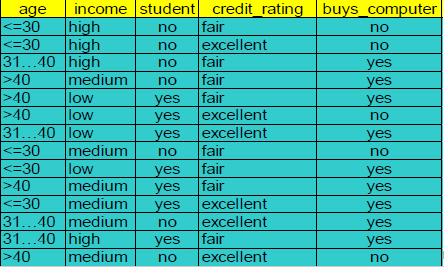
**Programming Assignment #2 report**

한양대학교 컴퓨터전공

2012004087

이기준

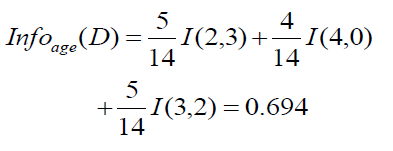
**Summary of your algorithm**

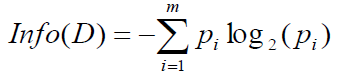


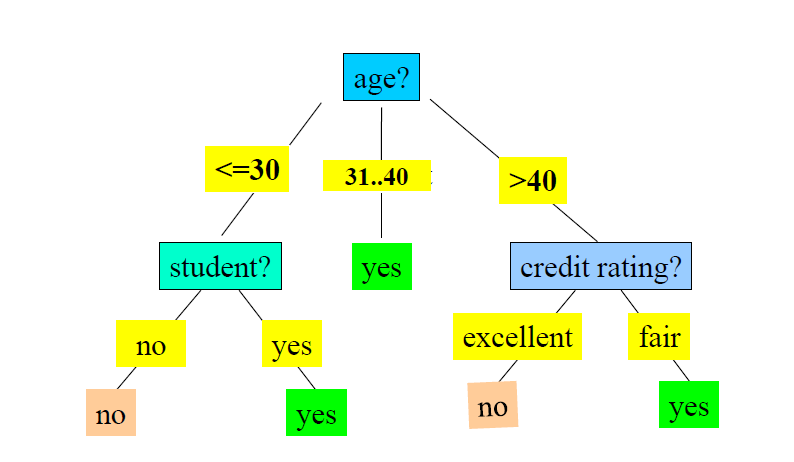
총 attribute가 4개가 있다고 할 때 (buys\_computer는 제외) 각각 attribute에 따라 information gain을 구한뒤 가장 큰 값을 선정하면 되는데



식을 보았을 때, Info(D)는 고정값이므로 Infox(D)값 중 가장 작은 값을 구하면 된다.





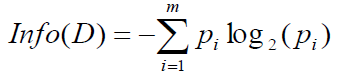
위의 계산과정을 통해서 구해보면 4개의 attribute중에 age의 information gain값이 제일 크다.

Age를 통해 갈리는 데이터들에 대해서 재귀적으로 처음부터 완전히 데이터가 갈릴 때까지 반복한다.

**Detailed description of your codes (for each function)**

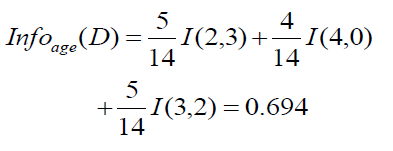
**import** numpy **as** np  
**import** math  
**import** sys

넘파이를 통해서 데이터를 쉽게 가공하였다.  
log 함수를 사용하기 위해 math, 프로그램 외부에서 변수를 받아오기위해 sys 이용

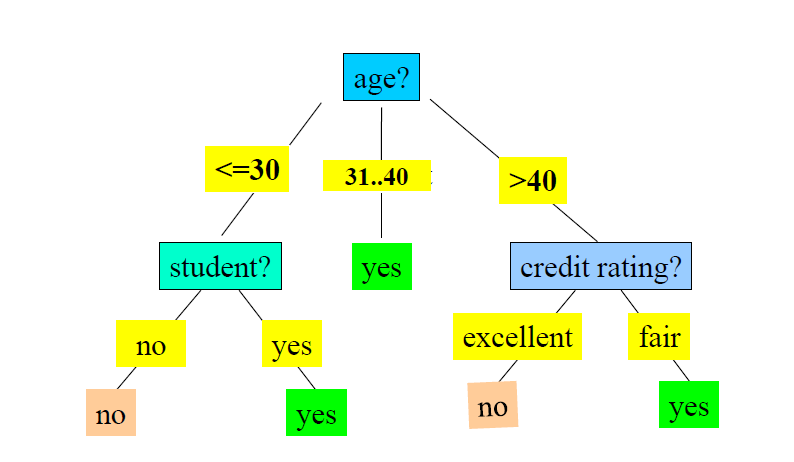


**def** Info (count\_list):  
 ret = 0  
 **for** i **in** count\_list :  
 **if** i == 0 :  
 **return** 0  
 ret -= i/sum(count\_list) \*math.log(i/sum(count\_list),2)  
 **return** ret

엔트로피 값을 구하는 함수이다. 위의 식을 그대로 구현하였다.

  
**def** Info\_A (list):  
 ret = 0  
 rows = np.unique(list[0])  
 cols = np.unique(list[1])  
  
 Tr\_list = list.T  
 x\_list = [[0 **for** x **in** range(cols.size)] **for** y **in** range(rows.size)]  
 **for** i **in** Tr\_list :  
 arr1 = np.where(i[0]== rows)  
 arr2 = np.where(i[1]== cols)  
 **if**(x\_list[arr1[0][0] ][arr2[0][0] ] == 0) :  
 x\_list[arr1[0][0]][arr2[0][0]] = 1  
 **else** :  
 x\_list[arr1[0][0]][arr2[0][0]] += 1  
**for** i **in** x\_list :  
 ret += (sum(i)/list[0].size \* Info(i))  
 **return** ret

특정 attribute에 대한 expected information 값을 구한다. 내부에서는 info() 함수를 호출한다.

  
**def** func (list, idx\_list, node) :  
 *# 모든 어트리뷰트를 다 탐색했을 때 재귀 종료* **if** list.shape[1] - 1 == len(idx\_list) :  
 node.append([list.T[list.shape[1] - 1][0]])  
 **return** *# 완벽히 갈렸을 때* **if** np.unique(list.T[list.shape[1] - 1]).size == 1 :  
 node.append([list.T[list.shape[1] - 1][0]])  
 **return** idx = -1  
 a = list[:, 0:-1]  
 c = list[:, -1]  
 min = 1  
 count = 0  
  
 **for** i **in** a.T:  
 *# print(i, "a")* **if** count **in** idx\_list:  
 count += 1  
 **continue** b = np.array([i, c])  
 *# print(b, "b")* **if** min > Info\_A(np.array(b)):  
 min = Info\_A(np.array(b))  
 idx = count  
 count += 1  
 *# 모두 만족한 경우기 때문에 더이상 진행하지않는다* idx\_list += [idx]  
  
attr = np.unique(list[:, idx])  
 *# xx\_list = np.empty([attr.size,], dtype=object)* xx\_list = [[] **for** y **in** range(attr.size)]  
  
 **for** i **in** list:  
 arr1 = np.where(i[idx] == attr)  
 *# np.append(xx\_list[arr1[0][0]], i)* xx\_list[arr1[0][0]].append(i)  
 **for** i **in** xx\_list:  
node.append([idx, i[0][idx]])  
func(np.array(i), idx\_list[:], node[len(node) - 1])

트레이닝 리스트를 통해서 의사 결정트리를 만드는 함수이다. 처음 information gain값이 가장 큰 attribute를 찾은다음 attribute의 각 속성마다 리스트들을 여러 개로 분리한다. 분리된 리스트들은 다시 재귀적으로 호출되어 완전히 갈라지거나, 모든 속성에 관해 나눠질 때까지 진행한다.

**def** testfunc (test,tree) :  
 count = 0  
 *# print(tree)* **if** len(tree) == 1:  
 **return** tree[0][0]  
  
 **while** count != len(tree):  
 *# print(test[tree[count][0]])* **if** test[tree[count][0]] == tree[count][1]:  
 **return** testfunc(test, tree[count][2:])  
 count += 1  
 count = 0  
   
 *# 동일한 조건을 못찾았으면 비슷한걸로 가게한다* **while** count != len(tree):  
 *# print(test[tree[count][0]])* **if** test[tree[count][0]] < tree[count][1]:  
 **return** testfunc(test, tree[count][2:])  
 count += 1  
  
 **return** testfunc(test, tree[-1][2:])  
  
트레이닝 리스트를 기반으로 만들어진 의사결정트리를 통해 한 개의 튜플의 테스트 값을 결정한다. 의사 결정트리에서 제대로 필터링이 되면 그 값을 반환하고, 찾지 못했을경우 가장 비슷한 값으로 반환한다.

**def** supertest (test\_list, tree):  
 ret\_list = np.insert(test\_list, len(test\_list[0]), values=0, axis=1)  
 print(ret\_list)  
 print(**"zz"**)  
 count =0  
 **for** i **in** test\_list:  
 print(i)  
 ret\_list[count][-1] = testfunc(test\_list[count], tree)  
 count += 1  
 **return** ret\_list

모든 테스트 리스트값에 대해서 결정한다. 위의 testfunc의 반복문이다.

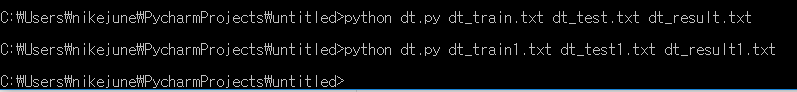
**Instructions for compiling your source codes at TA's computer (e.g. screenshot)**

OS : Windows10, 개발환경: pycharm 2017.2 사용언어:python (ver 3.6.4)

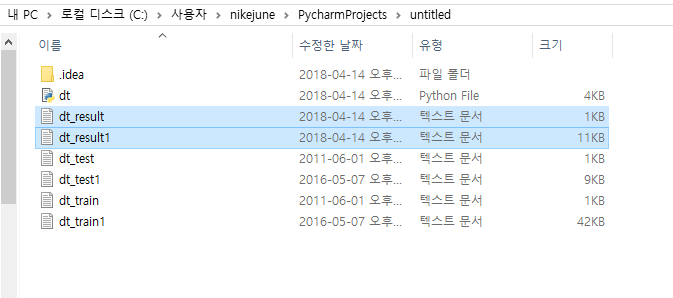
Numpy를 사용하기 때문에 numpy가 설치되어 있지않으면

Pip3 install numpy

를 통해 설치를 해야한다.

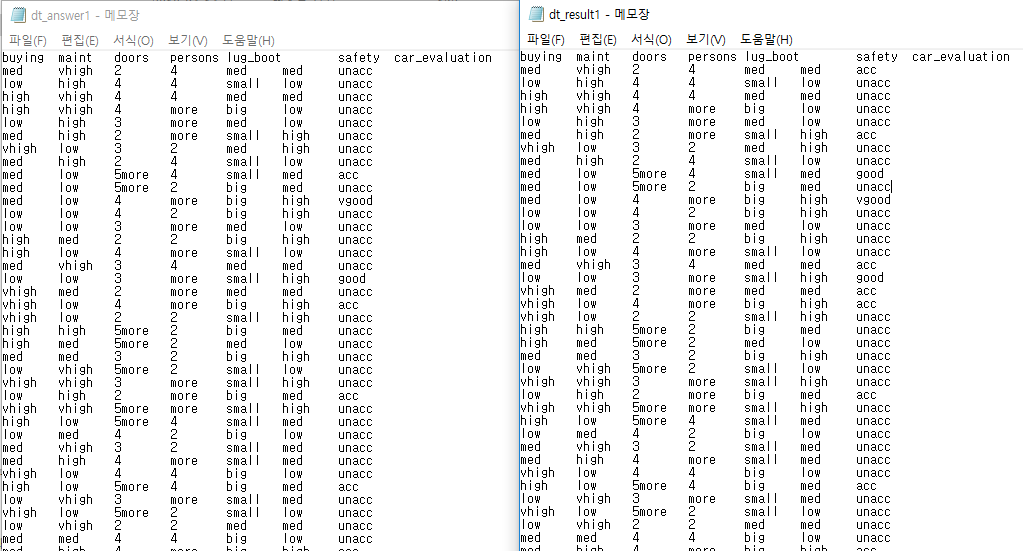
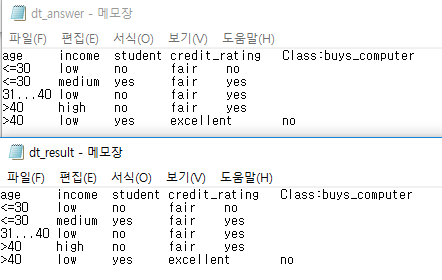


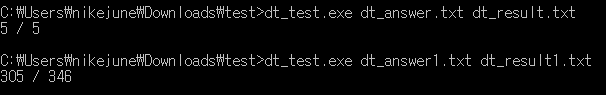
Python dt.py [train\_file] [test\_file] [result\_file] 순으로 입력



**Any other specification of your implementation and testing**

윈도우10에서 실행





테스트결과이다. 첫번째는 100% , 두번째는 약 88%정도 나온다.