머신러닝 데이터 수집과 분석 시각화

KDT 5기
심민혜

[목차]

---

1. 시카고 샌드위치 맛집 분석

1.1 분석 목적

- 시카고 매거진 홈페이지에 접속해서 샌드위치 가게 정보를 수집 후 지도에 표현한다.
- OOP형식으로 프로그래밍한다.

★인터넷에서 웹 페이지의 내용을 가져오기(크롤링) 위해 BeautifulSoup이라는 라이브러리를 사용한다.

> *BeautifulSoup*는 *HTML*과 *XML* 파일에서
> 데이터를 읽어내는 파이썬 라이브러리이다

2. 프로그래밍

2.2 시카고 샌드위치 맛집 소개 사이트에 접근하기

필요한 라이브러리를 설치한다.

```
from bs4 import BeautifulSoup
from urllib.request import urlopen, Request
from urllib.parse import urljoin
import re
import pandas as pd
from tqdm import tqdm
import folium
import pandas as pd
import googlemaps
import numpy as np
```

클래스를 생성한 후 __init__에 웹 크롤링할 사이트를 기재한다.

```
class Chicago:

    def __init__(self) -> None:
        self.url_base = 'https://www.chicagomag.com/'
        self.url_sub = '/Chicago-Magazine/November-2012/Best-Sandwiches-Chicago/'
        self.url = self.url_base + self.url_sub
        response = Request(self.url, headers={"User-Agent": "Mozilla/5.0"})
        html = urlopen(response)
        self.soup = BeautifulSoup(html, "html.parser")
```

2.2 접근한 웹페이지에서 원하는 데이터 추출하고 정리하기

★__init__에서 선언한 변수를 사용하기 위해서는 self를 이용한다.

랭킹 정보 얻기 (1위)

```
    def data_extract(self):
        soup = self.soup
        tmp_one = soup.find_all('div', 'sammy')[0]
        tmp_one.find(class_='sammyRank').get_text()
```

정규식을 사용하여 메뉴이름과 가게이름 분리

```
        tmp_string = tmp_one.find(class_='sammyListing').get_text()
        re.split(('\n|\r\n'),tmp_string)
```

순위, 메인 메뉴, 카페 이름, url 주소 저장

```
        rank = []
        main_menu = []
        cafe_name = []
        url_add = []

        list_soup = soup.find_all('div', 'sammy')

        for item in list_soup:
            rank.append(item.find(class_='sammyRank').get_text())
            tmp_string = item.find(class_='sammyListing').get_text()
            main_menu.append(re.split(('\n|\r\n'), tmp_string)[0])
            cafe_name.append(re.split(('\n|\r\n'), tmp_string)[1])
            url_add.append(urljoin(self.url_base, item.find('a')['href']))
```

데이터 프레임 만들기 (데이터 정리)

```
        data = {'Rank': rank, 'Menu': main_menu, 'Cafe': cafe_name, 'URL': url_add}
        self.df = pd.DataFrame(data, columns=['Rank', 'Cafe', 'Menu', 'URL'])
        self.df.to_csv("./data/best_sandwiches_list_chicago.csv", sep=',', encoding='UTF-8')
```

★컬럼 순서 정리

import pandas as pd

df = pd.DataFrame(data,
columns=['Rank','Cafe','Menu','URL'])

★데이터 프레임을 csv로 저장하는 라이브러리와 코드

import pandas as pd

df = pd.to_csv('저장장소/파일명',
sep=',',encoding='UTF-8')

>> print 결과

	Rank	Menu	Cafe	URL
0	1	BLT	Old Oak Tap	https://www.chicagomag.com/Chicago-Magazine/No...
1	2	Fried Bologna	Au Cheval	https://www.chicagomag.com/Chicago-Magazine/No...
2	3	Woodland Mushroom	Xoco	https://www.chicagomag.com/Chicago-Magazine/No...
3	4	Roast Beef	Al's Deli	https://www.chicagomag.com/Chicago-Magazine/No...
4	5	PB&L	Publican Quality Meats	https://www.chicagomag.com/Chicago-Magazine/No...

2.3 다수의 웹 페이지에 자동으로 접근해서 원하는 정보 가져오기

3페이지에서 가격과 주소 가져오기

```
    def get_information(self):
        df = self.df
        price = []
        address = []

        for n in df.index[:3]:

            response = Request(df['URL'][0], headers={"User-Agent": "Mozilla/5.0"})
            html = urlopen(response)
            self.soup_tmp = BeautifulSoup(html, 'lxml')
            gettings = self.soup_tmp.find('p', 'addy').get_text()

            price.append(gettings.split()[0][:-1])
            address.append(' '.join(gettings.split()[1:-2]))
```

2.4 50개 웹 페이지에 대한 정보 가져오기

tqdm을 적용하여 50페이지 핸들링

```
    def page_handling(self):
        df = self.df
        price = []
        address = []

        for n in tqdm(df.index):

            response = Request(df['URL'][0], headers={"User-Agent": "Mozilla/5.0"})
            html = urlopen(response)
            self.soup_tmp= BeautifulSoup(html, 'lxml')
            gettings = self.soup_tmp.find('p', 'addy').get_text()

            price.append(gettings.split()[0][:-1])
            address.append(' '.join(gettings.split()[1:-2]))
```

★작업 진행률 표시

from tqdm import tqdm

>> print 결과

```
43%|                                                  | 257/599 [00:27<00:36,  9.35it/s]
```

df에 가격과 주소 추가

```
        df['Price'] = price
        df['Address'] = address
        df = df.loc[:, ['Rank', 'Cafe', 'Menu', 'Price', 'Address']]
        df.set_index('Rank', inplace=True)
```

★데이터프레임에 열 추가

df = df.loc[:, ['추가할 열1', '추가할 열2']

결과 저장

```
        df.to_csv("./data/best_sandwiches_list_chicago2.csv", sep=',', encoding='UTF-8')
```

2.5 맛집 위치를 지도에 표시하기

데이터 불러오기

```
    def gamps(self):
        df = pd.read_csv("./data/best_sandwiches_list_chicago2.csv", index_col=0)
```

googlemaps 읽어오기

```
        gmaps_key = '******************'
        gmaps = googlemaps.Client(key=gmaps_key)
```

50개 맛집의 위도, 경도 정보 받아오기

```
        lat = []
        lng = []

        for n in tqdm(df.index):

            if df['Address'][n] != 'Multiple':
                target_name = df['Address'][n]+', '+'Cicago'
                gmaps_output = gmaps.geocode(target_name)
                location_output = gmaps_output[0].get('geometry')
                lat.append(location_output['location']['lat'])
                lng.append(location_output['location']['lng'])

            else:
                lat.append(np.nan)
                lng.append(np.nan)
```

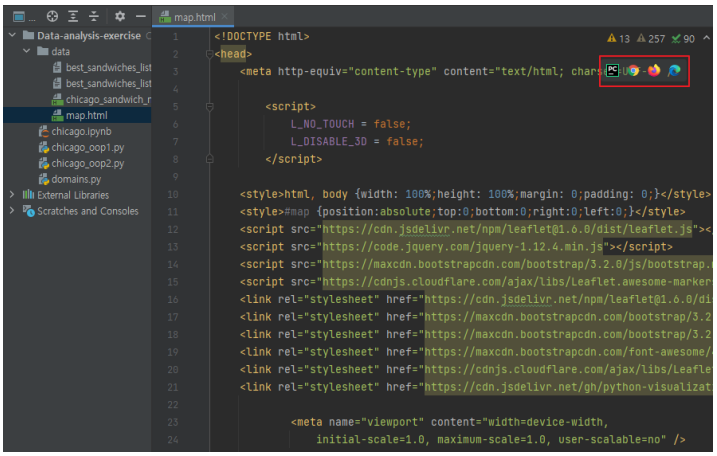
위도,경도 컬럼 추가

```
        df['lat'] = lat
        df['lng'] = lng
```

50개 맛집의 위도, 경도를 지도에 표시

```
mapping = folium.Map(location=[df['lat'].mean(), df['lng'].mean()], zoom_start=11)
for n in df.index:
    if df['Address'][n] != 'Multiple':
        folium.Marker([df['lat'][n], df['lng'][n]], popup=df['Cafe'][n]).add_to(mapping)
mapping.save('./data/map.html')
```

지도 열기



3. 결과