**Assignment2**

컴퓨터소프트웨어학부

2018007983

김성애

* 알고리즘

Train파일에서 train data들을 읽어 저장하는데, attribute의 개수만큼 리스트가 있고 리스트의 index는 attribute의 index입니다. 이 리스트 안에는 딕셔너리가 들어있는데, 딕셔너리에는 해당 attribute가 가지는 attribute value의 종류가 key로 들어가고 attribute value가 총 번 나오는지는 딕셔너리의 value에 해당합니다.

처음에는 이렇게 구한 리스트에서 마지막 인덱스에 들어있는 label 을 보고 label의 값들을 가지고 엔트로피(info) 를 구하고 이 엔트로피와 attribute value들, 트레이닝 데이터를 가지고 gain과 splitinfo를 구해 나눔으로써 gainRatio를 계산합니다. 각 attribute 에 대한 gainRatio를 보고 가장 좋은 attribute를 뽑아 내려갑니다. 더이상 나눌 attribute가 없거나 data들이 없거나, data label 이 모두 같다면 path를 찾은 것이므로 이를 저장해 둡니다.

최종적으로 만들어진 path들을 가지고 test data의 label을 정합니다. 이 때, 트레이닝 과정에서 학습되지 않은 path를 test data가 가지고 있다면 가능한 곳까지 내려가고 majority voting을 통해서 label을 정해줍니다.

* 코드 설명
* **def** cal\_train\_data(train\_data, remove)

**def** cal\_attribute(attribute, remove)

**def** cal\_attribute\_value(train\_data, attribute\_count)

Attribute를 뽑아 내려갈 때마다 이전에 뽑은 attribute 를 제외하고 train data와 attribute, attribute value를 가지게 하였는데, 위 함수들에 해당 기능을 구현하였습니다. 함수 내에서

deepcopy를 이용해 파라미터로 받은 train\_data나 attribute를 복사하고 복사본에서 이미

뽑힌 attribute에 관한 변수인 remove를 이용해 제거합니다.

* **def** find\_label\_count(train\_data)

위 함수를 통해 가지고 있는 train data 들의 label의 종류에 따른 개수를 구합니다. 예를

들어 label이 yes가 5개, no가 9개가 나왔다면 이를 key와 value값으로 하여 딕셔너리로 저  
장한 후에, 그 value들만을 리스트 [5,9] 로 하여 리턴하는 함수입니다. Info를 구하기 위해 사용됩니다.

* **def** cal\_gain\_ratio(info, attribute\_value, train\_data)

다음은 파라미터로 들어온 attribute\_value (리스트인데, 여기서 인덱스는 attribute의 인덱스를 뜻합니다. 예를 들어 buying 이라는 attribute가 attribute 리스트에서 0번째에 있다면 attribute\_value 의 0번째에는 buying의 값들이 딕셔너리 형태로 들어오게 됩니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

For 문을 통해 class label 있는 마지막을 제외하고 attribute\_value 리스트의 길이만큼 반복하는데, 각각 attribute 의 값을 가지는 attribute\_value 리스트에서 value들의 합을 구해 총 몇개가 있는지 알 수 있습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Attribute들을 모두 돌며 ratio를 구하기 위하는 과정인데, attribute의 값과 개수를 꺼내어

label 을 보고 info를 구한 후에 이를 가지고 해당 attribute로 나누었을 때 엔트로피 info\_a 을 구합니다. 이 둘의 차를 구해 gain을 구하고 ratio까지 구합니다.

여기서 엔트로피를 구하는 함수는 다음과 같습니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* **def** make\_decision\_tree

마지막으로 복잡한 decision tree를 계산하기 위한 path 를 구하는 함수입니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

각각 인자로 받은 train data, attribute, attribute value 값을 앞서 설명한 함수들을 이용해 원본을 복사하고 거기서 뽑힌 attribute 에 대한 정보를 제외합니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Label\_count는 find\_label\_count를 이용해 현재 가지고 있는 train data의 label의 종류와 값을 구하는데, 만약 이를 통해 엔트로피를 구했을 때 0이 나온다면 더이상 나누지 않아도 되기 때문에 path를 찾았다고 생각하고 total\_path에 저장합니다. 이때 path는 현재까지 내려오면서 거쳐온 attribute에 관한 정보를 가지고 있는데 (attribute 이름, attribute 값) 형식의 튜플로 되어있습니다. 만약 buying이 low라면 car\_evaluation 이 acc 이라고 하면

[(‘buying’, ‘low), (car\_evaluation,acc)] 으로 표현할 수 있습니다.

마찬가지로 나눌 attribute가 더이상 존재하지 않는다면 label을 정해야 하는데, label 들의 값을 보고 majority voting을 하여 많이 나온 label에 대하여 label을 정해줍니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

위 과정을 통해 attribute를 뽑아 내려가는 것을 반복합니다. 안에 있는 for문은 해당 attribute 의 값으로 가질 수 있는 attribute\_value 에 따라 나누어 split\_attribute\_list에 담는데, 예를 들어 buying이 low, med, high의 value를 가진다면 split\_attribute\_list에는 [‘low’, ‘med’, ‘high’] 가 들어가게 됩니다. 이를 가지고 train data 를 나누고 재귀적으로 반복하여 내려갑니다.

* Main에서는 test data에 대한 label를 결정하는 과정에서 학습된 path가 저장된 total path 안에 있는 각각의 path들을 봅니다.

트레이닝된 path가 존재한다면 path의 끝에 있는 label 튜플을 보고 값을 가져옵니다.

텍스트이(가) 표시된 사진

자동 생성된 설명

* 그렇지 않다면 (path가 존재하지 않는다면) 존재하는 path의 길이만큼 저장하고 해당 길이까지 동일한 path들의 결과를 majority voting을 통해 찾아내고 label로 설정해줍니다.

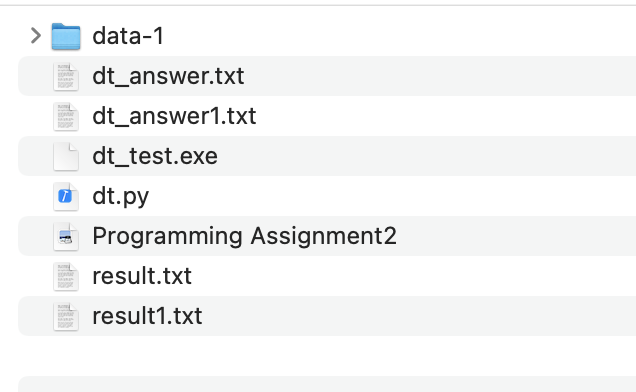
텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

* 컴파일 방법
* answer 파일, result 파일, test.exe, dt.py를 같은 위치 폴더에 넣습니다.



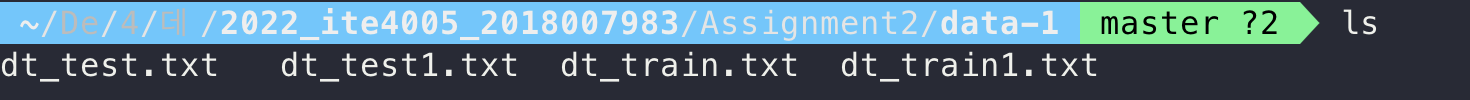
텍스트이(가) 표시된 사진

자동 생성된 설명

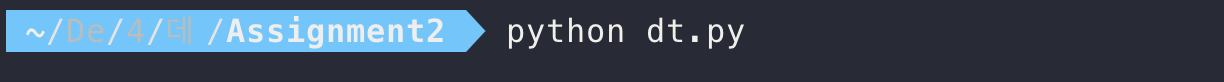
* data-1 폴더 안에는 train 파일과 test 파일이 들어있습니다.

테이블이(가) 표시된 사진

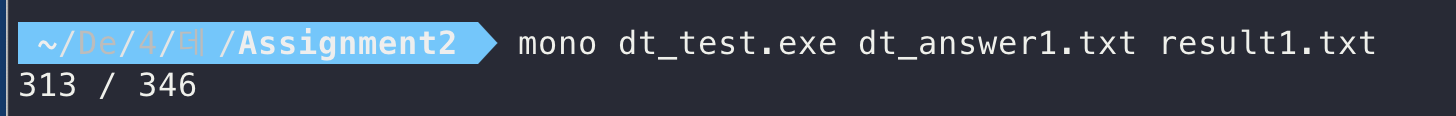
자동 생성된 설명



* Python dt.py 로 컴파일합니다.



* Mono 를 통해 실행합니다.



* 이외 시도한 것

Decision tree를 class node 를 통한 linked list 가 아닌 path를 기억하는 형태로 만들었는데, d처음에는 딕셔너리를 통해 key는 attribute의 이름이고 value는 다음으로 가야할 attribute의 인덱스를 주는 방법을 시도했습니다. 테스트할 때는 key를 보고 value를 따라가게 되었는데, value가 int가 아니라면 인덱스가 아니고 label임으로 label을 주는 방법이었습니다. 다만 여기서 이미 불린 attribute 가 다른 곳에서 다시 사용된다면 (예를 들어 buying이 low면 person 을 다음 attribute로 선택했는데, maint에서 person을 다시 선택한 경우) 딕셔너리의 value가 덮어 씌어지는 문제가 발생하여 정확도가 많이 떨어짐을 알 수 있었습니다.

제가 선택한 방법은 트레이닝 과정에서 학습되는 path가 attribute의 이름과 값을 튜플로 표현하여 가지고 있는 리스트 형태인데 path가 조금 많아진다는 단점과 학습되지 않은 path를 처리하는 문제가 발생하였습니다. 이때 구할 수 있는 path들의 majority voting을 이용해 label 하였습니다.