제출일 : 2019.12.15

# 고등학생 식단 점검 및 추천 시스템


과목명 : 데이터마이닝분석

학과명 : 정보융합학부

조이름 : 4조

팀 원 : 2017204002 김영서

2017204016 이지환

2017204068 김동영

# I. 주제 설명 및 선택 배경

실시한 프로젝트의 주제는 고등학생 식단 점검 및 추천 시스템이다. 고등학생을 대상으로 급식 식단과 간식이 포함된 일주일치 식단과, 학생 개인의 몸무게, 키, 나이, 성별, 영양소 권장량(신체 데이터에 따라 권장량이 다름), 알레르기 여부 등의 데이터를 가지고 하루 식단 점검, 일주일 식단 총평, 식단 추천을 해주는 것이 이 주제의 목표이다. 즉 한마디로 정의하자면 고등학생 급식 식단을 기준으로 식단 평가 및 추천을 해주는 것이다.

고등학생을 대상으로 선정한 이유는 성장을 위한 영양소 섭취가 중요한 나이이므로 영양소 섭취를 올바르게 하고 있는지를 평가하면 좋겠다 생각하였다. 급식 식단으로 식단을 한정한 이유는 실제 학생이 먹은 식단을 직접 input 받으면 좋을 것이라 생각하였지만, 실제 식단 데이터를 얻기가 어려워 그나마 얻기 쉬운 급식 식단으로 한정하게 되었다. 또한 급식 식단은 영양소 총량 정보가 잘 정리되어 있어 다루기 쉬웠고, 실제로 고등학생은 학교 급식으로 끼니를 먹는 경우가 많기에 급식 식단으로 선정하는 것이 타당하다 생각하였다. 물론 급식 이외에도 간식과 같은 군것질거리를 통해 영양소 섭취를 하기 때문에 현실성을 고려하여 식단에 간식 데이터도 추가하여 주제를 진행하였다.

이 주제를 선택하게 된 배경으로는 현대인이 바쁜 삶으로 인해 매 끼니를 거르고 인스턴트 음식을 자주 먹는 등 건강에 대해 소홀하므로 이러한 사람들에게 식습관 분석을 통해 건강의 중요성을 부각해주면 좋겠다 생각하였기 때문이다. 실제로 최근 이용자의 정보 입력을 통해 건강 정보를 제공하는 서비스와 시스템이 다양하게 출시되고 있어 조원들 모두 건강 데이터 분석에 대해 평소 관심을 가지고 있었다. 이번 기회를 통해 조금이나마 건강 데이터 분석에 대한 경험을 할 수 있을 것이라 생각하여 이 주제를 고르게 되었다.

# II. 데이터 수집 방법 및 데이터 구성 요소 설명

## A. 인체 치수 데이터

실제로 사용자의 키, 나이, 성별, 몸무게 등과 같은 정보를 입력 받으면 좋겠지만, 그 많은 데이터들을 얻기가 마땅치 않아 실제 한국인의 인체 치수 데이터를 사용하여 주제를 진행하기로 하였다.

인체 치수 데이터는 국가통계 마이크로데이터(https://mdis.kostat.go.kr/index.do)에서 제공하는 보건/복지 항목에 '한국인인체치수조사' 2015년 데이터를 다운받아 사용했다. 100개가 넘는 항목에서 식단 분석 시 사용될 만한 총 9개의 항목만 선택하여 다운받았다.

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 성별 | 골격근량 | 체지방량 | BMI | 체지방율 | 복부지방율 | 몸무게 | 나이 | 키 |
| 2 | 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 |
| 3 | 1 | 22.1 | 7 | 16.83558 | 14.7 | 0.7 | 47.8 | 17 | 168.5 |
| 4 | 1 | 22.3 | 6.7 | 17.79882 | 14.3 | 0.8 | 47 | 17 | 162.5 |
| 5 | 1 | 22.4 | 5.9 | 18.05486 | 12.7 | 0.8 | 46.8 | 17 | 161 |
| 6 | 1 | 22.5 | 6.2 | 16.5024 | 13.2 | 0.7 | 47.3 | 17 | 169.3 |
| 7 | 1 | 22.6 | 8 | 17.89867 | 16.3 | 0.8 | 49.5 | 17 | 166.3 |
| 8 | 1 | 22.7 | 8.6 | 18.66659 | 17.2 | 0.8 | 49.9 | 17 | 163.5 |
| 9 | 1 | 22.8 | 3.5 | 17.15937 | 7.8 | 0.8 | 45.2 | 17 | 162.3 |
| 10 | 1 | 22.9 | 6.9 | 16.70823 | 14.1 | 0.7 | 49.2 | 17 | 171.6 |

고등학생을 대상으로 선정하였기 때문에 나이가 17세에서 19세인 데이터만 선택하였고, 성별 1은 남자 성별 2는 여자를 의미한다. 이후 데이터 분석에 사용되는 열은 '성별', '몸무게', '나이', '키' 값이다. 이 데이터를 '고등학생 인

체치수.csv' 로 저장하였고 1405 row x 9 column으로 이루어져 있다.

### B. 급식 식단 데이터

급식 식단 데이터는 '나이스 교육정보 개방 포털 (https://open.neis.go.kr/portal/mainPage.do)'에서 가져왔다. 데이터셋 메뉴의 급식식단 정보에 들어가면 교육부와 17 개 시도교육청에서 제공하는 데이터 sheet 을 조회하고 다운 받을 수 있다. 시도교육청을 선택하고 학교명을 입력하여 검색을 한 후 csv 파일을 다운 받을 수 있는데 고등학생 대상이기 때문에 학교명에 '고등학교'를 입력하고 검색하여 고등학교 식단만 조회되도록 설정하였다.

```
from selenium import webdriver
from bs4 import BeautifulSoup

driver = webdriver.Chrome('../driver/chromedriver')
driver.get("https://open.neis.go.kr/portal/data/service/selectServicePage.do?page=1&rows=10&sortColu

office_list_raw = driver.find_element_by_xpath("""//*[@id="sheet-filter-ATPT_OFCDC_SC_CODE"]""")
office_list = office_list_raw.find_elements_by_tag_name("option")

office_names = [option.text for option in office_list] # 시도교육청코드 문자열 가져오기
del office_names[0] #'(필수)선택하세요' 항목 삭제

import time
from tqdm import tqdm_notebook
from selenium.webdriver.common.alert import Alert

alert = Alert(driver)

school_names = driver.find_element_by_xpath("""//*[@id="sheet-filter-SCHUL_NM"]""")
school_names.send_keys("고등학교") # 학교명에 '고등학교' 입력

# 각 시도교육청 선택하여 데이터 검색 후 csv 파일 다운받기
for office in tqdm_notebook(office_names):
    element = driver.find_element_by_id("sheet-filter-ATPT_OFCDC_SC_CODE")
    element.send_keys(office)

    driver.find_element_by_xpath("""//*[@id="sheet-search-button"]""").click()

    time.sleep(3)

    driver.find_element_by_xpath("""//*[@id="sheet-csv-button"]""").click()

    alert.accept()

    time.sleep(5)

driver.close()
```

총 17개의 시도교육청 파일을 수작업으로 하나하나 다운받기에는 힘드므로, 코드를 사용하여 자동으로 데이터를 가져왔다. 위 코드와 같이 ChromeDriver, Beautifulsoup, Selenium을 이용하여 사이트의 코드를 파싱 한 후 옵션 선택 및 버튼 클릭을 명령하여 csv 파일을 다운받았다.

```python
#각 시도교육청 17개의 csv파일 하나로 합치기
import pandas as pd
from glob import glob

school_diet_files = glob('C:/Users/dudtj/Desktop/데이터마이닝분석/급식식단 데이터/급식식단정보*csv')

tmp_raw = []                    ## 엑셀 파일에서 값 읽어서 tmp_raw에 저장

for file_name in school_diet_files:
    tmp = pd.read_csv(file_name, header=0)
    tmp_raw.append(tmp)

school_diet_raw = pd.concat(tmp_raw)    ##  pandas DataFrame으로 모든 엑셀 파일의 정보를 묶음

school_diet_raw.to_csv("급식식단 데이터.csv", encoding = 'utf-8-sig')
```

이렇게 다운받아진 17개의 시도교육청 급식식단 csv파일을 glob와 pandas 모듈을 이용하여 하나로 합쳐 '급식식단 데이터.csv'로 저장하였다. 열의 구성은 아래와 같으며 167540 row x 14 column이고, 이 중 데이터 분석에 필요한 열은 학교명, 식사명, 급식일자, 요리명, 칼로리정보, 영양정보 열이다.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 시도교육청 | 시도교육청 | 표준학교코 | 학교명 | 식사코드 | 식사명 | 급식일자 | 급식인원수 | 요리명 | 원산지정보 | 칼로리정보 | 영양정보 | 적재일시 |
| 2 | 0 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20200205 | 358 | 숙주나물닭 | 쌀 : 국내산 | 98.7 Kcal | 탄수화물(g | 2.02E+13 |
| 3 | 1 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20200204 | 358 | 참치야채브 | 쌀 : 국내산 | 1120.1 Kca | 탄수화물(g | 2.02E+13 |
| 4 | 2 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20200203 | 358 | 김가루볶음 | 쌀 : 국내산 | 1415.7 Kca | 탄수화물(g | 2.02E+13 |
| 5 | 3 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20191226 | 358 | 햄김치볶음 | 쌀 : 국내산 | 1745.6 Kca | 탄수화물(g | 2.02E+13 |
| 6 | 4 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20191224 | 358 | 땡초고기주 | 쌀 : 국내산 | 1835.0 Kca | 탄수화물(g | 2.02E+13 |
| 7 | 5 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20191223 | 358 | 현미밥(가) | 쌀 : 국내산 | 1318.4 Kca | 탄수화물(g | 2.02E+13 |
| 8 | 6 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20191220 | 358 | 나물비빔밥 | 쌀 : 국내산 | 1427.4 Kca | 탄수화물(g | 2.02E+13 |
| 9 | 7 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20191219 | 358 | 흑미밥(가) | 쌀 : 국내산 | 1153.8 Kca | 탄수화물(g | 2.02E+13 |
| 10 | 8 | C10 | 부산광역시 | 7150119 | 가야고등학 | 2 | 중식 | 20191218 | 358 | 김치참치김 | 쌀 : 국내산 | 1580.5 Kca | 탄수화물(g | 2.02E+13 |

### C. 병원 식단 데이터

급식 식단 외에도 병원 식단 데이터를 가져왔다. 병원 식단 데이터는 후에 식단 추천 시 올바르지 않은 식단을 먹은 학생에게 추천할 식단으로, 병원 식단은 환자들을 위한 식단이기에 아주 세밀하게 영양소가 짜여져 있다고 가정하였다. '공공데이터포털 (https://www.data.go.kr/dataset/15017299/fileData.do)'에 병원 급식을 검색한 결과 '국립소록도병원 영양급식 정보 20181231'이라는 제목의 URL이 검색되었다. 이를 통해 보건복지부에서 운영하는 국립소록도병원의 2018년 식단을 다운받았다.

데이터는 아래와 같이 일자와 구분, 메뉴명으로 되어있으며, 총 5627 row x 3 column이다. 이 데이터는 'hospital.csv'로 저장하였다.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | 일자 | 구분 | 메뉴명 | |
| 2 | 2018-01-02 | 조식 | 오징어무국 | |
| 3 | 2018-01-02 | 조식 | 스크램블에그 | |
| 4 | 2018-01-02 | 조식 | 베이컨감자채볶음 | |
| 5 | 2018-01-02 | 조식 | 쑥갓나물 | |
| 6 | 2018-01-02 | 조식 | 배추김치 | |
| 7 | 2018-01-02 | 중식 | 톳밥 | |
| 8 | 2018-01-02 | 중식 | 추어탕 | |
| 9 | 2018-01-02 | 중식 | 돈육계란장조림 | |
| 10 | 2018-01-02 | 중식 | 해물파전 | |
| 11 | 2018-01-02 | 중식 | 깻잎지 | |
| 12 | 2018-01-02 | 중식 | 알타리김치 | |

### D. 간식 데이터

간식 데이터는 '식품안전나라(https://www.foodsafetykorea.go.kr/main.do)'의 식품영양성분DB 메뉴의 가공식품 메뉴에 들어가 다운받았다. 가공식품군 옵션을 선택할 수 있는데, 전체를 전부 다운받으면 데이터가 너무 많고 쓸모 없을 것 같은 row들이 너무 많으므로, 고등학생들이 즐겨먹을 것 같은 가공식품군(빵 또는 떡류, 커피, 탄산음료류, 면류, 아이스크림류, 가공유류, 즉석섭취·편의식품류, 과자류)만 선택하여 다운 받았다.

```
import pandas as pd
from glob import glob

# 간식 데이터 엑셀 raw 파일 8개 하나로 합치기
snack_data_files = glob('간식 데이터/간식 데이터*xls')

tmp_raw = []                    ## 엑셀 파일에서 값 읽어서 tmp_raw에 저장

for file_name in snack_data_files:
    tmp = pd.read_excel(file_name, header=0)
    tmp_raw.append(tmp)

snack_data = pd.concat(tmp_raw)   ##  pandas DataFrame으로 모든 엑셀 파일의 정보를 묶음
snack_data.to_csv("간식 데이터.csv", encoding = 'utf-8-sig') ##  하나로 묶은 데이터를 '간식 데이터.csv' 파일로 저장
```

총 8개 각각의 가공식품군 csv파일을 glob와 pandas를 이용하여 하나로 합쳐 '간식 데이터.csv'로 저장하였다.

데이터의 구성요소는 아래와 같으며 9430 row x 15 column으로 이루어져 있다. 식품이름 열에는 실제 시중에서 판매하고 있는 가공식품이름 값이 기재되어 있다. 이 중 '식품이름, 열량, 탄수화물, 단백질, 지방, 당류, 나트륨, 포화지방산' 열이 분석에 사용되었다.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 번호 | 식품군 | 식품이름 | 1회제공량 | 열량 (kcal) | 탄수화물 ( | 단백질 (g) | 지방 (g) | 당류 (g) | 나트륨 (m | 콜레스테롤 | 포화지방산 | 트랜스지방 | 년도 |
| 2 | 0 | 1 | 빵 또는 떡 | 1,000원의 | 70 | 202.54 | 36.92 | 6.98 | 3 | 2.69 | 326.35 | 1.51 | 1.09 | 0 | 2013 |
| 3 | 1 | 2 | 빵 또는 떡 | 1,000원의 | 70 | 141.78 | 25.84 | 4.88 | 2.1 | 2.45 | 228.45 | 1.06 | 0.76 | 0 | 2017 |
| 4 | 2 | 3 | 빵 또는 떡 | 100% 우리 | 70 | 242.31 | 31.23 | 4.31 | 11.85 | 6.46 | 280 | 0 | 5.38 | 0 | 2012 |
| 5 | 3 | 4 | 빵 또는 떡 | 100% 우리 | 70 | 260.95 | 33.63 | 4.64 | 12.76 | 4.52 | 301.54 | 0 | 5.8 | 0 | 2017 |
| 6 | 4 | 5 | 빵 또는 떡 | 100% 우리 | 70 | 215.38 | 31.23 | 3.23 | 8.62 | 5.38 | 258.46 | 0 | 4.85 | 0 | 2012 |
| 7 | 5 | 6 | 빵 또는 떡 | 100% 우리 | 70 | 239.32 | 34.7 | 3.59 | 9.57 | 3.77 | 287.18 | 0 | 5.38 | 0 | 2017 |
| 8 | 6 | 7 | 빵 또는 떡 | 100% 우리 | 70 | 231.54 | 32.31 | 3.23 | 10.77 | 7.54 | 247.69 | 0 | 5.38 | 0 | 2012 |
| 9 | 7 | 8 | 빵 또는 떡 | 100% 우리 | 70 | 249.35 | 34.79 | 3.48 | 11.6 | 5.28 | 266.75 | 0 | 5.8 | 0 | 2017 |
| 10 | 8 | 9 | 빵 또는 떡 | 100%우리 | 70 | 236.92 | 32.31 | 4.31 | 11.85 | 6.46 | 258.46 | 0 | 6.46 | 0 | 2012 |

## III. 데이터 전처리 과정 및 방법

### C. 인체 데이터

가) '권장 섭취량', ;탄수화물', '단백질', '지방' column 추가

학생의 나이, 키, 체중 값을 이용하여 각 학생이 필수로 섭취하도록 권장되는 칼로리와 영양분의 양을 계산하여 column 에 추가하였다. 이때, 권장섭취량(kcal), 탄수화물(g), 단백질(g), 지방(g) 총 4 개의 값을 계산하였다.

권장섭취량(kcal)은 키와 몸무게에 기반한 것이 아니라 15~18 세 기준인 여성은 2000kcal, 남성은 2700kcal 를 추가하였다. 그리고 일반적인 비율에 기반해 권장섭취량에서 탄수화물은 50%, 단백질은 30%, 지방은 20%로 설정하였다. 그 뒤 탄수화물은 1g 당 4kcal, 단백질은 1g 당 4kcal, 지방은 1g 당 9kcal 이기 때문에 각 1g 당 해당 칼로리로 나누어 칼로리를 g 으로 변경하였다.

| 영양소 | 공식 |
|---|---|
| 권장 섭취량(kcal) | If(성별=1, 2700, 2000) |
| 탄수화물(g) | 권장섭취량(kcal)의 50% / 4 |
| 단백질(g) | 군장섭취량(kcal)의 30% / 4 |
| 지방(g) | 권장섭취량(kcal)의 20% / 9 |

나) '비타민', '티아민', '리보플라빈', '칼슘', '철분' column 추가

급식 식단 데이터에 기록되어 있는 추가 영양소 양의 비교도 포함하기 위해 '2010 한국인 영양섭취 기준'문서의 기준을 참고하여 추가하였다.

```
import pandas as pd

data = pd.read_csv("청소년 인체치수+하루권장량.csv", encoding = 'cp949')

for i in range(len(data)):

    if data.loc[i, '성별'] == 1: # 남자
        if data.loc[i, '나이'] == 17 or data.loc[i, '나이'] == 18: # 17, 18세
            data.loc[i, '비타민A'] = 850
            data.loc[i, '티아민'] = 1.3
            data.loc[i, '리보플라빈'] = 1.7
            data.loc[i, '비타민C'] = 110
            data.loc[i, '칼슘'] = 900
            data.loc[i, '철분'] = 15
        else: # 19세
            data.loc[i, '비타민A'] = 750
            data.loc[i, '티아민'] = 1.2
            data.loc[i, '리보플라빈'] = 1.5
            data.loc[i, '비타민C'] = 100
            data.loc[i, '칼슘'] = 750
            data.loc[i, '철분'] = 10
    else: # 여자
        if data.loc[i, '나이'] == 17 or data.loc[i, '나이'] == 18: # 17, 18세
            data.loc[i, '비타민A'] = 600
            data.loc[i, '티아민'] = 1.0
            data.loc[i, '리보플라빈'] = 1.2
            data.loc[i, '비타민C'] = 100
            data.loc[i, '칼슘'] = 900
            data.loc[i, '철분'] = 17
        else: # 19세
            data.loc[i, '비타민A'] = 650
            data.loc[i, '티아민'] = 1.1
            data.loc[i, '리보플라빈'] = 1.2
            data.loc[i, '비타민C'] = 100
            data.loc[i, '칼슘'] = 750
            data.loc[i, '철분'] = 14
```

| 영양소 | 성별 | 나이 | 필수 섭취량 |
|---|---|---|---|
| 비타민 A | 남성 | 17,18 | 850 |
| | | 19 | 750 |
| | 여성 | 17,18 | 600 |
| | | 19 | 650 |
| 티아민 | 남성 | 17,18 | 1.3 |
| | | 19 | 1.2 |
| | 여성 | 17,18 | 1.0 |
| | | 19 | 1.1 |
| 리보플라빈 | 남성 | 17,18 | 1.7 |
| | | 19 | 1.5 |
| | 여성 | 17,18 | 1.2 |
| | | 19 | 1.2 |

| 영양소 | 성별 | 나이 | 필수 섭취량 |
|---|---|---|---|
| 비타민 C | 남성 | 17,18 | 110 |
| | | 19 | 100 |
| | 여성 | 17,18 | 100 |
| | | 19 | 100 |
| 칼슘 | 남성 | 17,18 | 900 |
| | | 19 | 750 |
| | 여성 | 17,18 | 900 |
| | | 19 | 750 |
| 철분 | 남성 | 17,18 | 15 |
| | | 19 | 10 |
| | 여성 | 17,18 | 17 |
| | | 19 | 14 |

다) '알레르기' 보유 여부를 확률로 설정하여 알레르기 번호 추가

수집한 인체 데이터에는 알레르기에 관련된 정보가 포함되어 있지 않다. 하지만, 현실에는 그 수는 적지만 알레르기를 가지고 있는 학생들이 있다. 그로 인해, 알레르기 보유 여부를 확률로 설정하여 18 개의 알레르기 종류 중에서 랜덤하게 선택하여 부여하였다. 알레르기 종류는 난류, 우유, 메밀, 땅콩, 대두, 밀, 고등어, 게, 새우, 돼지고기, 복숭아, 토마토, 아황산염, 호두, 닭고기, 쇠고기, 오징어, 조개류(굴,전복,홍합 등)으로 총 18 개의 종류로 나눠서 생각하였다. 그리고 비율은 알레르기가 없는 사람을 전체의 80%, 한종류의 알레르기를 가지고 있는 사람을 전체의 15%, 두종류를 가지고 있는 사람을 5%로 지정하였다.

```
import random
import numpy as np
import pandas as pd

allergy = [str(i) for i in range(1,19)]

for i in range(len(data)):

    prob = random.random()

    if prob <= 0.8: # 80%의 확률로 알레르기값 없음
        data.loc[i, '알레르기'] = np.nan
    elif 0.8 < prob and prob <= 0.95: # 15%의 확률로 알레르기값 1개 있음
        data.loc[i, '알레르기'] = random.choice(allergy)
    else: # 5%의 확률로 알레르기값 2개 있음
        value = random.sample(allergy, 2)
        data.loc[i, '알레르기'] = str(value[0])+","+str(value[1])

data.to_csv("인체치수 최종본.csv", encoding = 'utf-8-sig')
```

라) '인체치수 최종본.csv'에 저장

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 성별 | 골격근량 | 체지방량 | BMI | 체지방율 | 복부지방율 | 몸무게 | 나이 | 키 | 기초대사량 | 탄수화물 | 단백질 | 지방 | 비타민A | 티아민 | 리보플라빈 | 비타민C | 칼슘 | 철분 | 알레르기 |
| 2 | 0 | 1 | 21.7 | 6 | 16.9542 | 13.2 | 0.7 | 45.6 | 17 | 164 | 1398.55 | 209.783 | 104.891 | 15.5394 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 3 | 1 | 1 | 22.1 | 7 | 16.8356 | 14.7 | 0.7 | 47.8 | 17 | 168.5 | 1451.3 | 217.695 | 108.848 | 16.1256 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 4 | 2 | 1 | 22.3 | 6.7 | 17.7988 | 14.3 | 0.8 | 47 | 17 | 162.5 | 1410.3 | 211.545 | 105.773 | 15.67 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 5 | 3 | 1 | 22.4 | 5.9 | 18.0549 | 12.7 | 0.8 | 46.8 | 17 | 161 | 1400.05 | 210.008 | 105.004 | 15.5561 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 6 | 4 | 1 | 22.5 | 6.2 | 16.5024 | 13.2 | 0.7 | 47.3 | 17 | 169.3 | 1448.43 | 217.264 | 108.632 | 16.0936 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | 18 |
| 7 | 5 | 1 | 22.6 | 8 | 17.8987 | 16.3 | 0.8 | 49.5 | 17 | 166.3 | 1463.68 | 219.551 | 109.776 | 16.2631 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 8 | 6 | 1 | 22.7 | 8.6 | 18.6666 | 17.2 | 0.8 | 49.9 | 17 | 163.5 | 1455.18 | 218.276 | 109.138 | 16.1686 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | 10 |
| 9 | 7 | 1 | 22.8 | 3.5 | 17.1594 | 7.8 | 0.8 | 45.2 | 17 | 162.3 | 1384.55 | 207.683 | 103.841 | 15.3839 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 10 | 8 | 1 | 22.9 | 6.9 | 16.7082 | 14.1 | 0.7 | 49.2 | 17 | 171.6 | 1486.05 | 222.908 | 111.454 | 16.5117 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 11 | 9 | 1 | 23.4 | 2.9 | 16.7493 | 6.3 | 0.7 | 45.6 | 17 | 165 | 1403.55 | 210.533 | 105.266 | 15.595 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 12 | 10 | 1 | 23.5 | 5.2 | 17.4628 | 11 | 0.7 | 47.6 | 17 | 165.1 | 1431.55 | 214.733 | 107.366 | 15.9061 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | |
| 13 | 11 | 1 | 23.5 | 9 | 19.5649 | 17.3 | 0.8 | 51.6 | 17 | 162.4 | 1473.05 | 220.958 | 110.479 | 16.3672 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | 4 |

### B. 급식 식단 데이터

가) 알레르기 column 추가

급식 식단 데이터의 '요리 이름'열에 요리명과 해당 요리를 섭취한 경우 생길 수 있는 알레르기 종류가 함께 작성되어 있다. 알레르기 종류를 분석에 사용하고자 하기 때문에 열에서 알레르기 번호를 중복 없이 나열한 뒤, 18개의 알레르기를 각각의 column으로 생성하여 해당하는 알레르기가 포함되면 1, 포함되지 않으면 0을 추가하였다. 알레르기 번호들을 가져오는 과정에서는 정규 표현식을 사용하였다.

```
import re
from tqdm import tqdm_notebook

col_name = [str(i) for i in range(1,19)]

for i in tqdm_notebook(range(len(data))):
    s = data.loc[i,'요리명']
    allergy = re.findall('\d+', s) # 문자열에서 숫자만 빼오기
    allergy = list(set(allergy)) # 리스트 값 중복 제거
    for num in col_name:
        if num in allergy:
            data.loc[i, num] = 1
        else:
            data.loc[i, num] = 0
```

```
data.rename(columns = {"1":"1.난류","2":"2.우유","3":"3.메밀","4":"4.땅콩","5":"5.대두","6":"6.밀","7":"7.고등어","8":"8.게",
                       "9":"9.새우","10":"10.돼지고기","11":"11.복숭아","12":"12.토마토","13":"13.아황산염","14":"14.호두",
                       "15":"15.닭고기","16":"16.쇠고기","17":"17.오징어","18":"18.조개류(굴,전복,홍합 등)"}, inplace = True)
```

나) '요리 이름' 문자열 변환

위에 가)에서 설명했듯이 '요리 이름'열에 요리명과 알레르기 번호를 제외하고 학교마다 식단을 작성하는 방법이 다르기 때문에 다양한 문자열들이 포함되어 있다. 이 문자열들은 오로지 식단만을 보고 싶은 경우에 방해가 되기 때문에 필요 없는 문자열을 제거하고 요리명만을 남겼다. 이 때, 정규 표현식을 이용하여 알레르기 번호와 모든 식단에서 공통되는 <br/>문자를 제거하였고 학교마다 사용되는 문자열들은 replace함수를 이용하여 제거하였다.

```
import re
l=[]
for i in range(len(data["요리명"])):
    a=re.sub("[ㄱ-ㅣ가-힣|-??|ㅠ([^)]+ㅠ)","",data["요리명"][i])
    a=a.replace("<br/>",",").replace("*","").replace("★","").replace("#","").replace("!","").replace("^","")
    a=a.replace("H","").replace("h","")
    b=re.sub(r'ㅠ([^)]+ㅠ)', '', a)
    b=b.replace("()","").replace("ㅠㅠ","").replace("@","").replace("*","").replace("^","")
    l.append(b)
data["요리 이름"]=l
```

다) '칼로리정보', '영양정보' column 값 변환

'칼로리정보' column속 데이터에 단위 'kcal'가 포함되어 데이터들이 문자로 인식되어 분석에 효율적이지 않다. 따라서 모든 데이터에서 단위를 제거하여 데이터의 자료형을 숫자로 바꾸었다.

또한, '영양정보' column에 '요리 이름'column과 비슷하게 모든 영양분과 양이 한 column에 작성되어 있다. 이 영양분들을 분석에 따로따로 사용하기 위해 각각의 column으로 나눠서 저장하였다. 이때, 데이터 속에 단위를 모두 제거하고 column이름에 단위를 함께 작성하여 각 column에 따른 정보를 보기 쉽도록 바꾸었다.

```
nutrient=['']*len(data)

for n in data.index:
    nutrient[n]=data['영양정보'][n]
    nutrient[n]=nutrient[n].replace('탄수화물(g) :','').replace('단백질(g) :','').replace('지방(g) :','').replace('티아민(mg) :','')
                    .replace('비타민A(R.E) :','').replace('리보플라빈(mg) :','').replace('비타민C(mg) :','').replace('칼슘(mg) :','')
                    .replace('철분(mg) :','')
```

```
total=[]
for i in range(len(nutrient)):
    sen=nutrient[i]
    total.append(sen.split('<br/>')) #<br/> 기준으로 나누기

for i in range(len(total)):
    for j in range(len(total[0])):
        total[i][j]=total[i][j].strip() #공백제거
        total[i][j]=float(total[i][j]) #형변환

all_list=[[],[],[],[],[],[],[],[],[]]

for i in range(len(total)):
    for j in range(9):
        all_list[j].append(total[i][j])
```

```
#딕셔너리 형태로 저장
nutrient_split={'탄수화물(g)':all_list[0],'단백질(g)':all_list[1],'지방(g)':all_list[2],'비타민A(mg)':all_list[3],
                '티아민(mg)':all_list[4],'리보플라빈(mg)':all_list[5],'비타민C(mg)':all_list[6],'칼슘(mg)':all_list[7],
                '철분(mg)':all_list[8]}
```

```
data2=pd.DataFrame(nutrient_split)

#중복정보 삭제
data=data.drop(columns=['영양정보'])

#data 합치기
result=pd.concat([data,data2],axis=1)
```

```
cal=[]
for i in range(len(result["칼로리정보"])):
    cal.append(float(result["칼로리정보"][i].split(" ")[0]))
result["칼로리(Kcal)"]=cal
result=result.drop(columns=["요리명","칼로리정보"])
# 열의 순서 바꾸기
result=result[['학교명', '식사코드', '식사명', '급식일자','요리 이름', '칼로리(Kcal)' ,'1.난류', '2.우유', '3.메밀',
               '4.땅콩', '5.대두','6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
               '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
               '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
               '칼슘(mg)', '철분(mg)']]
```

라) '한끼, 두끼 식사 / 세끼 식사' 로 나누기

급식 식단 데이터에서 급식이 제공된 날짜별로 합쳐서 하루의 식단을 만든 뒤, 하루에 제공되는 식단의 개수를 기준으로 한번, 두 번 제공되는 경우와 세번 모두 제공되는 경우로 나눠서 csv파일로 저장하였다.

우선, 조식, 중식, 석식을 모두 제공하는 고등학교를 찾는다. 이 고등학교들을 '삼식고등학교.csv'에 저장하여 나중에 쉽게 재사용할 수 있게 만든다. 그 다음에는 3끼를 모두 제공한 날의 식단을 모두 합쳤다. 3끼의 여부는

식사코드의 합이 6인 경우를 이용하여 찾는다. 1끼, 2끼 여부는 식사코드의 합이 6보다 작은 경우로 찾는다. 주의할 점은 세끼를 모두 제공하는 학교라고 선택된 학교에서도 하루에 한번 또는 두번 식사를 주는 경우가 있기 때문에 '삼식고등학교.csv' 속 고등학교 속에서도 식사코드를 비교해야 한다.

### 1) 삼식

```
#중복된 학교명들 받아오기
school=['']*len(result)

for n in result.index:
    school[n]=result['학교명'][n]

# 중복된 식사명들 받아오기
eat_time=['']*len(result)

for n in result.index:
    eat_time[n]=result['식사명'][n]
```

```
#조식인 index추출
one_time=[]
for i in range(len(eat_time)):
    if '조식' in eat_time[i]:
        one_time.append(i)

# 조식을 먹는 학교만 찾아낸다. 이를 중복
one_eat=[]
for i in range(len(one_time)):
    one_eat.append(school[one_time[i]])

# 학교이름 중복 없애기
one_eat_unique=[]

for i in one_eat:
    if i not in one_eat_unique:
        one_eat_unique.append(i)

#중식도 동일하게
two_time=[]

for i in range(len(eat_time)):
    if '중식' in eat_time[i]:
        two_time.append(i)
```

```
# 중식을 먹는 학교만 찾아낸다.
two_eat=[]
for i in range(len(two_time)):
    two_eat.append(school[two_time[i]])

two_eat_unique=[]

for i in two_eat:
    if i not in two_eat_unique:
        two_eat_unique.append(i)
#석식
three_time=[]
for i in range(len(eat_time)):
    if '석식' in eat_time[i]:
        three_time.append(i)

# 석식을 먹는 학교만 찾아낸다.
three_eat=[]
for i in range(len(three_time)):
    three_eat.append(school[three_time[i]])

three_eat_unique=[]

for i in three_eat:
    if i not in three_eat_unique:
        three_eat_unique.append(i)
```

```
# 해당하는 행 가져오기
row=[]
for i in range(len(df)):
    for j in range(len(result)):
        if result.iloc[j,0]==df.iloc[i,0]:
            row.append(result.loc[j].tolist())
d=pd.DataFrame(row,columns=result.columns)
# 열 순서 바꾸기
d2=d[['학교명', '급식일자','식사코드','요리 이름', '칼로리(Kcal)',  '1.난류', '2.우유', '3.메밀',
       '4.땅콩', '5.대두', '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아',
       '12.토마토', '13.아황산염', '14.호두', '15.닭고기', '16.쇠고기', '17.오징어',
       '18.조개류(굴,전복,홍합 등)', '탄수화물(g)', '단백질(g)', '지방(g)', '비타민A(mg)',
       '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)', '칼슘(mg)', '철분(mg)' ]]
# type 바꾸기
d2=d2.astype({'급식일자':int,'식사코드':int, '칼로리(Kcal)':np.float, '탄수화물(g)':np.float, '단백질(g)':np.float,
              '지방(g)':np.float,'비타민A(mg)':np.float, '티아민(mg)':np.float, '리보플라빈(mg)':np.float, '비타민C(mg)':np.float,
              '칼슘(mg)':np.float,'철분(mg)':np.float,'1.난류':np.float, '2.우유':np.float, '3.메밀':np.float, '4.땅콩':np.float,
              '5.대두':np.float,'6.밀':np.float, '7.고등어':np.float, '8.게':np.float, '9.새우':np.float, '10.돼지고기':np.float,
              '11.복숭아':np.float,'12.토마토':np.float, '13.아황산염':np.float, '14.호두':np.float, '15.닭고기':np.float,
              '16.쇠고기':np.float, '17.오징어':np.float,'18.조개류(굴,전복,홍합 등)':np.float})

nd=pd.DataFrame(0,columns=['급식일자', '식사코드', '칼로리(Kcal)',  '1.난류', '2.우유', '3.메밀','4.땅콩', '5.대두',
                           '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
                           '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
                           '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
                           '칼슘(mg)', '철분(mg)', '식단'],index=range(1))
for hn in df["삼시세끼"]:
    s=d2[d2["학교명"]==hn]
```

```
    # 요리명 제외하고 나머지 열 더하기
    s2=s[['급식일자', '식사코드', '칼로리(Kcal)',  '1.난류', '2.우유', '3.메밀','4.땅콩', '5.대두',
          '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
          '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
          '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
          '칼슘(mg)', '철분(mg)']]

    group=s2.groupby(s2["급식일자"]).sum()
    row2=[]
    idx=[]
    for i in range(len(group)):
        if group.iloc[i,0]==6:
            idx.append(group.index[i])
            row2.append(group.iloc[i].tolist())

    group2=pd.DataFrame(row2,columns=group.columns,index=idx)
```

```
    # 요리명 합치기
    totaln=[]
    for j in range(len(idx)):
        n=[]
        for k in range(3):
            n.append(s[s2["급식일자"]==idx[j]].iloc[k,3])
        totaln.append("//".join(n))

    # 학교와 요리명, 그외의 정보 합치기
    g=group2.reset_index()
    g["식단"]=totaln

    # index를 급식일자로 바꾸기
    g.rename(columns= {"index": "급식일자"}, inplace = True)
    nd=pd.concat([nd,g])
```

```
nd=nd.reset_index(drop=True)
# 열 순서 바꾸기
nd=nd[['식단', '칼로리(Kcal)', '1.난류', '2.우유', '3.메밀', '4.땅콩', '5.대두',
       '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
       '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
       '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
       '칼슘(mg)', '철분(mg)']]
nd=nd[1:]
nd.to_csv("삼식이.csv",encoding='utf-8-sig')
```

## 2) 일이식

```
# 삼시세끼 제공하는 학교를 제외한 학교
schoolname=result["학교명"].unique().tolist()
threeschool=df["삼시세끼"].tolist()
onetwo=[]
for i in range(len(schoolname)):
    if schoolname[i] not in threeschool:
        onetwo.append(schoolname[i])
```

```
row2=[]
for j in range(len(result)):
    if result.iloc[j,0] in onetwo:
        row2.append(result.loc[j].tolist())
```

```
d3=pd.DataFrame(row2,columns=result.columns)
```

```
# 열 순서 바꾸기
d4=d3[['학교명', '급식일자','식사코드','요리 이름', '칼로리(Kcal)',  '1.난류', '2.우유', '3.메밀',
       '4.땅콩', '5.대두', '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아',
       '12.토마토', '13.아황산염', '14.호두', '15.닭고기', '16.쇠고기', '17.오징어',
       '18.조개류(굴,전복,홍합 등)', '탄수화물(g)', '단백질(g)', '지방(g)', '비타민A(mg)',
       '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)', '칼슘(mg)', '철분(mg)' ]]
```

```
# type 바꾸기
d4=d4.astype({'급식일자':int,'식사코드':int, '칼로리(Kcal)':np.float, '탄수화물(g)':np.float, '단백질(g)':np.float,
              '지방(g)':np.float,'비타민A(mg)':np.float, '티아민(mg)':np.float, '리보플라빈(mg)':np.float, '비타민C(mg)':np.float,
              '칼슘(mg)':np.float,'철분(mg)':np.float,'1.난류':np.float, '2.우유':np.float, '3.메밀':np.float, '4.땅콩':np.float,
              '5.대두':np.float,'6.밀':np.float, '7.고등어':np.float, '8.게':np.float, '9.새우':np.float, '10.돼지고기':np.float,
              '11.복숭아':np.float,'12.토마토':np.float, '13.아황산염':np.float, '14.호두':np.float, '15.닭고기':np.float,
              '16.쇠고기':np.float, '17.오징어':np.float,'18.조개류(굴,전복,홍합 등)':np.float})
col=['급식일자', '식사코드', '칼로리(Kcal)',  '1.난류', '2.우유', '3.메밀','4.땅콩', '5.대두',
    '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
     '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
     '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
     '칼슘(mg)', '철분(mg)', '식단']
nd2=pd.DataFrame(0,columns=col,index=range(1))
nd3=pd.DataFrame(0,columns=col,index=range(1))
```

( '삼식고등학교.csv'파일 속 고등학교가 아닌 경우 )

```
for hn in onetwo:
    s=d4[d4["학교명"]==hn]

    # 요리명 제외하고 나머지 열 더하기
    s2=s[['급식일자', '식사코드', '칼로리(Kcal)',  '1.난류', '2.우유', '3.메밀','4.땅콩', '5.대두',
          '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
          '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
          '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
          '칼슘(mg)', '철분(mg)']]

    group=s2.groupby(s2["급식일자"]).sum()
    row4=[]
    idx=[]
    for i in range(len(group)):
        idx.append(group.index[i]) # 날짜
        row4.append(group.iloc[i].tolist())

    group2=pd.DataFrame(row4,columns=group.columns,index=idx)

    # 요리명 합치기
    totaln=[]
    for j in range(len(idx)):
        n=[]
        b=s[s2["급식일자"]==idx[j]]
        for k in range(len(b)):
            n.append(b.iloc[k,3])
        totaln.append("//".join(n))

    # 학교와 요리명, 그외의 정보 합치기
    g=group2.reset_index()
    g["식단"]=totaln

    # index를 급식일자로 바꾸기
    g.rename(columns= {"index": "급식일자"}, inplace = True)
    nd2=pd.concat([nd2,g])
```

( '삼식고등학교.csv'속 고등학교 중 3 끼를 주지 않은 날 선택

```
for hn in df["삼시세끼"]:
    s=d2[d2["학교명"]==hn]

    # 요리명 제외하고 나머지 열 더하기
    s2=s[['급식일자', '식사코드', '칼로리(Kcal)',  '1.난류', '2.우유', '3.메밀','4.땅콩', '5.대두',
          '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
          '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
          '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
          '칼슘(mg)', '철분(mg)']]

    group=s2.groupby(s2["급식일자"]).sum()
    row2=[]
    idx=[]
    for i in range(len(group)):
        if group.iloc[i,0]!=6:
            idx.append(group.index[i])
            row2.append(group.iloc[i].tolist())

    group2=pd.DataFrame(row2,columns=group.columns,index=idx)

    # 요리명 합치기
    totaln2=[]
    for j in range(len(idx)):
        n=[]
        b=s[s2["급식일자"]==idx[j]]
        for k in range(len(b)):
            n.append(b.iloc[k,3])
        totaln2.append("//".join(n))

    # 학교와 요리명, 그외의 정보 합치기
    g=group2.reset_index()
    g["식단"]=totaln2

    # index를 급식일자로 바꾸기
    g.rename(columns= {"index": "급식일자"}, inplace = True)
    nd3=pd.concat([nd3,g])
```

```
nd2=nd2.reset_index(drop=True)
nd3=nd3.reset_index(drop=True)
# 열 순서 바꾸기
nd2=nd2[['식단', '칼로리(Kcal)', '1.난류', '2.우유', '3.메밀', '4.땅콩', '5.대두',
       '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
       '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
       '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
       '칼슘(mg)', '철분(mg)']]
nd2=nd2[1:] # 1행 삭제
# 열 순서 바꾸기
nd3=nd3[['식단', '칼로리(Kcal)', '1.난류', '2.우유', '3.메밀', '4.땅콩', '5.대두',
       '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
       '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '탄수화물(g)',
       '단백질(g)', '지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)',
       '칼슘(mg)', '철분(mg)']]
nd3=nd3[1:] # 1행 삭제
nd4=pd.concat([nd2,nd3])
nd4=nd4.reset_index(drop=True)
nd4.to_csv("일이식이.csv",encoding='utf-8-sig')
```

마) 하루에 3끼를 먹은 데이터는 '삼식이.csv'에 저장, 하루에 한끼, 두끼를 먹은 데이터는 '일이식이.csv'에 저장

(삼식이.csv)

| | 식단 | 칼로리(Kc | 1.난류 | 2.우유 | 3.메밀 | 4.땅콩 | 5.대두 | 6.밀 | 7.고등어 | 8.게 | 9.새우 | 10.돼지고 | 11.복숭아 | 12.토마토 | 13.아황산 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 보리밥-개성,참치 | 2676.8 | 2 | 1 | 0 | 0 | 3 | 3 | 1 | 1 | 3 | 1 | 0 | 1 | 3 |
| 2 | 보리밥-개성,김치 | 3029.7 | 3 | 3 | 0 | 1 | 3 | 3 | 0 | 0 | 3 | 3 | 1 | 2 | 3 |
| 3 | 현미밥-개성,스팸 | 2994.7 | 3 | 3 | 0 | 0 | 3 | 3 | 0 | 1 | 3 | 3 | 0 | 2 | 3 |
| 4 | 기장밥-개성,건새 | 2950.7 | 3 | 1 | 0 | 0 | 3 | 3 | 0 | 1 | 3 | 3 | 0 | 1 | 3 |
| 5 | 기장밥-개성,갈릭 | 2869 | 1 | 3 | 0 | 0 | 3 | 3 | 0 | 0 | 3 | 2 | 0 | 1 | 3 |
| 6 | 보리밥-개성,애호 | 2785.8 | 1 | 2 | 0 | 0 | 3 | 3 | 0 | 1 | 3 | 2 | 0 | 1 | 3 |
| 7 | 흑미밥-개성,한우 | 3082.8 | 1 | 2 | 0 | 1 | 3 | 3 | 0 | 2 | 3 | 3 | 0 | 0 | 3 |

(일이식이.csv)

| | 식단 | 칼로리(Kc | 1.난류 | 2.우유 | 3.메밀 | 4.땅콩 | 5.대두 | 6.밀 | 7.고등어 | 8.게 | 9.새우 | 10.돼지고 | 11.복숭아 | 12.토마토 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 땡초고기볶음밥 | 1830.2 | 1 | 1 | 1 | 0 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 0 |
| 1 | 보리밥,조랭이떡 | 1366.9 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 1 | 0 | 1 |
| 2 | 나물비빔밥,두부 | 1267.6 | 2 | 2 | 0 | 1 | 2 | 2 | 0 | 0 | 2 | 1 | 0 | 0 |
| 3 | 하이라이스,뉴욕 | 1104.6 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 4 | 흑미밥,닭개장,깍 | 636.4 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 5 | 간장계란밥,돈육 | 1589 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 0 | 2 | 2 | 0 | 2 |
| 6 | 현미밥,스프&크 | 1585.5 | 2 | 1 | 0 | 0 | 2 | 2 | 1 | 0 | 2 | 2 | 1 | 1 |

## C. 간식 데이터

가) 식품 이름이 중복인 행 제거

나) 수집한 데이터를 확인하면 식품이름이 같은 행이 두개 씩 존재한다. 그 이유는 2013년도 데이터와 2017년도 데이터가 함께 존재하기 때문이다. 하지만 이번 프로젝트에는 최신 데이터를 사용하기 위해 2013년도 데이터는 제거하고 2017년도 데이터만을 남겨두었다.

다) 식품군이 '면류'인 행 중 식품이름에 '분유', '이유식'이 들어간 행 제거

선택한 식품군 중에서 필요 없는 행을 추가적으로 삭제하였다. 고등학생이 일반적으로 분유와 이유식 식품을 먹을리가 없다고 생각하여 간식 데이터 중에서 식품이름으로 분유와 이유식이 들어가 있는 행을 제거하였다.

라) NaN값이 포함된 행 제거

간식 데이터의 null값을 확인 해본 결과 당류, 나트륨, 콜레스테롤, 포화지방산, 트랜스지방산에서 null값이 다량으로 발견되었다. 이러한 null값은 데이터 분석에 문제가 된다고 생각하여 제거하였다.

마) 필요 없는 열 지우기

추가적으로 데이터를 분석하는 과정에서 용량만 차지하며 사용되지 않을 column들을 미리 삭제하였다. 이번 프로젝트에서 '번호, 콜레스테롤(mg), 1회 제공량(g), 트랜스지방산(g), 년도'열들이 필요 없다고 생각되어 해당하는 열들을 삭제하였다. 그 뒤에, index의 순서를 순서대로 초기화하였다.

```
# 식품이름이 중복인 행 제거 (년도가 2017인 행을 남기기)
snack_data = snack_data.drop_duplicates('식품이름', keep='last')

# 식품군이 '면류'인 행 중 식품이름에 '분유', '이유식'이 들어간 행 제거
drop_list = []

for i in range(len(snack_data)):
    if snack_data.iloc[i, 1] == '면류':
        if '분유' in snack_data.iloc[i, 2]:
            drop_list.append(snack_data.index[i])
        elif '이유식' in snack_data.iloc[i, 2]:
            drop_list.append(snack_data.index[i])

snack_data.drop(drop_list, inplace = True)

# NaN 값이 있는 행 제거
snack_data.dropna(inplace = True)

# 필요없는 열 지우기
snack_data.drop(['번호','콜레스테롤 (mg)', '1회제공량 (g)', '트랜스지방산 (g)', '년도'], axis='columns', inplace=True)

# index 순서 0부터 순서대로 초기화
snack_data.reset_index(drop=True, inplace = True)

# csv 파일로 저장
snack_data.to_csv("간식 데이터 최종본.csv", encoding = 'utf-8-sig')
```

바) '간식 데이터 최종본.csv'에 저장

| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | 식품군 | 식품이름 | 열량 (kcal) | 탄수화물 ( | 단백질 (g) | 지방 (g) | 당류 (g) | 나트륨 (m | 포화지방산 (g) |
| 2 | 0 | 빵 또는 떡 | 1,000원의 행복 토스트 식빵 | 141.78 | 25.84 | 4.88 | 2.1 | 2.45 | 228.45 | 0.76 |
| 3 | 1 | 빵 또는 떡 | 100% 우리밀로 만든 미니땅콩샌드 | 260.95 | 33.63 | 4.64 | 12.76 | 4.52 | 301.54 | 5.8 |
| 4 | 2 | 빵 또는 떡 | 100% 우리밀로 만든 미니복숭아샌드 | 239.32 | 34.7 | 3.59 | 9.57 | 3.77 | 287.18 | 5.38 |
| 5 | 3 | 빵 또는 떡 | 100% 우리밀로 만든 미니옥수수샌드 | 249.35 | 34.79 | 3.48 | 11.6 | 5.28 | 266.75 | 5.8 |
| 6 | 4 | 빵 또는 떡 | 100%우리밀로 만든 미니딸기샌드 | 255.15 | 34.79 | 4.64 | 12.76 | 4.52 | 278.34 | 6.96 |
| 7 | 5 | 빵 또는 떡 | 100%우리밀로 만든 미니생크림샌드 | 255.15 | 34.79 | 3.48 | 11.6 | 6.79 | 289.94 | 5.8 |
| 8 | 6 | 빵 또는 떡 | 100%우리밀로 만든 생크림 샌드 | 254.87 | 34.7 | 3.59 | 11.97 | 6.79 | 299.14 | 5.98 |

### D. 식단 데이터 만들기

가) "5개 균형 식단, 2개 불균형 식단, 2개 간식"으로 구성된 일주일치 식단

학생 한명 당 일주일치 식단 조합을 배치한다. 이를 위해서는 우선 일주일치 식단을 만든다. 일주일치 식단은 5개의 3끼 꼬박꼬박 먹은 균형적인 식단과 2개의 1끼 또는 2끼를 먹은 불균형적인 식단으로 구성한다. 균형과 불균형으로 나눈 이유는 균형적인 식단으로만 일주일치 식단을 짜는 것은 비현실적이고 분석결과에 특이점이 없을 것이라고 생각되기 때문이다. 학생의 일주일치 식단에서 2개의 불균형적인 식단이 outlier라고 볼 수 있지만 사람의 개성이므로 삭제하지 않고 프로젝트를 진행한다.

일주일 치 식단을 만들 때 랜덤하게 '삼식이.csv'와 '일이식이.csv'에서 5개 2개 선택하여 합친다. 이때 sampling_func함수를 이용하여 랜덤하게 샘플링한다. 이 조합을 5000개를 만들어 놓는다.

```
import pandas as pd
import numpy as np

df1=pd.read_csv('삼식이.csv',sep=',',encoding='utf-8-sig',engine="python", index_col=[0]) # 삼식이
df2=pd.read_csv('일이식이.csv',sep=',',encoding='utf-8-sig',engine="python", index_col=[0]) # 일이식이
df1['class']='규칙'
df1=df1[:54693] # df2 개수 만큼 슬라이싱
df2['class']='불규칙'
df3=pd.concat([df1,df2],ignore_index=True) # 규칙, 불규칙 합치기

def sampling_func(data):
    N = len(data)
    sample_n=5 #5개씩 뽑기
    sample = data.take(np.random.permutation(N)[:sample_n])
    return sample

df4= df3.groupby('class', group_keys=False).apply(sampling_func)
df4['num']=1
df4['sort']=np.arange(10)
df4.sort_index()
df3=df3.drop(df3.index[df4.index])

# 조합 5000개 뽑기
for i in range(5000):
    sample_set = df3.groupby('class', group_keys=False).apply(sampling_func)
    sample_set['num']=(i+2)
    sample_set['sort']=np.arange(10)
    df4=pd.concat([df4,sample_set],ignore_index=True)

index_num=df4[df4['sort']>=7].index
df5=df4.drop(index_num)
del df5['sort']
df5.to_csv('five two.csv',sep=',',encoding='utf-8-sig')
```

그 다음으로는 학생들이 하루에 간식을 두번씩 먹는다고 가정하여, 매일의 식단에 '간식 데이터 최종본.csv'에서 랜덤하게 뽑은 간식 2개를 함께 배치한다. 랜덤하게 뽑은 간식 2개는 'five two.csv'에 새로운 column에 추가되며 각 영양소에 따라 값들이 합해진다.

```
shickdan=pd.read_csv("five two.csv",encoding="utf-8-sig")
ganshick=pd.read_csv("간식 데이터 최종본.csv",encoding="utf-8-sig",engine='python',index_col=[0])
ganshick=ganshick.drop(columns=['식품군'])
shickdan=shickdan.drop(columns=["Unnamed: 0"])

def sampling_func(data):
    N = len(data)
    sample_n=2 # 2개씩 뽑기
    sample = data.take(np.random.permutation(N)[:sample_n])
    return sample

b=['열량 (kcal)','탄수화물 (g)','단백질 (g)','지방 (g)','당류 (g)','나트륨 (mg)','포화지방산 (g)','간식']
c=pd.DataFrame(0,columns=b,index=[0])
for i in range(len(shickdan)):
    a=sampling_func(ganshick)
    a["idx"]=i
    b=a.groupby("idx").sum()
    b["간식"]=", ".join(a["식품이름"])
    c=pd.concat([c,b])
c=c.reset_index(drop=True)
c=c.drop(0,axis=0)
c=c.reset_index(drop=True)
```

```
a=pd.concat([shickdan,c],axis=1)
a["총 열량(kcal)"]=a["칼로리(Kcal)"]+a["열량 (kcal)"]
a["총 탄수화물(g)"]=a["탄수화물(g)"]+a["탄수화물 (g)"]
a["총 단백질(g)"]=a["단백질(g)"]+a["단백질 (g)"]
a["총 지방(g)"]=a["지방(g)"]+a["지방 (g)"]
a=a[['num','식단', '간식','칼로리(Kcal)', '총 열량(kcal)', '총 탄수화물(g)', '총 단백질(g)', '총 지방(g)','비타민A(mg)', '티아민(mg)',
     '리보플라빈(mg)', '비타민C(mg)', '칼슘(mg)', '철분(mg)','당류 (g)', '나트륨 (mg)', '포화지방산 (g)', '1.난류', '2.우유',
     '3.메밀', '4.땅콩', '5.대두', '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
     '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)']]

a.to_csv("일주일 식단.csv",encoding='utf-8-sig')
```

완성된 데이터는 '일주일 식단' csv 파일에 저장한다.

| num | 식단 | 간식 | 칼로리(Kc | 총 열량(kc | 총 탄수화 | 총 단백질( | 총 지방(g) | 비타민A(m | 티아민(mg | 리보플라빈 | 비타민C(m | 칼슘(mg) | 철분(mg) | 당류 (g) | 나트륨 (m | 포화지방산 | 1.난류 | 2.우유 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 조개들깨죽 | 미니 트위 | 3455.3 | 3527.3 | 471.9 | 129.98 | 68.19 | 1526.3 | 2.3 | 2.8 | 210.8 | 895.3 | 23.5 | 3.66 | 155.1 | 0.6 | 3 | 2 |
| 1 | 데리야끼쭈 | 플라워모양 | 2965.9 | 3382.89 | 492.69 | 154.34 | 77.33 | 1396.7 | 3.1 | 2.2 | 208.8 | 707.7 | 20.2 | 28.25 | 41.7 | 1.03 | 3 | 2 |
| 1 | 쌀밥,해물 | 스카이플라 | 2680.8 | 2768.1 | 444.16 | 96.74 | 54.4 | 878.2 | 1.7 | 1.5 | 174.4 | 888.3 | 17.9 | 3.42 | 107.1 | 2.16 | 3 | 3 |
| 1 | 친환경찹 | 산코이네노 | 2559.9 | 2668.62 | 358.52 | 124.84 | 64.29 | 1556.5 | 2.6 | 1.7 | 105.7 | 979.5 | 16.8 | 3.76 | 142.02 | 1.91 | 3 | 3 |
| 1 | 완두콩밥, | 텐돈 3컬러 | 3254.4 | 3302.51 | 477.47 | 142.86 | 85.4 | 1260.9 | 1.8 | 2.8 | 133.9 | 1091.4 | 27.2 | 5.07 | 15.31 | 0.9 | 3 | 3 |
| 1 | 현미밥,해 | 트리플 볕 | 731.9 | 1282.35 | 172.43 | 44.2 | 41.24 | 317.3 | 0.3 | 0.2 | 45.1 | 471.1 | 7.9 | 8.04 | 1736 | 9.08 | 1 | 0 |
| 1 | 우리밀밥, | 알로하망고 | 728.9 | 1439.9 | 198.8 | 70.4 | 40.3 | 263.2 | 0.8 | 0.4 | 43.7 | 298.3 | 6.4 | 30.38 | 1938 | 5.3 | 1 | 1 |

# IV.알고리즘 설명 및 선택 배경 (장단점 등)

## A. K-means Algorithm

1)알고리즘 설명

주어진 데이터를 k개의 클러스터로 묶는 알고리즘으로, 각 클러스터와 거리 차이의 분산을 최소화하는 방식으로 동작한다. 알고리즘은 다음과 같은 단계로 진행된다.

(1) 전체 클러스터의 개수 K를 정한다.

(2) 데이터 집합에서 K개의 instance를 무작위로 골라서,이것을 K개 클러스터 각각에 대한 시작 클러스터 센터로 사용한다.

(3) 유클리디언 거리(Euclidean Distance) 를 사용하여 나머지 instance들을 K개의 센터 와 비교하여 제일 가까운 센터의 클러스터에 배정한다.

(4) 각 클러스터에 대하여 그 클러스터에 배정된 instance들을 가지고 평균 센터를 새로 계산한다.

(5) 만약 그 센터가 이전 센터와 똑같다면 프로세스를 멈추도록 하고, 그렇지 않다면 그 새로운 센터를 사용하여 step3-5를 반복한다.

2) 선택 배경

일주일치 식단 data를 몇개의 label로 나눌지 고민하다 k-means algorithm의 elbow method를 참고하면 최적의 label 갯수를 찾을 수 있을 것 같아 k-means clustering 을 진행하였다.

## B. PCA(Principal Component Analysis)

1) 알고리즘 설명

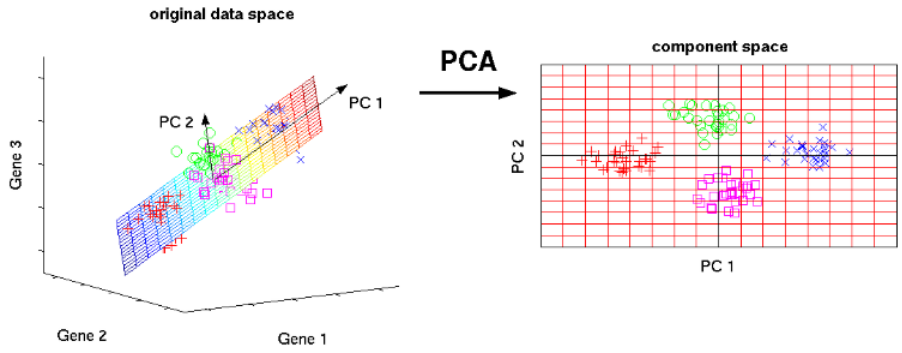
주성분 분석(PCA, Principal Component Analysis)은 가장 대표적인 차원 축소 알고리즘이다. PCA는 다음과 같은 단계로 이루어진다.

(1) 학습 데이터셋에서 분산이 최대인 축(axis)을 찾는다.

(2) 이렇게 찾은 첫번째 축과 직교(orthogonal)하면서 분산이 최대인 두 번째 축을 찾는다.

(3) 첫 번째 축과 두 번째 축에 직교하고 분산을 최대한 보존하는 세 번째 축을 찾는다.

(4) 1~3과 같은 방법으로 데이터셋의 차원(특성 수)만큼의 축을 찾는다.




[차원이 축소된 모습]

2) 선택배경

식단 data에는 칼로리, 탄수화물, 단백질 ,지방, 비타민a, 티아민, 리보플라빈,비타민c, 칼슘,철분 총 9개의 feature가 있는데, feature가 많아지면 차원 역시 증가하기 때문에 데이터들 간의 거리가 증가하게 된다. 이는 데이터를 training 시킬 때 overfitting이 될 위험이 커진다. 따라서 9개의 feature들을 pca를 통해 2개의 feature로 변환시키고자 해당 알고리즘을 선택하게 되었다.

3) pca의 장점 및 단점

< 장점 >

(1) 다중공선성 제거

pca를 쓰면 데이터의 큰 손실 없이 다중공선성이 없는 서로 무관한(수직) 새로운 변수들로 변환해줄 수 있는 장점이 있다.

(2) 학습 집합의 크기 줄여줌

학습 집합의 크기가 줄어들었기 때문에 학습 속도도 빨라지게 된다.

< 단점 >

(1) 특징 벡터의 label을 고려하지 않는다.

(2) 결과값이 특징구분을 좋게 한다는 보장이 없다.

### C. KNN algorithm

1) 알고리즘 설명

KNN algorithm은 특정공간 내에서 입력과 제일 근접한 k개의 요소를 찾아, 더 많이 일치하는 것으로 분류하는 알고리즘이다.


데이터들 간의 distance를 측정하는 방법은 Euclidean Distance, Manhattan Distance, Correlation Distance 등 여러 방법이 있는데, 분석 프로그램에는 Euclidean Distance를 적용하였다.

2) 선택배경

K-means algorithm으로 생성된 label의 개수들을 가지고 식단 data들이 각각 어느 label에 속하는지 알기 위해 KNN algorithm을 적용하였다. KNN algorithm을 통해 모든 데이터들의 label들을 결정하고, 각 label에 맞는 식단들을 추천해주는 단계로 이어 나갔다.

3) KNN algorithm의 장점 및 단점

< 장점 >

(1) 간단하고 효과적이다.

(2) 분석하는 데이터에 대한 기본적인 분포 가정이 없다.

(3) 분류와 숫자 예측 모두에 적용 가능하다.

(4) 학습과정이 빠르다.

< 단점 >

(1) 모든 데이터와 거리계산을 해야 하므로, 데이터 크기가 커지면 비용이 커진다.

(2) 데이터 구조에 대한 정보를 구할 수 없다.

(3) 명목변수와 결측값은 따로 처리해야 한다.

(4) 모델 구축을 하지 않아서 관계에서 새로운 인사이트를 얻는데 제한이 있다.

# V. 분석 프로그램 설명 및 캡처 화면

## 1. 식단 평가

식단 평가는 총 2가지인 하루 식단 평가와 전체 일주일의 식단 총평가로 나눠진다. 따라서 평가에 사용되는 데이터는 두개로 나뉜다. 하나의 행에 일주일치 식단이 모두 통합된 데이터는 일주일 식단을 총 평하는데 사용되며 하나의 행에 각 요일마다 하루치의 식단만 포함한 데이터는 하루 식단 점검할 때 사용된다. 사용되는 데이터는 '신체식단조합.csv'와 '신체식단조합(하루).csv'에 저장되어 있다.

'신체식단조합.csv'을 만들 때 하루하루의 식단은 "@@"를 사용하여 구별한다. 그리고 인체 데이터 개수인 1404개의 일주일 식단을 랜덤하게 뽑아서 인체 데이터와 조합을 만든다.

```
h=pd.read_csv("인체치수 최종본.csv",encoding='utf-8-sig',engine="python",index_col=[0])
a=pd.read_csv("일주일 식단.csv",encoding='utf-8-sig',engine="python",index_col=[0])
f=pd.DataFrame(0,columns=a.columns,index=[0])

def sampling_func(data):
    N = len(data)
    sample_n=1404 # 1404개 뽑기 (학생 수)
    sample = data.take(np.random.permutation(N)[:sample_n])
    return sample

for i in range(1,5002):
    g=a[a['num']==i].iloc[:,3:].sum() # 일주일 식단
    g=pd.DataFrame(g).T
    g["num"]=i
    g["일주일 식단"]="@@".join(a[a['num']==i].iloc[:,1]) # 식단 합치기
    g["일주일 간식"]="@@".join(a[a['num']==i].iloc[:,2]) # 간식 합치기
    f=pd.concat([f,g])

f=f.reset_index(drop=True)
f=f[1:] # 첫번째 행 제거하기
sample=sampling_func(f)
sample=sample.reset_index(drop=True)
week=sample.num
g=pd.concat([h,sample],axis=1)

# 열이름 바꾸기
g.columns=['성별','골격근량','체지방량', 'BMI', '체지방율', '복부지방율', '몸무게', '나이', '키', '기초대사량', '탄수화물', '단백질',
           '지방', '비타민A','티아민', '리보플라빈', '비타민C', '칼슘', '철분', '알레르기', '1.난류', '10.돼지고기', '11.복숭아',
           '12.토마토', '13.아황산염', '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)', '2.우유',
           '3.메밀', '4.땅콩', '5.대두', '6.밀', '7.고등어', '8.게', '9.새우', 'num', '간식','나트륨(mg)','당류(g)','리보플라빈(mg)',
           '비타민A(mg)', '비타민C(mg)', '식단', '일주일 간식', '일주일 식단', '철분(mg)', '총 단백질(g)', '총 열량(kcal)',
           '총 지방(g)', '총 탄수화물(g)', '칼로리(Kcal)', '칼슘(mg)', '티아민(mg)', '포화지방산(g)']
```

```
# 열 순서 바꾸기
g=g[['성별', '골격근량', '체지방량', 'BMI', '체지방율', '복부지방율', '몸무게', '나이', '키', '기초대사량', '탄수화물', '단백질',
     '지방','비타민A', '티아민', '리보플라빈', '비타민C', '칼슘', '철분', '알레르기', '일주일 간식', '일주일 식단','총 열량(kcal)',
     '총 탄수화물(g)','총 단백질(g)','총 지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)', '칼슘(mg)',
     '철분(mg)', '나트륨(mg)', '당류(g)', '포화지방산(g)','1.난류', '2.우유', '3.메밀', '4.땅콩', '5.대두', '6.밀', '7.고등어',
     '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염', '14.호두', '15.닭고기', '16.쇠고기', '17.오징어',
     '18.조개류(굴,전복,홍합 등)',]]

g.to_csv("신체식단조합.csv",encoding="utf-8-sig")
```

| 성별 | 골격근량 | 체지방량 | BMI | 체지방율 | 복부지방율 | 몸무게 | 나이 | 키 | 기초대사량 | 탄수화물 | 단백질 | 지방 | 비타민A | 티아민 | 리보플라빈 | 비타민C | 칼슘 | 철분 | 알레르기 | 일주일 간 | 일주일 식 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 | 2700 | 209.7825 | 104.8913 | 15.53944 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 데니쉬 트 | 쌀밥,돼지 |
| 1 | 22.1 | 7 | 16.83558 | 14.7 | 0.7 | 47.8 | 17 | 168.5 | 2700 | 217.695 | 108.8475 | 16.12556 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 바나나향 | 서리태밥 |
| 1 | 22.3 | 6.7 | 17.79882 | 14.3 | 0.8 | 47 | 17 | 162.5 | 2700 | 211.545 | 105.7725 | 15.67 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | 5,7 | 토스트 비 | 찹쌀밥,조 |
| 1 | 22.4 | 5.9 | 18.05486 | 12.7 | 0.8 | 46.8 | 17 | 161 | 2700 | 210.0075 | 105.0038 | 15.55611 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 청정원 요 | 혼합잡곡밥 |
| 1 | 22.5 | 6.2 | 16.5024 | 13.2 | 0.7 | 47.3 | 17 | 169.3 | 2700 | 217.2638 | 108.6319 | 16.09361 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 그대로카 | 쌀밥,쇠고 |
| 1 | 22.6 | 8 | 17.89867 | 16.3 | 0.8 | 49.5 | 17 | 166.3 | 2700 | 219.5513 | 109.7756 | 16.26306 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 몬스터 먼 | 친환경쌀밥 |
| 1 | 22.7 | 8.6 | 18.66659 | 17.2 | 0.8 | 49.9 | 17 | 163.5 | 2700 | 218.2763 | 109.1381 | 16.16861 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | 15,1 | 링키바클 | 쌀밥,벌집 |
| 1 | 22.8 | 3.5 | 17.15937 | 7.8 | 0.8 | 45.2 | 17 | 162.3 | 2700 | 207.6825 | 103.8413 | 15.38389 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 뷰티카로 | 백미밥,얼 |
| 1 | 22.9 | 6.9 | 16.70823 | 14.1 | 0.7 | 49.2 | 17 | 171.6 | 2700 | 222.9075 | 111.4538 | 16.51167 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 불고기버 | 백미밥,평 |
| 1 | 23.4 | 2.9 | 16.74931 | 6.3 | 0.7 | 45.6 | 17 | 165 | 2700 | 210.5325 | 105.2663 | 15.595 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 네이처밸 | 친환경찹 |

"신체식단조합(하루).csv"는 전처리 과정에서 생성한 '일주일 식단.csv'속 데이터를 모두 사용한다. 모든 일주일 식단을 인체 데이터와 조합을 만들기 위해서 같은 일주일을 배정받은 인체의 데이터는 7번 반복하여 7개의 row에 배정한다.

```
week=week.tolist()

row=[]
for j in range(len(a)):
    if a.iloc[j,0] in week:
        row.append(a.iloc[j])

everyday=pd.DataFrame(row)
everyday=everyday.reset_index(drop=True)

k=pd.DataFrame(pd.np.repeat(h.values,7,axis=0),columns=h.columns)
everyday2=pd.concat([k,everyday],axis=1) # 식단과 인체 조합

# 열 이름 바꾸기
everyday2.columns=['성별', '골격근량', '체지방량', 'BMI', '체지방율', '복부지방율', '몸무게', '나이', '키', '기초대사량',
       '탄수화물', '단백질', '지방', '비타민A', '티아민', '리보플라빈', '비타민C', '칼슘', '철분', '알레르기',
       'num', '하루 식단', '하루 간식', '칼로리(Kcal)', '총 열량(kcal)', '총 탄수화물(g)', '총 단백질(g)',
       '총 지방(g)', '비타민A(mg)', '티아민(mg)', '리보플라빈(mg)', '비타민C(mg)', '칼슘(mg)',
       '철분(mg)', '당류(g)', '나트륨(mg)', '포화지방산(g)', '1.난류', '2.우유', '3.메밀',
       '4.땅콩', '5.대두', '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아',
       '12.토마토', '13.아황산염', '14.호두', '15.닭고기', '16.쇠고기', '17.오징어',
       '18.조개류(굴,전복,홍합 등)']
# 열 순서 바꾸기
everyday2=everyday2[['성별', '골격근량', '체지방량', 'BMI', '체지방율', '복부지방율', '몸무게', '나이', '키', '기초대사량',
      '탄수화물', '단백질', '지방', '비타민A', '티아민', '리보플라빈', '비타민C', '칼슘', '철분', '알레르기',
      '하루 식단', '하루 간식','총 열량(kcal)','총 탄수화물(g)','총 단백질(g)','총 지방(g)', '비타민A(mg)', '티아민(mg)',
      '리보플라빈(mg)', '비타민C(mg)', '칼슘(mg)', '철분(mg)', '나트륨(mg)', '당류(g)', '포화지방산(g)','1.난류', '2.우유',
      '3.메밀', '4.땅콩', '5.대두', '6.밀', '7.고등어', '8.게', '9.새우', '10.돼지고기', '11.복숭아', '12.토마토', '13.아황산염',
      '14.호두', '15.닭고기', '16.쇠고기', '17.오징어', '18.조개류(굴,전복,홍합 등)',]]

everyday2.to_csv("신체식단조합(하루).csv",encoding="utf-8-sig")
```

| 성별 | 골격근량 | 체지방량 | BMI | 체지방율 | 복부지방율 | 몸무게 | 나이 | 키 | 기초대사량 | 탄수화물 | 단백질 | 지방 | 비타민A | 티아민 | 리보플라빈 | 비타민C | 칼슘 | 철분 | 알레르기 | 하루 식단 | 하루 간식 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 | 2700 | 209.7825 | 104.8913 | 15.53944 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 흰밥,육개장,아이카 감 | |
| 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 | 2700 | 209.7825 | 104.8913 | 15.53944 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 친환경발아웰드콘 커 | |
| 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 | 2700 | 209.7825 | 104.8913 | 15.53944 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 방울토마토맥시조코우 | |
| 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 | 2700 | 209.7825 | 104.8913 | 15.53944 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 쌀밥,콩나물죠나마 바 | |
| 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 | 2700 | 209.7825 | 104.8913 | 15.53944 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 쌀밥,김치볶스나이더스 | |
| 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 | 2700 | 209.7825 | 104.8913 | 15.53944 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 잡쌀밥,부다나포리바닐 | |
| 1 | 21.7 | 6 | 16.95419 | 13.2 | 0.7 | 45.6 | 17 | 164 | 2700 | 209.7825 | 104.8913 | 15.53944 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 잡곡밥,돈육쯔마미 이 | |
| 1 | 22.1 | 7 | 16.83558 | 14.7 | 0.7 | 47.8 | 17 | 168.5 | 2700 | 217.695 | 108.8475 | 16.12556 | 850 | 1.3 | 1.7 | 110 | 900 | 15 | | 버터가장초우기농 딸 | |

모든 평가의 기준은 다음과 같다. 열량, 탄수화물, 단백질, 지방, 각종 비타민, 칼슘, 철분은 권장 섭취량의 5%이내의 양을 섭취하지 못했으면 지적하고 나트륨은 300mg이상, 당류는 20mg이상, 포화지방산은 12mg 이상 먹었을 시 지적한다.

| 영양소 | 평가 기준 |
|---|---|
| 열량(kcal) | 권장 섭취량의 5% 이내 양을 섭취하지 못했으면 지적 |
| 탄수화물(g) | |
| 단백질(g) | |
| 지방(g) | |
| 각종 비타민(mg) | |
| 칼슘(mg) | |
| 철분(mg) | |
| 나트륨(mg) | 300mg 이상 먹었을 시 지적 |
| 당류(g) | 20mg 이상 먹었을 시 지적 |
| 포화지방산(g) | 12mg 이상 먹었을 시 지적 |

1) 하루 식단 평가

하루 식단 평가는 하루치 식단에 들어가 있는 각종 영양소와 각 학생 개인의 영양분 권장량을 비교하여 문제가 있는 영양분을 지적한다. 모든 영양분을 비교하여 얼마만큼 더 먹거나 덜 먹어야 하는지 알려준다. 또한, 학생이 가지고 있는 알레르기 종류가 들어있는 음식을 먹었을 때도 함께 알려준다.

```
data=pd.read_csv("신체식단조합(하루).csv",encoding='utf-8-sig',index_col=[0])
data["알레르기"]=data["알레르기"].fillna(0)
allergy={1:"난류", 2:"우유", 3:"메밀", 4:'땅콩', 5:'대두', 6:'밀', 7:'고등어', 8:'게', 9:'새우', 10:'돼지고기', 11:'복숭아',
         12:'토마토', 13:'아황산염', 14:'호두', 15:'닭고기',16:'쇠고기', 17:'오징어', 18:'조개류(굴,전복,홍합 등)'}

def get_info_daybyday(index):
    result_list=[]
    iteration=10 # 10개의 권장량 비교
    info=['기초대사량','탄수화물','단백질','지방','비타민A','티아민','리보플라빈','비타민C','칼슘','철분']
    measure=['kcal','g','g','g','mg','mg','mg','mg','mg','mg']
    for i in range(iteration):
        value=data.iloc[index,(i+9)]
        value1=value*0.95 # -5% 허용범위
        value2=value*1.05 # +5% 허용범위
        stand=data.iloc[index,(i+22)] #비교 대상

        if stand<value1:#덜먹은 것
            string=info[i]+'을 '+str(round(value-stand,2))+measure[i]+' 만큼을 더 먹으세요'
            result_list.append(string)
        elif stand>value2:#더 먹음
            string=info[i]+'을 '+str(round(stand-value,2))+measure[i]+' 만큼을 덜 먹으세요'
            result_list.append(string)
        else:
            string=info[i]+'은(는) 이 식습관 유지하세요'
            result_list.append(string)
```

```
    if data.loc[index,'나트륨(mg)']>300:
        result_list.append('나트륨(mg)을 '+str(round(data.loc[index,'나트륨(mg)']-300,2))+' 만큼  덜드세요')
    if data.loc[index,'당류(g)']>20:
        result_list.append('당류(g)를 '+str(round(data.loc[index,'당류(g)']-20,2))+' 만큼  덜드세요')
    if data.loc[index,'포화지방산(g)']>12:
        result_list.append('포화지방산(g)을 '+str(round(data.loc[index,'포화지방산(g)']-12,2))+' 만큼  덜드세요')

    if data.loc[index,"알레르기"] !=0:
        idx=index
        idx2=data.loc[idx,"알레르기"].split(",")
        column=[]
        alnum=[]
        allergyname=[]
        for i in range(18):
            if data.iloc[idx,35+i] != 0:
                column.append(data.columns[35+i])
        for j in range(len(idx2)):
            for k in range(len(column)):
                if idx2[j] in column[k]:
                    alnum.append(idx2[j])
        for i in alnum:
            allergyname.append(allergy[int(i)])
            allergyname=list(set(allergyname))
            if allergyname != []:
                result_list.append("알레르기를 유발하는 음식을 먹었어요 : "+", ".join(allergyname))

    totalstring=", ".join(result_list)
    return totalstring
```

```
totaldbd=[] #최종결과는 이중리스트 형태
for i in range(len(data)):
    totaldbd.append(get_info_daybyday(i))
data["평가결과"]=totaldbd
data.to_csv("하루평가.csv",encoding="utf-8-sig")
```

결과 예시)

왼쪽과 같은 식단을 먹은 경우 오른쪽과 같은 평가가 나온다.

**아침**

흰밥,시나몬토스트,쇠고기무국,닭가슴살장조림,두부매운조림,배추김치,우유,옥수수콘버터구이

**점심**

현미밥,콩나물수제비국,등갈비찜,풋마늘골뱅이초회,마늘돈까스,양배추샐러드,오렌지

**저녁**

기장밥,쇠고기콩나물국,참나물오이무침,닭날개 당호박조림,김치떡볶음,배추김치,얼라이브쥬스

**간식**

망고맛 푸딩, 무기후아바닐라맛웨이퍼

'권장섭취량은(는) 이 식습관 유지하세요',

'탄수화물을 1038.1g 만큼을 덜 드세요',

'단백질을 100.85g 만큼을 덜 드세요',

'지방을 373.0g 만큼을 덜 드세요',

'비타민A을 475.3mg 만큼을 덜 드세요',

'티아민을 15.7mg 만큼을 덜 드세요',

'리보플라빈을 9.4mg 만큼을 덜 드세요',

'비타민C은(는) 이 식습관 유지하세요',

'칼슘을 1432.7mg 만큼을 덜 드세요',

'철분을 30.8mg 만큼을 덜 드세요'

2) 일주일치 총 평가

일주일 식단 총평은 일주일치 식단에 들어 있는 총 영양분과 각 학생 개인의 (일일 영양분 권장량 X7)을 비교하여 모든 영양소 총평을 한다.

```
df1=pd.read_csv('신체식단조합.csv',sep=',',encoding='utf-8-sig',engine='python',index_col=[0])

def get_info(index):
    result_list=[]
    iteration=10 # 10개의 권장량 비교
    info=['기초대사량','탄수화물','단백질','지방','비타민A','티아민','리보플라빈','비타민C','칼슘','철분']
    measure=['kcal','g','g','g','mg','mg','mg','mg','mg','mg']
    for i in range(iteration):
        value=df1.iloc[index,(i+9)]*7
        value1=value*0.95 # -5% 허용범위
        value2=value*1.05 # +5% 허용범위
        stand=df1.iloc[index,(i+22)] #비교 대상

        if stand<value1:#덜먹은 것
            string=info[i]+'을 '+str(round(value-stand,2))+measure[i]+' 만큼을 더 드세요'
            result_list.append(string)
        elif stand>value2:#더 먹음
            string=info[i]+'을 '+str(round(stand-value,2))+measure[i]+' 만큼을 덜 드세요'
            result_list.append(string)
        else:
            string=info[i]+'은(는) 이 식습관 유지하세요'
            result_list.append(string)
```

```
    if df1.loc[index,'나트륨(mg)']>300*7:
        a='나트륨(mg)을 '+str(round(df1.loc[index,'나트륨(mg)']-300,2))+' 만큼 덜드세요'
        result_list.append(a)
    if df1.loc[index,'당류(g)']>20*7:
        b='당류(g)를 '+str(round(df1.loc[index,'당류(g)']-20,2))+' 만큼 덜드세요'
        result_list.append(b)
    if df1.loc[index,'포화지방산(g)']>12*7:
        c='포화지방산(g)을 '+str(round(df1.loc[index,'포화지방산(g)']-12,2))+' 만큼 덜드세요'
        result_list.append(c)

    totalstring=", ".join(result_list)

    return totalstring

total=[]
for i in range(len(df1)):
    total.append(get_info(i))

df1['평가결과']=total
data.to_csv("일주일총평가.csv",encoding="utf-8-sig")
```

결과 예시)

왼쪽과 같은 식단으로 일주일을 먹은 경우에 오른쪽과 같은 평가가 출력된다.

**<일주일 조식 또는 중식 또는 석식>**

친환경쌀발아현미밥,콩나물국저염식,한우버섯장조림,수제소시지구이,유기농배추김치,파스타발사믹샐러드,파인애플과일,우리밀크린베리쿠키빵//친환경쌀수수밥,냉이콩국저염식,통삼겹살구이,보쌈김치,파절이생채나물,양배추숙회,물미역숙회나물,수제머핀//친환경쌀낙지고추장비빔밥,봄동된장국저염식,연두부, 섭산적샐러드,잡채,유기농배추김치,파인애플쥬스 … 생략

**<일주일 간식>**

큰컵간짬뽕, 롤스토리 골든 롤 초코@@초코팡, 하이랜드스톰샤워(크라운제과)@@오리지널단팥크림빵, 로아커웨하스 나폴리타너@@인디 라운드 나쵸칩, 매운꿀꽈배기(농심)@@맥시 카사바 칩(사워크림), 청정원우리밀찰진국수 (대정농산㈜)@@진라면 컵 매운맛(오뚜기라면), 치오 스틱-감자맛@@세일럼 클래식 슈가 쿠키, 초이스엘 나쵸칩

'권장섭취량을 1676.95kcal 만큼을 더 드세요',

'탄수화물을 890.17g 만큼을 덜 드세요',

'단백질을 50.47g 만큼을 더 드세요',

'지방을 351.7g 만큼을 덜 드세요',

'비타민A을 635.7mg 만큼을 덜 드세요',

'티아민을 4.4mg 만큼을 덜 드세요',

'리보플라빈을 1.5mg 만큼을 더 드세요',

'비타민C은(는) 이 식습관 유지하세요',

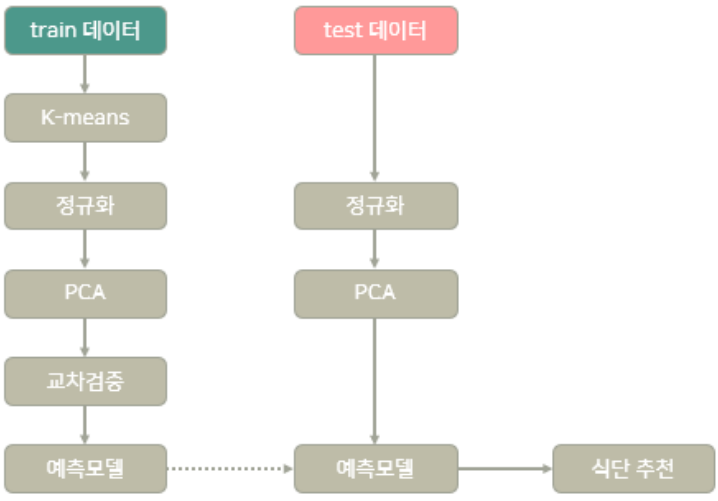
'칼슘을 1495.0mg 만큼을 더 드세요',

'철분을 23.1mg 만큼을 덜 드세요'

'나트륨(mg)을 2968.21 만큼 덜 드세요'

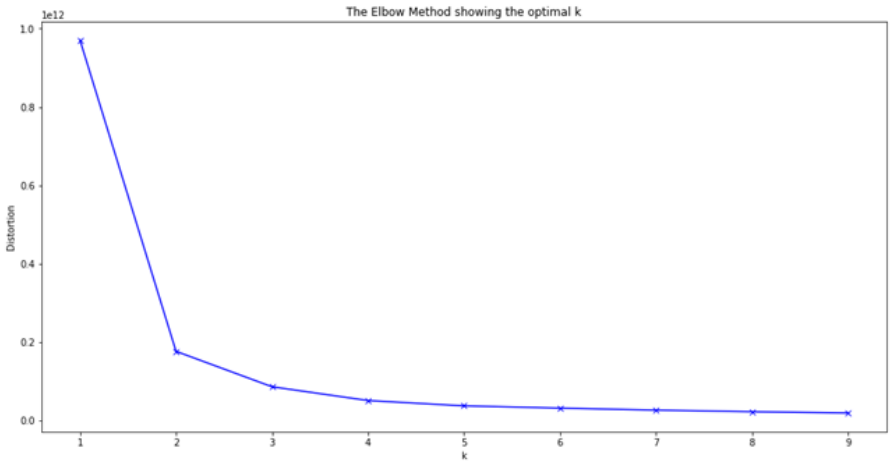
## 2. 식단 추천

식단 추천에는 다음과 같은 알고리즘을 순차적으로 적용하였다.




다음 알고리즘을 적용할 데이터셋은 일주일동안 세끼를 꼬박꼬박 먹은 식단과 일주일에 다섯번은 세끼를 먹고, 두번 정도는 한끼나 두끼를 먹은 식단(고등학생은 평일 동안에는 학교에 있기 때문에 일주일에 5번정도는 세끼를 먹는다고 가정하였다.)을 섞은 10000개의 데이터로 이루어져 있다. 이 10000개의 데이터를 4:1 비율로 train데이터와 test 데이터로 나누었다.

(1) k-means algorithm 적용




K-means 알고리즘의 elbow method 그래프를 통해 최적의 label 개수를 탐색하였다. 그래프를 보면 k가 2일 때와 3일 때가 가장 적절해 보이는데, 그 중 홀수인 3으로 label 개수를 선정하였다.

(2) 정규화

PCA를 적용하기 이전에 9개의 영양소마다 (칼로리, 탄수화물, 단백질 ,지방, 비타민a, 티아민, 리보플라빈,비타민c, 칼슘,철분) 단위가 전부 다 다르기 때문에, 크기가 큰 단위 값이 큰 값이 나오는 것을 막기 위해 정규화를 진행하였고 정규화를 통해 영양소 마다 동일한 단위를 가지는 결과를 가진다.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(value)
value_transform=scaler.transform(value)
```

| | 칼로리(Kcal) | 탄수화물(g) | 단백질(g) | 지방(g) | 비타민A(mg) | 티아민(mg) | 비타민C(mg) | 리보플라빈(mg) | 칼슘(mg) | 철분(mg) |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.234927 | -0.901105 | -1.198286 | -0.032841 | -0.312273 | 0.790511 | -0.637530 | -0.366321 | -1.317727 | -0.543711 |
| 1 | -0.149315 | 0.097261 | -0.515773 | -0.132695 | 0.087358 | -0.448208 | 0.040284 | -0.264731 | 0.666130 | -0.053567 |
| 2 | -0.123845 | -0.230331 | -0.336077 | -0.034330 | -0.213755 | -0.442903 | -0.680212 | -0.357855 | -0.572266 | -0.000196 |
| 3 | -0.197394 | -0.681058 | -0.915264 | -0.029034 | -0.152285 | -0.493300 | -1.205135 | -0.346567 | -1.264799 | -0.147239 |
| 4 | -0.019752 | -0.264617 | 0.009173 | -0.086904 | -0.007463 | 3.037182 | -0.085851 | 3.341712 | 0.119358 | 0.489948 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 7995 | -0.051822 | -0.119705 | 0.314490 | -0.043092 | 0.271467 | -0.347413 | 1.509942 | -0.225224 | -0.375244 | -0.112384 |
| 7996 | 0.094170 | 0.883161 | 0.543270 | -0.033482 | 0.518152 | -0.387200 | 1.206709 | -0.225224 | 1.040413 | -0.049210 |
| 7997 | -0.200605 | -1.003133 | -0.584656 | 0.014363 | -0.442683 | 1.021279 | -1.213417 | -0.010756 | -0.490868 | -0.414095 |
| 7998 | -0.085196 | -0.207295 | 0.490027 | -0.093688 | 0.836544 | 0.058420 | -0.315186 | -0.225224 | 0.113224 | 0.163185 |
| 7999 | -0.111021 | -0.571584 | 0.144777 | -0.036497 | 0.044827 | 2.565037 | 0.023083 | 2.876092 | 0.059842 | -0.165756 |

(3) PCA

9개의 모든 feature를 한꺼번에 고려하기 어렵기 때문에 PCA 통해 9개의 영양정보 feature들을 2개의 feature로 차원을 축소시켰다.

```
pca = PCA(n_components=2)
pca.fit(value_transform)
X_pca = pca.transform(value_transform)
```

| | 0 | 1 |
|---|---|---|
| 0 | -1.785358 | 0.360137 |
| 1 | -0.057914 | -0.448880 |
| 2 | -0.787412 | -0.561652 |
| 3 | -1.599988 | -0.584167 |
| 4 | 0.444868 | 4.456463 |
| ... | ... | ... |
| 7995 | 0.287034 | -0.329894 |
| 7996 | 1.376286 | -0.413287 |
| 7997 | -1.332372 | 0.721748 |
| 7998 | 0.398626 | -0.149083 |
| 7999 | 0.190439 | 3.825443 |

[ 두개의 feature로 변경된 모습 ]

(4) 교차검증

KNN 알고리즘에서의 k 값을 정하기 위해 교차검증(k-fold)를 실시함

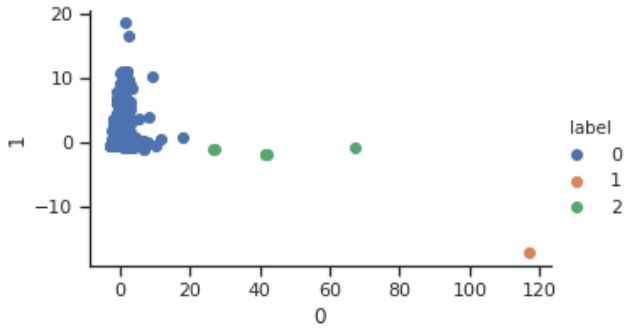
```
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

model = KNeighborsClassifier(n_neighbors=6) # 모델 객체 생성
kfold = KFold(n_splits=5)
scores = cross_val_score(model, value, label, cv=5)
print('cross validation score: {}'.format(scores))
print("고차검증의 평균 : %.3f" %(scores.mean()))
```

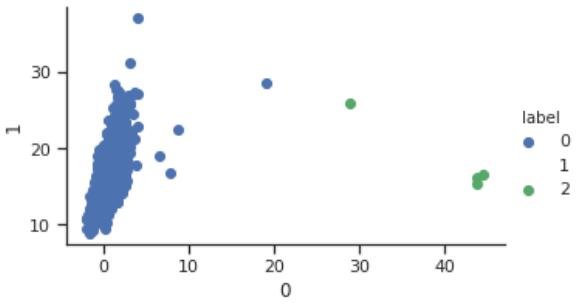
k 가 4,5,6일 때, 정확도 평균값을 측정한 결과 세 값 모두 거의 동일한  결과 값이 나와 그 중 가운데  값인 5로 k를 선정함

(5) KNN 알고리즘 적용한 예측 모델 생성

Test 데이터도 위와 같은 결과를 거친 뒤에 최종으로 KNN 알고리즘을 적용해 예측 모델을 생성한다.




[train data 결과]




[test data 결과]

(6) 식단 추천

모델링 결과 label 0에 분류된 학생은 올바른 식단을 먹은 학생이기 때문에 현 식습관을 유지하도록 삼식이 data에서  식단 column만 불러와 랜덤으로 7개를 가져와 일주일치를 추천해준다.(label 0 추천 식단:삼식이.csv)

Label 2로 분류된 학생은 올바르지 않은 식단을 먹은 학생이기 때문에 급식 식단보다 더 건강하게 짜여진 병원 식단들 중에 랜덤으로 하나를 가져와 일주일치를 추천해준다.(label 2 추천 식단: label2 추천.csv)

추천 예시)

**Label 0으로 분류된 학생**

| | | |
|---|---|---|
| 월 | 조식 | 찰현미밥,설렁탕,돈육볶음,깍두기,호두과자,우유 |
| | 중식 | 차수수밥,콩나물쫄면,팽이버섯된장국,도토리묵상추무침,치킨바켄,배추김치 |
| | 석식 | 치볶음밥,콩나물쫄면,미소된장국,단무지무침,불고기함박스테이크,요구르트 |
| 화 | 조식 | 잡쌀밥,북어무국,떡갈비견과류조림,청경채나물,배추김치,바나나 |
| | 중식 | 찰보리밥,부대찌개,오이고추무침,오리불고기,감자전,배추겉절이 |
| | 석식 | 잡곡밥,오징어국,혼제오리&머스터드소스,두부구이,깍두기,부추겉절이 |
| 수 | 조식 | 케찹,누룽지탕,눈꽃치즈토마토,배추김치,불고기타코,초코에몽 |
| | 중식 | 강황쌀밥,우리밀삼색수제비국,콩나물파채무침,통삼겹구이,배추김치,수박,상추 |
| | 석식 | 후랑크소시지도시락,근대된장국,깍두기,회오리감자,청포도에이드 |

**Label 2로 분류된 학생**

| | | |
|---|---|---|
| 월 | 조식 | 삼색수제비국,삼치카레구이, 진미채볶음, 냉이된장무침,배추김치 |
| | 중식 | 오징어무국, 스크램블에그, 베이컨감자채볶음, 쑥갓나물, 배추김치 |
| | 석식 | 톳밥, 추어탕, 돈육계란장조림, 해물파전, 깻잎지, 알타리김치 |
| 화 | 조식 | 동태알탕, 닭감자조림, 톳두부무침, 애호박새우젓볶음,배추김치 |
| | 중식 | 육개장, 두부전&양념장, 코다리순살무조림, 유채나물, 배추김치 |
| | 석식 | 매생이굴국, 돼지갈비떡찜, 숙주맛살무침, 참나물무침, 배추김치 |
| 수 | 조식 | 쇠고기떡국, 참치야채볶음, 감자메추리알조림, 유채나물,배추김치 |
| | 중식 | 누룽지, 쇠고기계란장조림, 표고버섯새우살볶음, 어묵고추장볶음, 깍두기 |
| | 석식 | 찰현미밥, 두부호박청국장국, 삼치카레구이, 돌김&양념장,오이소박이, 파김치 |

## VI. 총평 및 한계점

### A. 기대효과

1. 식습관 검증 및 식습관 개선 방안에 대해 알 수 있다.
2. 부족한 영양소가 무엇인지 알고, 보충할 수 있다.
3. 현재 실시하고 있는 급식 식단의 영양소가 잘 짜여 있는지 알 수 있다.
4. 나에게 맞는 올바른 식단을 추천 받을 수 있다.

### B. 추후연구

1. 고등학생 대상에서 전체 연령으로 대상을 확대하여 분석을 실시한다.

이번 연구에서는 급식 식단 데이터로 식단을 한정 지었기에 고등학생을 대상으로만 분석을 실시하였지만 추후 연구의 범위가 넓어진다면 전체 연령을 대상으로 분석을 실시하면 좋을 것이다.

2. 실제 일주일동안 먹은 식단을 input 받아 데이터로 사용한다.

1번과 연결하여 급식 데이터가 아닌 실제로 이용자가 일주일동안 먹은 식단, 메뉴를 상세하게 input 받아 연구를 진행하면 좋을 것이다.

3. 보유한 알레르기가 포함되는 식단은 제외하고 식단을 추천해준다.

식단 추천 시 보유한 알레르기가 포함된 식단은 아예 추천 대상에서 제외하여 추천해주면 좋을 것이다.

### C. 한계점

1. 간식 데이터의 함량 영양소 값에 오류가 있는 경우가 있다.

간식 데이터의 영양소 값이 터무니없이 작거나 큰 경우가 간혹 있어 정확한 값인지 의심되었는데, 데이터 제공 기관의 측정 오류이거나 단순한 의심일 뿐이라 이를 해결할 방도가 없었다.

2. 실제 섭취한 식단을 input 하지 않고 급식 식단으로 대체하였다.

실제 학생들이 섭취한 식단을 input 받은 데이터를 사용했으면 결과가 더욱 정확 해졌겠지만 그러한 데이터를 얻기가 어려워, 비교적 얻기 쉬운 데이터인 급식 식단을 이용하였다. 문제는 급식 식단을 영양사가 각 영양성분의 섭취를 고려하여 짠 것이므로 이 데이터를 이용하여 분석 시 결과가 균형적으로 분포되거나, general하게 나올 수 있다는 단점이 있다.

3. 각 개인의 알레르기 정보 데이터를 얻지 못했다.

인체 치수 데이터에는 알레르기 여부를 나타내는 항목이 없어 실제 데이터를 찾지 못했다. 때문에 알레르기를 가지고 있을 확률을 임의로 정하여 정보를 생성했으므로 실제와 데이터가 크게 차이가 날 위험이 있다.

4. 병원 식단 데이터의 영양소가 건강하게 잘 짜여져 있다 가정하였다.

식단 추천 결과로 올바르지 않은 식단을 먹은 학생에게는 올바른 영양섭취를 장려하는 의미에서 병원 식단을 추천하였는데, 실제 병원 식단 데이터에는 영양소 항목이 없으므로 환자를 위한 식단이라는 이유로 영양소가 잘 짜여져 있다 가정하였다. 병원 식단 데이터에도 영양소 column이 있었다면 실제 병원 식단 데이터의 영양소가 고르게 짜여 있는지 비교해 보았을 것이다.