

Assignment2

Data Science
컴퓨터소프트웨어학부 2017029589 류지범

#0 Environment

- Python 3.7.11 에서 진행했다.
- 사용한 모듈은 numpy와 pandas이다.
- 실행 환경은 macOS Monterey 12.3 이다.
- 패키지 목록은 requirements.txt 에 담아두었고 pip install -r requirements.txt 로 설치하면 된다.

#1 Run

- 패키지는 pip install -r requirements.txt 로 설치하면 된다.

- 

- 소스 코드와 input 파일이 있는 폴더에서 python decision.py dt_train.txt dt_test.txt dt_result.txt 의 형태로 실행하면 된다.

- 

#2 Code & Function Description

- get_info(dataframe) : dataframe에서 class_label에 해당하는 decision의 entropy를 구해주는 함수이다.

```
def get_info(dataframe):  
    header = dataframe.columns  
    class_label = header[-1]  
  
    class_label_list = dataframe.groupby(class_label).size().values  
    info = get_entropy(class_label_list)  
  
    return info
```

- `get_entropy(target_value)` : 하나의 attribute_name에 해당하는 class_label의 decision을 이용해서 공식에 따라 entropy를 구해주는 함수이다.

```
def get_entropy(target_value):
    entropy = 0
    total = target_value.sum()

    for value in target_value:
        p = value / total
        if p == 0.: continue
        entropy += p * np.log2(p)

    return -entropy
```

- `get_gain_ratio(crosstable, info)` : 공식에 따라 gain_ratio를 구해주는 함수이다. 각 target_value에 대해서 split_info와 info_a를 구해주고 gain_ratio를 리턴한다.

```
def get_gain_ratio(crosstable, info):
    info_a = 0
    split_info = 0

    target_values = crosstable.values
    total = target_values.sum()

    for target_value in target_values:
        d = target_value.sum() / total
        info_a += d * get_entropy(target_value)
        if d == 0.: continue
        split_info += d * np.log2(d)

    return (info - info_a) / -split_info
```

- `get_max_gain_ratio(dataframe, info)` : class_label을 제외한 모든 header에 대해서 [attribute_name, class_label] 로 이루어진 crosstable을 생성하고 gain_ratio를 구해서 dictionary에 넣는다. dictionary에서 가장 큰 gain_ratio 값을 가지는 attribute를 리턴해준다.

```
def get_max_gain_ratio(dataframe, info):
    header = dataframe.columns
    class_label = header[-1]

    gain_ratio_list = dict()

    for attribute in header[:-1]:
        crosstable = pd.crosstab(dataframe[attribute], dataframe[class_label])
        gain_ratio_list[attribute] = get_gain_ratio(crosstable, info)

    # return attribute
    return max(gain_ratio_list, key=gain_ratio_list.get)
```

- `class Node` : decision tree를 만들기 위한 class이다. class의 method들의 설명은 아래와 같다.
- `__init__(self, dataframe)` : dataframe, attribute_name과 child, classification, class_label을 instance로 가진다. child는 attribute를 key로 가지는 dictionary이다.

```
def __init__(self, dataframe):
    self.dataframe = dataframe
    self.attribute_name = None
    self.child = dict()
    self.classification = None
    self.class_label = dataframe.columns[-1]
```

- is_leaf(self)** : decision을 구할 때 leaf node인지 판단하는 함수이다. classification이 정해져있으면 true를 리턴하고 그렇지 않으면 false를 리턴한다.

```
def is_leaf(self):
    if self.classification == None:
        return False
    return True
```

- `has_to_split(self)` : tree를 생성할 때 split을 할지 판단하는 함수이다. 현재 가지고 있는 dataframe의 info 값이 0이면 entropy가 0인 것이므로 더이상 분류할 필요가 없으므로 false를 리턴한다.

```
def has_to_split(self):
    info = get_info(self.dataframe)
    if info == 0.:
        return False
    return True
```

- `split_data(self)` : 더 이상 split할 필요가 없으면 entropy가 0인 것이므로 class_label의 unique한 0번째 값을 classification으로 결정한다.
 - 나뉘야할 경우 max_gain_ratio에 해당하는 attribute_name을 찾은 후 이것을 Node의 attribute_name으로 설정한다.
 - 그후 attribute_name에 해당하는 attribute들의 unique한 값을 뽑아내서 이를 기준으로 group으로 묶어내서 split_table을 형성한다.
 - split_table에서 attribute_name에 해당하는 column은 drop하고 child node를 생성해서 dataframe을 넘겨준다.
 - child node는 현재 node의 자식으로 설정해준다. Child node도 마찬가지로 재귀적으로 `split_data()` 를 실행하도록 한다.

```
def split_data(self):
    if not self.has_to_split():
        self.classification = self.dataframe[self.class_label].unique()[0]
        return

    split_attribute = get_max_gain_ratio(self.dataframe,
get info(self.dataframe))
```

```

self.attribute_name = split_attribute
split_list = self.dataframe[split_attribute].unique()

for attr_name in split_list:
    split_table =
self.dataframe.groupby(split_attribute).get_group(attr_name)
    new_node = Node(split_table.drop(split_attribute, axis=1))
    self.child[attr_name] = new_node
    new_node.split_data()
return

```

- `decision(self, query)` : test data에 대해서 decision을 구해주는 함수이다. 재귀적으로 tree를 탐색하면서 값을 결정해준다.
 - 현재 node가 leaf node이면 classification값을 리턴해준다.
 - leaf node가 아닐 경우 attribute_name에 해당하는 attribute를 query에서 찾고 child node를 탐색해보도록 한다. 이 때 해당 child node가 존재하지 않는 경우는 majority voting을 통해서 classification 값을 정해주도록 한다.
 - 이 과정을 재귀적으로 탐색해서 decision 값을 찾도록 한다.

```

#query is Series
def decision(self, query):
    if self.is_leaf():
        return print_result(query, self.classification)
    else:
        search = query[self.attribute_name]
        try:
            child_node = self.child[search]
        except:
            self.classification = self.dataframe.iloc[:,
-1].value_counts().idxmax()
            return print_result(query, self.classification)
        return child_node.decision(query)

```

- `print_result(query, decision)` : 원래의 query에 결정된 decision 값을 추가하여 result file의 양식에 맞게 리턴해주는 함수이다.

```

def print_result(query, decision):
    result = ''
    for item in query:
        result += str(item) + '\t'
    result += str(decision) + '\n'
    return result

```

- `read_input(name)`: `input.txt` 를 읽어서 pandas의 dataframe으로 변환시켜 리턴해주는 함수이다.

```
def read_input(name):
    df = pd.read_table(name, sep='\t')
    return df
```

- `write_output(node)`: test file을 읽어서 각 query를 줄 별로 뽑아낸 후, decision tree 를 탐색하며 값을 찾아내고 output file에 기록해주는 함수이다.

```
def write_output(node):
    test = read_input(sys.argv[2])
    out = open(sys.argv[3], 'w')
    header = print_result(node.dataframe.columns, '')
    out.write(header)

    for i in test.index:
        temp = node.decision(test.loc[i])
        out.write(temp)

    out.close()
```

- `__main__`: decision tree를 생성하고 결과를 만들어주는 main 함수이다.

```
if __name__ == '__main__':
    root = Node(read_input(sys.argv[1]))
    root.split_data()

    write_output(root)
```

#3 Result

- `dt_test.txt`는 다음과 같은 형태로 출력됐다.

```
age    income  student  credit_rating  Class:buys_computer
<=30   low    no      fair      no
<=30   medium yes     fair      yes
31...40 low    no      fair      yes
>40    high   no      fair      yes
>40    low    yes     excellent no
```

```
lordly@JiBeomui-MacBookPro:~/Desktop/Hanyang/2022_1학기/
> mono dt_test.exe dt_answer.txt dt_result.txt
5 / 5
```

- 정답과 비교해보았을 때 100%의 accuracy를 보여주고 있다.

- dt_test1.txt는 다음과 같은 형태로 출력됐다.

The screenshot displays a macOS desktop environment. At the top, a window titled 'dt_result1.txt' shows a table of car evaluation data. The table has seven columns: 'buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', and 'car_evaluation'. The data rows consist of categorical values for the first six columns, followed by a numerical 'car_evaluation' value. Below the text file, a terminal window is open, showing the command 'mono dt_test.exe dt_answer1.txt dt_result1.txt' being executed. The terminal output shows '318 / 346'.

buying	maint	doors	persons	lug_boot	safety	car_evaluation
med	vhhigh	2	4	med	med	acc
low	high	4	4	small	low	unacc
high	vhhigh	4	4	med	med	acc
high	vhhigh	4	more	big	low	unacc
low	high	3	more	med	low	unacc
med	high	2	more	small	high	acc
vhhigh	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	vhhigh	3	4	med	med	acc
low	low	3	more	small	high	unacc
vhhigh	med	2	more	med	med	acc
vhhigh	low	4	more	big	high	acc
vhhigh	low	2	2	small	high	unacc
high	high	5more	2	big	med	unacc
high	med	5more	2	med	low	unacc
med	med	3	2	big	high	unacc
low	vhhigh	5more	2	small	low	unacc
vhhigh	vhhigh	3	more	small	high	unacc
low	high	2	more	big	med	acc
vhhigh	vhhigh	5more	more	small	high	unacc
high	low	5more	4	small	med	unacc
low	med	4	2	big	low	unacc

Terminal output:

```

> mono dt_test.exe dt_answer1.txt dt_result1.txt
318 / 346

```

- 정답과 비교해보았을 때 91.9075144508671%의 accuracy를 보이고 있다.
- attribute가 2개일 때 decision을 결정해버리는 pruning 방식을 사용해보았을 때 더 낮은 정확도가 나와서 이 방법은 사용하지 않았다.