



[데이터 사이언스 2023-1 ]

컴퓨터소프트웨어학부

2019016735 현수빈

Assignment 2

## 1. 컴파일 환경 및 실행 방법

### A. 컴파일 환경

- Python 3.7.8
- Visual studio code로 개발(v.1.58.2)

### B. 실행 환경

- Mac OS 12.6에서 실행하고 개발하였다.

### C. 실행방법

- 먼저 해당 파일은 numpy 라이브러리를 사용하기에 라이브러리 설치를 진행한다.
- 해당 dt.py 파일이 존재하는 위치까지 terminal로 이동한다.
- dt\_train.txt와 dt\_test.txt를 같은 위치에 삽입한다.
- 아래의 형식에 맞춰 명령어를 입력한다.

python3 dt.py {train\_file\_name} {test\_file\_name} {output\_file\_name}

Ex) python3 dt.py dt\_train.txt dt\_test.txt dt\_result.txt

```
hyeonsubin-ui-MacBookPro-2:data hyunsubin$ python3 dt.py dt_train.txt dt_test.txt dt_result.txt
done
hyeonsubin-ui-MacBookPro-2:data hyunsubin$
```

## 2. 구현 방법

### A. 전체 구조

- 이번 과제는 dt.py 단일 소스코드로 작성하였다.
- Decision tree를 쉽게 만들기 위해 **Feature, Node**라는 class를 생성하였다.
- **Feature class**은 feature을 쉽게 접근하기 위해 만든 구조체로 해당 feature의 이름과 feature values와 db에서의 순서를 저장하고 있다.

```
class Feature:
    def __init__(self, feature, i):
        self.featureName = feature #해당 feature의 이름
        self.featureValueList = [] #feature이 value로 가질 수 있는 값들의 리스트
        self.tag = i #디비에서 feature의 순서
```

- **Node class**는 decision tree를 만들기 위해 각 트리를 구성하는 노드의 역할을 하는 구조체이다. 각 노드는 feature값과 자식 노드와 leaf node의 경우 class label의 값, 부모 노드에 의해 나뉘질 때의 해당 feature value값을 저장하고 있다.

```
class Node:
    def __init__(self):
        self.feature = None #feature 종류, nonleaf이면 값이 있고 아니면 None임
        self.children = [] #child 노드
        self.class_label = None #leaf node일 때 class_label
        self.feature_value = None # 부모노드 feature에 의해 나뉘질 때의 해당 feature value
```

- Node class는 6개의 멤버함수를 가지고 있다.

```
# split하기 전의 entropy값
def calculate_before_entropy(self, class_labels, data_set):-
# split 후의 entropy 값
def calculate_after_entropy(self, class_labels, data_set, feature):-
# splitinfo를 계산
def calculate_splitInfo(self, data_set, feature):-
#리프노드에서 majority voting하는 방법
def majority_voting(self, class_labels, data_set):-
#decision tree 학습
def fit(self, data_set, remainfeatureList, class_label):-
#decision tree으로 예측
def predict(self, data):-
```

- Gain ratio를 계산하기 위해 필요한 **calculate\_before\_entropy, calculate\_after\_entropy, calculate\_splitInfo** 함수가 있다.
- 리프 노드일 경우 majority voting을 위한 **majority\_voting** 함수가 있다
- 해당 decision tree를 만드는 함수인 **fit**과 만든 decision tree로 예측하는 **predict** 함수가 존재한다.
- 메인 코드는 아래 사진으로 구성되어 있다.

```
if __name__ == "__main__":
    if len(sys.argv) != 4:
        print("argv is not correct! please check argv one more time")
        quit()

    train_file_name = sys.argv[1]
    test_file_name = sys.argv[2]
    output_file_name = sys.argv[3]

    feature_list, class_label, transaction_list = read_db_file(train_file_name)

    model = Node()
    model.fit( transaction_list, feature_list, class_label)

    create_output_file(model, output_file_name, test_file_name, feature_list, class_label)
    print("done")
```

- 가장 중요한 우선적으로 3개의 인자를 받는 것이 조건이기에 sys.argv 의 수가 4개가 아니면 경고 문구를 출력하고 Quit하였다.
- 4개일 경우에 read\_db\_file 함수로 train\_file를 읽은 후에 model을 생성하고 train\_file를 이용하여 fit 함수로 decision tree를 학습한다. 그 후에 create\_output\_file함수로 test\_file을 읽은 후에 predict하여 output file에 저장한다.

## B. 주요 함수 설명

### i. Fit

- 먼저 해당 노드의 entropy를 계산한다.
- 그 후, 아래의 두 경우인 경우 더이상 split하지 않고 종료한다.

```
#stop 조건
if before_entropy == 0:
    self.class_label = data_set[0][-1]
    return
elif remain_feature_list.size == 0 :
    self.class_label = self.majority_voting(class_label, data_set)
    return
```

- 첫번째 경우, 해당 노드의 entropy가 0이면 해당 노드의 class label이 모두 같기에 리프노드로 만들고 종료한다.
- 두번째 경우, remain\_feature\_list가 사이즈가 0이라면 더 사용할 수 있는 feature가 없기에 majority\_voting하여 리프노드로 만든 후 종료한다.
- remain\_feature\_list에 있는 feature마다 gain\_ratio를 계산하여 가장 max값 일 때의 feature을 찾는다. 같은 Max값이 있다면 제일 뒤의 값으로 결정했다.

```
max_feature = remain_feature_list[0]
max_gain_ratio = sys.float_info.min

for feature in remain_feature_list:
    info_gain = before_entropy - self.calculate_after_entropy(class_label, data_set, feature)
    split_info = self.calculate_splitInfo(data_set, feature)
    gain_ratio = info_gain / split_info
    if gain_ratio >= max_gain_ratio:
        max_gain_ratio = gain_ratio
        max_feature = feature

self.feature = max_feature
```

- 그 후 선택한 feature에 있는 feature\_value\_list마다 child 노드를 만든다. 이 때 해당하는 child 노드를 학습하기 전에 child node에 들어가는 data set의 크기가 0이라면 현재 노드에서의 majority voting으로 class\_label을 만들어 리프노드로 만든다.

```
for feature_value in max_feature.feature_value_list:
    data_subset = data_set[data_set.T[max_feature.tag]==feature_value]

    new_remain_feature_list = np.delete(remain_feature_list, np.where(remain_feature_list == max_feature))
    new_leaf = Node()
    new_leaf.feature_value = feature_value

    if data_subset.size != 0:
        new_leaf.fit(data_subset, new_remain_feature_list, class_label)
    else: #해당 노드에 데이터가 없는 경우
        new_leaf.class_label = self.majority_voting(class_label, data_set)
    self.children.append(new_leaf)
```

## ii. Predict

```
#decision tree으로 예측
def predict(self, data):
    if self.feature == None:
        return self.class_label

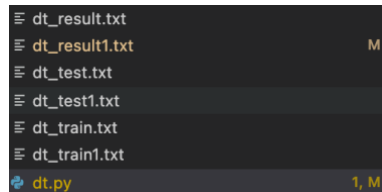
    for child in self.children:
        if child.feature_value == data[self.feature.tag]:
            return child.predict(data)
```

- 해당 노드의 feature가 None이면 리프노드이기 때문에 해당 노드의 class\_label값을 반환한다.
- 아니라면 해당 노드의 child의 feature\_value 중에 data와 같은 값인 child로 predict를 재귀적으로 호출한다.

## c. 실행 결과

위의 방식대로 제공된 database를 이용하여 실행한 경우 아래와 같이 dt\_result.txt(dt\_result1.txt)가 생성된다.

```
hyeonsubin-ui-MacBookPro-2:data hyunsubin$ python3 dt.py dt_train1.txt dt_test1.txt dt_result1.txt
done
hyeonsubin-ui-MacBookPro-2:data hyunsubin$ python3 dt.py dt_train.txt dt_test.txt dt_result.txt
done
```



Terminal에서 테스트 파일을 실행했을 때 각 100%, 93.06% 결과가 나왔다.

```
hyeonsubin-ui-MacBookPro-2:test program hyunsubin$ mono dt_test.exe dt_answer.txt dt_result.txt
5 / 5
hyeonsubin-ui-MacBookPro-2:test program hyunsubin$ mono dt_test.exe dt_answer1.txt dt_result1.txt
322 / 346
```