빅데이터 기말 과제

201844078 박상렬

프로젝트 요약

주제 : 전국 전기차 충전소 현황 데이터 가공 및 출력

주요 내용 : 시군구 검색 시 해당 지역의 충전소 현황 출력 (Folium Marker 활용)
해당 지역 충전소 분포도 출력 (Folium Choropleth 활용)
해당 지역 상세 지역별 충전소 분포도 그래프 출력 (Seaborn Countplot 활용)

사전 작업 - 전체 데이터 저장

전국 충전소 분포도를 구현하기 위해

전국 충전소 자료를 csv에 저장

(총 40만건이 넘는 자료를 매번 불러오기에 애로사항이 있으므로)

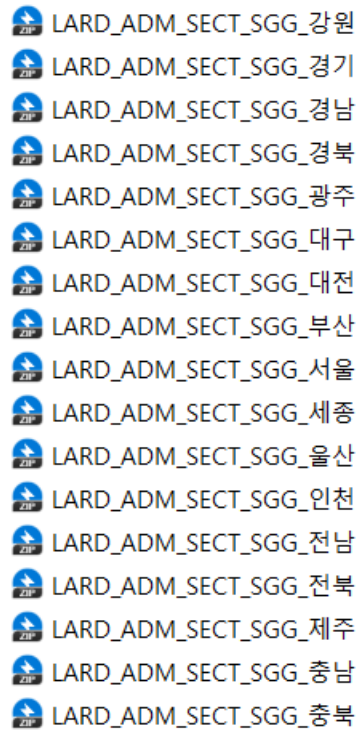
```
def getTotalData():
  total_data = []
  area_array_temp = df_area.drop_duplicates(['zcode'])
  area_array = list(map(str, area_array_temp['zcode'].to_numpy()))
  for z in area_array:
    service_url = 'https://apis.data.go.kr/B552584/EvCharger/getChargerInfo'
    parameters = '?serviceKey=' + ServiceKey   #인증키
    parameters += '&pageNo=1&numOfRows=9999&zcode=' + z
    detail_area_array_temp = df_area[df_area['zcode'] == int(z)]
    detail_area_array = list(map(str, detail_area_array_temp['zscode'].to_numpy()))
    print(detail_area_array)
    for zs in detail_area_array:
      parameters += '&zscode=' + zs
      url = service_url + parameters
      response = urllib.request.urlopen(url)
      results = response.read().decode('utf-8')
      results_to_json = xmltodict.parse(results)
      data = json.loads(json.dumps(results_to_json))
      total_data.extend(data['response']['body']['items']['item'])
      parameters = parameters[:-13]

  df_total = pd.DataFrame(total_data)
  df_total.to_csv('/content/total_data.csv')
```

사전 작업 - GeoJSON 파일 생성 (1)

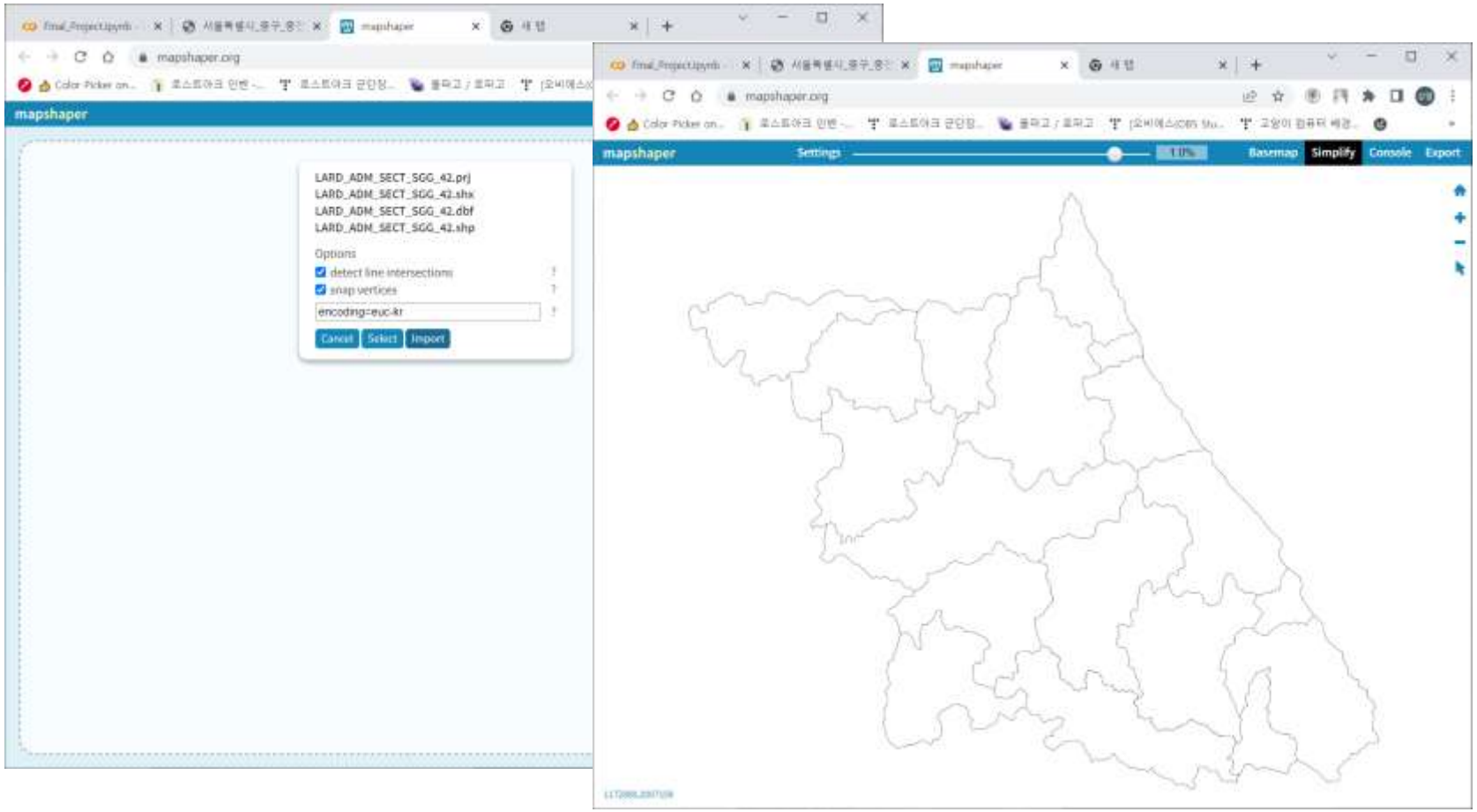
전국 충전소 분포도를 구현하기 위해 각 행정구역 GeoJSON 파일 생성

1. 국가공간정보포털 오픈마켓(http://data.nsdi.go.kr/dataset/15144)
   에서 각 지역별 SHF 다운로드



사전 작업 - GeoJSON 파일 생성 (2)

2. MapShaper(https://mapshaper.org)에서 JSON으로 변환



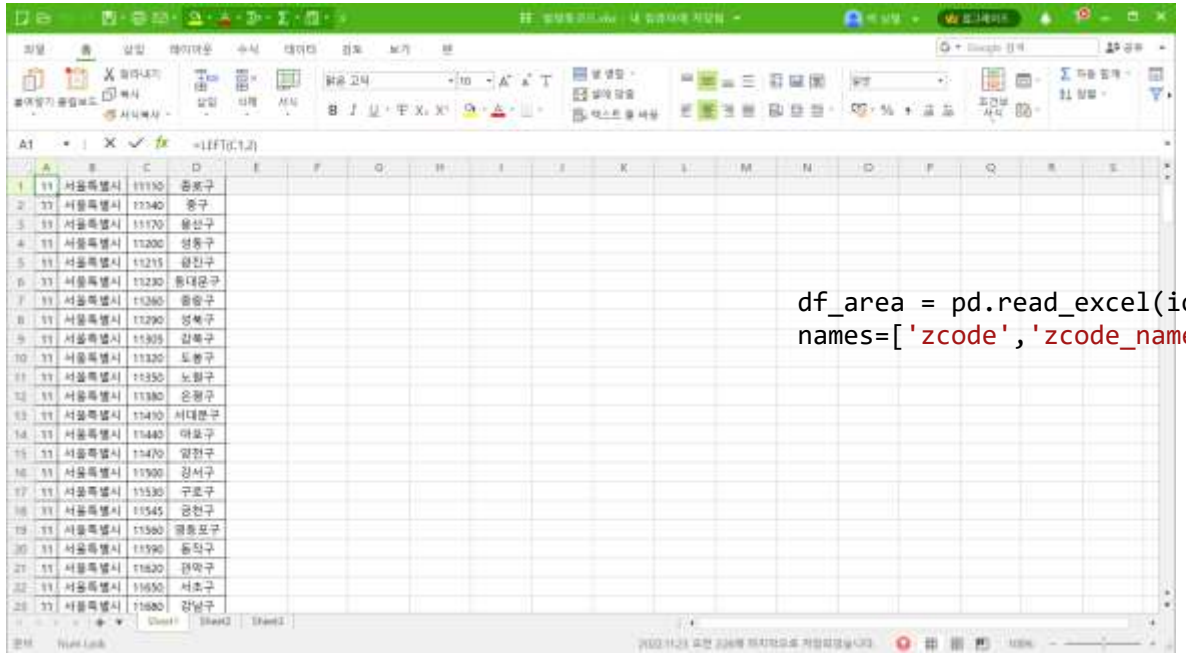
- 강원도.json
- 경기도.json
- 경상남도.json
- 경상북도.json
- 광주광역시.json
- 대구광역시.json
- 대전광역시.json
- 부산광역시.json
- 서울특별시.json
- 세종시.json
- 울산광역시.json
- 인천광역시.json
- 전라남도.json
- 전라북도.json
- 제주도.json
- 충청남도.json
- 충청북도.json

사전 작업 - 법정동 코드 배열

Python에서 일일히 전국 코드를 배열로 만들긴 어려움이 있어

인터넷에 정리된 표를 복사하여 엑셀 파일로 저장 후 불러와서

데이터 프레임 생성



```
df_area = pd.read_excel(io='/content/drive/MyDrive/Colab Notebooks/법정동코드.xlsx', header=None,
names=['zcode','zcode_name','zscode','zscode_name'])
```

희망 지역 검색

```
지역 목록 : 서울특별시 부산광역시 대구광역시 인천광역시 광주광역시 대전광역시 울산광역시 세종시 경기도 강원도 충청북도 충청남도 전라북도 전라남도 경상북도 경상남도 제주도
지역 :
    지역 목록 : 서울특별시 부산광역시 대구광역시 인천광역시 광주광역시 대전광역시 울산광역시 세종시 경기도 강원도 충청북도 충청남도 전라북도 전라남도 경상북도 경상남도 제주도 지역 : 서울특별시
    서울특별시 상세 지역 목록 : 종로구 중구 용산구 성동구 광진구 동대문구 중랑구 성북구 강북구 도봉구 노원구 은평구 서대문구 마포구 양천구 강서구 구로구 금천구 영등포구 동작구 관악구 서초구 강남구 송파구 강동구
    상세 지역 :
```

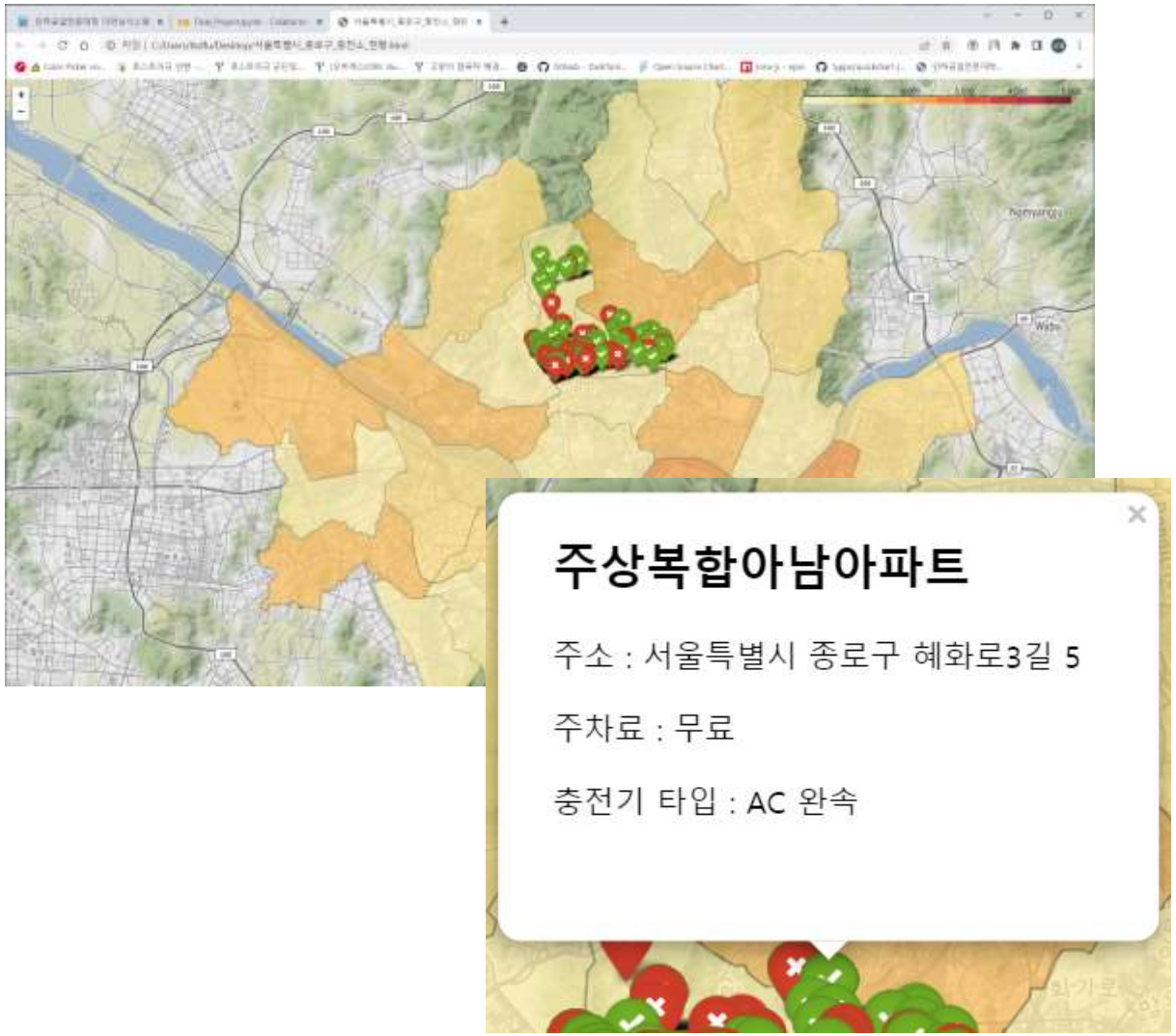
```python
# 지역 코드
  area_array_temp = df_area.drop_duplicates(['zcode'])
  area_array = area_array_temp['zcode_name'].to_numpy()
  print('지역 목록 :', end=' ')
  for a in area_array:
    print(a, end=' ')
  zcode_temp = input('지역 : ')
  if zcode_temp not in area_array:
    print('올바른 지역이 아닙니다.')
    return
  zcode = str(area_array_temp[area_array_temp['zcode_name'] == zcode_temp].i
at[0,0])
  if zcode_temp == '세종시':
    zscode = '36110'
    zscode_temp = ''
  else:
    # 상세 지역 코드
    detail_area_array_temp = df_area[df_area['zcode_name'] == zcode_temp]
    detail_area_array = detail_area_array_temp['zscode_name'].to_numpy()

    print(zcode_temp + ' 상세 지역 목록 :', end=' ')
    for a in detail_area_array:
      print(a, end=' ')
    zscode_temp = input('상세 지역 : ')
    if zscode_temp not in detail_area_array:
      print('올바른 상세 지역이 아닙니다.')
      return
    zscode = str(detail_area_array_temp[detail_area_array_temp['zscode_name'
] == zscode_temp].iat[0,2])
```

```python
def getData(zcode, zscode):
  total_data = []
  service_url = 'https://apis.data.go.kr/B552584/EvCharger/getChargerInfo'
  parameters = '?serviceKey=' + ServiceKey   #인증키
  parameters += '&pageNo=1&numOfRows=9999&zcode=' + zcode
  parameters += '&zscode=' + zscode
  url = service_url + parameters
  response = urllib.request.urlopen(url)
  results = response.read().decode('utf-8')
  results_to_json = xmltodict.parse(results)
  data = json.loads(json.dumps(results_to_json))
  total_data.extend(data['response']['body']['items']['item'])
  df_total_temp = pd.DataFrame(total_data)
  df_total = df_total_temp
  return df_total
```

검색한 지역의 데이터를 불러와 DataFrame에 저장함

특정 지역 충전소 현황 지도 (1)



검색을 희망하는 지역과 상세 지역을 입력하면 결과 지도를 html 파일로 생성

사용 가능한 충전소는 초록색 마커, 불가능한 충전소는 붉은색 마커로 표시

마커 클릭 시 주소와 주차료 여부 표시

해당 지역의 분포도도 확인할 수 있음

특정 지역 충전소 현황 지도 (2)

마커 생성 코드

```
def marker(data, color, icon, map):
  for name, addr, lat, lng, free, chtype in zip(data.statNm, d
ata.addr, data.lat, data.lng, data.parkingFree, data.chgerType
):
    free_val = ''
    if free == 'Y':
      free_val = '무료'
    else:
      free_val = '유료'
    chtype = int(chtype)
    ch_array = ['DC 차데모', 'AC 완속', 'DC 차데모 + AC3 상', 'D
C 콤보', 'DC 차데모 + DC 콤보', 'DC 차데모 + AC3 상 + DC 콤보',
'AC3 상', 'H2']
    charger_type = ch_array[chtype - 1]
    html = f'''
            <h2>{name}</h2>
            <p>주소 : {addr}</p>
            <p>주차료 : {free_val}</p>
            <p>충전기 타입 : {charger_type}</p>
            '''
    iframe = folium.IFrame(html, width=300, height=200)
    popup = folium.Popup(iframe, max_width=400)
    folium.Marker([lat, lng], popup=popup, tooltip=name, icon=
folium.Icon(color, icon=icon, prefix='fa')).add_to(map)
```

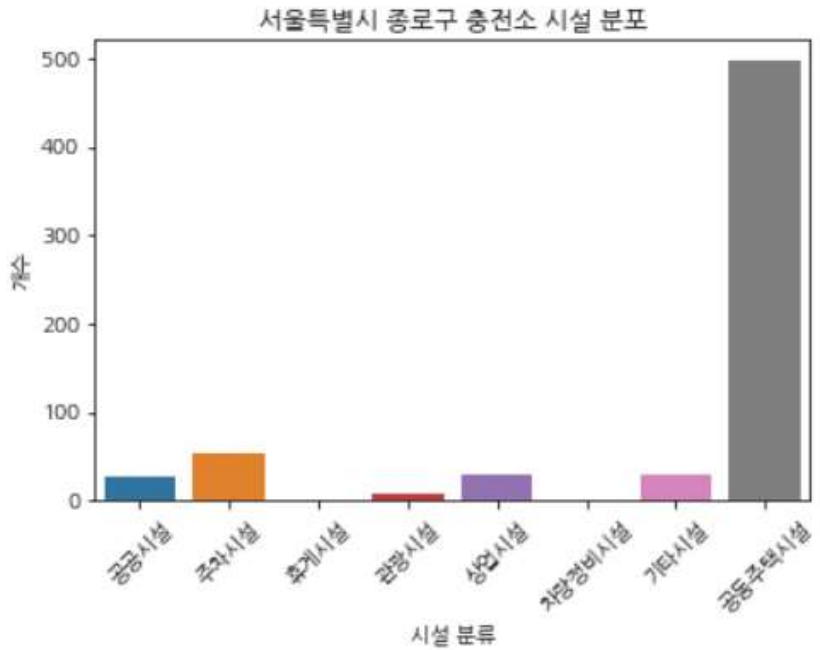
지도 생성 코드

```
smap = folium.Map(location=[37.4493812, 126.6573386], zoom_sta
rt=10, tiles='Stamen Terrain')
  geo='/content/drive/MyDrive/Colab Notebooks/GeoJSON/' + zcod
e_temp + '.json'
  total_data_temp = pd.read_csv('/content/drive/MyDrive/Colab
Notebooks/total_data.csv', low_memory=False)
  total_data = total_data_temp[total_data_temp['zcode'] == int
(zcode)].groupby('zscode', as_index=False).count()
  total_data['zscode'] = total_data['zscode'].astype(str)
  marker(data_avail, 'green', 'check', smap)
  marker(data_using, 'red', 'times', smap)
  folium.Choropleth(geo_data=geo, data=total_data, columns=['z
scode','lat'], fill_color='YlOrRd', fill_opacity=0.7, line_opa
city=0.3,
                    threshold_scale=[0, 500, 1000, 1500, 2000,
2500, 3000, 3500, 4000, 4500, 5000], key_on='feature.propertie
s.COL_ADM_SE').add_to(smap)

  smap.save('/content/' + zcode_temp + '_' + zscode_temp + '_
충전소_현황.html')
```

특정 지역 충전소 시설 분포 그래프

검색을 희망하는 지역과 상세 지역을 입력하면 해당 지역의 충전소가 어느 시설에 많이 분포되어 있는지 확인할 수 있음




```
ax = sns.countplot(data=data, x='kind', order=['A0','B0','C0',
'D0','E0','F0','G0','H0'])
  ax.set_title('{0} {1} 충전소 시설 분포'.format(zcode_temp, zs
code_temp))
  ax.set_xlabel('시설 분류')
  ax.set_ylabel('개수')
  ax.set_xticklabels(labels=['공공시설', '주차시설', '휴게시설',
'관광시설', '상업시설', '차량정비시설', '기타시설', '공동주택시설'
], rotation=45)
```