날씨 데이터를 활용한 태양광 발전량 예측

5강. 데이터 전처리

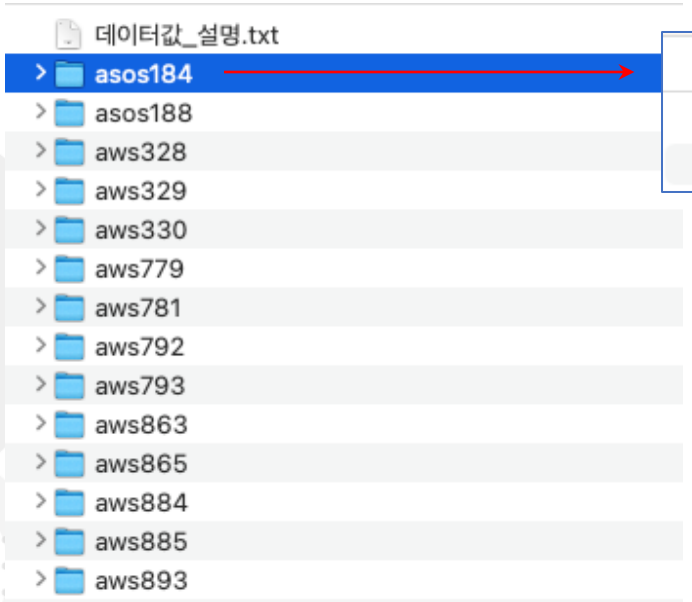
1. 수집된 데이터의 시간 간격 동일하게 맞추기
2. 위성 데이터의 nc파일을 csv파일로 변환하기
3. 위성 이미지 좌표를 위도 경도로 변환하여 DSR값 가져오기
4. 주소별로 정리된 csv파일
5. UTC시간을 서울 시간으로 변경, 시간 데이터 타입 변경
6. 하향 단파 복사 관련 데이터만 골라서 저장
7. 대체 위도와 경도
8. 세 개의 csv파일 합치기

1 실습용 데이터파일 소개

실습을 위해 제공되는 파일들을 먼저 살펴보겠습니다. 크게 위성 데이터와 지상 데이터 폴더가 있습니다.

지상 데이터(ASOS, AWS)

폴더명 ground_weather_data



일시	1분 강수량(mm)
2020.1.1 0:01	0
2020.1.1 0:02	0
2020.1.1 0:03	0
2020.1.1 0:04	0
2020.1.1 0:05	0
2020.1.1 0:06	0
2020.1.1 0:07	0
2020.1.1 0:08	0

기준시간	기온	풍향	풍속	습도
2020.1.1 0:10	2.41	288.97	2.55	62.44
2020.1.1 0:20	2.53	204.75	2.05	61.8
2020.1.1 0:30	2.52	280.6	2.08	61.21
2020.1.1 0:40	2.59	300.17	2.39	61.99
2020.1.1 0:50	2.65	170.1	2.47	63.21
2020.1.1 1:00	2.62	307.55	2.16	63.3
2020.1.1 1:10	2.62	265.3	1.7	62.55

1 실습용 데이터파일 소개

기상청 Korea Meteorological Administration

위성 데이터(단파복사)

폴더명 sat_data

위도	경도	동이름	유형	지점번호	가까운 지점	대체위도	대체경도
33.2218157	126.252995	대정읍	aws	793	대정	33.241	126.2263
33.24016	126.3806	하예동	aws	328	중문	33.2494	126.406
33.24695	126.48927	강정동	aws	884	기상(과)	33.2593	126.5176
33.24679	126.56396	서귀동	aws	884	기상(과)	33.2593	126.5176
33.26177	126.38678	상예동	aws	328	중문	33.2494	126.406
33.2640931	126.412	색달동	aws	328	중문	33.2494	126.406
33.262002	126.428862	중문동	aws	328	중문	33.2494	126.406

jeju_geo_match_info.csv

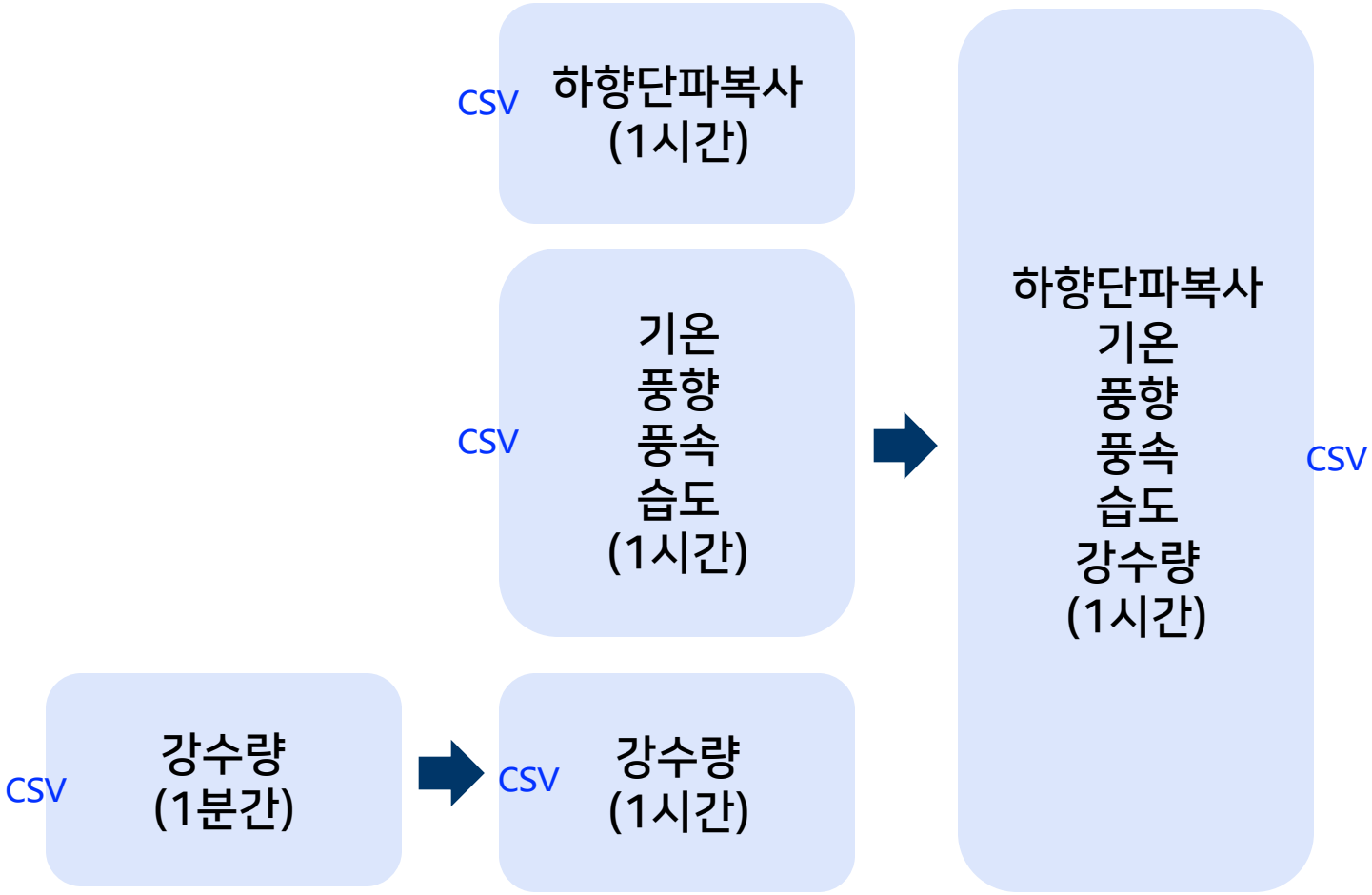
sat_data폴더의 지역별 단파복사 파일들은 위의 csv 파일을 참조하여 생성된 파일입니다.

date		dim_y	dim_x	ASR	ASR_DQF1	DSR	DSR_DQF1	RSR	RSR_DQF1	SW_DQF	gk2a_imager_proj
2020-03-01 0:00	636771	707	471	122.4000015	1	141.6000061	1	312.2000122	1	1	
2020-03-01 1:00	636771	707	471	183.3000031	1	212.1000061	1	435.8000183	1	1	
2020-03-01 2:00	636771	707	471	128.1000061	1	148.1999969	1	632	1	1	
2020-03-01 3:00	636771	707	471	224.4000092	1	259.7000122	1	612.9000244	1	1	
2020-03-01 4:00	636771	707	471	209.6000061	1	242.5	1	646.2000122	1	1	
2020-03-01 5:00	636771	707	471	210.6000061	1	243.6999969	1	598.5	1	1	

2 수집된 데이터의 시간 간격 동일하게 맞추기

기상청 Korea Meteorological Administration

여러 종류의 데이터를 수집했는데 가장 먼저 살펴볼 것 중에 하나는 데이터의 시간 간격입니다.



2 수집된 데이터의 시간 간격 동일하게 맞추기

기상청 Korea Meteorological Administration

```
import re

for dirName in dirNames:
    csv_list = glob(os.path.join(data_dir,dirName,'*.csv'))
    for csvFile in csv_list:
        # 파일명에 rain을 포함하는 파일을 찾습니다.
        count = csvFile.find('rain')
        if count>0:
            fileName = os.path.basename(csvFile)
            print(csvFile)

            # 파일 읽기
            df = pd.read_csv(csvFile, encoding='cp949')
            fileFront = fileName.split('_')[0]
            filterStr = re.sub(r'[0-9]', '',fileFront)
            if filterStr=='aws':
                # 지점 컬럼 삭제(불필요)
                df.drop(columns='지점', inplace=True)

            # 결측치를 0으로 채워준다.
            df.fillna(0, inplace=True)

            # 한 시간단위로 강수량을 수정한다.
            time_list, hour_rain = rain_sum(df)

            # pandas DataFrame 생성
            value = {'일시':time_list, '강수량':hour_rain}

            # 시간당 강수량을 저장하는 데이터프레임 생성
            df2 = pd.DataFrame(value)

            # 데이터형을 시간 타입으로 변경
            df2['일시'] = pd.to_datetime(df2['일시'])

            # 데이터 수집 기간을 2020-3-1 0:00 ~ 2021-1-1 0:00
            df2 = df2[df2['일시']>'2020-3-1 8:00']

            df2.set_index(keys='일시', inplace=True, drop=True)
            df2.to_csv(save_dir+fileName, mode='w', encoding='utf-8-sig')
```

Glob함수는 확장자가 특정 파일만 골라서 리스트를 생성할 때 유용하게 사용됩니다. 여기서는 csv만 골라냅니다. 그리고 Aws나 asos폴더 안에는 2개의 파일이 있습니다.

강수량 데이터는 파일명에 rain이라는 문자열을 포함하고 있습니다. ‘rain’문자열을 포함하는 파일만 아래 코드를 실행합니다.

AWS는 ASOS보다 지점 컬럼이 더 존재

한 시간 단위로 강수량을 누적

Datetime으로 타입을 변경하고 날짜시간 조건으로 필터링합니다. 문자열로는 이 같은 조건을 사용하지 못함.

전처리를 잘 하시려면 파이썬이나 Pandas를 많이 사용해보시는 것을 권장드립니다. 데이터 사이언스에서 전처리는 전체 프로세스 중에 많은 시간을 들이는 비중이 높은 작업 입니다.

일시	1분 강수량(mm)
2020.1.1 0:01	0
2020.1.1 0:02	0
2020.1.1 0:03	0
2020.1.1 0:04	0
2020.1.1 0:05	0
2020.1.1 0:06	0
2020.1.1 0:07	0
2020.1.1 0:08	0

ASOS 1분당 강수량

지점	일시	1분 강수량(mm)
328	2020.1.1 0:01	0
328	2020.1.1 0:02	0
328	2020.1.1 0:03	0
328	2020.1.1 0:04	0
328	2020.1.1 0:05	0
328	2020.1.1 0:06	0
328	2020.1.1 0:07	0
328	2020.1.1 0:08	0
328	2020.1.1 0:09	0

AWS 1분당 강수량

3 위성 데이터의 nc파일을 csv파일로 변환하기

기상청 Korea Meteorological Administration

NC 파일에서 하향 단파 복사(일사량)를 활용하려면 csv파일로 변환을 해야 합니다.
변환을 위해 필요한 파이썬 패키지를 추가로 설치해야 합니다.

netcdf2csv, xarray, scipy

```
from netcdf2csv import convert_dir
import xarray as xr
import scipy

netcdf_dir ='./nc'            # nc 파일이 저장된 경로
csv_dir = './csv'             # csv 파일이 저장될 경로
clean_csv_dir='./csv/clean'   # clean_csv 파일이 저장될 경로

if not os.path.exists(csv_dir):
    os.makedirs(directory)

if not os.path.exists(clean_csv_dir):
    os.makedirs(directory)

# 1st : nc파일의 경로,  2nd : csv 파일이 저장될 경로,
# 3rd : clean_csv 파일이 저장될 경로, 4th : clean_csv파일을 생성할지 여부
convert_dir(netcdf_dir,csv_dir,cleaned_csv_dir=clean_csv_dir,clean_choice=1)
```

convert_file( ) 함수는
파일 하나만 변환해줍니다.

4 위성 이미지 좌표를 위도 경도로 변환하여 DSR값 가져오기

기상청 Korea Meteorological Administration

변환된 CSV파일을 열어서 확인해 봅니다.

* 본 파일은 2022년 3월 10일 10시 00분의 NC파일을 CSV로 변환 한 예입니다.

A	B	C	D
ydim	xdim	DSR	DQF_DSR
0	0	0.32304767	0
0	1	0.3250509	0
0	2	0.33303753	0
0	3	0.3223129	0
0	4	0.30429038	0
0	5	0.30381992	0
0	6	0.31197688	0
0	7	0.3082395	0
0	8	0.30686793	0
0	9	0.30975503	0
0	10	0.3164073	0
0	11	0.32586318	0
0	12	0.33321843	0
0	13	0.32638738	0

변환된 파일을 살펴보면 ydim과 xdim은 위성 이미지 상의 x,y 좌표 값이고, DSR은 하향 단파 복사(일사량) 데이터 값입니다. 여기서 우리가 알고자 하는 위치는 위도와 경도입니다. 위성 이미지 상의 x, y값을 위도와 경로도 바꿔주는 변환 테이블인 Coordinate_transform.csv파일을 활용합니다.

위도	경도	위성 이미지 좌표		
lat	long		dim_y	dim_x
38.01786	124.58534	403589	448	389
37.99936	124.58575	404489	449	389
37.98087	124.58617	405389	450	389
37.96238	124.58658	406289	451	389

coordinate_transform.csv 파일의 일부

5 주소별로 정리된 csv파일

기상청 Korea Meteorological Administration

date		dim_y	dim_x	ASR	ASR_DQF1	DSR	DSR_DQF1	RSR	RSR_DQF1	SW_DQF	_imager_projection
2020.3.1 0:00	637672	708	472	107.200005	1	121.200005	1	331	1	1	0
2020.3.1 1:00	637672	708	472	152.600006	1	172.600006	1	471.899994	1	1	0
2020.3.1 2:00	637672	708	472	129.199997	1	146.100006	1	634.400024	1	1	0
2020.3.1 3:00	637672	708	472	149.400009	1	168.900009	1	694.799988	1	1	0
2020.3.1 4:00	637672	708	472	204.5	1	231.199997	1	656.600037	1	1	0
2020.3.1 5:00	637672	708	472	226.5	1	256.100006	1	587.600037	1	1	0
2020.3.1 6:00	637672	708	472	93.8000031	1	106.099999	1	614.400024	1	1	0
2020.3.1 7:00	637672	708	472	133.699997	1	151.199997	1	412	1	1	0
2020.3.1 8:00	637672	708	472	154.199997	1	174.400009	1	186.5	1	0	0
2020.3.1 9:00	637672	708	472							0	0
2020.3.1 10:00	637672	708	472							0	0
2020.3.1 11:00	637672	708	472							0	0

강정동.csv

이 파일에는 ASR(흡수단파복사), DSR(하향단파복사), RSR(상향단파복사)데이터를 모두 포함하고 있으며,

각 데이터별로 '_DQF1'이라는 컬럼이 존재합니다. 이는 DQF1이 '1'인 경우에만 데이터가 유효함을 알려줍니다.

그리고 date 컬럼은 서울시간이 아닌 UTC시간으로 9시간이 차이가 나는 것에 주의 하셔야 합니다.

6 UTC시간을 서울 시간으로 변경, 시간 데이터 타입 변경

기상청 Korea Meteorological Administration

UTC시간을 서울 시간으로 변경하는 함수를 만듭니다.

```python
# UTC Time -> SEOUL Time
def convTime(dt_str):
    timeFormat = "%Y-%m-%d %H:%M"

    # Create datetime object in local timezone
    dt_utc = datetime.strptime(dt_str, timeFormat)
    dt_utc = dt_utc.replace(tzinfo=pytz.UTC)

    # Get local timezone
    local_zone = tz.tzlocal()

    # Convert timezone of datetime from UTC to local
    dt_local = dt_utc.astimezone(local_zone)
    conv_time = dt_local.strftime(timeFormat)
    return conv_time
```

예) “2022-10-10 13:24”

읽어올 시간의 포맷과 시간이 국제시간으로 설정

현재 시간 존을 읽어온다. (컴퓨터에 서울시간으로 설정된 내용을 읽어옴)

읽어온 시간과 동일한 포맷으로 시간 문자열 생성 (서울 시간)

7 하향 단파 복사 관련 데이터만 골라서 저장

위성 데이터의 csv파일에서 date, DSR, DSR_DQF1 컬럼만 선별적으로 가져옵니다.

```
data_dir = './sat_data/'
save_dir = './data/'

csv_list = glob(os.path.join(data_dir,'*.csv'))

for csvFile in csv_list:

    # 파일 읽기
    df = pd.read_csv(csvFile, encoding='cp949')
    dropCol = ['Unnamed: 1', 'dim_y', 'dim_x', 'ASR',
               'ASR_DQF1', 'RSR', 'RSR_DQF1', 'SW_DQF',
               'gk2a_imager_projection']

    df.drop(columns = dropCol, inplace=True)

    # 결측치를 0으로 채워준다.
    df.fillna(0, inplace=True)

    # UTC -> Seoul Time으로 변경한다.
    conv_time = list(map(convTime, df['date']))
    df['date']=conv_time
    # 컬럼명을 수정한다. 'date'-> '일시'
    df.rename(columns = {'date':'일시'},inplace=True)
    df['일시'] = pd.to_datetime(df['일시'])
    df.set_index(keys='일시', inplace=True, drop=True)
    # 파일명만을 가져온다.
    fileName = os.path.basename(csvFile)
    df.to_csv(save_dir+fileName, mode='w', encoding='utf-8-sig')
```

csv파일을 읽어서 불필요한 컬럼은 삭제 합니다.

결측치를 0으로 채우고 앞에서 만들었던 convTime함수를 사용하여 서울시간으로 변환한 뒤 다시 데이터 타입을 datetime 으로 변경합니다.

8 세 개의 csv파일 합치기

기상청 Korea Meteorological Administration

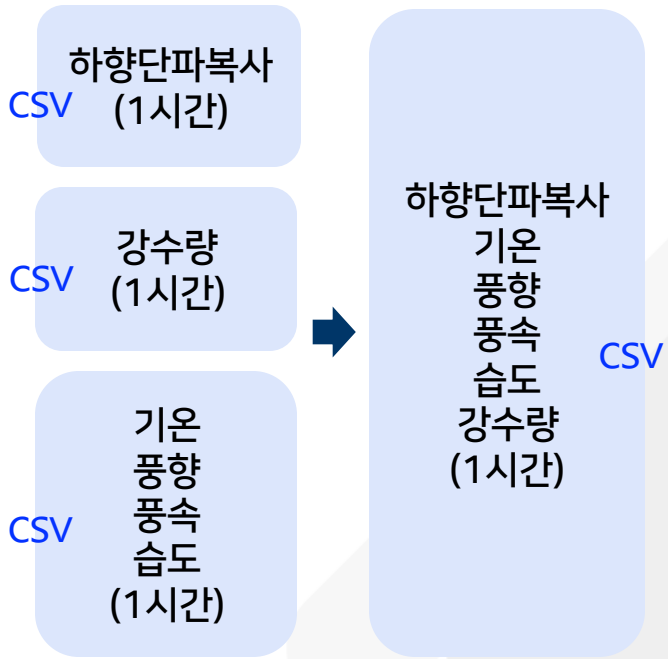
하향 단파 복사, 강수량, ASOS 또는 AWS 데이터 합치기

```
data_dir = './data/'
save_dir = './merge/'

for i in range(0, len(match_df)):
    # 하향 단파 복사
    mergeFile  = data_dir + match_df.iloc[i, 0]+'.csv'
    merge_df   = pd.read_csv(mergeFile, encoding='UTF-8')

    # 강수량 정보 파일 : '파일명2'
    precipFile = data_dir + match_df.iloc[i, 4]
    precip_df  = pd.read_csv(precipFile, encoding='UTF-8')

    # 기온, 풍향, 풍속, 습도 정보 파일 : '파일명1'
    awsFile = data_dir + match_df.iloc[i, 3]
    aws_df  = pd.read_csv(awsFile, encoding='UTF-8')

    # 데이터 프레임간 합치기
    # 위성 일사량 데이터가 중간 중간 누락되었다.
    # 따라서 AWS,ASOS데이터도 함께 삭제 해주어야 한다.
    # 그래서 데이터프레임을 합치는데 기준을 일사량 데이터의 인덱스로 한다.
    merge_df = pd.merge(merge_df, precip_df, how='left', left_on='일시', right_on='일시')
    merge_df = pd.merge(merge_df, aws_df, how='left', left_on='일시', right_on='일시')

    #merge_df.set_index(keys='일시', inplace=True, drop=True)
    merge_df.to_csv(save_dir + match_df.iloc[i, 0]+'.csv', mode='w', encoding='utf-8-sig')
```

하향단파복사 csv파일을 읽어 merge_df 생성

강수량 csv파일을 읽어 precip_df 생성

Aws, asos csv파일을 읽어 aws_df 생성



세 개의 csv파일을 합치고 하나의 csv파일로 저장합니다.

수고하셨습니다