**Decision Tree(ID3)**

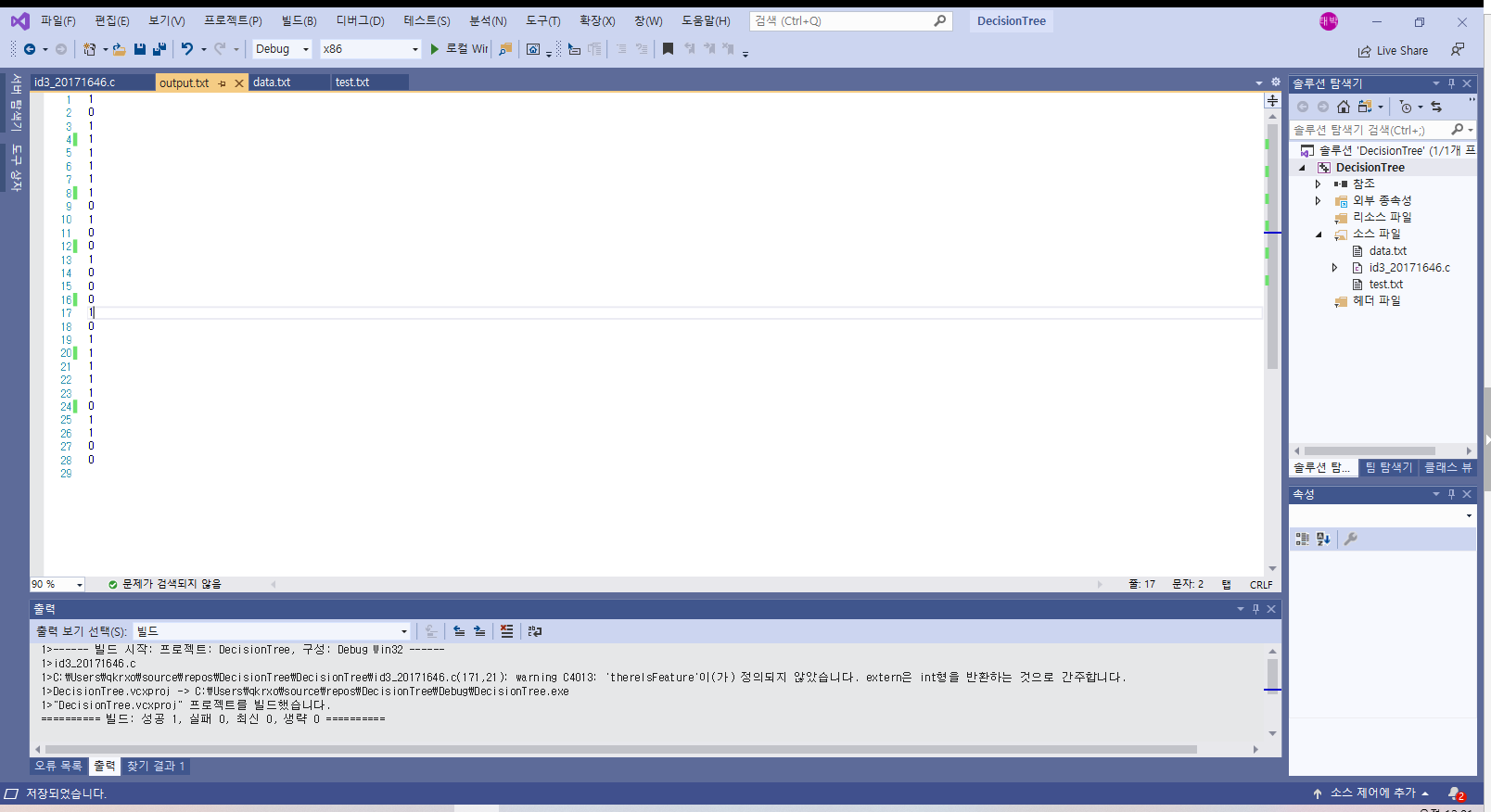
컴퓨터공학과 20171646 박태윤

**(코드 실행 결과)**

**-Input**

|  |  |  |
| --- | --- | --- |
|  |  |  |

**-output**

****

**(코드 / 알고리즘 설명)**

data.txt에 있는 데이터들을 읽는다. 첫 줄에는 데이터의 개수와 feature의 개수가 주어지고 그 밑으로는 데이터들의 feature와 class가 주어진다. feature의 개수가 n개라고 가정하면 (feature 1) (feature 2) … (feature n) (class)와 같은 형태로 주어진다. 이후 각 feature의 정보획득량을 계산하여 이를 통해 트리를 만든다. 트리를 만들다가 클래스가 0이나 1로 모두 같은 값이 나오거나 더 볼 feature가 없다면 leafnode를 만든다. 트리 생성이 완료되면 test.txt에 있는 데이터들의 클래스를 분류하여 output.txt를 만든다.

|  |  |
| --- | --- |
| 텍스트, 스크린샷, 노트북, 컴퓨터이(가) 표시된 사진  자동 생성된 설명 |  |

트리의 노드는 다음과 같이 만들었다. 트리의 노드를 treenode와 마지막에 연결되는 leafnode로 구분하기 위해 tagfield를 선언하였다. feature와 class의 값이 모두 0아니면 1이기 때문에 이진 트리이므로 treenode의 링크는 Llink와 Rlink로 2개를 선언하였으며 왼쪽 링크와 오른쪽 링크에 연결된 노드가 어떤 값을 나타내는지 보여주는 Ldata와 Rdata를 선언하였다. int\*\* arr에 data.txt에 있는 데이터들의 feature와 class가 저장되는데, treenode구조체 안의 int\*\* thisArr은 이 트리노드의 arr과 같은 역할을 한다. 즉, 앞에서 어떤 노드가 연결이 되었고 어느 데이터를 봐야하는지를 알려주는 역할을 한다. arrRow는 thisArr이 앞에서 언급한 역할을 하기 위해 필요한 변수이다. thisArr의 행 크기를 알려준다. 이 treenode가 트리노드인지 리프노드인지를 알려주는 tagfield형 변수 tag를 선언하였으며 공용체를 통해 트리노드라면 int index를 사용하고 리프노드라면 int tagNum를 사용할 수 있도록 구현하였다.

앞에서 언급했지만 int\*\* arr에는 data.txt에 있는 데이터들의 feature와 class가 들어간다. numOfData와 numOfFeature는 각각 데이터의 개수와 feature의 개수를 뜻한다.

함수들 역시 위의 코드와 같이 선언하였다. logB는 밑이 2인 로그를 계산하기 위해 존재한다. makeFirstNode는 Tree에서 첫 노드를 생성하는 함수이며 makeTree는 트리를 생성하는 함수이다. countT와 countF는 각각 feature + class의 1과 0의 개수를 구해주는 역할을 하며 thereIsFeature는 트리에 어떤 노드를 만드는 과정에서 더 고려할(계산할)feature가 존재하는지를 알려주는 역할을 한다. calEntropy와 calEntropy2는 엔트로피를 계산해주는 함수이다. makeNodeArr은 해당 노드의 thisArr을 만들어주는 역할을 하며 makeLeafNode는 리프노드를 생성해주는 함수이다. makeOutput은 출력을 해주는 함수이며 freeTree와 PostOrder는 트리의 메모리 해제를 해주는 함수이다.

텍스트, 스크린샷, 노트북, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

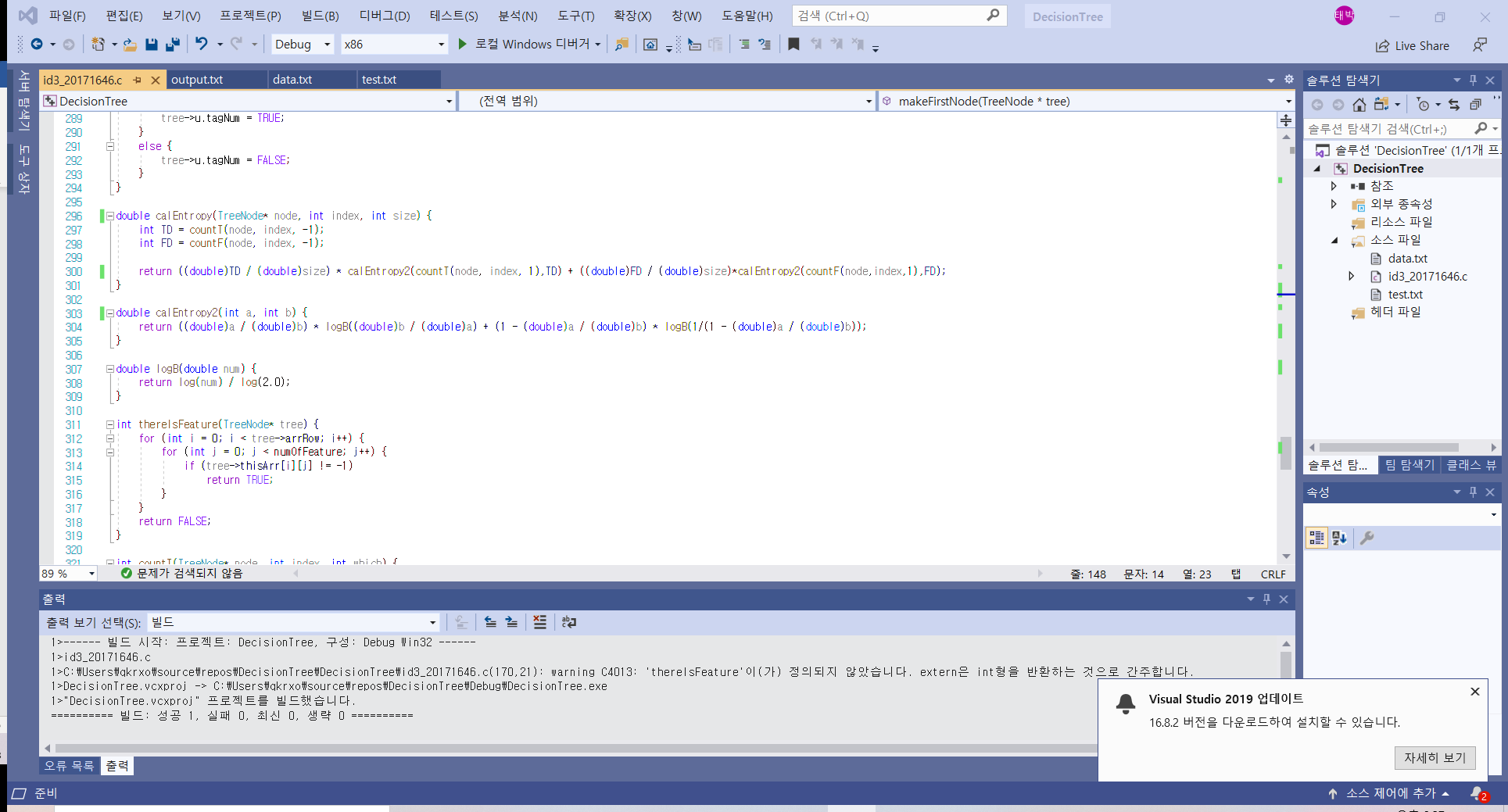
위와 같이 메인 함수를 구현하였다. data.txt를 읽어 첫 줄에 쓰여진 데이터의 개수와 feature의 개수를 각각 numOfData, numOfFeature에 저장하였다. 트리의 첫 노드인 tree를 메인에서 동적할당 하였으며 그 이후 arr에 앞에서 받은 numOfData와 numOfFeature을 활용하여 data.txt에 있는 데이터들의 feature와 class를 assign하였다. 이후 makeFirstNode를 통해 index등 트리 첫 노드의 값들을 할당하였고 makeTree를 통해 전체 트리를 구성하였으며 makeOutput으로 출력파일을 만들었고 마지막으로 PostOrder를 통해 트리 메모리 해제를 진행하였다. 또한 동적할당을 사용한 arr역시 메모리 해제를 진행하였다.

|  |  |
| --- | --- |
|  |  |

tree의 첫 노드를 만들어주는 함수이다. Llink와 Rlink는 모두 NULL로 초기화 시켜줬으며 tag는 treenode로 지정하였다. 이후 tree의 thisArr을 만들어줬는데 트리의 첫 노드이므로 arr과 똑같이 만들어주었다. 엔트로피 계산을 위해 TD와 FD라는 변수를 선언하였는데 TD는 TRUE(1)의 개수, FD는 FALSE(0)의 개수를 의미한다. 정보획득량 계산은 클래스의 엔트로피를 계산한 이후 각 feature의 엔트로피를 계산하여 클래스의 엔트로피에서 feature의 엔트로피를 뺀 값이 어느 feature에서 가장 큰지를 결정하여 그 feature를 트리노드의 값으로 지정해줘야한다. 즉, 각 feature에서 엔트로피를 계산했을 때 어느 엔트로피가 가장 작은지를 고르면 된다. 때문에 처음에 minEntropy를 -1로 지정하였고(Entropy는 음수가 나올 수 없다.) 이후 반복문을 통해 모든 feature의 엔트로피를 계산해가며 어느 feature에서 엔트로피가 가장 작은지를 구하였으며 이 정보를 index라는 변수에 저장하였다. 반복문이 종료되면 tree의 u.index에 index값을 넣어주었고 Ldata와 Rdata를 각각 FALSE, TRUE로 지정하였다.

|  |  |
| --- | --- |
|  |  |

TRUE(1)과 FALSE(0)의 개수를 세주는 함수이다. node와 index, which라는 매개변수가 존재하는데 node는 현재 보고 있는 노드를 가리키며 index는 몇 번째 feature를 볼 지, which는 어느 작업을 할 지 결정해주는 역할을 한다. which가 -1일 때 랑 그렇지 않을 때 함수의 기능이 달라진다. -1인 경우에는 노드의 thisArr에서 해당 index를 봤을 때 1이 몇 개인지 혹은 0이 몇 개인지를 세준다. 반면 -1이 아닌 경우는 thisArr에서 함수에 따라 해당 index의 1(counT) 또는 0(countF)만을 보고 그 안에서 클래스의 1(counT)이 몇 개인지 0(countF)이 몇 개인지를 세준다.



엔트로피 계산 관련 함수들이다. logB는 num을 받아 밑이 2인 로그로 계산하여 반환하는 역할을 한다. 식은

로 표현할 수 있기 때문에 calEntropy와 calEntropy2를 위와 같이 구현하였다.

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

트리를 만들어주는 makeTree함수이다. 매개변수로는 현재 보고 있는 노드인 tree와 왼쪽 자식 노드인 Lnode, 오른쪽 자식 노드인 Rnode를 가진다. 우선 왼쪽 자식노드인 Lnode를 동작할당하여 tree의 Llink와 연결시켜준다. 이후 Lnode의 link들을 모두 NULL로 초기화 한 뒤 Lnode에 대해 makeNodeArr를 실행한다. makeNodeArr은 해당 노드의 thisArr를 만들어주는 역할을 한다. Lnode의 thisArr를 만든 뒤 Lnode를 leafnode로 만들어야할지 treenode로 만들어야할지 판단하기 위해 조건문을 실행한다. !thereIsFeature(더 볼 feature가 없음)이거나 Lnode->arrRow가 countT또는 countF와 값이 같을 때, 즉 class가 TRUE 또는 FALSE밖에 없을 때 또는 arrRow가 0일 때

leafnode로 만들어준다. 그렇지 않다면 treenode로 만들어주는데 각 feature의 엔트로피를 계산하고 비교하는 방식은 makeFirstNode와 크게 다르지 않다. 한 가지 다른점은 Lnode의 thisArr의 열 중에서 모든 값이 -1인 열이 존재할 수 있는데 그 열은 제외하고 계산을 한다. thisArr에서 모든 값이 -1인 열은 이미 앞에서 정보획득량이 가장 높은 feature로 선택되어 트리에 연결됐음을 의미한다. Lnode뿐 아니라 Rnode도 위와 같은 과정을 통해 생성하며 마지막에는 트리 생성을 재귀적으로 진행하기 위해 Lnode의 tag가 treenode라면 makeTree(Lnode, Lnode->Llink, Lnode->Rlink)를 하였고 역시 Rnode의 tag가 treenode라면 makeTree(Rnode, Rnode->Llink, Rnode->Rlink)를 하였다.

텍스트, 스크린샷, 노트북, 컴퓨터이(가) 표시된 사진

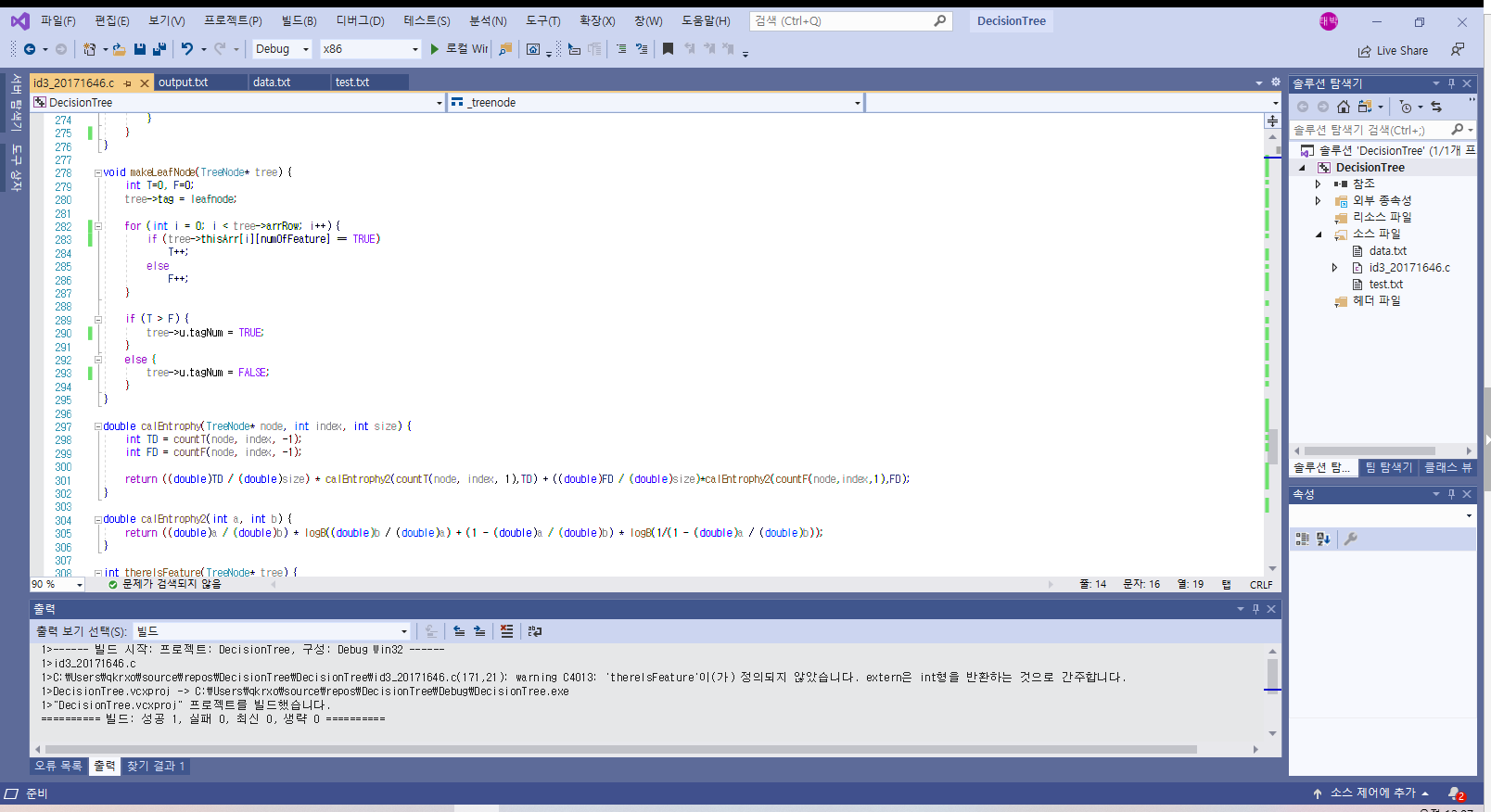
자동 생성된 설명

현재 보고 있는 노드의 thisArr을 만들어주는 함수이다. 매개변수로 현재 보고 있는 노드인 child와 부모 노드인 parent, 왼쪽 자식노드인지 오른쪽 자식노드인지를 알려주는 tf를 받는다. thisArr은 각 노드에 대해 이 배열을 토대로 노드를 구성했음을 의미한다. 반복문을 i=0 ~ 부모노드의 arrRow만큼 돌아 부모노드의 thisArr[i][부모노드의 index]가 tf랑 같을 때 count++을 해주는데, 자식 노드의 arrRow를 결정하기 위함이다. 예를 들어 부모 노드의 index가 1이고 왼쪽 자식 노드를 보고 있다면 부모 노드의 thisArr[1][???]에서 0이 몇 개인지를 세고, 이를 자식 노드 thisArr의 행 크기로 정해주는 것이다. 자식 노드의 thisArr의 크기를 동적 할당을 통해 정해주는데, 크기는 thisArr[count][numOfFeature+1]이 된다. 이후 child->thisArr에 값을 넣어주기 위해 반복문을 실행한다. 반복문은 (parent->arrRow) \* (numOfFeature+1)만큼 실행한다. 예를 들면 결과적으로 다음과 같이 자식 노드의 thisArr이 정해진다.

테이블이(가) 표시된 사진

자동 생성된 설명

즉, 자식노드에서 부모 노드의 index에 해당하는 부분은 -1이며 왼쪽 자식 노드인지 오른쪽 자식 노드인지에 따라 부모 노드 index에서 0또는 1에 해당하는 값만 골라 값을 넣어준다. 만약 앞에서 모든 feature에 대해 노드가 만들어졌다면 thisArr이 클래스를 제외한 부분에서 모두 -1값을 가질 것이다.



리프노드를 만들어주는 함수이다. 해당 노드의 thisArr에서 클래스에 TRUE가 더 많은지 FALSE가 더 많은지를 판단하여 tagNum을 결정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

더 계산할 Feature가 존재하는지를 알려주는 함수이다. class를 제외한 thisArr에서 -1이 아닌 값이 나온다면 TRUE를 반환하고 그렇지 않다면 모든 Feature에 대해 이미 앞에서 노드 생성이 완료된 것이므로 FALSE를 반환한다.

텍스트, 스크린샷, 노트북이(가) 표시된 사진

자동 생성된 설명

출력 결과인 output.txt를 만들어주는 함수이다. test.txt에서 테스트 케이스들을 읽은 뒤 이를 토대로 트리에서 어떤 리프노드에 도달하는지를 출력해준다. ary배열은 한 테스트 케이스를 저장해주는 배열이다. numOfTest만큼 반복문을 실행한다. 반복문 처음에 tmp가 트리의 첫 노드인 tree를 가리키게 하고 ary배열에 test.txt 한 줄에 써져 있는 데이터를 넣어준다. 이후 while(1)반복문을 실행하는데, tmp가 리프노드에 도달한 경우 해당 리프노드의 tagNum을 fprintf해주고 break를 한다. 리프노드에 도달하지 않았다면 tmp를 ary[tmp->u.index]에 따라 이동시킨다. 즉, 현재 보고 있는 노드가 어떤 index값을 가지고 있으며 그에 따라 ary배열의 값을 읽어 왼쪽 링크를 탈 지 오른쪽 링크를 탈 지 결정해준다. for반복문이 끝났다면 동적할당을 해준 ary배열의 메모리 해제를 진행하고 fclose를 해준다.

텍스트, 스크린샷, 노트북, 컴퓨터이(가) 표시된 사진

자동 생성된 설명

트리의 메모리 해제를 담당하는 함수들이다. 트리를 postorder로 순회하면서 메모리 해제를 진행한다. freeTree에서는 해당 노드와 노드 안에 있는 thisArr까지 메모리 해제를 해준다.