Programming Assignment #2

2017030328 조지훈

Compilation method and environment

사용 에디터: Visual Studio Code & VIM editor

개발 OS: Ubuntu 20.04.1 LTS (Windows 10 PRO 21H1 버전에서 WSL2 이용)

언어: Python 3.9.7

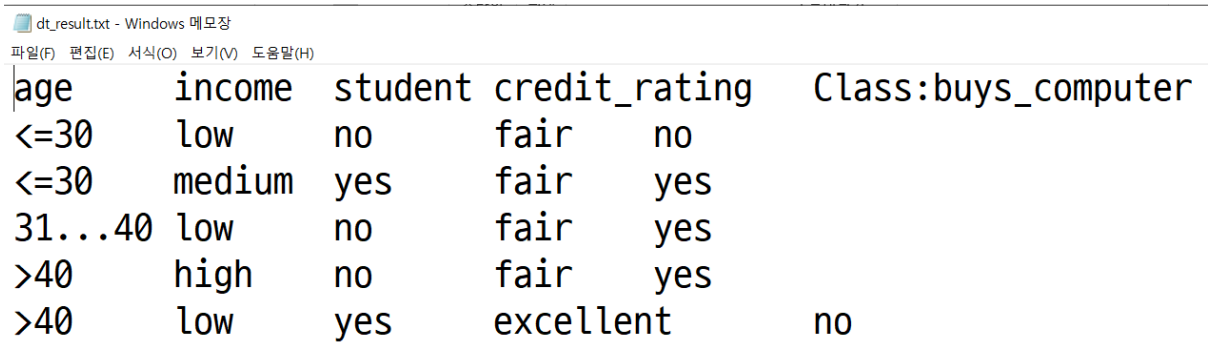
장치 사양

프로세서	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
RAM	16.0GB
시스템 종류	64 비트 운영 체제, x64 기반 프로세서

실행 방법

```
Assignment02 > master ?1 ▒▒▒ python3 dt.py dt_train.txt dt_test.txt dt_result.txt
```

리눅스 우분투 기준으로 과제 명세서와 동일하게 3개의 arguments를 받아 작동하고 결과물에 해당하는 arguments에 대한 파일로 결과가 출력된다. 다음은 과제에서 주어진 테스트에 대한 프로그램 실행 결과물의 일부이다.



Algorithm 설명

decision tree를 엔트로피에 개념을 두고 있는 information gain을 기반으로 구현하였다.

입력 변수를 바탕으로 목표 변수의 값을 예측하는 모델을 생성하는 것을 목표로 하고, 트리 구조에서, 각 내부 노드들은 하나의 입력 변수에, 자녀 노드들로 이어지는 가지들은 입력 변수의 가능한 값에 대응된다. 자료 집합을 적절한 분할 기준 또는 분할 테스트에 따라 부분 집합들로 나누고, 부분 집합들을 더 이상 나눌 수 없을 때까지 해당 과정을 진행하며 '학습'을 하고 결정

트리를 생성하게 된다. 이후 예측을 해야 할 데이터가 입력이 된다면, 만들어진 결정 트리를 기준으로 값을 예측하게 된다.

Code 설명

해당 코드는 크게 2부분으로 구분할 수 있다. 트레이닝 데이터를 이용해서 학습을 하는 부분과 학습을 해서 나온 룰을 이용해 들어온 테스트 데이터의 값을 분류하는 부분으로 나눌 수 있다. 메인 함수를 기준으로 107-118에 해당하는 부분이 학습하는 부분이고, 125-139에 해당하는 부분이 학습한 규칙을 이용해서 결과를 구하고 파일로 출력하는 부분이다.
해당 코드에는 5가지의 함수가 존재하고, 이 중에서 4가지가 데이터 학습에 사용되는 함수이고, 1가지가 데이터를 입력하여 결과를 출력할 때 이용하는 함수이다.

initdata(data, num)

각 속성이 보유한 값을 추출하는 함수이다. Attribute value가 처음부터 정해져 있지 않더라도 트레이닝 데이터를 이용하여 보유한 Attribute value를 구해서 따로 저장을 한다. Class label도 해당 함수를 이용하여 나올 수 있는 값을 추출하여 저장한다.

Entropy(tree, target)

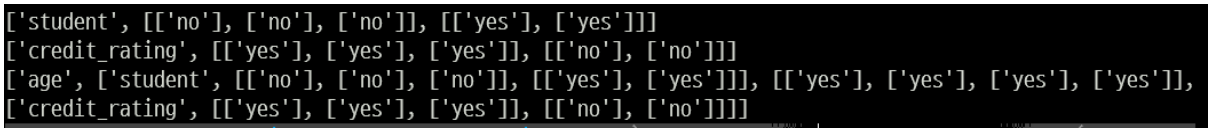
입력으로 주어지는 tree에 대한 entropy를 구하는 함수이다. 2번째 argument로 주어지는 target는 Class label에서 나올 수 있는 값을 의미하는 것으로, tree 내부의 각각의 Class label의 개수를 구하여 해당 비율을 이용해 entropy를 계산하여 반환하는 함수이다. numpy를 이용해 log를 연산.

infoGain(subdata, infoma, attribute, target)

information gain을 기준으로 argument로 주어지는 subdata(=subtree)를 최적의 결과로 분할하여 분할된 tree와 분할한 기준이 되는 Attribute를 반환하는 함수이다. 초기에 최고의 tree는 입력으로 들어온 subdata로 설정하고, 최고의 information gain은 0(지금 엔트로피 = 나중 엔트로피)으로 최고의 기준 Attribute는 ''(값이 없음)으로 설정한다. 이후 현재의 엔트로피를 구하고, infoma로 주어진 Attribute 목록을 이용하여 각각의 Attribute에 대해 분류하여 information gain을 연산. information gain이 가장 큰 값이 나온 경우 기존의 값을 수정하여 값을 반환하게 된다.

makeRule(data, infoma, attribute, target)

위에서 설명한 3개의 함수를 이용하여 decision tree의 rule를 생성하는 함수이다. rule은 list의 형태로 이루어지게 된다.



위 그림은 실제로 테스트를 진행할 때 rule를 출력한 결과이다. list의 0번째 위치에는 해당 시점에서의 분류 기준(Attribute)이 자리를 하고 있고, 그 이후에 Attribute가 가질 수 있는 값의 개수만큼 list가 존재한다. 그리고 각각의 위치에는 Attribute value의 값이 같은 data들이 들어가게 되고, Attribute value의 값이 같은 data들끼리 모인 집합에서 다시 makeRule 함수를 적용하여 다시 분할을 진행한다. 이 때, 더 이상 분할이 진행되지 않는다면, list에 Class label만 남겨서 결과를 알 수 있도록 구현한다.

infoGain 함수를 통해 얻은 분할된 데이터를 이용해서 다시 makeRule 함수를 호출하는 재귀적 형태로 구현되어 있고, 트레이닝 데이터가 가지고 있는 모든 Class label을 하나도 지우지 않고, rule를 나타내는 list에 저장한다는 특징을 가지고 있다.

(함수 종료 시점) 함수 내부적으로는 infoGain의 반환 값인 criteria(분류 기준)을 확인하여 분류 기준이 존재하지 않을 경우 데이터를 더 이상 분할하지 않는 것으로 판단하여 해당 시점에서 함수를 종료하고 반환을 진행하게 된다.

testing(rule, data, infoma, attribute)

앞선 함수들과 다르게 구한 rule를 이용하여 data로 들어온 값의 Class label을 결정하는 함수이다. rule를 만들 때 list의 0번째에는 분류 기준이 들어가기 때문에 해당 위치의 data가 분류 기준이 아니라면, 해당 부분 list는 leaf node로 판단하게 된다. 해당 leaf node에는 같은 Attribute value를 가지는 data들의 결과가 저장되어 있을 것이기 때문에, 함수에서는 그 값 중에서 random으로 값 1개를 반환 시킨다. 이렇게 하는 이유는 같은 Attribute value를 가진 data들끼리도 Class label이 다를 수 있기 때문에 다수결의 원칙이 아닌 확률적인 원리를 이용하기 위해서 다음과 같이 구현하였다. 그렇게 때문에 실제로 테스트를 진행했을 때도 조금씩 값이 다르게 나온 것을 확인할 수 있었다.

다음은 테스트 결과이다.

```
C:\Users\msi\Project_Ubuntu\Data Science\20
dt_test.exe dt_answer.txt dt_result.txt
5 / 5

C:\Users\msi\Project_Ubuntu\Data Science\20
dt_test.exe dt_answer1.txt dt_result1.txt
312 / 346

C:\Users\msi\Project_Ubuntu\Data Science\20
dt_test.exe dt_answer1.txt dt_result1.txt
313 / 346

C:\Users\msi\Project_Ubuntu\Data Science\20
dt_test.exe dt_answer1.txt dt_result1.txt
314 / 346

C:\Users\msi\Project_Ubuntu\Data Science\20
dt_test.exe dt_answer.txt dt_result.txt
5 / 5
```

dt_result.txt의 경우에는 사이즈도 작고 분류도 잘 되어서 오차가 없는 것을 확인할 수 있었지만, 그보다 복잡한 data인 dt_result1.txt의 경우에는 약 10% 정도의 오차가 발생하였고, 또한 그 오차가 프로그램을 진행해서 dt_result1.txt을 생성할 때마다 달랐는데, 이는 트레이닝을 진행한 data set에서 같은 Attribute value를 가진 data들끼리도 Class label이 다르기 때문에 매번 결과가 조금씩 다르게 나오는 것으로 판단하였다.

```
['maint', [['acc'], ['acc'], ['acc']], [['acc'], ['acc']], [['unacc'], ['unacc'], ['unacc']], [['acc'], ['acc']]]
['doors', [['unacc'], ['unacc']], [['unacc'], ['unacc'], ['unacc']], [['acc'], ['unacc'], ['acc']], [['acc'], ['acc']]]
```

또한, 실제로 그러한 data가 존재하는 것을 확인하였다.