< Data Science (ITE4005) >

Programming Assignment#2 – Decision tree

소프트웨어 전공

2013010926 강은석

1. Introduction

: This is an assignment that uses one of methods 'information gain, gain ratio, gini index' for making decision tree. So I select 'Information Gain' algorithm that is based by entropy theory. I think that entropy theory is well express purity. and making decision tree is process for finding purity value.

2. Summary of My algorithm

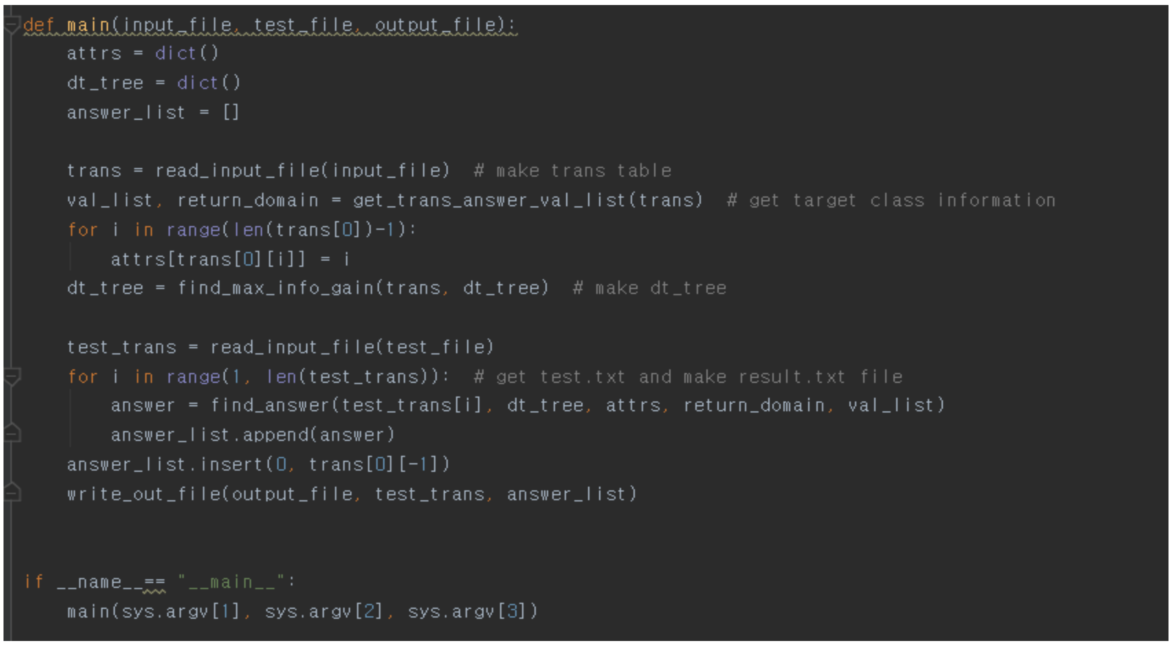
: Information gain theory is based by entropy theory. Entropy mean degree of impurity.

To compare the difference of impurity degrees between an original dataset S and its split sub set Ss.

Gain (S,A) = Entropy(S) – sigma |Ss|/|S|*Entropy(Ss) [sigma domain is attribute]

Entropy = sigma - P * log2P [P values of probability of class]

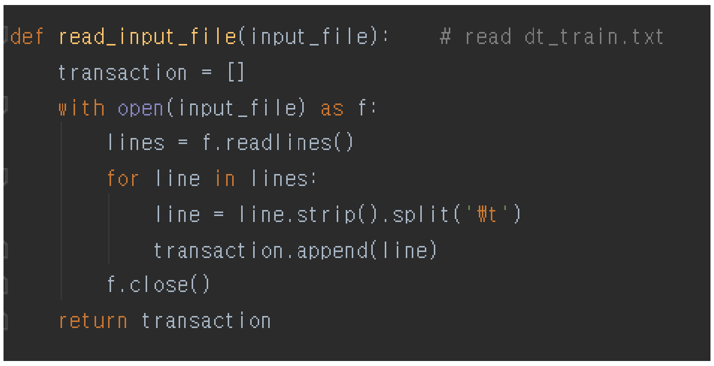
3. Description of codes



```python
def main(input_file, test_file, output_file):
    attrs = dict()
    dt_tree = dict()
    answer_list = []

    trans = read_input_file(input_file)  # make trans table
    val_list, return_domain = get_trans_answer_val_list(trans)  # get target class information
    for i in range(len(trans[0])-1):
        attrs[trans[0][i]] = i
    dt_tree = find_max_info_gain(trans, dt_tree)  # make dt_tree

    test_trans = read_input_file(test_file)
    for i in range(1, len(test_trans)):  # get test.txt and make result.txt file
        answer = find_answer(test_trans[i], dt_tree, attrs, return_domain, val_list)
        answer_list.append(answer)
    answer_list.insert(0, trans[0][-1])
    write_out_file(output_file, test_trans, answer_list)


if __name__ == "__main__":
    main(sys.argv[1], sys.argv[2], sys.argv[3])
```

그림 1 - main function

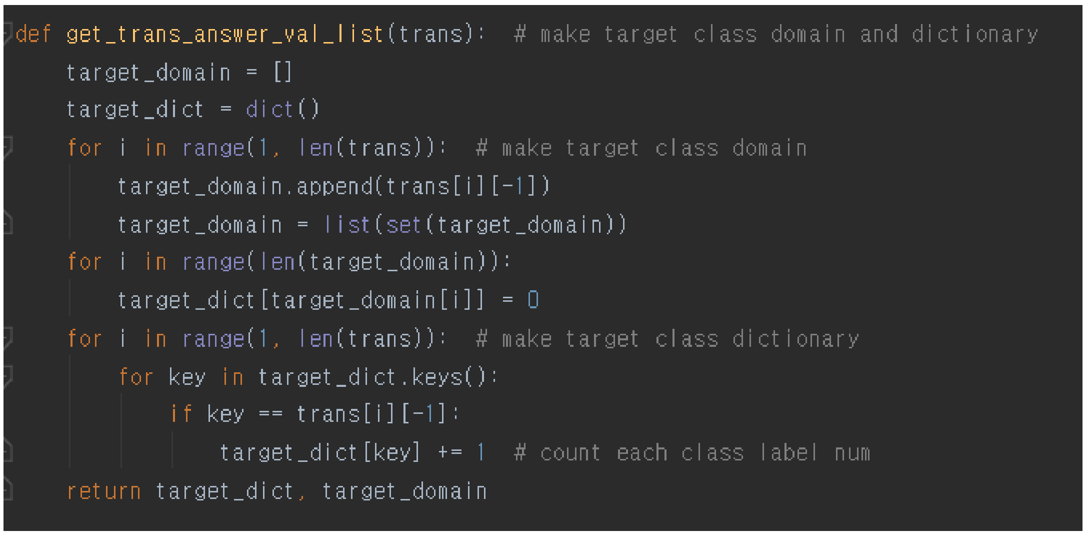
: Main function takes three arguments that are two input file name and one output file name. First. Read input file and make trans table that save train information. Second. Make decision tree through 'find_max_info_gain()' function. Then, Read test file and run on decision tree through 'find_answer()' function. Finally make output file that name is dt_result.txt.



```python
def read_input_file(input_file):    # read dt_train.txt
    transaction = []
    with open(input_file) as f:
        lines = f.readlines()
        for line in lines:
            line = line.strip().split('\t')
            transaction.append(line)
        f.close()
    return transaction
```

그림 2 - read txt file function

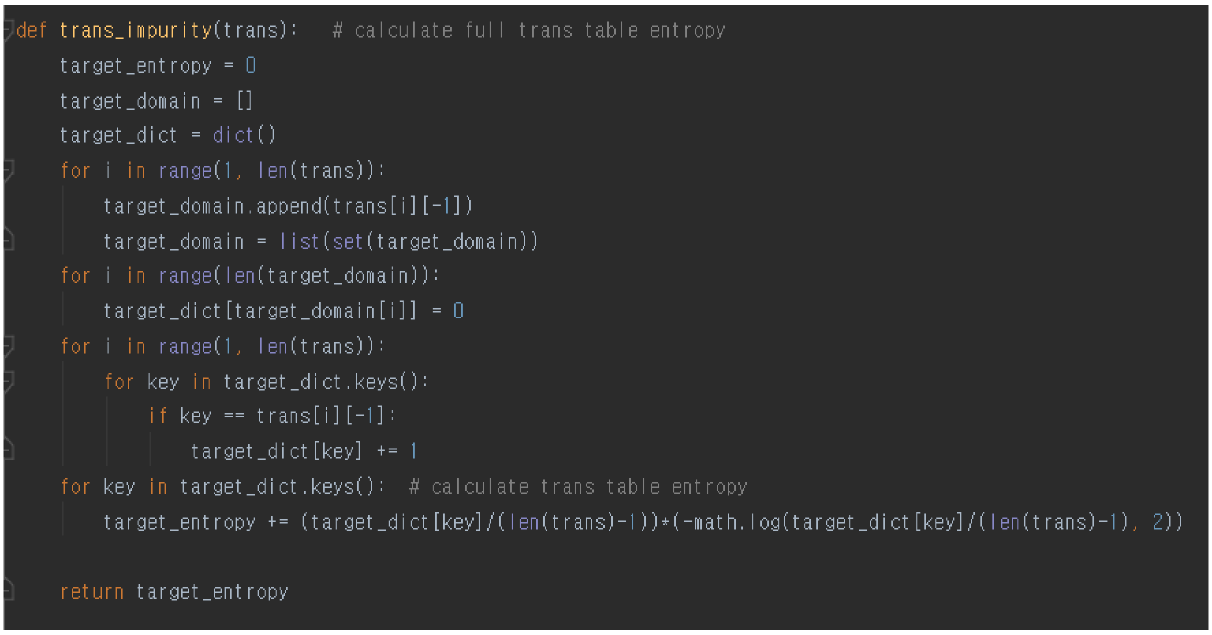
: This function is just read txt file. In this assignment, read 'dt_train.txt' and 'dt_test.txt' and return table.



```python
def get_trans_answer_val_list(trans):  # make target class domain and dictionary
    target_domain = []
    target_dict = dict()
    for i in range(1, len(trans)):  # make target class domain
        target_domain.append(trans[i][-1])
        target_domain = list(set(target_domain))
    for i in range(len(target_domain)):
        target_dict[target_domain[i]] = 0
    for i in range(1, len(trans)):  # make target class dictionary
        for key in target_dict.keys():
            if key == trans[i][-1]:
                target_dict[key] += 1  # count each class label num
    return target_dict, target_domain
```

그림 3 - get answer target class label and count function

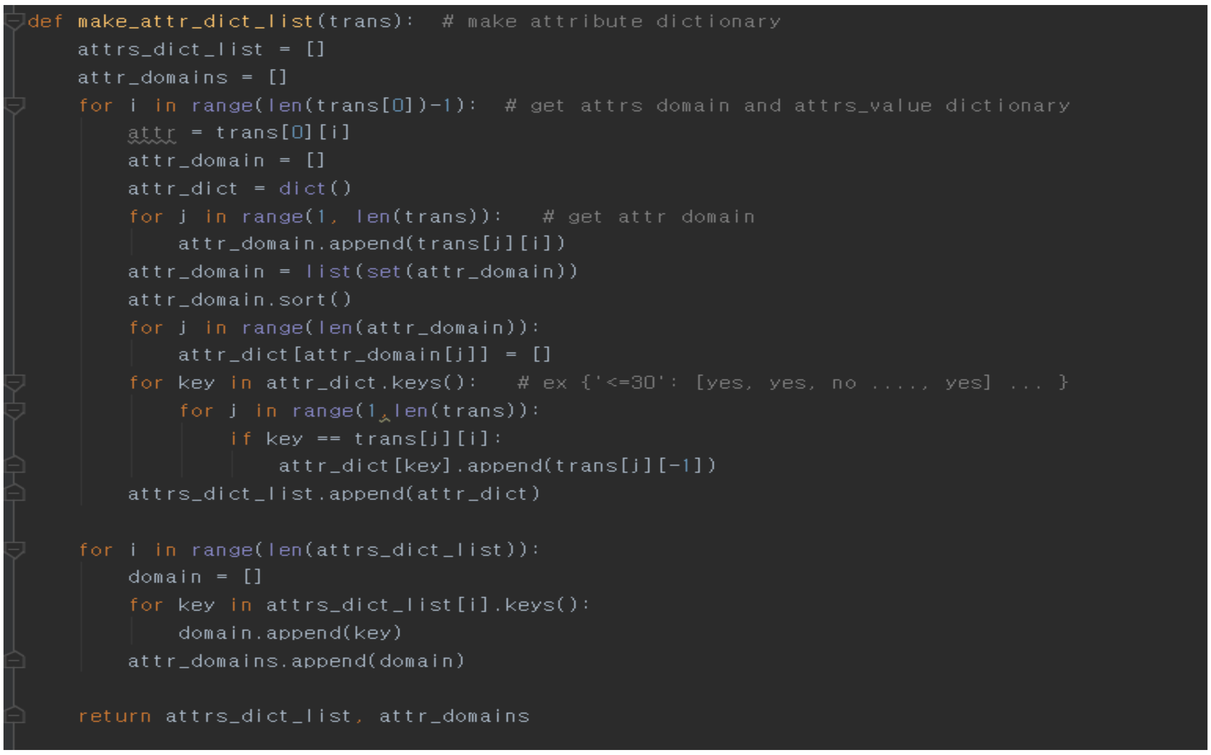
: This function read transaction table and get target class's value domain. And count amount of each target class's value. This return value is used for searching decision tree. (ex. In 'dt_train.txt', target class is "Class:buys_computer" and target class domain is ['yes', 'no'])



```python
def trans_impurity(trans):  # calculate full trans table entropy
    target_entropy = 0
    target_domain = []
    target_dict = dict()
    for i in range(1, len(trans)):
        target_domain.append(trans[i][-1])
        target_domain = list(set(target_domain))
    for i in range(len(target_domain)):
        target_dict[target_domain[i]] = 0
    for i in range(1, len(trans)):
        for key in target_dict.keys():
            if key == trans[i][-1]:
                target_dict[key] += 1
    for key in target_dict.keys():  # calculate trans table entropy
        target_entropy += (target_dict[key]/(len(trans)-1))*(-math.log(target_dict[key]/(len(trans)-1), 2))

    return target_entropy
```

그림 4 - calculate table's entropy function

: This function calculate target entropy about input "trans"(table). Count amount of each target class's value. And calculate 'target_entropy' based by information gain algorithm



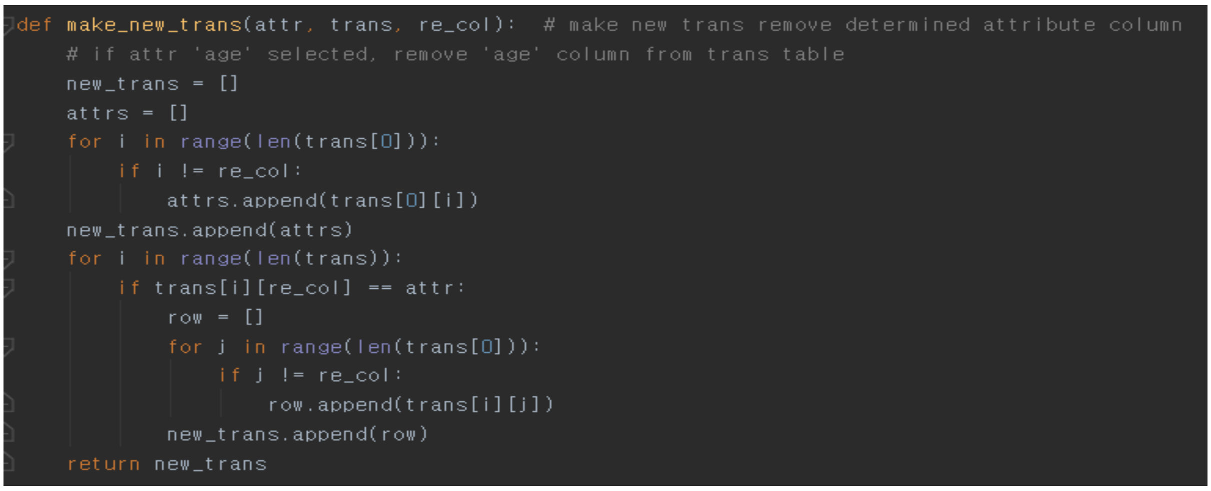
```python
def make_attr_dict_list(trans):  # make attribute dictionary
    attrs_dict_list = []
    attr_domains = []
    for i in range(len(trans[0])-1):  # get attrs domain and attrs_value dictionary
        attr = trans[0][i]
        attr_domain = []
        attr_dict = dict()
        for j in range(1, len(trans)):   # get attr domain
            attr_domain.append(trans[j][i])
        attr_domain = list(set(attr_domain))
        attr_domain.sort()
        for j in range(len(attr_domain)):
            attr_dict[attr_domain[j]] = []
        for key in attr_dict.keys():   # ex {'<=30': [yes, yes, no ....., yes] ... }
            for j in range(1,len(trans)):
                if key == trans[j][i]:
                    attr_dict[key].append(trans[j][-1])
        attrs_dict_list.append(attr_dict)

    for i in range(len(attrs_dict_list)):
        domain = []
        for key in attrs_dict_list[i].keys():
            domain.append(key)
        attr_domains.append(domain)

    return attrs_dict_list, attr_domains
```

그림 5 - get each attribute information function

: This function get each attribute information. This process is get each attribute's domain that are key of 'attr_dict'(dictionary)'s key. And 'attr_dict's value is related target class's value.

Ex. attribute 'Age' ➔ domain = ['<=30', '30..40', '>40']

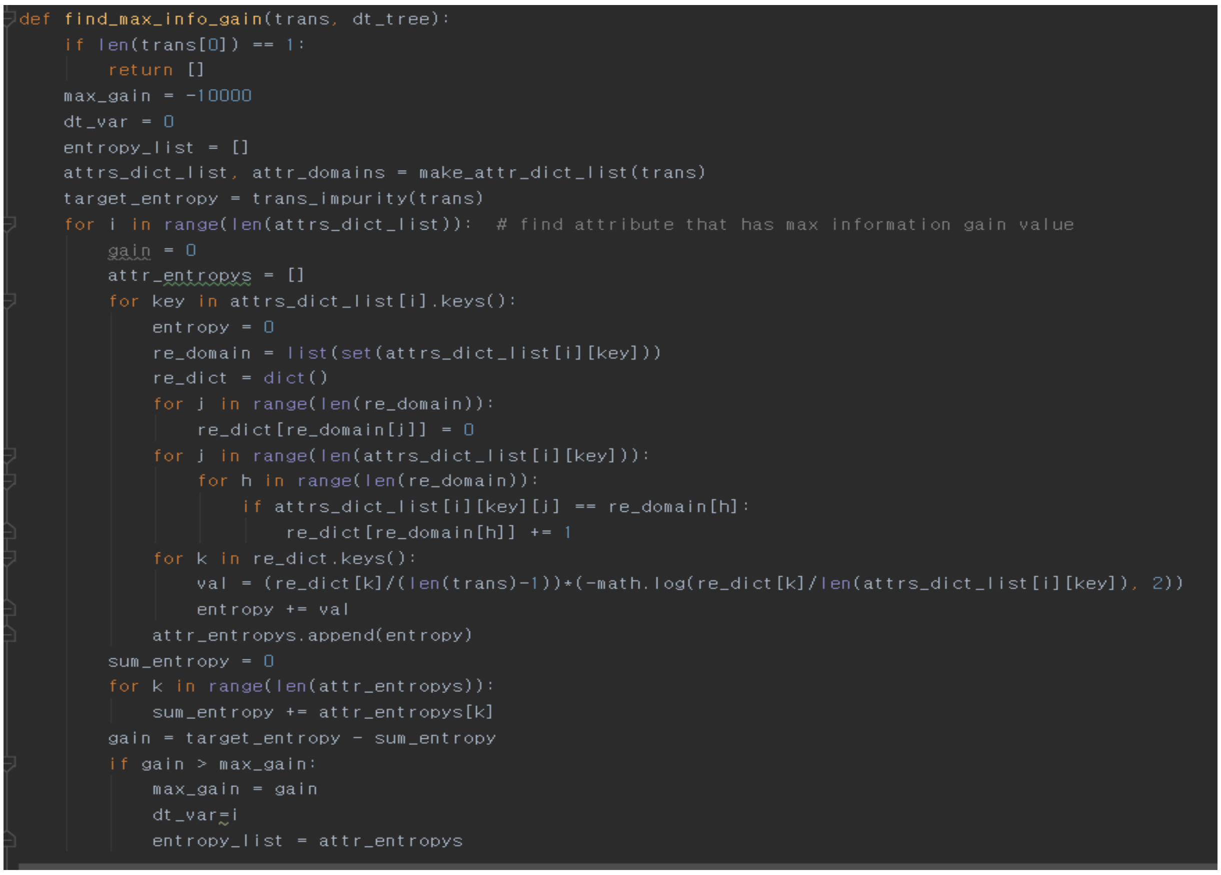
'attr_dict' ➔ {'<=30': [yes, yes, … yes], '30..40': [no, no …, yes], '>40': [yes, no…, yes]}



```python
def make_new_trans(attr, trans, re_col):  # make new trans remove determined attribute column
    # if attr 'age' selected, remove 'age' column from trans table
    new_trans = []
    attrs = []
    for i in range(len(trans[0])):
        if i != re_col:
            attrs.append(trans[0][i])
    new_trans.append(attrs)
    for i in range(len(trans)):
        if trans[i][re_col] == attr:
            row = []
            for j in range(len(trans[0])):
                if j != re_col:
                    row.append(trans[i][j])
            new_trans.append(row)
    return new_trans
```

그림 6 - make new trans table function

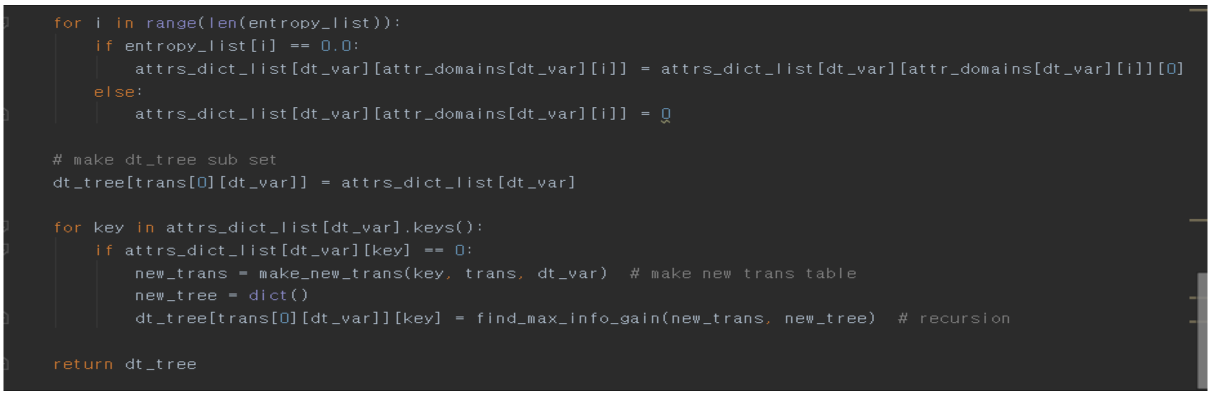
: This function removes selected column(re_col) in trans table.



```python
def find_max_info_gain(trans, dt_tree):
    if len(trans[0]) == 1:
        return []
    max_gain = -10000
    dt_var = 0
    entropy_list = []
    attrs_dict_list, attr_domains = make_attr_dict_list(trans)
    target_entropy = trans_impurity(trans)
    for i in range(len(attrs_dict_list)):  # find attribute that has max information gain value
        gain = 0
        attr_entropys = []
        for key in attrs_dict_list[i].keys():
            entropy = 0
            re_domain = list(set(attrs_dict_list[i][key]))
            re_dict = dict()
            for j in range(len(re_domain)):
                re_dict[re_domain[j]] = 0
            for j in range(len(attrs_dict_list[i][key])):
                for h in range(len(re_domain)):
                    if attrs_dict_list[i][key][j] == re_domain[h]:
                        re_dict[re_domain[h]] += 1
            for k in re_dict.keys():
                val = (re_dict[k]/(len(trans)-1))*(-math.log(re_dict[k]/len(attrs_dict_list[i][key]), 2))
                entropy += val
            attr_entropys.append(entropy)
        sum_entropy = 0
        for k in range(len(attr_entropys)):
            sum_entropy += attr_entropys[k]
        gain = target_entropy - sum_entropy
        if gain > max_gain:
            max_gain = gain
            dt_var=i
            entropy_list = attr_entropys
```

그림 7 - calculate attribute's information gain and get max gain attribute

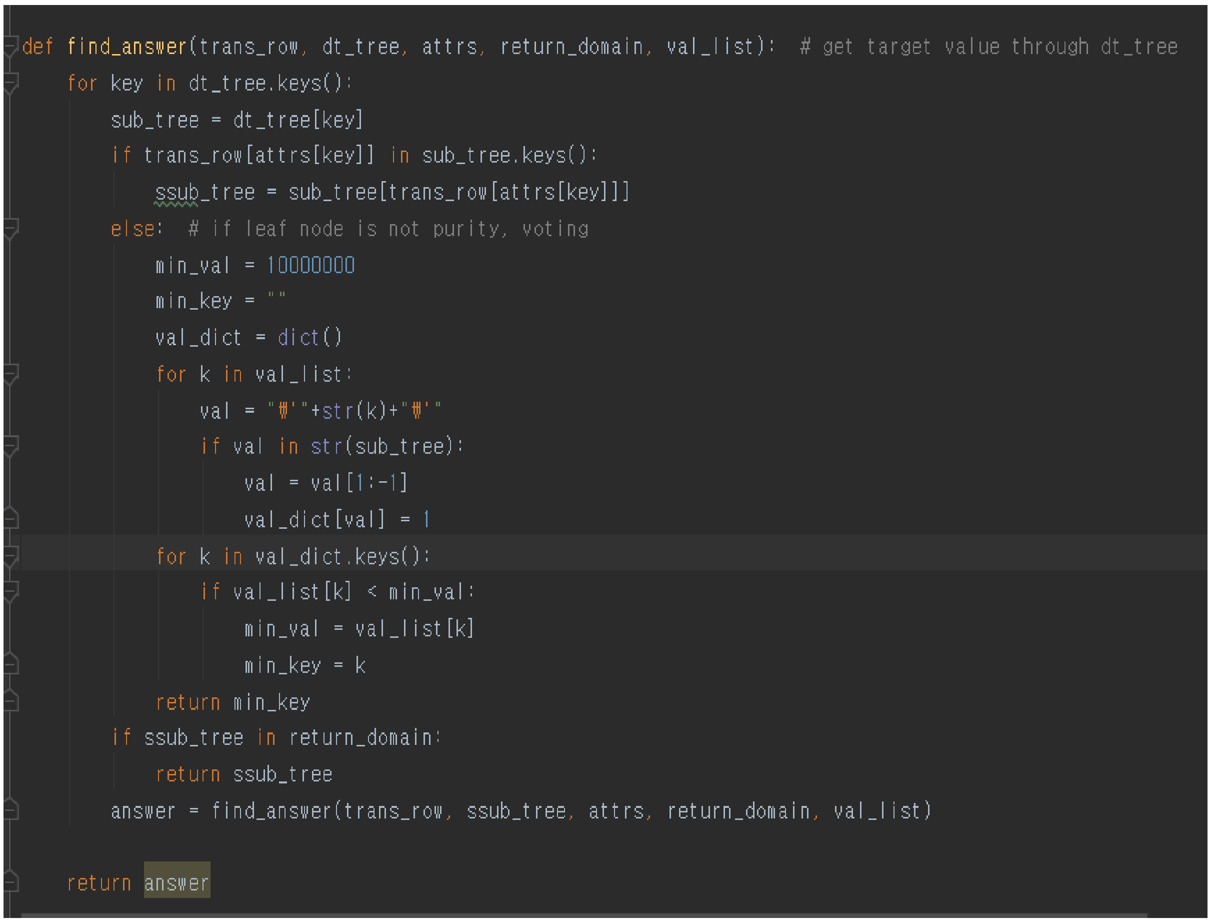
: This function calculates each attribute's information gain value and get maximum information gain attribute. In this function, I use "attr_domain_dict_list()" for attribute's dictionary and "trans_impurity()" for calculate trans table entropy.



```python
    for i in range(len(entropy_list)):
        if entropy_list[i] == 0.0:
            attrs_dict_list[dt_var][attr_domains[dt_var][i]] = attrs_dict_list[dt_var][attr_domains[dt_var][i]][0]
        else:
            attrs_dict_list[dt_var][attr_domains[dt_var][i]] = 0

    # make dt_tree sub set
    dt_tree[trans[0][dt_var]] = attrs_dict_list[dt_var]

    for key in attrs_dict_list[dt_var].keys():
        if attrs_dict_list[dt_var][key] == 0:
            new_trans = make_new_trans(key, trans, dt_var)  # make new trans table
            new_tree = dict()
            dt_tree[trans[0][dt_var]][key] = find_max_info_gain(new_trans, new_tree)  # recursion

    return dt_tree
```

그림 8 - make decision tree by recursively

: About maximum information gain attribute, make decision tree node. If attribute sub tree is not pure, make new trans table that remove max info gain attribute's column, and call "find_max_info_gain()" function for recursive.



```python
def find_answer(trans_row, dt_tree, attrs, return_domain, val_list):  # get target value through dt_tree
    for key in dt_tree.keys():
        sub_tree = dt_tree[key]
        if trans_row[attrs[key]] in sub_tree.keys():
            ssub_tree = sub_tree[trans_row[attrs[key]]]
        else:  # if leaf node is not purity, voting
            min_val = 10000000
            min_key = ""
            val_dict = dict()
            for k in val_list:
                val = "₩'"+str(k)+"₩'"
                if val in str(sub_tree):
                    val = val[1:-1]
                    val_dict[val] = 1
            for k in val_dict.keys():
                if val_list[k] < min_val:
                    min_val = val_list[k]
                    min_key = k
            return min_key
        if ssub_tree in return_domain:
            return ssub_tree
        answer = find_answer(trans_row, ssub_tree, attrs, return_domain, val_list)

    return answer
```

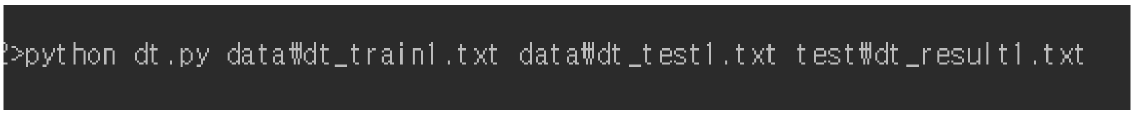
그림 9 – get target value through dt_tree

: This function is search decision tree by trans row.

Trans row is 'dt_test.txt' table's row data. If leaf node is not purity, go parent node. In parent node voting target value. Voting system process to find less frequency target value.

4. How to compile code

: Project structure is “dt.py” python file, data dictionary that includes train and test data set, and test dictionary that includes answer file and result file.



```
?>python dt.py data₩dt_train1.txt data₩dt_test1.txt test₩dt_result1.txt
```

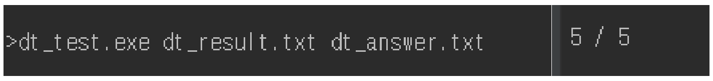
그림 10 - how to run dt.py file

: python dt.py [train dataset] [test dataset] [result file path name]

Train dataset and test dataset are in ‘data’ dictionary.

Result file path is ‘test’ dictionary for running dt_test.exe that calculate accuracy.

5. Result of test

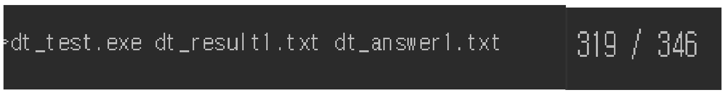


```
>dt_test.exe dt_result.txt dt_answer.txt
```

```
5 / 5
```

그림 11 - result of input dt_train.txt

: result of input dt_trains.txt that associate “buy_computer”



```
>dt_test.exe dt_result1.txt dt_answer1.txt
```

```
319 / 346
```

그림 12 - result of input dt_train1.txt

: result of input dt_trains1.txt that associate “car evaluation”