think there’s nothing to mention for any specifications.