**Programming Assignment 2**

2014005014

Jimin Lee

**1. Environment**

* OS: Windows
* Language: Python 3.7

**2. Goal** : Build a **decision tree**, and then classify the test set using it.

**3. Representation of decision tree**

* Each internal node tests an attribute variable.
* Each branch corresponds to an attribute value.
* Each leaf node assigns a classification value.

**4. Implementation of decision tree**

* **Basic algorithm**
* Tree is constructed in top-down recursive divide-and-conquer manner.
* The data are partitioned recursively based on selected attribute.
* Test attributes are selected on **information gain**.
* At start, all the training data are at the root.
* Node is splitted recursively by the selected attribute and has children corresponds to an attribute value.
* Also, each splitted nodes have the partitioned data.
* **Conditions for stopping partitioning and classification**
* If all samples for a given node belong to the same class, the leaf is classified for the class.
* If there are no remaining attributes for further partitioning, majority voting is employed for classifying the leaf.
* **Prediction**
* Starting with root, visit recursively the child node according to the attribute value of the test tuple.
* If the node is leaf, classify by the classification value.
* If the node has no branch corresponds to the attribute value, determine using majority voting.

**4. Description of code**

* **DecisionTree class**
* Member
* *data*

- store training data using 'dict' data type

- the key is each attribute variable of training data and class label.

- the value is list of the corresponding attribute value of each tuple.

- ex) { ‘age’:[‘<=30’,’ 31...40’…], ‘income’:[‘high’,’ low’…], ‘student’:[‘no’,’no’…], … ‘Class:buys\_compute’:[‘no’,’yse’…] }

* *target*

- store the class label

* *depth*

- current depth of the node

* *attribute*\_var

- selected attribute for splitting

* *classification*\_val

- classification value

* *entropy*

- entropy for class variable in data the node has

* *children*

- store child nodes using 'dict' data type

- the key is attribute value and the value is corresponding node

* *isLeaf*

- if it is True, the node is leaf

* Method
* **split(self,depth)**

- split the node recursively and set depth

- if 'self.entropy == 0' that means all samples for a given node belong to the same class, the recursion stopped.

- if self.depth == max\_depth' that means there are no remaining attributes for further partitioning, the recursion stopped.

- search in the remaining attribute for best splitting

- generate splitted data for the number of attribute value types with the selected attribute

- each splitted data is a subset of self.data that contains the attribute value and has no the attribute variable

- generate child nodes for splitted data and call split() function in children

* **\_\_search\_best\_split(self)**

- search an attribute in the remaining for best splitting using information gain

- return the attribute if it makes the highest information gain

* **\_\_information\_gain (self,*a*)**

- return information gain of the argument *a* in self.data

* **predict(self,test)**

- if the node is leaf, return self.classification\_val

- if the node has no branch corresponds to the attribute value, return most common value

- visit recursively the child node according to the attribute value of the test tuple.

* **entropy(c\_data)**

**-** c\_data is list of class value

**-** calculate entropy in c\_data

- Entropy =

* **read\_file(path)**

- receive training file name as an argument

- read the file and create ‘dict’ data

- set max\_depth that is global value to (the number of attributes – 1)

- return ‘dict’ data and string of class label

* **create\_result(test\_path,result\_path,dt,target)**

**-** receive test file name, result file name, root node, and target

**-** classify test file and write result file

* **main()**

**-** receive training file name, test file name and result file name as an argument

**-** call read\_file() and store train\_data

**-** set initial\_depth to 0

**-** create object of DecisionTree class

**-** call split () method and start initial\_depth

**-** create result file by create\_result ()

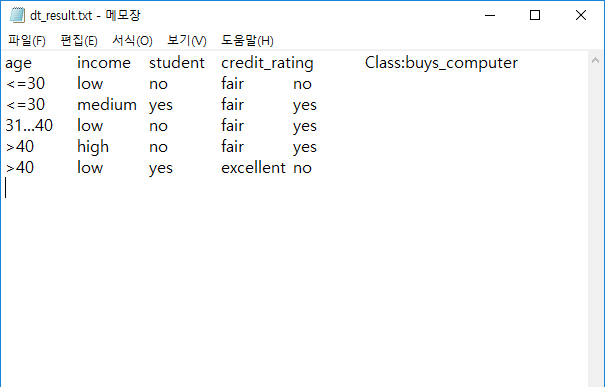
**5. Execution**

**Ex) dt\_train.txt**

**[Command line]**

****

**[dt\_result.txt]**

****

**[Testing program]**

****

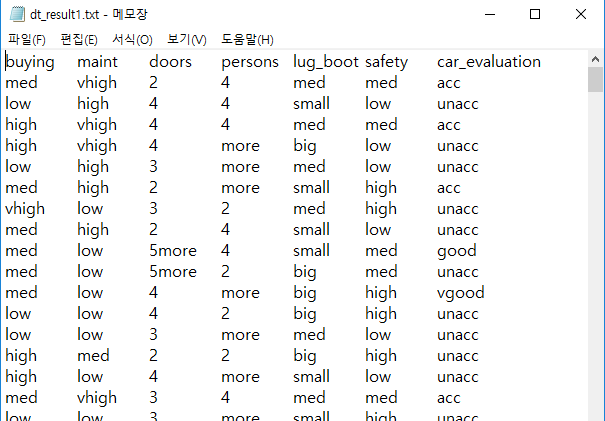
**Accuracy rate: 100%**

**Ex) dt\_train1.txt**

**[Command line]**

****

**[part of dt\_result.txt]**

****

**[Testing program]**

****

**Accuracy rate: 91.04%**