20806 범준환

소방서로부터 가장 가까운 고등학교는?

2021-07-16 (금)

0. 동기

8교시 소방훈련을 하다 문득 든 생각이다. 만일 학교에 불이 난다면? 뭐, 상상하기도 싫을만큼 끔찍한 일이다. 일과 중이었으면 사상자가 날 수도 있고, 화재로 연소된 건물이 복구될 때까지 정상적인 학교 일과를 진행하기가 어렵기 때문이다. 특히 우리 학교처럼 오래된 학교는 불에 더욱 취약할 것이다. 여기서 궁금증이 생겼다. 만일 우리 학교에서 불이 나면, 소방차가 빨리 달려올 수 있을까? 가장 가까운 소방서와 우리 학교는 거리가 얼마나 될까? 전국에서 우리 학교는 소방서에서 얼마나 먼 정도에 속할까?

궁금하니까 이제 알아보자.

1. 자료 준비

먼저, 소방서와 119 안전센터에 대해 알아야 한다. 소방서는 우리가 잘 아는 그대로다. 소방차가 있고, 소방관들이 있다. 그런데 119 안전센터는 생소할 수도 있다. 소방서가 시청이라면, 119 안전센터는 주민센터다. 우리가 더 쉽게 볼 수 있는 건 더 수가 많은 119 안전센터다. 필요한 자료가 정해졌다. 전국의 고등학교 현황과 119 안전센터와 소방서의 현황. 두 자료 모두 위도와 경도를 포함해야 두 지점 간의 거리를 계산할 수 있다.

이러한 공공데이터는 '공공데이터포털(data.go.kr)'에 있다. 공공데이터포털에는 '전국초중등학교위치표준데이터'가 있지만, 한국교원대학교에서 제공하는 자료이기 때문에 이 학교에서 관리하는 고등학교만 있다. 우리 광양제철고등학교도 (당연히) 없어서 사용하기 어렵다. 다른 대안을 찾던 중, 학교알리미에서 제공하는 '학교기본정보' API를 발견했다. 위도와 경도가 명확히 표시되어 있어 목적에 부합했다.

119 안전센터의 위치는 공공데이터포털에서 찾을 수 있었다. 소방청이 직접 제공하는 '전국 소방서 및 119 안전센터 정보(2018)' csv 파일을 다운 받았다. 하지만 이 자료에는 도로명 주소만 있을 뿐, 위도와 경도는 없다.

2. 자료 전처리

119 안전센터 목록에 위도와 경도를 추가해야 한다. 그러기 위해서는 도로명 주소로 위도와 경도를 얻을 수 있는 API를 사용해야 한다. 첫 번째로 고려한 것은 행정안전부의 '도로명 주소 API'였지만,

도로명 주소의 코드를 얻는 엔드포인트를 한 번 더 거쳐야 하고, 얻는 정보도직접적인 위도/경도가 아닌 다른 좌표라서 후보에서 제외했다. 두 번째는 카카오 맵 API다. 다행히 중학생 때 어플리케이션 등록을 해놓아서 사용 신청도 거칠 필요 없고, 검색 API 사용도 자유로워서 선택했다. 119 안전센터 현황 파일은 csv인데, 파이썬의 기본 csv 라이브러리를 사용하기보단 1학년 때 읽은 '모두의 데이터 과학 with 파이썬'에서 습득한 판다스를 사용하기로 한다.

```
import requests
import pandas as pd

def get_latitude_longtitude(query: str):
    param = {"query": query}

    res = requests.get("https://dapi.kakao.com/v2/local/search/address.json",
            headers={"Authorization": "KakaoAK [token]"},
            params=param)

    documents = res.json()["documents"]
    if len(documents) <= 0:
        print("documents is 0")
        return (0, 0)

    road_address = documents[0]["road_address"]
    if road_address is None:
        print("road_address is None")
        return (0, 0)

    longtitude = road_address["x"]
    latitude = road_address["y"]

    return (latitude, longtitude)

centers = pd.read_csv("119centers.csv", header=0)
centers = centers.drop('전화번호', axis=1) # 전화번호 행을 삭제한다

latitudes = [None] * len(centers)
longtitudes = [None] * len(centers)

for i, row in centers.iterrows():
    latitudes[i], longtitudes[i] = get_latitude_longtitude(row["주소"])

centers["latitude"] = latitudes
centers["longtitude"] = longtitudes

centers.to_csv("119centers_with_ll.csv", index=False)
```

판다스로 csv 파일을 불러와 각 행의 도로명 주소를 인자로 해서 카카오 맵 API를 호출해 위도/경도를 가져와 새로운 csv 파일로 저장한다. 하지만 API의 결과가 나오지 않는 119 안전센터가 있다.

```
609,충북소방본부,충주소방서,주덕,충북 충주시 주덕읍 산양로 35,0,0
611,충북소방본부,충주소방서,연수,충북 충주시 연수동 454-30,0,0
627,충북소방본부,괴산소방서,청천,충북 괴산군 청청면 선평길 78-9,0,0
636,충남소방본부,천안동남소방서,독립,충남 천안시 동남구 목천읍 삼방로 95,0,0
```

```
638,충남소방본부,천안동남소방서,풍세,2018년 개서 예정,0,0
646,충남소방본부,천안서북소방서,직산,2018년 개서 예정,0,0
647,충남소방본부,천안서북소방서,불당,2018년 개서 예정,0,0
```

'2018년 개서 예정(위에도 적었듯이 이 데이터는 2018년 기준이다)'이여서 위치가 나오지 않는 곳과, 어떤 곳은 주소에 오타가, 어떤 곳은 잘못된 주소가, 등등 문제가 있는 데이터가 있어서 일단 위도/경도를 0으로 삽입했다. 이 행들은 뒤에 가서 없애기로 하자.

3. 거리 구하기

수학 시간에 좌표평면 위의 두 점이 주어졌을 때 두 점간의 거리를 구하는 방법을 배웠다. x좌표끼리 y좌표끼리 뺀 값을 각각 제곱해 더해서 제곱근을 취하면 된다. 다만, 지구는 둥글고, 위도와 경도를 빼어 계산한 값의 단위는 무엇인지의 문제가 있었다. 추가 자료를 찾아 구현하는 것보단 haversine이라는 라이브러리를 이용하기로 했다.

학교 위치와 119 안전센터 위치 파일을 각각 불러온다.

```
# load schools
with open("schoolList_2020_04_hig.json", "r") as f:
    school_json = json.load(f)

schools = pd.DataFrame(school_json["list"])
schools = schools[['SCHUL_RDNMA', 'SCHUL_NM', 'LTTUD', 'LGTUD']]
schools["LTTUD"] = pd.to_numeric(schools["LTTUD"], downcast="float")
schools["LGTUD"] = pd.to_numeric(schools["LGTUD"], downcast="float")

# load 119 centers
centers = pd.read_csv("119centers_with_ll.csv")
centers["latitude"] = pd.to_numeric(centers["latitude"], downcast="float")
centers["longtitude"] = pd.to_numeric(centers["longtitude"], downcast="float")
centers = centers.loc[centers["latitude"] != 0]
```

학교알리미에서 받은 파일은 json이고 'list'로 감싸져 있어 한 겹을 풀어내느라 json 라이브러리를 사용했다. 위도와 경도는 소수점 이후로 숫자가 많아서 판다스는 문자열로 인식한다. 그래서 pd.to_numeric으로 타입 캐스팅을 float로 해주어야 한다. 마지막 줄은 위도가 0인 119 안전센터를 목록에서 제거한다.

```
def line_distance(school, center):
    school_location = (school["LGTUD"], school["LTTUD"])
    center_location = (center["longtitude"], center["latitude"])
    return haversine(school_location, center_location)
```

거리를 구하는 함수를 정의한다. haversine의 wrapper다.

```
least_diss = [None] * len(schools)
least_centers = [None] * len(schools)
i = 0
for _, school in schools.iterrows():
    least_dis = 100000
    least_center = ""
    for j, center in centers.iterrows():
        dis = line_distance(school, center)
        if dis < least_dis:
            least_dis = dis
            least_center = center

    least_diss[i] = least_dis
    least_centers[i] = least_center['119안전센터명']

    i += 1
```

가장 짧은 거리(킬로미터)와 가장 가까운 119 안전센터를 저장할 리스트를 만들고, 방금 정의한 함수를 이용해 119 안전센터와의 거리를 구한다. 이전의 거리보다 가까우면 그 거리와 센터를 변수에 저장하고 계속 나아간다.

```
schools["least_dis"] = least_diss
schools["least_center"] = least_centers

schools.to_csv("schools_w_least.csv", index=False)
```

각각 가장 가까운 거리와 센터를 학교 정보에 추가한다음 새 csv 파일로 저장한다.

4. 결과 분석

손쉬운 분석을 위해서 저장된 csv 파일을 엑셀로 불러온다. utf-8으로 불러와야 한다. 가장 가까운 센터와의 거리를 나타내는 least_dis(킬로미터)로 오름차순으로 정렬한다.

SCHUL_NM	LTTUD	LGTUD	least_dis(km)	least_center
달성고등학교	35.858932	128.55534	0.035913966	내당
인월고등학교	35.46496	127.60272	0.046928828	인월
영덕고등학교	37.252453	127.068504	0.063022624	영통
영북고등학교	38.08626	127.276	0.085336207	영북
한양공업고등학교	37.565872	127.011444	0.086188769	을지로

최소 거리 상위 5개 목록이다. 내 자신이 짠 코드기 때문에 믿기 어려운 데이터다. 각각 길이가 맞는지, 실제로 가까운지 확인해보자.

달성고등학교와 내당119안전센터의 모습이다. 그냥 보아도 정말 가깝다는 걸 알 수 있는데, 거리도 31m 뿐이다. 자료의 0.035km(=35m)와 거의 비슷한 수치다. 인월고등학교 직선거리는 달성고등학교보다는 길지만, 사이에 도로가 없다. 46m로 정말 가까운 거리다.
우리 광양제철고등학교는 몇 순위일까?

순번	SCHUL_NM	LTTUD	LGTUD	least_dis	least_center
760	광양제철고등학교	34.956808	127.72849	0.78154633	금호

전국 2365개교 중 760등이다. 이런 거에도 9등급제를 적용해보자면, 32%로 4등급이다. 가장 가까운 119안전센터는 금호119안전센터고, 직선거리로 781m, 차량 이동거리는 6분이다. '소방차 및 효율적인 골든타임 확보 방안에 관한 연구'에 따르면 국내에서는 소방차의 출발 시간에서 도착 시간까지의 골든 타임을 5분 이내로 정의하고 있다. 이 정의에 따르면 6분이라는 거리는 좋은 시간은 아니다. 우리 학교에서 불이 났을 때, 빠른 화재 진압은 기대하기 어렵다.

소방서로부터 가장 가까운 고등학교는? 20806 범준환

5. 느낀 점

주제를 선발하고, 자료를 가져오고, 위도/경도를 채우는 과정과 직선 거리를 계산하는 과정까지 전부 '자료'를 중심으로 한 활동이었다. 특히 csv와 파이썬 자료처리 라이브러리인 pandas를 이용해 자료를 다루는 경험은 처음이라 새로운 도구를 배우는 경험이 될 수 있었다. 게다가 우리 학교가 119안전센터와(주변 환경상 그래보이긴 하지만) 소방 골든타임을 넘길만큼 거리가 멀다는 점은 예상하긴 했지만 살짝 충격이었다.

아쉬웠던 점은 첫째, 자료에 대한 아쉬움이다. 119안전센터의 주소가 오타가 있거나 잘못 되어있어 아쉽게 해당 센터를 제거해야했고, 2018년 자료라 2021년 자료인 학교 현황과 다소 맞지 않을 것이다. 소방청 공식 자료임에도 이런 문제점이 있다는 게 너무나 아쉬웠다.

둘째는 자신이 능숙하지 않음의 문제점이다. 위도/경도는 소수점 뒤 숫자가 무려 6자리를 넘는다. 따라서 파이썬과 pandas가 자료형을 잘못 인식하는 등의 오류로 위도와 경도가 350321.3과 1282145.5 등으로 인식되어 최단 거리가 18,415km로 나온 학교도 있었다.

앞으로 추가적인 데이터 분석 활동을 할 때 더욱 정확한 자료를 찾고 문제 있는 자료를 보정하는 활동을 자료 전처리 부분에 추가해야겠다는 생각과 데이터 과학의 이론적인 부분을 더 공부해야겠다는 생각을 했다.

참고자료

- 공공데이터포털: 소방청_전국 소방서 및 119안전센터 정보(2018.11.28) https://www.data.go.kr/data/15048242/fileData.do
- 학교알리미: 학교기본정보 https://www.schoolinfo.go.kr/ng/go/pnnggo_a01_l0.do
- 모두의 데이터 과학 with 파이썬 (드미트리 지노비에프 저)
- 카카오 디벨로퍼스: 로컬 API https://developers.kakao.com/docs/latest/ko/local/dev-guide#address-coord
- 황의홍, 최지훈, 최돈묵. "소방차 출동 시 효율적인 골든타임 확보 방안에 관한 연구." 한국방재학회 연구집. Vol. 18, No. 5 (Aug. 2018), pp.119~126