기상청

날씨데이터를 활용한 태양광 발전량 예측

- 실습자료

기상청

contents

기상청

contents

분석개요

위성데이터 및 ASOS, AWS데이터를 활용하여 태양광 발전량을 예측하는 방법을 알아봅니다.
기상청은 서울 기상관측소를 비롯하여 전국 103개의 종관기상관측소(ASOS)와 무인으로 운영되는 510개소의 자동기상관측장비(AWS)를 이용하여 지상기상관측 업무를 수행하고 있습니다.

ASOS(Automated Synoptic Observing System)

종관 기상관측이란 종관 규모의 날씨를 파악하기 위해 정해진 시각에 모든 관측소에서 같은 시각에 실시하는 지상관측을 말합니다.
종관규모란 일기에도 표현된 보통의 고기압이나 저기압의 공간적 크기 및 수명을 말하며, 매일 날씨 현상을 뜻합니다.
제공기간 : 1904년~(지점별, 요소별 다름)
제공요소 : 기온, 강수, 바람, 기압, 습도, 일사, 일조, 눈, 구름, 시정, 지면상태, 지면 · 초상온도, 일기현상, 증발량, 현상번호
제공지점 : 103개 (원하는 지점이 없는 경우, 제공요소가 AWS에 존재한다면, AWS로 보완합니다)

AWS (Automatic Weather System)

- ASOS지점의 관측 공백 해소 및 국지적인 기상 현상을 파악하기 위해 설치되었습니다.
- AWS는 과거에 사람이 직접 관측하던 것을 자동으로 관측할 수 있도록 설계한 장비로, 실시간 측정, 연산, 저장, 표출 등 모든 과정을 자동으로 처리합니다.
- 제공기간 : 1997년~(지점별, 요소별 다름)
- 제공요소 : 기온, 강수, 바람, 기압, 습도 등
- 제공지점 : 510개

위성데이터 - 천리안위성 2A호

- 천리안위성 2A호는 천리안위성 1호의 기상관측 역할을 승계하는 차세대 정지궤도 기상위성으로,2019.7.25. 정식 서비스를 시작하였습니다.
- 총 16개 채널을 통해 다양한 관측이 가능하며, 동아시아는 10분, 한반도 영역은 2분 간격으로 관측자료가 생산되고 있습니다.

위성데이터 - 천리안위성 2A호

- 천리안위성 2A호는 천리안위성 1호의 기상관측 역할을 승계하는 차세대 정지궤도 기상위성으로,2019.7.25. 정식 서비스를 시작하였습니다.
- 총 16개 채널을 통해 다양한 관측이 가능하며, 동아시아는 10분, 한반도 영역은 2분 간격으로 관측자료가 생산되고 있습니다.

하향 단파 복사

- 태양으로부터 지구로 도달하는 복사를 태양복사 또는 단파복사라고 하는데, 이는 일사량과 매우 높은 상관관계를 지닙니다.

환경 설정

실습을 진행하기 위해 아나콘다(Anaconda)를 설치하고, 가상환경을 생성합니다.

- conda create -n weather python=3.8

생성된 가상환경으로 들어가서 필요한 패키지를 설치합니다. 그리고 jupyter notebook을 실행합니다.

- conda activate weather
- pip install numpy pandas matplotlib jupyter seaborn scikit-learn tensorflow geopy folium graphviz

기상청

Ⅰ. ASOS, AWS, 위성데이터 전처리

1. 데이터 폴더 확인하기
2. ASOS, AWS 데이터를 시간당 강수량으로 변경
3. 한 시간 단위로 데이터 주기 변경
4. 위성데이터의 UTC시간을 서울시간으로 변환하는 함수
5. 위성 데이터 중에 단파복사 데이터만 남기고 삭제
6. 발전소와 가장 근접한 관측지점으로 대체하기
7. 전처리한 세 개의 데이터 파일을 하나로 합치기
8. 제주 태양광 발전량 데이터 구하기
9. 날짜와 시간을 합하여 일시 컬럼을 생성

1. 데이터 폴더 확인하기

- Ground_weather_data 폴더안에는 아래와 같이 제주 지역별 날씨 정보를 포함하고 있습니다.

```
import os
from glob import glob
import pandas as pd
import numpy as np
```

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

dirNames = os.listdir(data_dir)
print(dirNames)
```

```
['aws330', '데이터값_설명.txt', '.DS_Store', 'aws893', 'asos188', 'aws779', 'aws792', 'aws793',
'aws885', 'aws884', '제주도태양광발전시설-기상관측소_매칭정보.csv', 'asos184', 'aws863', 'aws865', 'aw
s328', 'aws329', 'aws781']
```

2. ASOS, AWS 데이터를 시간당 강수량으로 변경(1/2)

- 분당 강수량으로 되어 있는 것을 한 시간 단위의 누적 강수량으로 데이터 주기를 변경해주는 함수를 만듭니다.

```
from datetime import datetime, tzinfo
from dateutil import tz
import pytz
import time

# 한 시간 단위의 누적 강수량으로 데이터를 변환한다.
def rain_sum(df):
    prevStr = ''
    timeStr = ''
    sum = 0.0

    time_list = []
    hour_rain = []

    for i in range(0, len(df)):
        timeStr = df.iloc[i,0].split(':')[0]
        # 이전 행과 현재 행의 '년.월.일 시'문자열이 같다면
        # 같은 '시'에 해당하므로 누적으로 더 해줍니다.
        if prevStr==timeStr:
            sum+=df.iloc[i,1]

        # 문자열이 다르다면 '시'가 변했으므로 누적된 값을 리스트에 저장하고
        # sum값을 새롭게 누적하기 시작합니다.
        else:
            time_list.append(df.iloc[i,0])
            hour_rain.append(sum)
            sum = df.iloc[i,1]

        #print(df.iloc[i,0],sum)
        prevStr = timeStr
    return time_list, hour_rain
```

2. ASOS, AWS 데이터를 시간당 강수량으로 변경(2/2)

- ground_weather_data폴더에 있는 파일중에 rain문자열을 포함하는 파일을 읽어서 결측치를 0으로 채우고 한 시간 누적 강수량으로 변환하고, 시간 문자열을 datetime타입으로 변환합니다.

```
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')
```

3. 한 시간 단위로 데이터 주기 변경(1/2)

- 현재 10분 단위로 되어 있는 데이터의 주기를 1시간 단위로 변경합니다.

```
def oneHour_avg(df):
    prevStr = ''
    timeStr = ''

    temp_sum = 0.0
    wdir_sum = 0.0
    wspd_sum = 0.0
    humi_sum  = 0.0

    time_list = []
    temp_list = []
    wdir_list = []
    wspd_list = []
    humi_list = []

    for i in range(0, len(df)):
        timeStr = df.iloc[i,0].split(':')[0]
        # 이전 행과 현재 행의 '년.월.일 시' 문자열이 같다면
        # 같은 '시'에 해당하므로 누적으로 더 해줍니다.
        if prevStr==timeStr:
            temp_sum += df.iloc[i,1]
            wdir_sum += df.iloc[i,2]
            wspd_sum += df.iloc[i,3]
            humi_sum += df.iloc[i,4]

        # 문자열이 다르다면 '시'가 변했으므로 누적된 값을 리스트에 저장하고
        # sum값을 새롭게 누적하기 시작합니다.
        else:
            time_list.append(df.iloc[i,0])
            temp_list.append(temp_sum/6)
            wdir_list.append(wdir_sum/6)
            wspd_list.append(wspd_sum/6)
            humi_list.append(humi_sum/6)
            temp_sum = df.iloc[i,1]
            wdir_sum = df.iloc[i,2]
            wspd_sum = df.iloc[i,3]
            humi_sum = df.iloc[i,4]

        prevStr = timeStr

    return time_list, temp_list, wdir_list, wspd_list, humi_list
```

3. 한 시간 단위로 데이터 주기 변경(2/2)

```
for dirName in dirNames:
    csv_list = glob(os.path.join(data_dir,dirName,'*.csv'))
    for csvFile in csv_list:

        count = csvFile.find('rain')

        # 파일명에 rain을 포함하지 않는 파일을 찾습니다.
        if count<0:
            fileName = os.path.basename(csvFile)
            #print(csvFile)

            # 파일 읽기
            df = pd.read_csv(csvFile, encoding='cp949')

            # 결측치를 0으로 채워준다.
            df.fillna(0, inplace=True)
            # 한 시간단위로 강수량을 수정한다.
            time_list, temp_list, wdir_list, wspd_list, humi_list = oneHour_avg(df)

            # pandas DataFrame 생성
            value = {'일시':time_list, '기온':temp_list, '풍향':wdir_list, '풍속':wspd_list, '습도'
            # 시간당 강수량을 저장하는 데이터프레임 생성
            df2 = pd.DataFrame(value)

            df2['일시'] = pd.to_datetime(df2['일시'])
            df2 = df2[df2['일시']>'2020.3.1 8:00']
            #print(df2)
            df2.set_index(keys='일시', inplace=True, drop=True)
            df2.to_csv(save_dir+fileName, mode='w', encoding='utf-8-sig')
```

4. 위성데이터의 UTC시간을 서울시간으로 변환하는 함수

```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
```

5. 위성 데이터 중에 단파복사 데이터만 남기고 삭제

```
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')
```

6. 발전소와 가장 근접한 관측지점으로 대체하기

- 현재 10분 단위로 되어 있는 데이터의 주기를 1시간 단위로 변경합니다. 우리가 필요로 하는 위도와 경도에 관측지점이 없는 경우 가장 가까운 대체 지점으로 합니다.

```
matchFile ='./jeju_geo_match_info.csv'

# 파일 읽기
match_df = pd.read_csv(matchFile, encoding='UTF-8')
match_df
```

	위도	경도	동이름	유형	지점번호	가까운 지점	대체위도	대체경도
0	33.221816	126.252994	대정읍	aws	793	대정	33.2410	126.2263
1	33.240160	126.380600	하예동	aws	328	중문	33.2494	126.4060
2	33.246950	126.489270	강정동	aws	884	기상(과)	33.2593	126.5176
3	33.246790	126.563960	서귀동	aws	884	기상(과)	33.2593	126.5176
4	33.261770	126.386780	상예동	aws	328	중문	33.2494	126.4060

- 필요한 정보만 남기고 나머지는 삭제합니다.

```
dropCol = ['위도', '경도', '대체위도', '대체경도', '가까운 지점']
match_df.drop(columns = dropCol, inplace=True)
```

	동이름	유형	지점번호
0	대정읍	aws	793
1	하예동	aws	328
2	강정동	aws	884
3	서귀동	aws	884
4	상예동	aws	328

-동별로 가장 가까운 관측지점의 번호로 파일명을 확인합니다.

```
fileList1 = []
fileList2 = []
for i in range(len(match_df)):
    fileList1.append(match_df.iloc[i, 1]+str(match_df.iloc[i, 2])+'.csv')
    fileList2.append(match_df.iloc[i, 1]+str(match_df.iloc[i, 2])+'_rain.csv')
match_df['파일명1'] = fileList1
match_df['파일명2'] = fileList2
match_df
```

	동이름	유형	지점번호	파일명1	파일명2
0	대정읍	aws	793	aws793.csv	aws793_rain.csv
1	하예동	aws	328	aws328.csv	aws328_rain.csv
2	강정동	aws	884	aws884.csv	aws884_rain.csv
3	서귀동	aws	884	aws884.csv	aws884_rain.csv
4	상예동	aws	328	aws328.csv	aws328_rain.csv

7. 전처리한 세 개의 데이터 파일을 하나로 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')
```

8. 제주 태양광 발전량 데이터 구하기

- 아래의 파일을 통해 제주 태양광 총발전량 데이터를 얻을 수 있습니다. 아래 파일을 열고 불필요한 데이터는 삭제합니다.

```
solarPowerFile = '한국전력거래소_시간별 육지 제주 태양광 및 풍력발전량_2014년-2020년.csv'

# 파일 읽기
power_df = pd.read_csv(solarPowerFile, encoding='cp949')
power_df
```

	거래일	거래시간	육지태양광 총발전량	육지풍력 총발전량	제주태양광 총발전량	제주풍력 총발전량
0	2014-01-01	0	0.000000	250.214160	0.000000	37.235992
1	2014-01-01	1	0.000000	230.334882	0.000000	40.034720
2	2014-01-01	2	0.000000	224.563859	0.000000	36.944184
3	2014-01-01	3	0.000000	223.519520	0.000000	30.664892
4	2014-01-01	4	0.000000	216.955896	0.000000	24.566528
...	...	...	...	...	...	...
61363	2020-12-31	19	406.184674	823.291232	0.436184	169.120761
61364	2020-12-31	20	289.995063	772.872278	0.025248	151.415406
61365	2020-12-31	21	164.918523	740.541095	0.000224	152.391992
61366	2020-12-31	22	69.294937	734.738133	0.029552	150.711033
61367	2020-12-31	23	15.777139	617.367822	0.000216	137.878316

61368 rows × 6 columns

9. 날짜와 시간을 합하여 일시 컬럼을 생성

- 날짜와 시간을 합하여 일시 컬럼을 생성하고 제주 태양광 총 발전량을 제외하고 나머지 컬럼을 삭제합니다. 그리고 일시 컬럼의 타입을 datetime타입으로 변경하고, index로 변경합니다.

```
dtList = []

for i in range(len(power_df)):
    dtStr = power_df.iloc[i,0] + ' ' + str(power_df.iloc[i,1])+':00'
    dtList.append(dtStr)

power_df['일시'] = dtList
power_df
```

```
for i, timeStr in enumerate(power_df['일시']):
    power_df.iloc[i,6] = (timeStr.replace('-','.'))

power_df
```

```
delCol = ['거래일','거래시간','육지태양광 총발전량', '육지풍력 총발전량','제주풍력 총발전량']
power_df.drop(columns=delCol, inplace=True)
power_df['일시'] = pd.to_datetime(power_df['일시'])
power_df = power_df[power_df['일시']>'2020.03.1 8:00']
power_df.set_index(keys='일시', inplace=True, drop=True)
power_df
```

	제주태양광 총발전량
일시	
2020-03-01 09:00:00	5.594496
2020-03-01 10:00:00	11.325051
2020-03-01 11:00:00	17.675403
2020-03-01 12:00:00	22.734366
2020-03-01 13:00:00	27.707441
...	...
2020-12-31 19:00:00	0.436184
2020-12-31 20:00:00	0.025248
2020-12-31 21:00:00	0.000224
2020-12-31 22:00:00	0.029552
2020-12-31 23:00:00	0.000216

7335 rows × 1 columns

기상청

II. 탐색적데이터분석
- 태양광발전소위치_시각화하기

1. 파일 불러오기
2. 컬럼별 데이터 특성 확인하기
3. 결측치 삭제
4. 데이터 이상치 수정
5. 사업개시일을 시간순으로 정렬
6. 또 다른 csv파일 불러오기
7. 태양광 발전소만 출력하기
8. 데이터 신뢰성 확인
9. 주소 컬럼 생성하기
10. 주소 중복을 피하기 위해 unique
11. 주소에서 '시' 문자 삭제
12. 한글 주소를 위도와 경도로 변환
13. Geopy에서 한글 주소가 인식이 안될때 영문 주소 사용하여 변환
14. 주소와 경도, 위도를 사용하여 데이터프레임 생성
15. Folium을 사용하여 marker표시하기

1. 파일 불러오기

- 데이터를 처리 하는데 필요한 패키지를 불러옵니다.

```
import os
import pandas as pd
import numpy as np

pd.set_option('mode.chained_assignment',  None)
```

- 태양광 발전소 정보를 포함하고 있는 '제주특별자치도_태양광발전소현황_20220630.csv' 파일을 불러옵니다. Csv파일안에는 한글이 포함되어 있어 encoding을 'cp949'로 설정합니다.

```
# 공공데이터포털에서 제주특별자치도_태양광발전소현황 파일 다운로드
# https://www.data.go.kr/data/3082724/fileData.do
df = pd.read_csv('제주특별자치도_태양광발전소현황_20220630.csv', encoding='cp949')
df
```

	행정시	읍면동	허가일자	상호	설비용량(KW)	상태	사업개시일	데이터기준일자
0	제주시	구좌읍	1998-06-10	행원풍력발전단지(1차)	3480.0	사업개시	2000-04-05	2022-06-30
1	제주시	회천동	2002-12-09	파낙스에너지(주)	1000.0	사업개시	2002-12-18	2022-06-30
2	제주시	한경면	2003-04-14	한경풍력 1단계	6000.0	사업개시	2004-02-28	2022-06-30
3	제주시	한경면	2005-04-04	신창풍력발전단지	1700.0	사업개시	2006-03-03	2022-06-30
4	제주시	구좌읍	2005-08-30	제주월정풍력발전소	1500.0	사업개시	2006-07-20	2022-06-30
...	...	...	...	...	...	...	...	...
2143	제주시	한경면	2022-06-08	민재 태양광발전소	99.6	인허가	NaN	2022-06-30
2144	제주시	한림읍	2022-06-08	효일4호 태양광발전소	99.0	인허가	NaN	2022-06-30
2145	제주시	한림읍	2022-06-08	효일3호 태양광발전소	99.0	인허가	NaN	2022-06-30
2146	제주시	한림읍	2022-06-08	효일5호 태양광발전소	99.0	인허가	NaN	2022-06-30
2147	서귀포시	성산읍	2022-06-21	한라2호소수력발전소	210.0	인허가	NaN	2022-06-30

2148 rows × 8 columns

2. 컬럼별 데이터 특성 확인하기

- 읽어온 데이터는 판다스 데이터프레임 타입으로 저장되는데 데이터의 컬럼별 특성들을 확인하기 위해 info()함수를 사용하여 정보를 출력합니다.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2148 entries, 0 to 2147
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   행정시       2148 non-null   object 
 1   읍면동       2148 non-null   object 
 2   허가일자      2148 non-null   object 
 3   상호        2148 non-null   object 
 4   설비용량(KW)  2148 non-null   float64
 5   상태        2148 non-null   object 
 6   사업개시일     1571 non-null   object 
 7   데이터기준일자   2148 non-null   object 
dtypes: float64(1), object(7)
memory usage: 134.4+ KB
```

3. 결측치 삭제

- 2019년 12월까지의 태양광 발전소 데이터만 선택하기 위해서는 사업개시일 컬럼이 중요한데 결측치가 있습니다. 결측치 데이터를 삭제합니다.

```
# 사업개시일이 있는 데이터만 골라낸다.
# 사업개시일이 없는 데이터는 아직 인허가가 진행중인 것이므로 drop
df = df.dropna(axis=0)
df
```

4. 데이터 이상치 수정

- 사업개시일이 있지만 '상태'컬럼이 '사업개시'가 아닌 경우를 확인해봅니다.

```
# 사업개시일이 기록되었는데 상태가 '공사진행','인허가'인 경우
df[df['상태']!='사업개시']
```

	행정시	읍면동	허가일자	상호	설비용량(KW)	상태	사업개시일	데이터기준일자
6	제주시	한경면	2006-08-11	탐라해상풍력발전소	30000.0	공사진행	2017-09-16	2022-06-30
29	서귀포시	성산읍	2008-02-28	성산풍력 1단계	12000.0	공사진행	2009-03-30	2022-06-30
583	제주시	한경면	2016-11-16	성용유화발전	750.0	인허가	2016-11-16	2022-06-30

- 상태는 '공사진행'이거나 '인허가'로 되어 있지만 사업개시일이 존재하므로 사업개시로 수정

```
df.loc[6, '상태'] = '사업개시'
df.loc[6]
```

```
df.loc[29, '상태'] = '사업개시'
df.loc[29]
```

```
df.loc[583, '상태'] = '사업개시'
df.loc[583]
```

5. 사업개시일을 시간순으로 정렬

- 사업개시일을 시간순으로 정렬하려면 문자열타입에서 datetime타입으로 변경하고, sort_values함수로, 사업개시일 컬럼값을 기준으로 정렬합니다. 그리고 인덱스를 새로 생성합니다.

```
df['사업개시일'] = pd.to_datetime(df['사업개시일'])
df.sort_values(by='사업개시일', inplace=True)

# 데이터프레임의 인덱스를 새롭게 생성한다.
df.reset_index(drop=True, inplace=True)
```

- 그런데 여기서, 문제가 발생합니다. 파일명에는 분명 태양광 발전소 현황으로 되어 있지만 태양광만이 아닌 신재생 에너지 발전소 전체의 정보가 포함되어 있습니다. 상호만으로 태양광 발전소를 걸러 내기에는 무리가 있어 보입니다.

6. 또 다른 csv파일 불러오기

- '제주특별자치도_신재생에너지발전시설현황_20200120.csv'파일을 불러옵니다. 이 파일도 파일을 불러올 때, encoding을 'cp949'를 사용합니다.

```
df2 = pd.read_csv('제주특별자치도_신재생에너지발전시설현황_20200120.csv', encoding='cp949')
df2
```

	허가일자	상호	설비용량(KW)	설치행정시	설치지역	원동력종류	사업개시일	구분	데이터기준일자
0	2018.4.4	(유)나월(나월태양광발전소)	496.80	제주시	애월읍	태양광	2019.4.26	육상	2020.1.20
1	2014.10.7	(유)메가솔라 태양광발전소	1800.00	제주시	구좌읍	태양광	2015.7.1	없음	2020.1.20
2	2014.5.15	(유)무한에코에너지(서민행복8호태양광발전소)	516.00	서귀포시	남원읍	태양광	2014.10.16	없음	2020.1.20
3	2014.12.9	(유)용흥마을태양광발전소	99.00	서귀포시	강정동	태양광	2015.10.29	없음	2020.1.20
4	2014.10.14	(유)제주 늘해랑	2198.00	서귀포시	성산읍	태양광	2019.1.31	없음	2020.1.20
...	...	...	...	...	...	...	...	...	...
836	2018.9.19	후니3 태양광발전소	93.15	서귀포시	표선면	태양광	2019.3.4	육상	2020.1.20
837	2014.10.28	희윤2 태양광발전소	99.00	서귀포시	성산읍	태양광	2015.6.22	없음	2020.1.20
838	2014.10.29	희윤4 태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20
839	2014.10.29	희윤5 태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20
840	2014.10.29	희윤태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20

841 rows × 9 columns

7. 태양광 발전소만 출력하기

- 이 파일도 신재생에너지 발전소 전체를 담고 있는 파일이지만, 원동력종류 컬럼이 있어서 태양광을 구분할 수 있습니다.

```
df2 = df2[df2['원동력종류']=='태양광']
df2.reset_index(drop=True, inplace=True)
df2
```

	허가일자	상호	설비용량(KW)	설치행정시	설치지역	원동력종류	사업개시일	구분	데이터기준일자
0	2018.4.4	(유)나월(나월태양광발전소)	496.80	제주시	애월읍	태양광	2019.4.26	육상	2020.1.20
1	2014.10.7	(유)메가솔라 태양광발전소	1800.00	제주시	구좌읍	태양광	2015.7.1	없음	2020.1.20
2	2014.5.15	(유)무한에코에너지(서민행복8호태양광발전소)	516.00	서귀포시	남원읍	태양광	2014.10.16	없음	2020.1.20
3	2014.12.9	(유)용흥마을태양광발전소	99.00	서귀포시	강정동	태양광	2015.10.29	없음	2020.1.20
4	2014.10.14	(유)제주 늘해랑	2198.00	서귀포시	성산읍	태양광	2019.1.31	없음	2020.1.20
...	...	...	...	...	...	...	...	...	...
806	2018.9.19	후니3 태양광발전소	93.15	서귀포시	표선면	태양광	2019.3.4	육상	2020.1.20
807	2014.10.28	희윤2 태양광발전소	99.00	서귀포시	성산읍	태양광	2015.6.22	없음	2020.1.20
808	2014.10.29	희윤4 태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20
809	2014.10.29	희윤5 태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20
810	2014.10.29	희윤태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20

811 rows × 9 columns

8. 데이터 신뢰성 확인

- 본 파일의 태양광 발전소의 갯수가 811개인데 갯수가 앞에서 불러왔던 발전소의 갯수와 차이가 있어서 정보의 신뢰성을 확인합니다. 2019년 12월 기준으로 신뢰할 만한 제주도내 태양광 발전소의 수와 설비용량 정보를 확인하기 위해 아래 자료를 참조하였습니다.

〈표 Ⅴ-34〉 태양광 현황(총괄) (2019년 12월 기준)

구분	합 계		운전중		추진중	
	용량(kW)	개소	용량(kW)	개소	용량(kW)	개소
계	639,462	1,860	**245,321**	**811**	394,141	1,049

* 자료: 제주특별자치도 내부자료

9. 주소 컬럼 생성하기

- '설치행정시'컬럼과 '설치지역' 컬럼을 더하여 '주소'컬럼을 만듭니다.

```
for i in range(len(df2)):
    addr = df2.loc[i].설치행정시 + ' ' + df2.loc[i].설치지역
    df2.loc[i, '주소'] = addr
```

	허가일자	상호	설비용량(KW)	설치행정시	설치지역	원동력종류	사업개시일	구분	데이터기준일자	주소
0	2018.4.4	(유)나월(나월태양광발전소)	496.80	제주시	애월읍	태양광	2019.4.26	육상	2020.1.20	제주시 애월읍
1	2014.10.7	(유)메가솔라 태양광발전소	1800.00	제주시	구좌읍	태양광	2015.7.1	없음	2020.1.20	제주시 구좌읍
2	2014.5.15	(유)무한에코에너지(서민행복8호태양광발전소)	516.00	서귀포시	남원읍	태양광	2014.10.16	없음	2020.1.20	서귀포시 남원읍
3	2014.12.9	(유)용흥마을태양광발전소	99.00	서귀포시	강정동	태양광	2015.10.29	없음	2020.1.20	서귀포시 강정동
4	2014.10.14	(유)제주 늘해랑	2198.00	서귀포시	성산읍	태양광	2019.1.31	없음	2020.1.20	서귀포시 성산읍
...	...	...	...	...	...	...	...	...	...	...
806	2018.9.19	후니3 태양광발전소	93.15	서귀포시	표선면	태양광	2019.3.4	육상	2020.1.20	서귀포시 표선면
807	2014.10.28	희윤2 태양광발전소	99.00	서귀포시	성산읍	태양광	2015.6.22	없음	2020.1.20	서귀포시 성산읍
808	2014.10.29	희윤4 태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20	서귀포시 성산읍
809	2014.10.29	희윤5 태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20	서귀포시 성산읍
810	2014.10.29	희윤태양광발전소	99.28	서귀포시	성산읍	태양광	2019.7.11	없음	2020.1.20	서귀포시 성산읍

811 rows × 10 columns

- 사업개시일이 있지만 '상태'컬럼이 '사업개시'가 아닌 경우를 확인해봅니다.

10. 주소 중복을 피하기 위해 unique

- 주소가 중복되는 경우 하나만 출력하기 위해 unique()함수를 사용합니다.

```
geo_addr = df2['주소'].unique()

geo_pos = []

for addr in geo_addr:
    geo_pos.append(addr.replace('시',''))

print(geo_pos)
```

11. 주소에서 '시' 문자 삭제

- 뒤에서 geopy를 사용하여 위도와 경도로 변환시 주소에 '시'가 포함되어 있으면 정상적으로 변환이 되지 않습니다.

```
geo_addr = df2['주소'].unique()

geo_pos = []

for addr in geo_addr:
    geo_pos.append(addr.replace('시',''))

print(geo_pos)
```

12. 한글 주소를 위도와 경도로 변환

- 주소값을 geopy패키지를 활용하여 경도와 위도 값을 얻는 함수를 만듭니다. Geopy는 별도의 가입없이 사용할 수 있는 패키지이지만, 한글 주소에 대해 약간의 오류를 가지고 있습니다. 영문 주소는 정상적으로 변환이 되지만 한글 주소는 가끔 None값을 반환합니다.

```
# 가입 없이 주소->좌표 변환
from geopy.geocoders import Nominatim

unique_addr=[]
unique_lat=[]
unique_lng=[]

def geocoding(address):
    geolocoder = Nominatim(user_agent = 'South Korea', timeout=None)
    geo = geolocoder.geocode(address)
    if geo==None:
        print('geo is null')
        return 0, 0
    else:
        return geo.latitude, geo.longitude
```

```
for addr in geo_pos:
    print(addr)
    lat, lng = geocoding(addr)
    if lat==0 or lng==0:
        continue
    unique_addr.append(addr)
    unique_lat.append(lat)
    unique_lng.append(lng)
    print(lat, lng)
```

13. Geopy에서 한글 주소가 인식이 안될때

- 번거롭지만 geopy에서 한글 주소가 인식되지 않을 경우 영문 주소로 입력해 줍니다.

```
ko_addr  = ['서귀포 안덕면', '서귀포 남원읍', '서귀포 도순동']
eng_addr = ["Andeok-myeon, Seogwipo-si", 'Namwon-eup, Seogwipo-si', '29beon-gil, Seogwipo-si']

index = 0
for addr in eng_addr:
    print(ko_addr[index])
    lat, lng = geocoding(addr)
    if lat==0 or lng==0:
        continue
    unique_addr.append(ko_addr[index])
    unique_lat.append(lat)
    unique_lng.append(lng)
    print(lat, lng)
    index+=1
```

14. 주소와 경도, 위도를 사용하여 데이터프레임 생성

- 앞에서 생성된 주소와 경도 위도를 사용하여 데이터프레임을 생성합니다.

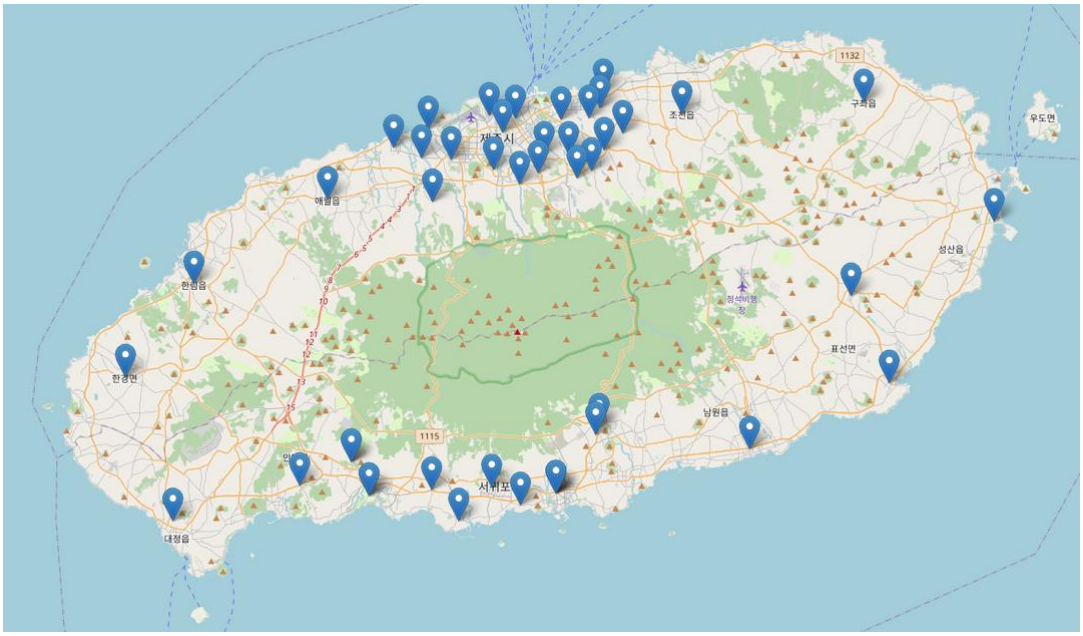
```
df_unique = pd.DataFrame({'주소': unique_addr,
                          '경도(x)': unique_lng,
                          '위도(y)': unique_lat})

df_unique
```

	주소	경도(x)	위도(y)
0	제주 애월읍	126.375200	33.450790
1	제주 구좌읍	126.810550	33.516620
2	서귀포 강정동	126.481362	33.232835
3	서귀포 성산읍	126.916280	33.435961

15. Folium을 사용하여 marker표시하기

- Folium을 사용하여 marker를 표시하고, html파일로 저장합니다.

```
import pandas as pd
import folium

jeju_map = folium.Map(location=[33.38, 126.55], zoom_start=11)

for i in range(len(pos_df)):
    pos_list = [pos_df.iloc[i][1], pos_df.iloc[i][0]]
    folium.Marker(pos_list, popup=pos_df.index[i]).add_to(jeju_map)

jeju_map.save('jeju.html')
```



기상청

Ⅲ. 데이터 상관관계 분석

1. 제주 전지역을 하나의 데이터프레임에 합하기
2. 제주태양광 총발전량 데이터도 합하기
3. Heatmap함수를 통해 상관관계 확인
4. Scatter를 통한 산점도 그래프로 특성 확인

1. 제주 전지역을 하나의 데이터프레임에 합하기

- 앞에서 전처리된 Csv파일을 읽어서 사용하지 않으려는 컬럼을 delCol리스트에 설정해주시면 됩니다. 아래 코드에서는 'DSR', 'DSR_DQF1', '강수량', '기온', '풍향', '풍속', '습도' 컬럼을 모두 사용합니다. 그리고 지역별 데이터들을 모두 가져와야 하는데 csv파일의 컬럼명이 동일합니다. 예를 들면 강정동에 '강수량'이 있고, 애월읍에도 '강수량'이 있는데 하나의 데이터프레임에는 동일한 컬럼명이 존재할 수 없습니다. 따라서 csv파일에서 컬럼을 읽어와서 통합된 데이터프레임에 추가할 때, 컬럼명을 변경해줘야 합니다. 아래의 코드에서는 "_숫자"를 컬럼명에 추가해서 컬럼명이 다르게 설정합니다.

```
# ['Unnamed: 0', 'DSR', 'DSR_DQF1', '강수량', '기온', '풍향', '풍속', '습도']
delCol = ['Unnamed: 0']
firstFlag = True
i=0

localName = []
newCol =[]

for dirName in dirNames:
    #지역별 컬럼 순서 리스트를 생성
    localName.append(dirName.split('/')[2].split('.')[0])

    if firstFlag== True:
        # 파일 읽기
        merge_df = pd.read_csv(dirName, encoding='UTF-8')
        print(merge_df.columns)
        merge_df.drop(columns = delCol, inplace=True)
        merge_df.set_index(keys='일시', inplace=True, drop=True)

        firstFlag = False
    else:
        temp_df = pd.read_csv(dirName, encoding='UTF-8')
        temp_df.drop(columns = delCol, inplace=True)
        temp_df.set_index(keys='일시', inplace=True, drop=True)
        merge_df = pd.merge(merge_df, temp_df, how='left', left_index=True, right_index=True)

    if 'DSR' not in delCol:
        newCol.append('DSR_{}'.format(i))
    if 'DSR_DQF1' not in delCol:
        newCol.append('DSR_DQF1_{}'.format(i))
    if '강수량' not in delCol:
        newCol.append('강수량_{}'.format(i))
    if '풍속' not in delCol:
        newCol.append('풍속_{}'.format(i))
    if '풍향' not in delCol:
        newCol.append('풍향_{}'.format(i))
    if '기온' not in delCol:
        newCol.append('기온_{}'.format(i))
    if '습도' not in delCol:
        newCol.append('습도_{}'.format(i))

    merge_df.columns = newCol
    i+=1

print(merge_df.columns)
print(newCol)
merge_df
```

2. 제주태양광 총발전량 데이터도 합하기

- 앞에서 제주도 전지역의 데이터를 하나의 데이터프레임으로 합한 후에 Matplotlib에서 한글을 출력하려면 아래와 같이 설정을 해줘야 합니다.

```
merge_df = pd.merge(merge_df, solar_df, how='left', left_index=True, right_index=True)
merge_df = merge_df.fillna(0)
print(localName)
merge_df.head(20)
```

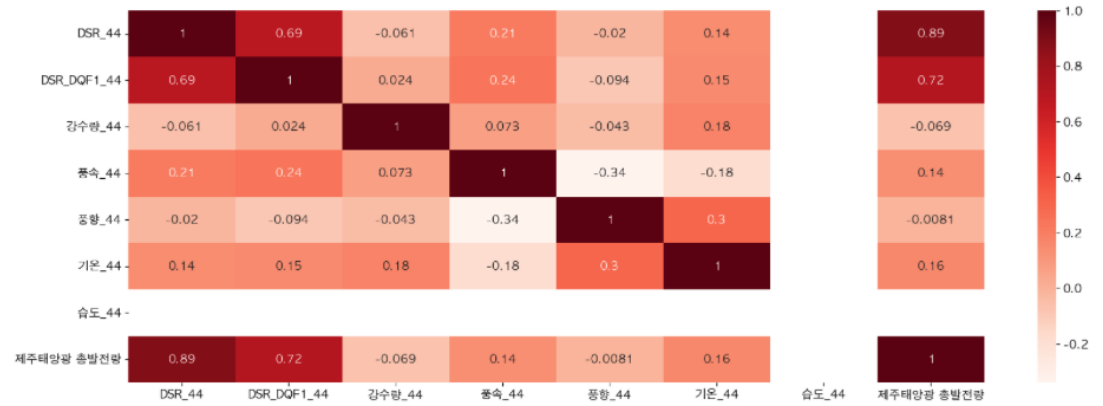
- 운영체제별로 폰트가 조금씩 차이가 나기 때문에 운영체제별로 다르게 설정했습니다.
- 운영체제를 식별하기 위해 platform패키지를 사용했습니다.

```
import platform

import matplotlib.pyplot as plt
from matplotlib import rc
import seaborn as sns
%matplotlib inline

if platform.system()=='Windows':
    plt.rc('font', family='NanumMyeongjo') # For Windows
    print(plt.rcParams['font.family'])
elif platform.system()=='Darwin':
    plt.rc('font', family='AppleGothic') # For MacOS
    print(plt.rcParams['font.family'])
else:
    plt.rc('font', family='NanumGothic')
    print(plt.rcParams['font.family'])

# 폰트 사이즈
plt.rcParams['font.size'] = 12.

# x축 라벨사이즈
plt.rcParams['xtick.labelsize'] = 10.

# y축 라벨사이즈
plt.rcParams['ytick.labelsize'] = 10.

#그래프 축 폰트사이즈
plt.rcParams['axes.labelsize'] = 10
plt.rcParams['axes.unicode_minus'] = False
```

3. Heatmap함수를 통해 상관관계

- heatmap함수를 사용하여 상관관계를 확인할 수 있습니다. DSR(하향단파복사)데이터는 0.89~0.92정도의 상관계수를 갖습니다. 그런데 의외로 높은 상관계수를 갖는 데이터가 DSR_DQF입니다. DSR_DQF는 DSR데이터의 유효성을 나타내는 데이터입니다. 기온과 풍속이 0.16, 0.14의 상관계수를 갖습니다. 상관계수로는 풍향과 강수량이 낮은 값을 갖는 것을 확인할 수 있습니다.

```
import matplotlib.pyplot as plt
import seaborn as sns

#Using Pearson Correlation
plt.figure(figsize=(20,7))
cor = merge_df.iloc[:,colNum-8:].corr()
heatmap = sns.heatmap(cor, annot=True, cmap=plt.cm.Reds)
plt.show()

fig = heatmap.get_figure()
fig.savefig("solar_dsr_corr.png")
```



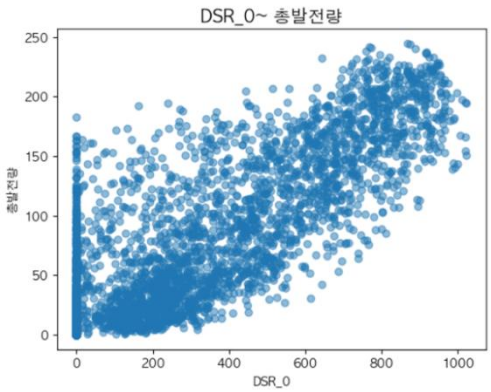
4. Scatter를 통한 산점도 그래프로 특성 확인(1/2)

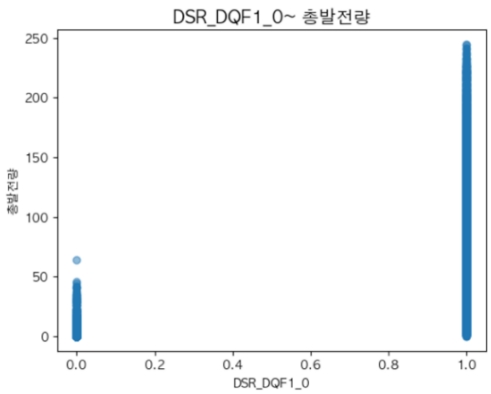
- 태양광 총 발전량과 각 컬럼 간의 특성을 그래프로 출력합니다.

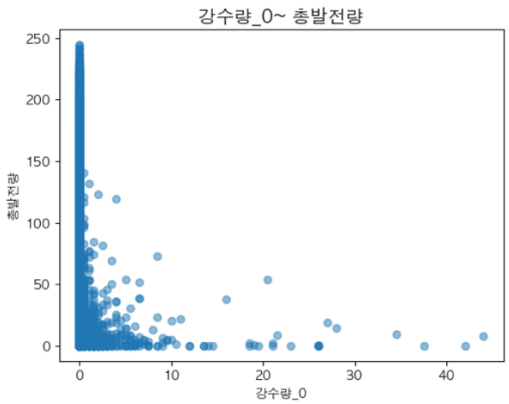
```
import matplotlib.pyplot as plt

i=0
for colName in merge_df.columns:

    X = merge_df[colName]
    Y = merge_df['제주태양광 총발전량']

    plt.scatter(X, Y, alpha=0.5)
    plt.title(colName +'~ 총발전량')
    plt.xlabel(colName)
    plt.ylabel('총발전량')
    plt.show()

    if i>20:
        break
    else:
        i+=1
```

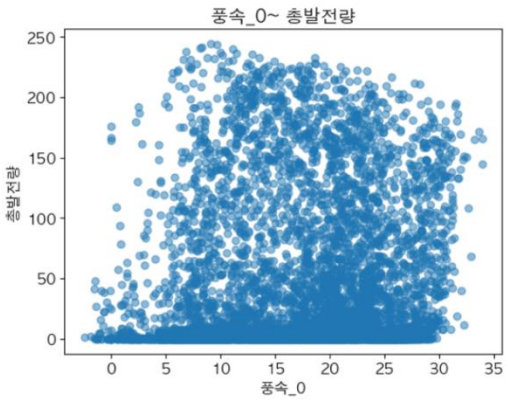
4. Scatter를 통한 산점도 그래프로 특성 확인(2/2)

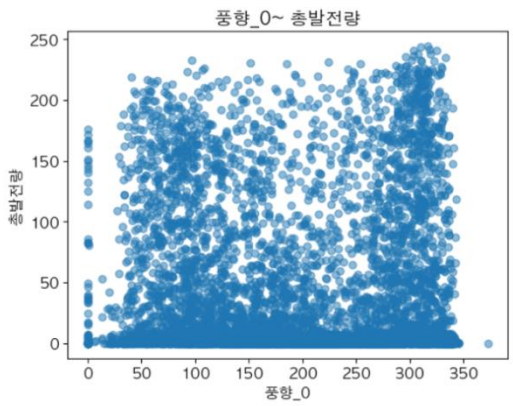
- 각 컬럼 데이터와 총 발전량과의 상관관계를 산점도로 확인할 수 있습니다.

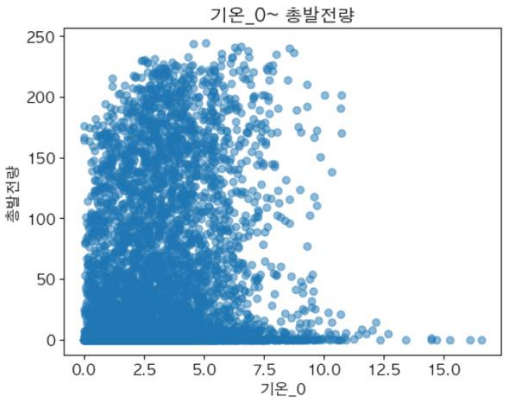












기상청

Ⅳ. 딥러닝 모델 학습

1. 모든 컬럼의 데이터 정규화
2. 다변량 함수의 시계열 데이터화
3. 시계열 데이터 모델링을 위해 LSTM모델 구성
4. 케라스 콜백함수 설정 및 학습
5. 테스트 데이터로 mae, mse값 확인

1. 모든 컬럼의 데이터 정규화

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler

scaler = MinMaxScaler(feature_range = (0,1))
#scaler = StandardScaler()

def normalize(X,y):
    X_norm = X.copy()

    for name in X:
        temp = X[name].to_numpy().reshape(-1,1)
        #print(temp)
        X_norm[name] = scaler.fit_transform(temp)


    temp = y.to_numpy().reshape(-1,1)
    y_norm = scaler.fit_transform(temp)

    return X_norm, y_norm

X_norm, ydata = normalize(X,y)
Xdata = X_norm.to_numpy()
```

2. 다변량 함수의 시계열 데이터화

- 시계열 모델을 학습 시키기 위해 일정한 시간길이 단위로 데이터를 끊어서 Train, test데이터를 준비합니다.

```
def multivariate_data(dataset, target, start_index, end_index, history_size, target_size, step,
    data = []
    labels = []

    start_index = start_index + history_size
    if end_index is None:
        end_index = len(dataset) - target_size

    for i in range(start_index, end_index):
        indices = range(i - history_size, i, step)
        data.append(dataset[indices])

        if single_step:
            labels.append(target[i + target_size])
        else:
            labels.append(target[i:i + target_size])
    return np.array(data), np.array(labels)
```

```
sequence_length = 24
# 다음 한시간을 예측
future_target = 1
STEP = 1

X_train, y_train = multivariate_data(Xdata, ydata, 0, TRAIN_SPLIT, sequence_length,
                                                    future_target, STEP, single_step=True)
X_test, y_test = multivariate_data(Xdata, ydata, TRAIN_SPLIT, None, sequence_length,
                                                  future_target, STEP, single_step=True)
X_train.shape, X_test.shape
```

```
((6476, 24, 315), (661, 24, 315))
```

3. 시계열 데이터 모델링을 위해 LSTM모델 구성

```
from tensorflow.keras import Input, Model
from tensorflow.keras.layers import Dense, SimpleRNN, Bidirectional, Dropout

h_units = 100

inputs = Input(shape=(X_train.shape[-2:]))
x = LSTM(h_units, return_sequences=False, recurrent_dropout=0.5, kernel_initializer='he_normal')
#x = LSTM(h_units, return_sequences=False, kernel_initializer='he_normal')(x)
x = Dropout(0.5)(x)

outputs = Dense(1)(x)

model = Model(inputs, outputs)

model.compile(loss='mse', metrics=['mae'], optimizer=Adam(learning_rate = 0.0006))
```

4. 케라스 콜백함수 설정 및 학습

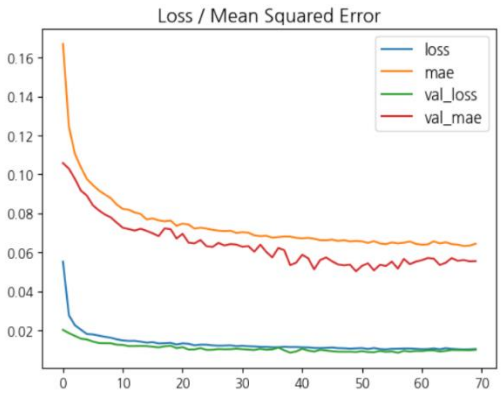
1. ReduceLROnPlateau : 'val_mae'를 기준으로 7epoch동안 최소값보다 낮아지지 않으면 학습율을 0.5로 줄여줍니다.
2. EarlyStopping : 'val_mae'를 기준으로 20epoch동안 최소값보다 낮아지지 않으면 학습을 중단합니다.
3. ModelCheckpoint : 'val_mae'를 기준으로 더 낮은 최소값이 나오면 그 때의 가중치값을 best.h5파일에 저장합니다.

- 콜백함수를 설정하고, 학습을 진행합니다.

```
from tensorflow.keras.callbacks import EarlyStopping,ReduceLROnPlateau, ModelCheckpoint


# 'val_mae'를 기준으로 7epoch동안 val_mae 값이 줄어들지 않을 경우 기존의 learning_rate를 반으로 줄인다.
reduce_lr = ReduceLROnPlateau(monitor='val_mae', factor=0.5,patience=7)

#'val_mae'가 20EPOCH동안 내려가지 않으면 학습을 중단한다.
es=EarlyStopping(monitor='val_mae', patience=20)

# 학습중에 'val_mae'가 가장 낮을 때 'best.h5'에 weight값을 저장한다.
mc = ModelCheckpoint('best.h5', monitor='val_mae', mode='auto', verbose = 1, save_best_only=True)
callback = [reduce_lr, es, mc]

history = model.fit(train_data, epochs=150, batch_size=BATCH_SIZE, validation_data=test_data, callbacks=[callback])
```

- 학습이 진행되는 동안 매epoch마다 loss(mse), val_loss(mse), mae, val_mae값을 기록하고 Pyplot함수를 활용하여 그래프로 확인합니다.

```
import matplotlib.pyplot as pyplot

if MODEL=='LSTM':
    model.load_weights('best.h5')

    pyplot.title('Loss / Mean Squared Error')
    pyplot.plot(history.history['mae'], label='mae')
    pyplot.plot(history.history['val_mae'], label='val_mae')
    #pyplot.plot(history.history['val_mae'], label='val_mae')
    pyplot.legend()
    pyplot.show()

    test_loss = model.evaluate(X_test, y_test)

    y_pred = model.predict(X_test)
```



5. 테스트 데이터로 mae, mse값 확인

```
y_pred = model.predict(X_test)

y_pred
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
from math import sqrt

mae_v  = mean_absolute_error(y_test, y_pred)
mse_v = mean_squared_error(y_test, y_pred)
mae_v, mse_v
```

본 문서의 내용은 기상청 날씨마루(https://bd.kma.go.kr)의
날씨데이터를 활용한 태양광 발전량 예측 실습 자료입니다.