

2020 2학기 자료구조 프로젝트 결과보고서

: 서강대학교 학생설계전공 분류 프로그램

20181202 김수미 / 20181228 김효진

1. 주제 소개

1) 주제 : 서강대학교 학생설계전공 분류 프로그램

서강대학교 내 학생설계전공들의 구성 과목 비율을 계산하여 분야별로 전공을 분류하고, 새로 입력되는 전공의 분야를 판별해주는 프로그램

2) 주제 선정 동기

두 팀원 모두 의사결정트리(Decision Tree)에 관심이 있어 이를 활용하여 만들 수 있는 프로그램을 먼저 고안해보기로 하였다. 주제를 논의하는 과정에서 실생활에 밀접하게 도움이 될 수 있는 프로그램을 만들고 싶다는 의견에 동의하였고, 서로 일상 속에서 느꼈던 불편함에 대해 얘기하던 중 서강대학교 학생설계전공 홈페이지가 불편하다는 의견이 나왔다. 학생설계전공의 특성상 수많은 전공이 존재하지만 카테고리별로 묶여있지 않고 단순히 나열되어 있어 설계 전공에 관심있는 학생들이 기존 설계 전공을 확인할 때 불편함을 겪는다는 현실에 공감하였다. 설계 전공의 종류나 구성 과목 등에 대한 정보는 홈페이지에서 충분히 주어져 있었기 때문에 주어진 정보를 바탕으로 전공들을 보기 쉽게 분류하여 설계 전공에 관심있는 학생들, 새로이 설계 전공을 만들고자 하는 학생들에게 도움이 될 수 있는 프로그램을 구현해보기로 하였다.

2. 사용하는 자료구조 및 알고리즘

1) 이진트리(binary tree) 자료구조

이진트리는 각 노드 별로 최대 두 개의 자식 노드를 가질 수 있는 트리이다. 따라서 각 노드 마다 참/거짓의 두 가지 기준에 따라 분류 또는 의사결정을 하는 데 용이한 자료 구조이다. 본 프로젝트에서는 전공을 설계할 때 어떤 기존 전공의 과목을 가져온 것인지에 따라 분류하기 위해 기존 전공들을 노드로 하고, 설계 전공이 해당 전공을 일정 비율 이상 포함하고 있는지의 여부를 참/거짓으로 나누는 이진트리를 구현하였다.

2) ID3 알고리즘의 활용

프로그램 제작을 위해 과제로 제시되기도 하였던 의사결정트리(Decision Tree), 그 중에서도 의사결정트리를 생성하는 데 사용되는 알고리즘 중 초창기 모델인 ID3 알고리즘을 사용하였다.

의사결정트리란 의사결정의 규칙을 트리구조로 나타내고 이를 활용해 전체 데이터를 소집단으로 분류하거나 새 데이터의 분류를 예측하는 분석기법이다. 의사결정트리를 생성할 때 노드로 규칙들을 선택하는 기준이 존재해야 하는데, 이는 기본적으로 불순도(Impurity)가 아래로 갈수록 낮아지는 방향을 택하도록 한다. 불순도를 수치화하는 지표로는 엔트로피(Entropy), 지니계수(Gini Index) 등이 있는데 알고리즘마다 사용하는 불순도 지표에 차이가 있다. ID3 알고리즘은 불순도의 지표로 엔트로피를 채택한 알고리즘이다. 엔트로피는 정보량의 기댓값이다. 정보량은 특정 사건(x)이 가지고 있는 정보의 양을 의미하며, 식으로는 다음과 같이 표현된다.

$$I(x) = \log_2 \frac{1}{p(x)}$$

불순도가 높을수록 엔트로피값 역시 커진다. 불순도가 높다는 것은 각 사건 별로 발생 확률이 비슷하다는 것을 의미한다. 엔트로피는 정보량의 기댓값이므로 식으로는 다음과 같이 표현된다.

$$Entropy = \sum_{i=1}^c p(x_i) I(x_i) = \sum_{i=1}^c p(x_i) \log_2 \frac{1}{p(x_i)}$$

ID3 알고리즘은 데이터들의 엔트로피를 계산하고 비교하여 엔트로피가 작아지는 방향으로 가지를 뻗어나가 의사결정트리를 생성한다.

ID3 이외에도 지니계수를 불순도로 활용하는 CART 알고리즘, C4.5 알고리즘 등 의사결정 트리를 구현할 수 있는 여러 알고리즘이 존재하지만 ID3가 가장 기초적인 알고리즘이어서 기계 학습에 있어 초심자인 본 팀에게 적합하다고 판단했다. 다만 수업시간에 주어진 과제보다 더 발전시켜 응용하는 것을 목표로 두었다. 모든 데이터가 0과 1의 두 가지 클래스로만 이루어졌던 기존 과제와 달리 본 프로젝트에서는 6가지의 클래스로 나누어 데이터를 분류해보고자 한다. 또 교내에 존재하는 기존 학부 전공들이 데이터의 feature로 활용되기 때문에 feature의 개수 또한 과제의 test 데이터보다 많다는 점에서 차이가 있다.

3. 프로젝트 진행 과정

1) 데이터 수집(서강대학교 학생설계전공 과목 리스트)

먼저 Decision Tree의 학습데이터를 만들기 위한 사전 데이터 수집을 진행했다. 해당 프로젝트의 목적은 기존에 존재하는 학생설계전공들을 바탕으로 Decision Tree 분류 프로그램을 만들고, 새롭게 개설하는 학생설계전공의 분야를 파악, 해당 분야에 개설되어 있는 학생설계전공을 안내해 주는 것이기 때문에 가장 먼저 기존에 존재하는 학생설계전공 데이터가 필요하다. 이를 위해 서강대학교 학생설계전공 홈페이지의 학생설계전공 승인 현황 탭 (http://oneself.sogang.ac.kr/front_b/html/002001.html)에서 데이터를 수집했다.

교과목 구성표



*학생설계전공명 : 문화예술컨텐츠학

번호	해당전공	필수/선택	과목번호	과목명	학점
1	교양학부	일반선택	LAU4006	문학과판타지	3
2	교양학부	일반선택	LAU4015	서양미술의역사와감상	3
3	교양학부	일반선택	LAU4021	서양음악의역사와감상	3

〈 그림 1. 서강대학교 학생설계전공 홈페이지에서 확인할 수 있는 데이터 〉

2020년도부터 2017년도까지 승인된 비교적 최근의 학생설계전공만을 대상으로 했으며 홈페이지에서 제공되는 표를 굿어와 엑셀 파일로 정리했다. 학생설계전공의 이름과 해당 설계 전공 이수를 위해 수강할 수 있는 과목 리스트(과목 이름, 과목코드, 과목 소속 전공 등)를 총 59개의 설계 전공을 대상으로 정보를 수집했다.

과학기술사회학(STS)				
수업계열	필수/선택	과목번호	과목명	학점
공공인재 연계전공	전공필수	PUB4002	환경과법	3
교양학부	공통선택	STS2010	과학사	3
교양학부	공통선택	STS2011	기초빅데이터프로그래밍	3
교양학부	공통선택	STS2002	생명과환경	3
교양학부	일반선택	SHU4060	생명의사랑학	3
교양학부	공통선택	STU4011	우주와원자시대	3
교양학부	공통선택	STS2001	자연과인간	3
교양학부	일반선택	ETU4001	환경윤리	3

〈 그림 2. 학생 설계 전공 데이터 : 초기 데이터〉

2) 분류기준 설정 및 데이터 가공

데이터를 저장한 다음, 각 설계전공들을 어떤 기준으로, 어떤 부류들로 분류할 수 있을지를 살펴보았다. 각 설계전공들을 상이한 교과목들로 구성되어 있었으며, 각자 목표하는 학문이나 실용적 가치가 달랐는데, 각 전공들의 특성은 해당 전공을 구성하고 있는 교과목에서 확연하게 나타났다. 특히 공학 계열의 설계 전공은 기존 공학부 소속 전공의 교과목들을 다량 포함하고 있었으며, 인문계열의 설계 전공은 기존 인문학부 소속 전공의 교과목들을 다량 포함하는 것을 확인할 수 있었다. 이를 통해 각 설계 전공을 구성하는 교과목들의 학부 전공 비율을 분류 기준으로 세우고 설계 전공을 약 6가지의 분야로 나눠보기로 하였다. 예를 들어 해당 전공을 구성하는 교과목 중 경영학부 과목의 비율이 높으

면 해당 전공은 경영 분야와 관련이 깊고, 공학부 과목의 비율이 높으면 기술, 공학 분야와 관련이 깊다고 간주하는 것이다.

이를 위해 모든 설계 전공의 구성 과목 중 연계전공(연계전공은 융합적인 전공이 많아 명확한 기준이 되기 힘들다고 생각하여 제외시킴) 및 교양과목으로 개설된 수업들은 모두 제외시키고, 기존 학부 전공만을 남겨 다시 데이터를 정리해 주었다.

과학기술사회학(STS)				
수업개열	필수/선택	과목번호	과목명	학점
국어국문학전공	전공선택	KOR4504	문학과문화콘텐츠	3
미디어&엔터테인먼트전공	전공선택	MAE4015	디지털 미디어 데이터 분석론	3
사회학전공	전공선택	SOC3031	과학사회학	3
사회학전공	전공필수	SOC2003	사회조사방법론	3
사회학전공	전공필수	SOC2001	사회학개론	3
사회학전공	전공선택	SOC3033	이미지와상상력의사회학	3
사회학전공	전공선택	SOC3023	환경사회학	3
아트&테크놀로지전공	전공선택	AAT3015	3D Modeling Studio	3
아트&테크놀로지전공	전공선택	ANT4012	Immersive Media Studio	3

〈 그림 3. 학생 설계 전공 데이터 : 1차 가공 〉

그리고 각 설계 전공을 구성하는 교과목들의 기존 소속 학부 전공(예 : 경영학과, 경제학과 등 기존 학부 소속 전공)비율 계산을 위해, 과목번호만을 제외한 나머지 정보들은 제거해 주었다.

문화예술컨텐츠학	과학기술사회학(STS)	중국통상외교	한국고전문학콘텐츠학
SOC3028	KOR4504	ECO2007	MGT3006
SOC2001	MAE4015	ECO4007	MGT4501
SOC3017	SOC3031	ECO2001	KOR2200
SOC3033	SOC2003	ECO2002	KOR3311
PSY3009	SOC2001	ECO3009	KOR3201
PSY2004	SOC3033	ECO3017	KOR3309
PSY2001	SOC3023	ECO3011	KOR3314
PSY3013	AAT3015	ECO3010	MAE4018

〈 그림 4. 학생 설계 전공 데이터 : 2차 가공 〉

과목 번호에 학과 정보가 들어있기 때문에, 이제 해당 데이터를 가지고 각 설계 전공을 구성하는 전공 과목의 기존 소속 학부 전공 비율(예 : 중국통상외교 전공은 중국문화전공 과목 30%, 정치외교학

전공 70% 로 이루어져 있다)을 계산해 줄 것이다. 엑셀 파일을 input 데이터로 입력 받아 문자열을 읽어 비율을 계산하는 코드를 작성하려고 했는데, C언어는 엑셀 파일을 읽고 문자열을 처리하는데 굉장히 비효율적이었기 때문에 이 부분에서만 따로 Python 언어를 사용해 데이터를 처리해 주었다. 먼저 각 설계 전공 이름과 전공을 구성하는 과목의 개수와 과목 코드들을 엑셀파일에서 추출해 txt 파일로 출력한 다음, 해당 txt 파일을 가지고서 학부 전공 비율을 계산했다. 비율 계산은 설계 전공과 기존 학부 전공을 가나다순으로 정리한 다음, 해당 순서대로 정보를 처리해 주었다. 따라서 최종 비율을 저장한 txt 파일에는 세로축이 가나다순의 설계 전공을 59개를, 가로축이 가나다순의 기존 학부 전공 26개를 나타낸다. 즉 해당 txt 파일의 첫번째 줄이 아래와 같은 경우,

→ 첫번째 줄 : 0 0 0 9 0 0 0 0 36 27 0 0 0 0 0 0 0 18 9 0 0 0 0 0 0

학생 설계 전공을 가나다순으로 정리했을 때 가장 첫번째가 되는 과목은 ‘공공외교전공’ 이므로 해당 줄의 데이터는 공공외교 전공의 정보를 나타내는 것이고, 4, 9, 10, 19, 20번째 예만 숫자가 기록되어 있으므로, 기존 학부 전공을 가나다순으로 정리했을 때 4, 9, 10, 19, 20번째 전공에 해당하는 것들의 교과목들만으로 해당 설계 전공은 이루어져 있는 것이다. 4, 9, 10, 19, 20번째 학부 전공은 글로벌한국학전공, 사학전공, 사회학전공, 정치외교학전공, 종교학전공 이므로 ‘공공외교전공’ 이라는 이름의 학생설계전공은 글로벌한국학전공 과목 9%, 사학전공 과목 36%, 사회학전공 과목 27%, 정치외교학전공 과목 18%, 종교학전공 과목 9% 로 이루어져 있는 것이다. 소수점 아래는 모두 버림을 했기 때문에 퍼센트의 합이 100%가 나오지 않는 경우도 있다.

```
import sys
from openpyxl import load_workbook
file_name = 'std_major.xlsx'
wb = load_workbook(filename = file_name, data_only = True)
ws = wb.active

# set hyperparameters
number = 65

# start iteration
for i in range(7):
    #row = chr(number)
    #row = 'A' + chr(number)
    row = 'B' + chr(number) # 이 때 range(7)
    #row = 'AZ'
    name = row + '1'
    st_num = 2
    fin_num = 33
    print(ws[name].value)

# get information from xlsx file
all_st = []
for i in range(st_num, fin_num + 1):
    cell_num = row + str(i)
    if ws[cell_num].value == None : break
    print(ws[cell_num].value)
    all_st.append(ws[cell_num].value)

# write text file
f = open(ws[name].value + '.txt', 'w')
f.write(str(len(all_st)) + '\n')
for i in range(len(all_st)):
    f.write(all_st[i] + '\n')
f.close()

# go to next step
number += 1

# get major korean name
major_KOR = []
file = open('[0]학생설계전공_리스트.txt', 'r', encoding = 'UTF8')
data = file.readlines()
for x in data:
    major_KOR.append(x[:-1])
file.close()

# start iteration
percent_file = open('[2]percentage.txt', 'w')
for y in range(len(major_KOR)):

    # get major name
    major_str = []
    file = open('[1]기존전공_리스트_Eng.txt', 'r')
    data = file.readlines()
    for x in data:
        major_str.append(x[:-1])
    file.close()

    # compare major name
    major_num = [0 for i in range(len(major_str))]
    file = open(major_KOR[y] + '.txt', 'r')
    line = file.readline()
    while True:
        line = file.readline()
        if not line: break
        for i in range(len(major_str)):
            if line.startswith(major_str[i]):
                major_num[i] += 1
            break

    # calculate percentage
    total = 0
    percent_num = [0 for i in range(len(major_str))]
    for i in range(len(major_num)):
        total += major_num[i]
    for i in range(len(major_num)):
        percent_num[i] = round(major_num[i] / total * 100)

    # save result in text file
    for i in range(len(major_num)):
        percent_file.write(str(percent_num[i]) + ' ')
    percent_file.write('\n')

percent_file.close()
print('calculation complete.')
```

< 그림 5. txt. 파일을 생성하고 비율을 계산하는 파이썬 코드 >



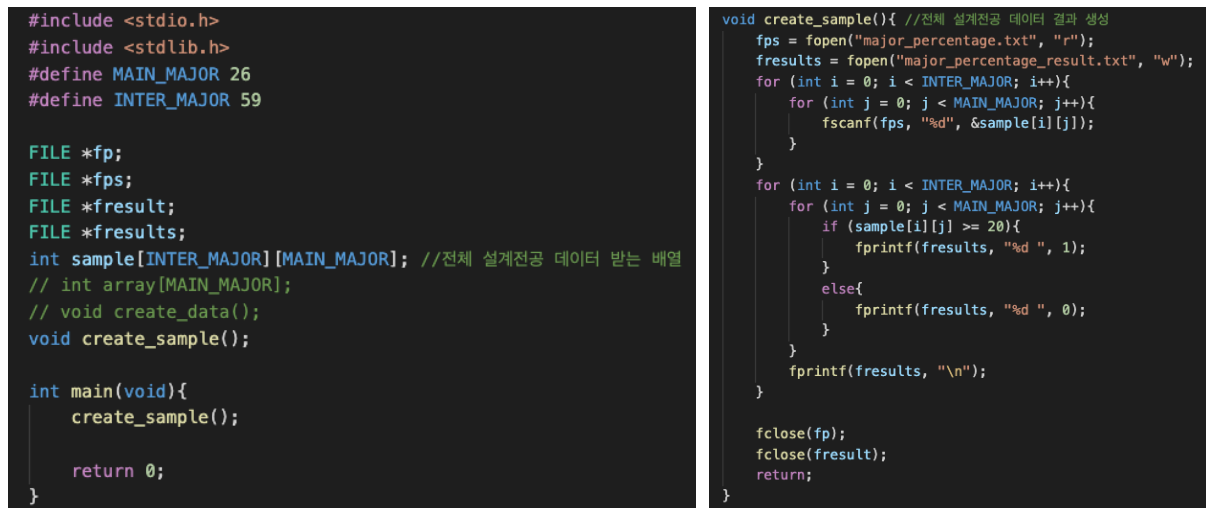
〈 그림 6. 학생 설계 전공 데이터 txt 파일 추출 〉

0	0	0	9	0	0	0	0	36	27	0	0	0	0	0	0	0	0	18	9	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	33	0	29	38	0	0	0	0
0	0	7	0	0	0	0	7	0	33	0	0	0	0	20	13	0	0	0	7	0	13	0	0	0	0
0	0	0	0	0	40	0	0	0	0	0	40	0	0	0	0	0	0	0	0	12	0	0	0	8	
0	0	33	0	0	0	0	22	0	0	0	0	0	33	0	0	0	0	0	0	0	11	0	0	0	
65	0	0	0	0	0	0	24	0	6	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	
0	29	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	
0	62	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	
33	17	0	0	0	0	17	0	0	8	0	0	0	8	0	0	0	0	0	17	0	0	0	0	0	
25	25	0	0	0	0	0	0	0	0	0	50	0	0	0	0	0	0	0	0	0	0	0	0	0	
50	50	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
48	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	7	0	43	0	0	0	14	7	0	0	0	0	0	21	7	0	0	0	
15	30	0	0	0	0	0	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	20	0	0	
0	35	0	0	0	0	0	0	0	59	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	

〈 그림 7. 학생 설계 전공별 학부 과목 비율 계산 txt 파일 내부 〉

이진 트리를 활용해 데이터를 분류할 예정이기 때문에 위 그림의 데이터를 이진 데이터로 변환하는 과정이 필요했다. 트리에서 데이터를 나누는 기준은 기존 학부 전공이 해당 설계 전공에 일정 비중 이상 포함되어 있는지, 일정 수준의 관련성을 갖는지 여부다. 비중이 있다면 참, 그렇지 않다면 거짓으로 나누고자 하였다. 수치로 명확한 기준을 설정하기 위해 학부 전공 과목 비율이 20퍼센트 이상 일 경우 참(1), 미만일 경우 거짓(0)으로 나누기로 하였다. 예를 들어 첫 번째 줄인 ‘공공외교전공’ 데이터의 경우, 과목 비율이 20퍼센트가 넘는 전공은 36%인 사학전공, 27%인 사회학전공이므로 이

데이터에서 1의 값을 갖는 열은 이 두 열이 된다. 이와 같은 과정을 위 데이터를 입력 받아 조건문을 활용해 0과 1로 데이터를 재구성한 후 그 결과를 새로운 txt 파일로 출력하였다. 이 코드는 c로 작성하였다.



```
#include <stdio.h>
#include <stdlib.h>
#define MAIN_MAJOR 26
#define INTER_MAJOR 59

FILE *fp;
FILE *fps;
FILE *fresult;
FILE *fresults;
int sample[INTER_MAJOR][MAIN_MAJOR]; //전체 설계전공 데이터 받는 배열
// int array[MAIN_MAJOR];
// void create_data();
void create_sample();

int main(void){
    create_sample();

    return 0;
}

void create_sample(){ //전체 설계전공 데이터 결과 생성
    fps = fopen("major_percentage.txt", "r");
    fresults = fopen("major_percentage_result.txt", "w");
    for (int i = 0; i < INTER_MAJOR; i++){
        for (int j = 0; j < MAIN_MAJOR; j++){
            fscanf(fps, "%d", &sample[i][j]);
        }
    }
    for (int i = 0; i < INTER_MAJOR; i++){
        for (int j = 0; j < MAIN_MAJOR; j++){
            if (sample[i][j] >= 20){
                fprintf(fresults, "%d ", 1);
            }
            else{
                fprintf(fresults, "%d ", 0);
            }
        }
        fprintf(fresults, "\n");
    }

    fclose(fp);
    fclose(fresult);
    return;
}
```

〈 그림 8. 이진 데이터로 변환하는 코드 〉

모든 작업을 마친 후 각 설계 전공을 개설 목적과 구성 교과목을 기준으로 6가지의 분야로 분류해 주었다. 분야는 교육, 국제정치 및 외교, 기술공학, 기업경영 및 금융, 응용과학, 인문/예술 6개로 구성되어 있으며, 각 분야에 A부터 F까지의 태그를 달아 각 설계 전공에 걸맞는 분야의 태그를 붙여 주었다. 아래는 태그까지 포함해 최종 학습 데이터 txt 파일을 완성한 것이다.

0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	B
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	F
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	C
0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	C
0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	A
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	D
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	B
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	B

〈 그림 9. 최종 학습 데이터 input.txt 파일 내부 〉

3) ID3 알고리즘 코드 작성

알고리즘은 수업시간 과제에서 다루었던 정보 이론의 정보량과 엔트로피를 이용해 Decision Tree를 구성하는 ID3 알고리즘을 활용했다.

학습데이터는 총 59개의 설계 전공 데이터와, 각 전공의 구성 학부 전공(26가지) 비율을 0과 1로 나 타낸 26개의 feature, 6가지의 class로 구성된다. 즉 총 데이터 개수는 59개, feature 개수는 26개,

class. 개수는 6개 이다. 트리를 구성하는 과정은 기존 ID3 알고리즘과 동일하며, 전체 데이터의 class 구성 비율과 각 데이터의 feature index가 0~25 이면서 class가 1~6인 비율을 계산해 각 feature로 데이터를 분류했을 때의 정보량을 계산하고, 가장 정보 획득량이 큰 feature로 매 기준점을 만들면서 Decision Tree를 구성한다.

Decision Tree가 완성되면 새로운 설계 전공의 정보를 입력 받아 해당 설계 전공이 6가지 클래스 중 어떤 부류에 속하는지와 해당 전공에 개설되어 있는 설계 전공 리스트를 보여줌으로써 설계 전공을 새롭게 개설하고자 하는 학생이 자신의 설계 전공이 어떤 특성을 가지는지, 비슷한 특성을 가지는 설계 전공은 어떤 것들이 개설되어 있는지를 살펴볼 수 있게 해 준다.

새로운 설계 전공의 정보를 입력 받는 과정은 다음과 같다. 먼저 새로운 설계 전공의 이름을 입력 받고, 해당 설계 전공이 어떤 전공 과목들로 이루어져 있는지, 그 개수를 질문한다(예 : 경영학 전공 과목의 개수를 입력해주세요). 총 26가지의 전공에 대해 같은 질문을 모두 응답하면, 코드는 답변을 바탕으로 기존 input 데이터와 동일하게 해당 전공의 구성 과목 학부 전공 비율을 계산해 20% 이상이면 1, 미만이면 0으로 반환해 새로운 설계 전공에 대한 데이터를 생성한다. 새롭게 만들어진 데이터를 Decision Tree에 넣어주면 만들어진 기준을 바탕으로 해당 설계 전공이 어떤 클래스 부류에 속하는지를 계산해 보여준다.

4) 코드 세부 설명

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <math.h>
4  #define log2(x) log(x)/log(2)
5  int data_num, feature_num;
6  int data[500][500];
7  int data_class[500];
8  int nodes[150000] = { NULL };
```

먼저 코드 작성에 필요한 헤더파일과 전역변수, 매크로 등을 선언해준다. 엔트로피 계산에 필요한 log2 매크로는 위와 같이 작성해 주었다. data_num과 feature_num은 전체 데이터와 feature 개수를 저장하고, 2차원 배열 data는 input.txt 파일의 모든 설계전공 데이터를, data_class 배열은 모든 설계전공의 class 를 저장한다. nodes 배열은 트리에 저장되는 모든 노드 데이터와 노드번호(1부터 시작)를 저장하는 배열이다. 이미 분류에 사용된 feature를 구별하기 위해, 인덱스로 접근해 정보를 얻을 수 있도록 하기 위해 일차원 배열에 노드 정보들을 따로 기록해 두었다.


```

10 // 엔트로피 계산함수 1
11 float s_ent(int num, int num_class) {
12     float output;
13     if (num == 0 || num_class == 0) return 0;
14     output = ((float)num_class / (float)num) * log2(((float)num / (float)num_class));
15     return output;
16 }
17 // 엔트로피 계산함수 2
18 float m_ent(int num, int by_class[]) {
19     float fin = (float)num / (float)data_num;
20     float sum = 0;
21     for (int i = 0; i < 6; i++)
22         sum += s_ent(num, by_class[i]);
23     sum *= fin;
24     return sum;
25 }
26 // 엔트로피 계산함수 3
27 float l_lent(int zero_num, int one_num, int zero_class[], int one_class[]) {
28     float output = 0;
29     output += m_ent(zero_num, zero_class);
30     output += m_ent(one_num, one_class);
31     return output;
32 }

```

위 세 함수는 엔트로피 계산에 사용되는 함수이다. 설계 전공 데이터의 클래스 개수가 6개나 되기 때문에 하나의 함수에 엔트로피 계산식을 작성하는 경우 너무 복잡해 질 수 있어 세 함수에 걸쳐 값을 계산해 주었다.

```

34 // ----- 0. 6개의 요소 중 MAX값을 return 하는 함수 -----
35 int checkMax(int a, int b, int c, int d, int e, int f) {
36     if (a >= b && a >= c && a >= d && a >= e && a >= f) return 1;
37     else if (b >= a && b >= c && b >= d && b >= e && b >= f) return 2;
38     else if (c >= a && c >= b && c >= d && c >= e && c >= f) return 3;
39     else if (d >= a && d >= b && d >= c && d >= e && d >= f) return 4;
40     else if (e >= a && e >= b && e >= c && e >= d && e >= f) return 5;
41     else if (f >= a && f >= b && f >= c && f >= d && f >= e) return 6;
42 }

```

6개의 매개변수 중 가장 큰 값의 class를 return 하는 함수이다. 더이상 Decision Tree의 노드를 생성할 수 없을 때(사용 가능한 feature가 없거나, 노드의 개수가 정해진 한도를 넘어가는 경우) 과반수의 클래스 번호로 리프 노드를 형성해 주는 데에 사용된다. 텍스트 파일에서는 A ~ F 의 6가지 데이터 클래스가 있었지만 이를 다루기 쉽게 1에서 6의 정수값으로 변환하여 사용하였다.

```

44 // ----- 1. 트리를 구성하는 노드. 리프노드는 자식노드가 NULL -----
45 typedef struct treeNode *treePtr;
46 □ typedef struct treeNode {
47     int data;
48     treePtr leftchild;
49     treePtr rightchild;
50 } treeNode;
51
52 // ----- 2. 트리에 새로운 노드를 추가하는 함수 -----
53 □ treePtr insert_node(treePtr node, int num) {
54     treePtr ptr = (treePtr)malloc(sizeof(*ptr));
55     ptr->data = num;
56     ptr->leftchild = ptr->rightchild = NULL;
57     if (node->leftchild == NULL) node->leftchild = ptr;
58     else node->rightchild = ptr;
59     return ptr;
60 }

```

트리를 구성하는 노드 구조체이다. 데이터 필드와 leftchild, rightchild 포인터로 구성되어 있다. 리프 노드는 leftchild, rightchild가 가리키는 것이 없이 NULL 이다.

```

62 // ----- 3. 모두 같은 class를 가지는지 check하는 함수 -----
63 □ int check_class(int data_idx[], int num_data) {
64     int major_class[6] = { 0 };
65     for (int i = 0; i < num_data; i++) {
66         for (int j = 0; j < 6; j++)
67             if (data_class[data_idx[i]] == j + 1) major_class[j]++;
68     }
69     if (major_class[1] == 0 && major_class[2] == 0 && major_class[3] == 0 && major_class[4] == 0 && major_class[5] == 0) return 1;
70     else if (major_class[0] == 0 && major_class[2] == 0 && major_class[3] == 0 && major_class[4] == 0 && major_class[5] == 0) return 2;
71     else if (major_class[0] == 0 && major_class[1] == 0 && major_class[3] == 0 && major_class[4] == 0 && major_class[5] == 0) return 3;
72     else if (major_class[0] == 0 && major_class[1] == 0 && major_class[2] == 0 && major_class[4] == 0 && major_class[5] == 0) return 4;
73     else if (major_class[0] == 0 && major_class[1] == 0 && major_class[2] == 0 && major_class[3] == 0 && major_class[5] == 0) return 5;
74     else if (major_class[0] == 0 && major_class[1] == 0 && major_class[2] == 0 && major_class[3] == 0 && major_class[4] == 0) return 6;
75     else return 0;
76 }

```

데이터 집합이 모두 같은 클래스를 가지는 데이터들로 구성되어 있는지를 check 해주는 함수이다. 모두 같은 클래스로 구성되어 있다면 해당 클래스 번호를 반환하고, 그렇지 않은 경우 0을 반환한다.

```

78 // ----- 4. 데이터 feature별 entropy 계산 함수 -----
79 float feature_entropy(int feature_class[], int data_idx[], int data_number) {
80     float yes = 0, no = 0, result;
81     float yes_yes = 0, yes_no = 0;
82     float no_yes = 0, no_no = 0;
83     int zero_class[6] = { 0 }, one_class[6] = { 0 };
84
85     // 현재 feature에서 참/불참이면서 class가 1~6
86     for (int i = 0; i < data_number; i++) {
87         for (int j = 0; j < 6; j++)
88             if (feature_class[i] == 0 && data_class[data_idx[i]] == j + 1) zero_class[j]++;
89         for (int j = 0; j < 6; j++)
90             if (feature_class[i] == 1 && data_class[data_idx[i]] == j + 1) one_class[j]++; }
91
92     // 0, 1 의 총 개수 count
93     for (int i = 0; i < 6; i++)
94         no += zero_class[i];
95     for (int i = 0; i < 6; i++)
96         yes += one_class[i];
97
98     // entropy를 계산한다
99     result = l_lent(no, yes, zero_class, one_class);
100     return result;
101 }

```

데이터의 feature별 entropy를 계산하는 함수이다. 매개변수로 데이터들의 각 feature(0~25/26가지)에 해당하는 값들을 입력 받으며 해당 값들과 해당 데이터의 클래스 값을 기준으로 엔트로피를 계산해서 return해준다. 위에서 선언한 엔트로피 계산함수들을 사용한다.

```

103 // ----- 5. 정보 획득량이 가장 큰 feature의 index를 찾는 함수 -----
104 int max_info(int data_idx[], int data_len, int for_node) {
105     float min_entropy = 100, flag = 0;
106     int min_feature_idx = 0, for_for = for_node;
107     int *feature = malloc(sizeof(int) * data_len);
108     for (int feature_idx = 0; feature_idx < feature_num; feature_idx++) {
109         for_for = for_node;
110         for (int i = 0; i < data_len; i++) {
111             feature[i] = data[data_idx[i]][feature_idx];
112         }
113         if (feature_entropy(feature, data_idx, data_len) < min_entropy) {
114             while (for_for != 0) {
115                 if (feature_idx == nodes[for_for]) {
116                     flag = 1;
117                     break; }
118                 for_for /= 2; }
119             if (flag == 1) { flag = 0; continue; }
120             min_entropy = feature_entropy(feature, data_idx, data_len);
121             min_feature_idx = feature_idx;
122         }
123     }
124     free(feature);
125     return min_feature_idx;
126 }

```

각 feature로 나누었을 때 정보획득량이 가장 큰 feature가 무엇인지 계산하는데 사용되는 함수이다. 매번 클래스의 엔트로피는 고정된 값 이므로, 정보획득량은 (클래스 엔트로피 - feature 엔트로피)

feature 엔트로피 값이 가장 작을 때 가장 크다. 따라서 26개의 feature 중 가장 엔트로피 값이 작은 feature를 Decision Tree의 결정 노드로 선택해주면 된다. 가장 작은 feature의 인덱스 값(0 ~ 25 중 하나)를 return 한다. 한 노드의 feature 번호를 결정할때, 해당 노드의 조상 노드에서 이미 분류 기준으로 사용된 feature 번호가 다시 선택되어서는 안된다. 이를 체크해 주기 위해 전역변수 선언에서 nodes 배열을 선언해 준 것이며, 함수 내부에서 해당 배열을 이용해 조상 노드에서 이미 분류 기준으로 사용된 feature 번호가 다시 선택되는 것을 피할 수 있도록 해주었다.

```

128 // ----- 6. leftchild와 rightchild를 만드는 함수 -----
129 void make_child(treePtr node, int feature_idx, int for_node, int data_idx[], int data_len) {
130     treePtr temp1, temp2;
131     int for_node1 = for_node, for_node2 = for_node;
132     int *yes_data = malloc(sizeof(int) * 500);
133     int *no_data = malloc(sizeof(int) * 500);
134     int check, y_idx = 0, n_idx = 0;
135     int left_feature = 0, right_feature = 0;
136     int y_class_num = 0, n_class_num = 0;
137     int major_class[6] = { 0 };
138     int max;
139
140     // leftchild 데이터 집합과 rightchild 집합을 분류
141     for (int i = 0; i < data_len; i++) {
142         if (data[data_idx[i]][feature_idx] == 1) { yes_data[y_idx] = data_idx[i]; y_idx++; }
143         else if (data[data_idx[i]][feature_idx] == 0) { no_data[n_idx] = data_idx[i]; n_idx++; }
144     }
145
146     // 모든 데이터의 class count
147     for (int i = 0; i < data_len; i++) {
148         for (int j = 0; j < 6; j++)
149             if (data_class[data_idx[i]] == j + 1) major_class[j]++;
150     }
151

```

본격적으로 트리의 노드들을 방울방울 생성해내는 함수이다. 이 코드에서 가장 많은 부분을 차지하며 재귀적인 형태를 띤다. 먼저 함수 선언에 필요한 변수들을 선언해준다.

해당 함수는 매번 현재 노드에서 분류 기준이 되는 feature의 인덱스와 분류 대상이 되는 데이터 집합을 매개변수로 받아온다(데이터 집합은 인덱스만 받아오며, 해당 인덱스를 통해 전체 데이터를 저장하고 있는 data 2차원 배열에서 값을 참조할 수 있다).

가장 먼저 하는 일은 받은 데이터 리스트에서 분류 기준이 되는 feature에서의 값들을 0과 1로 구분하는 것이다. 예를 들어 1, 3, 5번 인덱스 데이터를 받았는데 현재 분류 기준 feature 인덱스가 3 이라면, 1, 3, 5번 데이터의 3번 feature 값이 0인지 1인지를 확인하는 것이다. 모든 결정 노드는 데이터를 해당 노드의 feature 인덱스를 기준으로 하여 0의 값을 가지는지 1의 값을 가지는지로 분류하므로 해당 작업을 진행하는 것이다. 1의 값을 가지는 데이터 인덱스들은 yes_data 배열에, 0의 값을 가지는 데이터 인덱스들은 no_data 배열에 저장되며 배열의 길이는 y_idx와 n_idx에 저장된다.

```

152 // 전달 받은 데이터의 class가 모두 같은 경우
153 if (major_class[1] == 0 && major_class[2] == 0 && major_class[3] == 0 && major_class[4] == 0 && major_class[5] == 0) return;
154 else if (major_class[0] == 0 && major_class[2] == 0 && major_class[3] == 0 && major_class[4] == 0 && major_class[5] == 0) return;
155 else if (major_class[0] == 0 && major_class[1] == 0 && major_class[3] == 0 && major_class[4] == 0 && major_class[5] == 0) return;
156 else if (major_class[0] == 0 && major_class[1] == 0 && major_class[2] == 0 && major_class[4] == 0 && major_class[5] == 0) return;
157 else if (major_class[0] == 0 && major_class[1] == 0 && major_class[2] == 0 && major_class[3] == 0 && major_class[5] == 0) return;
158 else if (major_class[0] == 0 && major_class[1] == 0 && major_class[2] == 0 && major_class[3] == 0 && major_class[4] == 0) return;
159
160 // 배열 최대치를 초과하는 경우 트리 생성 중단
161 if (for_node * 2 > 65535) {
162     max = checkMax(major_class[0], major_class[1], major_class[2], major_class[3], major_class[4], major_class[5]);
163     temp1 = insert_node(node, max);
164     for_node1 *= 2;
165     nodes[for_node1] = max;
166     return;
167 }

```

그 다음 하는 작업은 매개변수로 전달받은 데이터들의 클래스가 모두 같은지를 확인하는 것이다. 데이터의 클래스가 모두 같은 경우 더이상 분류를 진행할 필요가 없으므로 함수를 중단한다.

또한 배열 최대치를 초과하는 경우에도 트리 생성을 중단한다. 여기서 배열 최대치라는 것은 nodes 배열의 최대치를 의미한다. 원래 설계전공데이터는 26개의 feature를 가지기 때문에 최대 2의 26제곱(67,108,864)개까지 노드가 생성될 수 있다. 하지만 1차원 배열의 최대 크기는 약 25만 이기 때문에 해당 Decision Tree의 최대 height를 17로 제한해 주었다. 최대치를 초과해 nodes 배열을 사용하지 못하는 경우 조상 노드에서 이미 사용된 feature number를 확인할 수 없기 때문에 배열의 크기에 트리의 크기를 어쩔 수 없이 맞춰 주었다.

```

169 else { // leftchild 노드 생성
170     check = check_class(yes_data, y_idx);
171     if (check != 0) { // 데이터의 class가 모두 같은 경우
172         temp1 = insert_node(node, check);
173         for_node1 *= 2;
174         nodes[for_node1] = check;
175         make_child(temp1, left_feature, for_node1, yes_data, y_idx);
176     }
177     else {
178         left_feature = max_info(yes_data, y_idx, for_node1);
179         for_node1 *= 2;
180         nodes[for_node1] = left_feature;
181         temp1 = insert_node(node, left_feature);
182         make_child(temp1, left_feature, for_node1, yes_data, y_idx);
183     }

```

전달받은 데이터들의 클래스가 모두 같지 않은 경우, 다시 데이터 분류 작업을 진행한다. 분류 기준 feature 인덱스에서 1의 값을 가지는 데이터들은 leftchild 노드로, 0의 값을 가지는 데이터들은 rightchild 노드로 전달된다. 전달되는 것은 데이터 인덱스이며 이를 이용해 다음 노드에서는 데이터의 세부 정보를 data 2차원 배열에서 참조할 수 있다.

먼저 1의 값을 가지는 데이터들을 leftchild노드로 전달해주기 위한 과정을 살펴보자. 1의 값을 가지는 데이터들을 모아봤는데 해당 데이터들이 모두 같은 클래스 번호를 가지고 있다면, 해당 데이터를 전달받는 leftchild 노드는 feature 번호가 아닌 클래스 번호를 전달받게 되며, 리프노드가 된다. 해당 작업을 check 함수를 이용해 수행해 주었다.

1을 값을 가지는 데이터들이 모두 같은 클래스 번호를 가지는게 아니라면, 해당 데이터들을 전달받는 leftchild의 분류 기준이 될 feature 번호를 결정해 주어야 한다. max_info 함수를 통해 해당 데이터 집합에서 가장 많은 정보획득량을 가지는 feature 번호를 결정하고, 해당 feature 번호를 data 필드로 가지는 노드를 생성한 뒤에, 해당 노드와 분류기준으로 결정한 feature 번호, 데이터 리스트, 데이터 개수 등을 매개변수로 가지는 make_child 함수를 재귀호출해준다. for_node 변수를 이용해 매번 생성되는 트리 노드 번호와 노드의 data field 정보를 nodes 배열에도 저장한다.

```

185 // rightchild 노드 생성
186 check = check_class(no_data, n_idx);
187 if (check != 0) { // 데이터의 class가 모두 같은 경우
188     temp2 = insert_node(node, check);
189     for_node2 = (for_node2 * 2) + 1;
190     nodes[for_node2] = check;
191     make_child(temp2, right_feature, for_node2, no_data, n_idx);
192 }
193 else {
194     right_feature = max_info(no_data, n_idx, for_node2);
195     for_node2 = (for_node2 * 2) + 1;
196     nodes[for_node2] = right_feature;
197     temp2 = insert_node(node, right_feature);
198     make_child(temp2, right_feature, for_node2, no_data, n_idx);
199 }
200 }
201 free(yes_data);
202 free(no_data);
203 }

```

다음은 0의 값을 가지는 데이터들을 rightchild노드로 전달해주기 위한 과정이다. 전체적인 구조는 1의 값을 가지는 데이터들을 leftchild노드로 전달해주는 과정과 동일하다. 1의 값을 가지는 데이터들이 모두 같은 클래스 값을 가지는지를 먼저 체크하고, 해당하지 않는다면 분류 기준이 될 feature 번호를 정해 노드를 새로 생성한 다음 다시 make_child 함수를 재귀호출한다.

```

205 // -----7. decisiontree를 생성하는 함수 -----
206 treePtr create_tree(int data_index[]) {
207     int check, feature_idx, for_node = 0;
208     treePtr rootNode = (treePtr)malloc(sizeof(*rootNode));
209
210     // 처음 입력받은 데이터의 클래스가 모두 같은 경우
211     check = check_class(data_index, data_num);
212     if (check != 0) {
213         printf("아무런 분류도 이루어지지 않았습니다.");
214         rootNode->data = check;
215         rootNode->leftchild = NULL;
216         rootNode->rightchild = NULL;
217         return rootNode;
218     }

```

이제 루트노드를 만들고 make_child 함수를 최초로 호출하여 Decision Tree를 본격적으로 생성하

는 create_tree 함수를 살펴보자. 먼저 처음 입력 받은 설계 전공 데이터의 클래스가 모두 같은 경우에는 이를 안내하는 문구를 출력해주도록 했다.

```
220      // decision tree의 루트노드 생성
221      feature_idx = max_info(data_index, data_num, for_node);
222      rootNode->data = feature_idx;
223      rootNode->leftchild = NULL;
224      rootNode->rightchild = NULL;
225      for_node++;
226      nodes[for_node] = feature_idx;
227
228      // 루트 노드 밑으로 새로운 트리 생성
229      make_child(rootNode, feature_idx, for_node, data_index, data_num);
230      return rootNode;
231  }
```

설계 전공 데이터의 클래스가 모두 같지 않은 경우, 루트노드를 생성해 Decision Tree 만드는 것을 시작한다. 맨 처음 루트노드의 분류 기준 feature 번호를 결정하고, 해당 feature 번호와 모든 데이터 리스트 등을 make_child 에 매개변수로 전달해 준다. Decision Tree의 생성이 끝나면 루트노드의 주소를 return 한다.

```
233      // ----- 8. Decision Tree 구성 노드 메모리 할당 해제 함수 -----
234  void postorderTraverse(treePtr np) {
235      if (np == NULL) return;
236      postorderTraverse(np->leftchild);
237      postorderTraverse(np->rightchild);
238  }
239  void delTree(treePtr np) {
240      if (np != NULL) {
241          postorderTraverse(np->leftchild);
242          postorderTraverse(np->rightchild);
243          free(np);
244      }
245  }
```

위 함수는 Decision Tree의 사용이 모두 끝난 다음 사용된 노드의 메모리를 모두 해제시키는 함수이다. post traversal을 이용해 모든 노드들을 방문하며, delTree 함수를 이용해 각 노드들을 free 시킨다.


```

247 // ----- 9. main 함수 -----
248 int main() {
249     int data_feature, feature_idx;
250     char major_name[20];
251     char basic_major[26][50];
252     char class_list[6][50];
253     char student_major[59][50];
254     int major_num = 0, total_major_num = 0;
255     float major_percent = 0;
256     int total_major[26];
257     int test[26];
258
259     // data.txt 파일 오픈 및 정보 저장
260     FILE *fp;
261     fp = fopen("input.txt", "r");
262     if (fp == NULL) {
263         fprintf(stderr, "file open error \n");
264         exit(1); }
265     fscanf(fp, "%d%d", &data_num, &feature_num);
266     for (int i = 0; i < data_num; i++) {
267         for (int j = 0; j < feature_num; j++)
268             fscanf(fp, "%d", &data[i][j]);
269         fscanf(fp, "%d", &data_class[i]);
270     } fclose(fp);
271

```

메인 함수 부분이다. 가장 먼저 설계전공 데이터 정보를 가지고 있는 input.txt 파일을 입력 받아 데이터 개수, feature 개수, 데이터 값들을 변수와 배열에 저장한다.

```

272 // 기존 학부 전공 리스트 저장
273 fp = fopen("major_list.txt", "r");
274 if (fp == NULL) {
275     fprintf(stderr, "file open error \n");
276     exit(1); }
277 for (int i = 0; i < 26; i++)
278     fscanf(fp, "%s", &basic_major[i]);
279 fclose(fp);

```

그 다음 기존 학부 전공 이름 리스트가 저장되어 있는 major_list.txt 파일을 입력 받아 basic_major 배열에 저장한다. 해당 문자열들은 이후 콘솔창에 질문을 출력하는 부분에서 사용될 것이다.

```

281 // 전공 분야 리스트 저장
282 fp = fopen("class_list.txt", "r");
283 if (fp == NULL) {
284     fprintf(stderr, "file open error Wn");
285     exit(1); }
286 for (int i = 0; i < 6; i++)
287     fscanf(fp, "%s", &class_list[i]);
288 fclose(fp);
289
290 // 설계 전공 리스트 저장
291 fp = fopen("student_major.txt", "r");
292 if (fp == NULL) {
293     fprintf(stderr, "file open error Wn");
294     exit(1); }
295 for (int i = 0; i < 59; i++)
296     fscanf(fp, "%s", &student_major[i]);
297 fclose(fp);

```

그 다음 전공 분야의 이름 리스트가 저장되어 있는 class_list.txt 파일을 입력 받아 class_list 배열에 저장한다. 해당 문자열들은 이후 사용자가 입력한 새로운 설계 전공의 분야를 출력해주는 데에 사용될 것이다.

```

299 // 초기 데이터 idx 배열 생성
300 int *data_index = malloc(sizeof(int) * data_num);
301 for (int i = 0; i < data_num; i++)
302     data_index[i] = i;
303
304 // 새로운 설계전공 데이터 받기
305 printf("새로운 설계전공의 이름을 입력해 주세요 (띄어쓰기제외/20자 이내)WnWn>> ");
306 scanf("%s", major_name);
307 printf("Wn다음은 해당 전공을 구성하는 학과 과목에 대한 질문입니다.WnWn");
308 for (int i = 0; i < 26; i++) {
309     printf("- %s전공 과목의 개수를 입력해주세요 : ", basic_major[i]);
310     scanf("%d", &major_num);
311     total_major[i] = major_num;
312     total_major_num += major_num; }
313 for (int i = 0; i < 26; i++) {
314     major_percent = (float)total_major[i] / (float)total_major_num;
315     if (major_percent > 0.2) test[i] = 1;
316     else test[i] = 0; }
317
318 // decisiontree 생성
319 treePtr rootNode, temp;
320 rootNode = create_tree(data_index);
321 temp = rootNode;
322 feature_idx = temp->data;

```

그 다음 초기 데이터 인덱스 배열을 생성한다. 루트 노드 생성에 사용되는 배열로, 루트노드에서 다루는 데이터 집합은 전체 데이터에 대한 집합이므로, 해당 배열은 모든 데이터의 인덱스값을 포함하고 있다. 이후 create_tree 함수를 호출할 때 사용할 것이다.

그리고 트리를 생성하기 전에, 먼저 새로운 클래스 타입을 지정받을 새로운 설계 전공에 대한 데이터

를 받기로 하자. 콘솔창에서 해당 설계 전공의 이름과 설계 전공을 구성하는 학부 전공별 과목 수를 입력받는다. 해당 값을 기준으로 새로운 설계 전공을 구성하는 과목들의 전공 비율을 계산하고, 20%가 넘는 경우 값을 1로 반환해 처음 기존 설계전공에 대한 정보를 저장하고 있는 input.txt 파일의 각 행과 동일한 형식(클래스 값만 빠진)으로 새로운 설계전공 데이터를 생성해준다. 해당 데이터는 test 배열에 차례대로 저장된다. 저장되는 값은 26개의 feature에 대한 0 또는 1의 값이다.

test 배열을 생성한 다음에는 create_tree 함수를 호출하여 Decision Tree를 생성해 주도록 한다. 그리고 루트 노드의 분류 기준으로 잡힌 feature 번호를 feature_idx라는 변수에 저장한다.

```

324 // test 데이터 분류 : 자식노드가 둘 다 NULL이 아니면 계속 반복
325 while (temp->leftchild != NULL && temp->rightchild != NULL) {
326     data_feature = test[feature_idx];
327     if (data_feature == 1) temp = temp->leftchild;
328     else if (data_feature == 0) temp = temp->rightchild;
329     feature_idx = temp->data;
330     //printf("%d %d %n", temp->leftchild->data, temp->rightchild->data);
331 }
332 // 자식노드의 둘 중 하나가 NULL인 경우 (무조건 left)
333 if (temp->leftchild != NULL && temp->rightchild == NULL) {
334     temp = temp->leftchild;
335     feature_idx = temp->data;
336 } // 리프노드의 데이터 값 반환
337
338 printf("\n< '%s'은 '%s' 분야 입니다. >\n", major_name, class_list[feature_idx-1]);
339 printf("< 아래는 해당 분야에 개설되어있는 설계전공 리스트 입니다. 확인해보세요. >\n\n");
340 for (int i = 0; i < 59; i++) {
341     if (data_class[i] == feature_idx) printf(">> %s\n", student_major[i]);
342 } printf("\n");
343
344 // 생성된 트리의 모드 노드 메모리 할당 해제
345 delTree(rootNode);
346
347 free(data_index);
348 system("pause");
349 return 0;
350 }

```

이제 생성한 test 배열에 저장되어 있는 값을 바탕으로 새로운 설계 전공의 클래스를 결정할 차례이다. 루트 노드의 feature 번호를 시작으로 1또는 0의값에 따라 Decision Tree의 left 또는 rightchild로 이동하고, 더 이상 이동할 수 있는 노드가 없을때 까지(리프노드에 도달할 때 까지) 이를 반복한다. 리프노드에 도달하면 리프노드의 데이터 값을 반환하며, 이는 새로운 설계전공의 클래스가 된다.

전달받은 클래스 번호를 바탕으로 새로운 설계전공의 분야를 출력해주고, 해당 분야에 개설되어 있는 기존 설계전공들의 리스트를 출력해 줌으로써 프로그램은 종료된다.

4. 실행 결과 및 활용 방안

1) 코드 사용 방법 및 실행 결과

코드를 실행시키면 제일 먼저 아래와 같은 문구를 콘솔창에서 확인할 수 있다.

```
새로운 설계전공의 이름을 입력해 주세요 (띄어쓰기제외/20자 이내)  
>>
```

〈 그림 10. 설계 전공 이름 입력 화면 〉

새롭게 개설하고자 하는 설계 전공의 이름을 띄어쓰기 없이, 20자 이내로 입력하면 아래와 같이 구성 과목에 대한 질문이 나오게 된다. 질문에 차례대로 답을 입력해 주면 된다. 학생이 설계 전공을 이수 하기 위해 수강해야 하는 과목 리스트 구성을 모두 마친 상태라고 가정한다.

```
새로운 설계전공의 이름을 입력해 주세요 (띄어쓰기제외/20자 이내)  
>> 토목공학전공  
다음은 해당 전공을 구성하는 학과 과목에 대한 질문입니다.  
- 경영학전공 과목의 개수를 입력해주세요 : _
```

〈 그림 11. 설계 전공 이름 입력 완료 시 나오는 질문 〉

```
- 경영학전공 과목의 개수를 입력해주세요 : 2  
- 경제학전공 과목의 개수를 입력해주세요 : 3  
- 국어국문학전공 과목의 개수를 입력해주세요 : 0  
- 글로벌한국학전공 과목의 개수를 입력해주세요 : 0  
- 기계공학전공 과목의 개수를 입력해주세요 : 10  
- 물리학전공 과목의 개수를 입력해주세요 : 9  
- 미국문화전공 과목의 개수를 입력해주세요 : 0  
- 미디어&엔터테인먼트전공 과목의 개수를 입력해주세요 : 0  
- 사학전공 과목의 개수를 입력해주세요 : 0  
- 사회학전공 과목의 개수를 입력해주세요 : 0  
- 생명과학전공 과목의 개수를 입력해주세요 : 0  
- 수학전공 과목의 개수를 입력해주세요 : 11  
- 신문방송학전공 과목의 개수를 입력해주세요 : 0  
- 심리학전공 과목의 개수를 입력해주세요 : 0  
- 아트&테크놀로지전공 과목의 개수를 입력해주세요 : 3  
- 영미어문전공 과목의 개수를 입력해주세요 : 0
```

〈 그림 12. 질문에 답변을 완료한 상태 〉

모든 질문에 대답을 완료하면 답변 내용을 바탕으로 프로그램이 0과 1로 이루어진 데이터를 생성하고, 이를 Decision Tree에 집어넣어 새롭게 입력한 설계전공의 분야를 판별해 준다. 그리고 해당 분야에 이미 개설되어 있는 설계전공(2020~2017년 기준)의 리스트를 콘솔창에 print 해 학생이 이를 확인할 수 있게 해준다.

```
< '토목공학전공'은 '기술공학' 분야 입니다. >
< 아래는 해당 분야에 개설되어있는 설계전공 리스트 입니다. 확인해보세요. >

>> 과학기술사회학(STS)
>> 과학적모델링
>> 물리화학
>> 전산&물리기반심화화학

계속하려면 아무 키나 누르십시오 . . .
```

〈 그림 13. 구성 전공 과목 질문에 답변한 후 확인할 수 있는 화면 〉

```
새로운 설계전공의 이름을 입력해 주세요 (띄어쓰기제외/20자 이내)
>> 언어학전공
다음은 해당 전공을 구성하는 학과 과목에 대한 질문입니다.
- 경영학전공 과목의 개수를 입력해주세요 : 0
- 경제학전공 과목의 개수를 입력해주세요 : 0
- 국어국문학전공 과목의 개수를 입력해주세요 : 10
- 글로벌학전공 과목의 개수를 입력해주세요 : 5
- 기계공학전공 과목의 개수를 입력해주세요 : 0
- 물리학전공 과목의 개수를 입력해주세요 : 0
- 미군문화전공 과목의 개수를 입력해주세요 : 9
- 미디어&엔터테인먼트전공 과목의 개수를 입력해주세요 : 0
- 사회학전공 과목의 개수를 입력해주세요 : 0
- 사회학전공 과목의 개수를 입력해주세요 : 0
- 생명과학전공 과목의 개수를 입력해주세요 : 0
- 수학교전공 과목의 개수를 입력해주세요 : 0
- 신문화전공 과목의 개수를 입력해주세요 : 2
- 심리학전공 과목의 개수를 입력해주세요 : 1
- 아트&테크놀로지전공 과목의 개수를 입력해주세요 : 0
- 영미어문전공 과목의 개수를 입력해주세요 : 11
- 유럽문화학전공 과목의 개수를 입력해주세요 : 8
- 전자공학전공 과목의 개수를 입력해주세요 : 0
- 정치외교학전공 과목의 개수를 입력해주세요 : 2
- 종교학전공 과목의 개수를 입력해주세요 : 0
- 중국문화전공 과목의 개수를 입력해주세요 : 7
- 철학전공 과목의 개수를 입력해주세요 : 0
- 커뮤니케이션학전공 과목의 개수를 입력해주세요 : 2
- 컴퓨터공학전공 과목의 개수를 입력해주세요 : 0
- 환경생태공학전공 과목의 개수를 입력해주세요 : 0
- 화학전공 과목의 개수를 입력해주세요 : 0
< '언어학전공'은 '인문/예술' 분야 입니다. >
< 아래는 해당 분야에 개설되어있는 설계전공 리스트 입니다. 살펴보세요. >
>> 공예예술인문학
>> 디지털스토리텔링
>> 문화예술컨텐츠학
>> 문화인류학
>> 문화콘텐츠학
>> 사회철학
>> 아시아공연예술문화학
>> 영상문화학
>> 영상서사학
>> 영상스토리텔링
>> 영화인문학
>> 인간과진리
>> 인문소통학
>> 인문학전문화콘텐츠학
>> 창의비주얼아트
>> 크리에이티브컬처디자인
>> 한국고전문헌학
계속하려면 아무 키나 누르십시오 . . .
```

```
새로운 설계전공의 이름을 입력해 주세요 (띄어쓰기제외/20자 이내)
>> 금융통계학
다음은 해당 전공을 구성하는 학과 과목에 대한 질문입니다.
- 경영학전공 과목의 개수를 입력해주세요 : 12
- 경제학전공 과목의 개수를 입력해주세요 : 13
- 국어국문학전공 과목의 개수를 입력해주세요 : 0
- 글로벌학전공 과목의 개수를 입력해주세요 : 0
- 기계공학전공 과목의 개수를 입력해주세요 : 0
- 물리학전공 과목의 개수를 입력해주세요 : 0
- 미군문화전공 과목의 개수를 입력해주세요 : 0
- 미디어&엔터테인먼트전공 과목의 개수를 입력해주세요 : 0
- 사회학전공 과목의 개수를 입력해주세요 : 0
- 사회학전공 과목의 개수를 입력해주세요 : 2
- 생명과학전공 과목의 개수를 입력해주세요 : 0
- 수학교전공 과목의 개수를 입력해주세요 : 8
- 신문화전공 과목의 개수를 입력해주세요 : 0
- 심리학전공 과목의 개수를 입력해주세요 : 0
- 아트&테크놀로지전공 과목의 개수를 입력해주세요 : 0
- 영미어문전공 과목의 개수를 입력해주세요 : 0
- 유럽문화학전공 과목의 개수를 입력해주세요 : 0
- 전자공학전공 과목의 개수를 입력해주세요 : 0
- 정치외교학전공 과목의 개수를 입력해주세요 : 3
- 종교학전공 과목의 개수를 입력해주세요 : 0
- 중국문화전공 과목의 개수를 입력해주세요 : 0
- 철학전공 과목의 개수를 입력해주세요 : 0
- 커뮤니케이션학전공 과목의 개수를 입력해주세요 : 0
- 컴퓨터공학전공 과목의 개수를 입력해주세요 : 5
- 환경생태공학전공 과목의 개수를 입력해주세요 : 0
- 화학전공 과목의 개수를 입력해주세요 : 0
< '금융통계학'은 '기업경영및금융' 분야 입니다. >
< 아래는 해당 분야에 개설되어있는 설계전공 리스트 입니다. 살펴보세요. >
>> 국제예술경영학
>> 금융공학
>> 금융투자학
>> 기업금융시장
>> 문화예술경영학
>> 문화콘텐츠경영학
>> 브랜드디자인
>> 빅데이터캐팅
>> 소비자커뮤니케이션학
>> 융합문화마케팅
>> 인공지능서비스기획
>> 인터랙션디자인
>> 지속가능경영
>> 콘텐츠경영학
>> 통합마케팅커뮤니케이션학
>> 행동경제학
>> 회계세무인재
>> IMC경영학
계속하려면 아무 키나 누르십시오 . . .
```

〈 그림 14. 다른 설계 전공 입력 예시 〉

직접 하나의 새로운 설계 전공을 입력해보면, 전공과 어울리는 분야가 적절하게 출력되는 것을 확인할 수 있다. 기존 설계 전공의 분야가 애초에 정확한 기준으로 분류되어 있지 않기 때문에 정확도를 정확한 수치로 계산해볼 수는 없었지만, 학습데이터를 바탕으로 정확한 Decision Tree가 생성되었기

때문에 새로운 설계 전공의 분야는 기존 설계 전공 데이터에 걸맞게 분류가 되었다고 이야기할 수 있을 것이다.

2) 활용 방안

현재 서강대학교 학생설계전공 홈페이지에서는 기존에 개설된 학생설계전공들을 살펴볼 수 있지만, 어떠한 분류 기준도 제시하고 있지 않고 단지 연도별로 승인된 설계전공만을 구분하고 있어 기존에 어떤 분야에 어떤 설계전공들이 존재하는지를 확인하기 어렵다. 위와 같은 분류 프로그램을 만들어 데이터베이스를 관리한다면 학생들이 설계 전공을 개설하고자 할 때, 본인이 관심있는 분야에 이미 개설되어 있는 전공을 살펴볼 수 있을 것이고, 새로운 설계 전공이 개설된 후에는 해당 전공을 어떤 분야로 분류해 정리해 두어야 할 지 쉽게 판별할 수 있을 것이다.

학생설계전공 승인현황

· 서강대 학생은 로그인 후, 학생설계전공명을 클릭하여 교과목구성표를 확인하실 수 있습니다.

승인연도	학생설계전공명	전공구성
2020년	<u>문화예술컨텐츠학</u>	교양학부, 사회학전공, 여성학 연계전공, 아트&테크놀로지전공, 심리학전공, 철학전공
2020년	<u>과학기술사회학 (STS)</u>	국어국문학전공, 융합소프트웨어연계전공, 교양학부, 미디어&엔터테인먼트전공, 사회학전공, 종교학전공, 공공인재 연계전공, 여성학 연계전공, 영미어문전공, 지식융합미디어학부, 아트&테크놀로지전공, 철학전공
2020년	<u>중국통상외교</u>	정치외교학전공, 경제학전공, 중국문화전공, 공공인재 연계전공
2020년	<u>한문고전문학콘텐츠학</u>	경영학전공, 국어국문학전공, 융합소프트웨어연계전공, 사학전공, 교양학부, 미디어&엔터테인먼트전공, 중국문화전공, 신문방송학전공, 아트&테크놀로지전공

< 그림 15. 서강대학교 학생설계전공 리스트 페이지. 카테고리가 없어 불편하다 >

5. 역할분담 및 느낀점

1) 역할분담

	김수미	김효진
프로젝트 주제 선정	50%	50%
데이터 수집 및 가공	50%	50%
코드 작성	70%	30%
보고서 작성	50%	50%
발표자료 제작	30%	70%
발표	50%	50%

2) 느낀점

- **김수미** : 직접 주제를 정해 데이터를 수집하고 원하는 결과를 이끌어낼 수 있는 코드를 작성하는 작업이 매우 흥미로웠다. 실제로 실생활 속에서 불편하거나 필요하다고 생각되는 부분들을 코딩을 통해서 해결하니 뿌듯했고, 실생활 속에서 개선할 수 있는 점들을 찾아 다른 프로젝트를 더 해보고 싶다는 생각이 들었다. 아쉬웠던 점은 처음에는 추천 프로그램을 이용한 앱이나 웹사이트를 만들어보고 싶었는데, 사용할 수 있는 프로그래밍 언어가 한정되어 있어 구현할 수 있는 서비스에 벽이 있다는 사실을 알게 되었다. 다양한 서비스를 개발하기 위해 다양한 프로그래밍 언어에 대한 지식의 필요성을 느낄 수 있었다.

- **김효진** : 수업에서 배우는 개념들이 실생활에서는 어떻게 활용되는 건지 감이 잘 오지 않아서 주제를 선정할 때도 처음에는 막막했는데, 생각보다 일상 속 가까이에서 사용될 수 있다는 점을 깨닫고 신기했다. 설계 전공의 분류기준이 명확한 지표를 바탕으로 세운 것이 아니었다는 점은 조금 아쉬웠고, 현재 프로그램은 설계 전공을 입력하면 분석해 주는 것에 초점이 맞춰져 있는데, 굳이 입력하지 않아도 분야별로 검색할 수 있는 시스템까지 있었으면 좋을 것 같다는 생각을 했다. 앞으로도 프로그래밍을 공부할 때 실생활의 불편함을 발견하고 거기에 배운 것을 적용할 수 있는 힘을 기르도록 노력해야겠다고 생각했다.