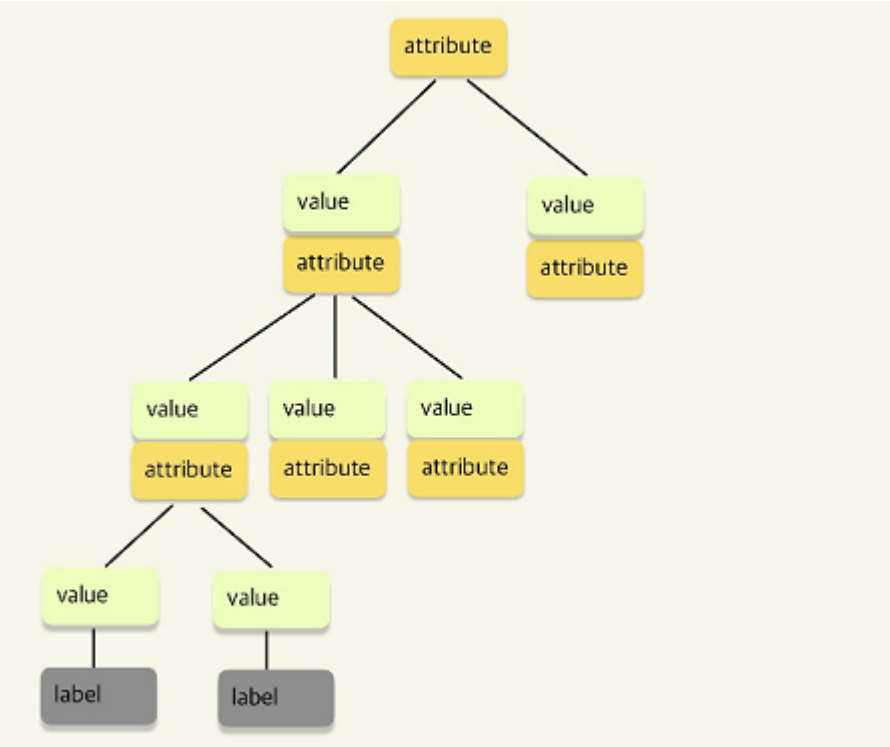
Build decision tree & classify the test set

summary

train set을 바탕으로 decision tree를 구축하고, 그것을 이용하여 test set을 classify한다.



Compilation method and environment

Windows: Python 3.7

execute

```
$ python decisionTree.py dt_train.txt dt_test.txt dt_result.txt
```

argv[1] = train set file

argv[2] = test set file

argv[3] = result file

check score

```
$ dt_test.exe dt_answer.txt dt_result.txt
```

argv[1] = answer file

argv[2] = experiment result file

decisionTree.py

class Tree

`__init__`

```
def __init__(self, _data, is_leaf):
        self.data = _data
        self.children = dict()
        self.is_leaf = is_leaf
        self.crt = None
```

init 함수는 Tree 객체(노드) 생성 시 초기화하는 함수이다. 각 노드는 train set에 대한 data를 가지고, children을 dictionary 형태로 가진다. `is_leaf`는 더 이상 split되지 않을 때 leaf임을 명시하며, `crt`는 해당 노드가 split 기준이 되는 attribute를 가질 경우 string 형식의 값을 가진다.

`buildTree`

```
def buildTree(self, _data):
    ...
    for i in range(len(_data.columns)-1):
                ...
                gain = p_info - c_info
                if Max < gain:
                    Max = gain
                    selected_attr = i

            self.crt = attrs[selected_attr]
            values = _data.iloc[:, selected_attr].unique()

            for i in range(len(values)):
                nd = _data[_data[attrs[selected_attr]] == values[i]]
                nd = nd.drop([attrs[selected_attr]], axis = 1)
                self.children[values[i]] = Tree(nd, 0)
                self.children[values[i]].buildTree(nd)
```

buildTree 함수는 train set을 기반으로 decision tree를 생성한다. parent node의 information gain (`p_info`)을 계산하고 parent node로부터 split가능한 attribute들에 대해 모두 information gain (`c_info`)을 계산하여, `p_info` - `c_info`가 최대가 되도록 하는 attribute를 다음에 split할 기준으로 선정한다.

선정된 attribute에 대하여 같은 value를 가지는 row끼리 새로운 dataframe `nd`을 생성하고, 이미 ancestor node에서 적용된 attribute는 더 이상 분류 기준이 될 필요가 없으므로 `nd`에서 해당 column을 삭제한다. attribute의 value 값을 key로, 새로운 Tree 노드를 value로 하여 children에 저장하고 `buildTree`함수를 재귀적으로 호출한다.

`classify`

```
def classify(self, _data):
        for i in range(len(_data)):
            err = 0
            case = _data.iloc[i, :]
            next = self
            while next.crt != None:
                if not (case[next.crt] in next.children.keys()):
                    maj = next.data.iloc[:, -1].value_counts(sort=True).idxmax(axis=0)
                    test.iloc[i, -1] = maj
                    err = 1
                    break
                next = next.children[case[next.crt]]
            if not err:
                next = next.children
                test.iloc[i, -1] = next
```

classify 함수는 decision tree를 이용하여 test set의 class label을 결정한다. 노드에 더이상 `crt`가 없으면 완전히 split 되었다는 뜻이므로, 해당 노드의 children을 class label로 가져온다.

탐색을 하는 도중 만약 현재 노드의 attribute에 해당하는 값들 중 자신의 값이 없어 더 이상 탐색할 수 없다면, train set을 기반으로 majority voting 방식을 통해 class label을 결정한다.

`make_crossTable`

```
def make_crossTable(fac, label):
    table = pd.crosstab(fac, label)
    table['total'] = table.iloc[:, :len(labels)].sum(axis=1)
    return table
```

make_crossTable 함수는 attribute와 class label 간의 cross table을 생성한다. class label 마다 attribute의 각 value 값들의 빈도를 계산해주고, 총 빈도를 계산하여 마지막 column으로 total index의 값으로 추가한다. information gain을 계산하기 위해 이용된다.

`entropy`

```
def entropy(crossTable):
    global setCnt
    ent = 0
    for i in range(len(crossTable)):
        total = crossTable.iloc[i, -1]
        info = sum(-p/total * math.log(p/total, 2) for p in crossTable.iloc[i, :-1] if p)
        prob = total/ setCnt * info
        ent = ent + prob
    return ent
```

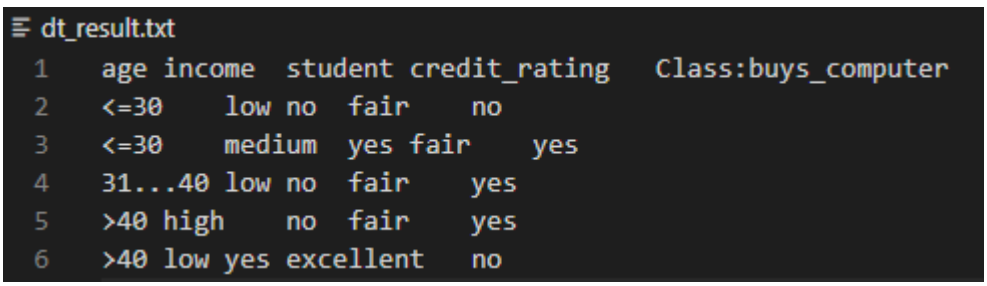
entropy 함수는 split되는 기준 attribute를 선택할 때 information gain를 계산하기 위해 사용된다.

`main`

```
if __name__ == '__main__':
    train_file = sys.argv[1]
    test_file = sys.argv[2]
    result_file = sys.argv[3]

    setCnt = 0

    train = pd.read_csv(train_file, sep='\t')
    labels = train.iloc[1:,len(train.columns)-1].unique()
    decisionTree = Tree(train, 0)

    decisionTree.buildTree(train)

    test = pd.read_csv(test_file, sep='\t')
    test[train.columns[-1]] = 'none'

    decisionTree.classify(test)

    test.to_csv(result_file, index=False, sep='\t')
```

command line으로 train set, test set 각각의 file name과 test set의 classification 결과를 저장할 result file name을 입력받고, train set과 test set을 dataframe 형태(`train`, `test`)로 읽어들인다. `train`을 바탕으로 `decisionTree`를 구축하고, 이것을 이용하여 `test`의 class label들을 결정하여 result file에 write한다.

result (ex)



dt_result1.txt

```
buying  maint   doors   persons lug_boot    safety  car_evaluation
med vhigh   2   4   med med acc
low high    4   4   small   low unacc
high    vhigh   4   4   med med acc
high    vhigh   4   more    big low unacc
low high    3   more    med low unacc
med high    2   more    small   high    acc
vhigh   low 3   2   med high    unacc
med high    2   4   small   low unacc
med low 5more   4   small   med good
med low 5more   2   big med unacc
med low 4   more    big high    vgood
low low 4   2   big high    unacc
low low 3   more    med low unacc
high    med 2   2   big high    unacc
high    low 4   more    small   low unacc
med vhigh   3   4   med med acc
low low 3   more    small   high    good
vhigh   med 2   more    med med acc
vhigh   low 4   more    big high    acc
vhigh   low 2   2   small   high    unacc
```