



서울 대기 오염 분석

컴퓨터공학전공

2018108278

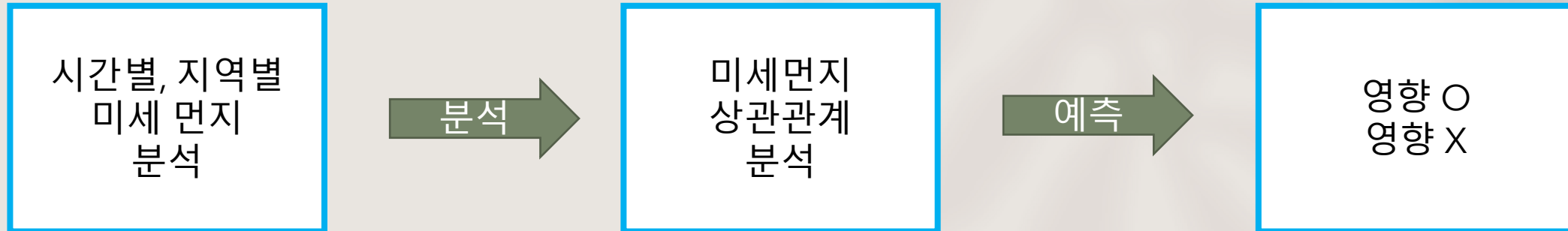
허재혁

목 차

1. 개요
2. 필요성
3. 사용 라이브러리
4. 사용 Dataset 설명
5. 상세 코드 분석(데이터 전처리, 시각화, 데이터 상관관계)
6. 테스트 결과
7. 결론



1. 개요



2. 필요성



(대기 오염으로 인한 질병)

대기 오염은 지구상에서도 심각한 환경 문제로도 꼽히고, 인간의 건강에도 악영향을 끼친다.
이에 따라 미리 공기의 오염도를 분석하여 미리 미세먼지 농도를 확인하여 예방하거나 대비할 수 있다.

데이터 소개

데이터 출처: 서울 열린 데이터 광장(<https://data.seoul.go.kr/dataList/OA-15526/S/1/datasetView.do>)

서울특별시

서울시 지방분권 전문가 포럼 개최 '지방시대, 서울의 미래와 지방분권 과제'

서울소식

응답소

정보공개

분야별정보

로그인 | 회원가입 | 사이트맵

서울 열린데이터 광장

공공데이터

통계

서울빅데이터

소식&참여

이용안내

데이터셋

Home > 공공데이터 > 공공데이터

찾고 싶은 데이터를 입력해 주세요.


Q

상세 검색

통합 검색

☐ 결과 내 재검색

공공데이터



환경

활용사례(갤러리) 등록

URL 복사

목록 이동






서울시 대기오염 측정정보

서울특별시 보건환경연구원 대기오염측정망시스템에서 제공하는 대기오염 측정정보입니다.
1시간평균 측정값을 보정 후 매시 5분에 제공합니다.
각 자치구의 측정소에서 측정된 결과를 바로 보정후 직접제공하여 보다 빠른 데이터를 제공합니다.
측정기 상태: "0": 정상 "1": 교정, "2": 비정상, "4": 전원단절, "8": 보수중, "9": 자료이상

[전체 설명보기](#)

파일내려받기

* 파일에 이상이 있는 경우 '오류신고'를 통해 운영자에게 알려주세요. [오류신고](#)

NO	항목	파일명	용량(MB)	수정일	내려받기
1	데이터	AIR_HOUR_2021.csv	55.76	2022.11.09.	
2	데이터	AIR_HOUR_2020.csv	55.96	2022.11.09.	
3	데이터	AIR_HOUR_2019.csv	57.59	2022.11.09.	
4	데이터	AIR_HOUR_2018.csv	53.34	2022.11.09.	
5	데이터	AIR_HOUR_2017.csv	52.96	2022.11.09.	

3.사용 라이브러리

In [1]:

```
'''라이브러리 불러오기'''  
import numpy as np  
import pandas as pd  
import matplotlib  
import matplotlib.pyplot as plt  
import seaborn as sns  
%matplotlib inline
```

In [10]:

```
import folium  
from folium.plugins import MarkerCluster
```

In [11]:

```
from datetime import datetime  
  
df['Measurement date'] = df['Measurement date'].astype('datetime64')  
df['hour'] = df.loc[:, "Measurement date"].dt.hour  
df = df.drop('Measurement date', axis=1)
```

3.사용 라이브러리

Numpy

: 배열 라이브러리
(다양한 수치연산 기능을 갖는다)

Pandas

: 데이터 분석을 위한 라이브러리
(데이터베이스에 비유)

Matplotlib

: 다양한 데이터를 많은 방법으로 도식화할 수 있는 파이썬 라이브러리

Seaborn

: 데이터를 멋지게 표시하는 모듈
(엑셀에 비유)

Folium

: 파이썬에서 지리 데이터를 효과적으로 시각화해주는 라이브러리

datetime

: 날짜와 시간을 표현하기 위한 라이브러리

4.사용 Dataset

- **MeasureMent date:** 측정 날짜
- **Station Code :** 측정장소 코드
- **Latitude :** 위도
- **Longitude :** 경도
- **PM2.5 :** 입자의 크기가 2.5um이하인 미세먼지 *PM(Particulate Matter)
- **PM10 :** 입자의 크기가 지름 10um이하인 미세먼지
- **SO2:** 이산화황
- **NO2:** 이산화질소
- **O3:** 오존
- **CO:** 일산화탄소

5.1 데이터 전처리



'''데이터 불러오기'''

```
df = pd.read_csv('/kaggle/input/airpollution/air_seoul.csv')  
df.head()
```

```
[2]:
```

	Measurement date	Station code	Address	Latitude	Longitude	SO2	NO2	O3	CO	PM10	PM2.5
0	2021-01-01 0:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.059	0.002	1.2	73	57
1	2021-01-01 1:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.058	0.002	1.2	71	59
2	2021-01-01 2:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.056	0.002	1.2	70	59
3	2021-01-01 3:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.056	0.002	1.2	70	58
4	2021-01-01 4:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.003	0.051	0.002	1.2	69	61

.head() 함수는 불러온 데이터의 상위 5개의 행을 출력한다.

5.1 데이터 전처리

```
[3]: '''데이터의 차원'''  
df.shape
```

```
[3]: (647511, 11)
```

```
[4]: '''Column 데이터 출력'''  
df.columns.tolist()
```

```
[4]: ['Measurement date',  
      'Station code',  
      'Address',  
      'Latitude',  
      'Longitude',  
      'SO2',  
      'NO2',  
      'O3',  
      'CO',  
      'PM10',  
      'PM2.5']
```

Shape : 행과 열의 개수를 튜플로 반환해준다. (647511, 11) -> 647511개의 행, 11개의 열

Columns: 열을 기준으로 데이터의 형태가 반환되었다.

5.1 데이터 전처리

[5]:

```
'''데이터 정보'''  
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 647511 entries, 0 to 647510  
Data columns (total 11 columns):  
#   Column                Non-Null Count  Dtype    
---  ---                  
0   Measurement date      647511 non-null object   
1   Station code          647511 non-null int64    
2   Address               647511 non-null object   
3   Latitude              647511 non-null float64   
4   Longitude             647511 non-null float64   
5   SO2                   647511 non-null float64   
6   NO2                   647511 non-null float64   
7   O3                    647511 non-null float64   
8   CO                    647511 non-null float64   
9   PM10                  647511 non-null int64     
10  PM2.5                 647511 non-null int64     
dtypes: float64(6), int64(3), object(2)  
memory usage: 54.3+ MB
```

.info() 함수는 데이터에 대한 전반적인 정보를 나타낸다. Df를 구성하는 행과 열의 크기, 컬럼명, 컬럼을 구성하는 값의 자료형등을 출력해준다.

647511 non-null -> 모든 열에 대하여 647511개의 데이터 중 647511개 전부 결측치가 아니므로 결측치가 없다는 뜻이다.

5.1 데이터 전처리

```
[6]: '''결측치 확인'''  
pd.isnull(df)  
df.isnull().sum()
```

```
[6]: Measurement date    0  
Station code          0  
Address               0  
Latitude              0  
Longitude             0  
SO2                   0  
NO2                   0  
O3                    0  
CO                    0  
PM10                  0  
PM2.5                 0  
dtype: int64
```

-> 결측 데이터는 흔히 발생하기 때문에 데이터를 가지고 결론을 내릴 때 상당한 영향을 끼칠 수도 있다. 결측값은 아예 제거를 해주거나, 특정 값으로 채워주거나 크게 두 가지 선택을 해주는 것이 좋다.

.isnull(), isnull().sum() 함수는 결측값을 확인할 때 사용하는 함수이다.

5.1 데이터 전처리

```
[7]: '''각 열들의 고유값 정보 출력'''  
df.nunique()
```

```
[7]: Measurement date    25911  
     Station code       25  
     Address           25  
     Latitude          25  
     Longitude         25  
     SO2               186  
     NO2               132  
     O3                253  
     CO                172  
     PM10              551  
     PM2.5             333  
     dtype: int64
```

.nunique()함수는 데이터에 고유값들의 수를 출력해주는 함수이다.

5.1 데이터 전처리

```
[8]: '''중복된 데이터 확인'''  
df.duplicated()
```

```
[8]: 0      False  
     1      False  
     2      False  
     3      False  
     4      False  
     ...  
     647506 False  
     647507 False  
     647508 False  
     647509 False  
     647510 False  
     Length: 647511, dtype: bool
```

.duplicated()함수는 중복 여부를 확인할 때 사용하는 함수이다.

5.2 데이터 시각화(위도 & 경도 데이터)

[9]:

```
# 위도 경도 DataFrame
location = df.groupby('Station code')['PM10'].agg([np.mean])
location['Latitude'] = df['Latitude'].unique()
location['Longitude'] = df['Longitude'].unique()
location.head()
```

[9]:

	mean	Latitude	Longitude
Station code			
101	37.965605	37.572016	127.005008
102	37.970469	37.564263	126.974676
103	35.539183	37.540033	127.004850
104	42.328468	37.609823	126.934848
105	41.437737	37.593742	126.949679

102 37.970469 37.564263 126.974676

104 42.328468 37.609823 126.934848

103 35.539183 37.540033 127.004850

5.2 데이터 시각화(위도 & 경도 데이터)

```
[58]:
import folium
from folium.plugins import MarkerCluster

# PM10에 따른 color 변화
def color_select(x):
    if x >= 45:
        return 'red'
    elif x >= 40:
        return 'yellow'
    else:
        return 'blue'

# Map
seoul = folium.Map(location=[37.55138077230307, 126.98712254969668], zoom_start=12)

# Circle
for i in range(len(location)):
    # 관측소
    folium.Circle(location=[location.iloc[i,1], location.iloc[i,2]], radius = location.iloc[i, 0]*30, color=color_select(location.iloc[i,0]), fill_color='#ffffgg').add_to(seoul)

# Marker / Sejong Univ.
folium.Marker([37.55195608145124, 127.07362532752212], icon=folium.Icon(popup='Sejoing Univ.', color='red', icon='glyphicon glyphicon-home')).add_to(seoul)
seoul
```

- Location(위도, 경도): 위,경도 좌표를 기준으로 지도를 그릴 수 있다.
- Zoom_start: 정보를 지정하여 확대의 정도를 지정할 수 있다.(최대 18까지)

5.2 데이터 시각화(시간별 미세먼지 농도)

```
from datetime import datetime

df['Measurement date'] = df['Measurement date'].astype('datetime64')
df['hour'] = df.loc[:, "Measurement date"].dt.hour
df = df.drop('Measurement date', axis=1)
```

datetime 라이브러리로 시간별 미세먼지 농도 측정

5.2 데이터 시각화(시간별 미세먼지 농도)

```
[13]: data = df.groupby('hour', as_index=False).agg({'SO2': 'mean', 'NO2': 'mean', 'O3': 'mean', 'CO': 'mean', 'PM10': 'mean', 'PM2.5': 'mean'})
data
```

```
[13]:
```

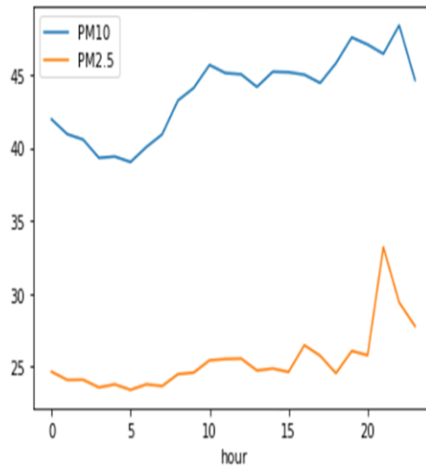
	hour	SO2	NO2	O3	CO	PM10	PM2.5
0	0	-0.001909	0.024584	0.011626	0.529891	41.944368	24.651817
1	1	-0.002126	0.021533	0.012065	0.524424	40.927181	24.091507
2	2	-0.002047	0.019609	0.012448	0.516841	40.558962	24.103003
3	3	-0.002203	0.018209	0.012198	0.509860	39.303351	23.577686
4	4	-0.002267	0.018247	0.011033	0.507101	39.394380	23.785717
5	5	-0.002303	0.020649	0.008226	0.515338	39.018339	23.413360
6	6	-0.001988	0.025026	0.005753	0.536940	40.044229	23.788876
7	7	-0.001966	0.027204	0.005840	0.562405	40.911421	23.670649
8	8	-0.001507	0.028721	0.008698	0.576376	43.231577	24.484415
9	9	-0.001398	0.027778	0.014076	0.559335	44.089890	24.593978
10	10	-0.001183	0.024440	0.019282	0.529488	45.661971	25.433010
11	11	-0.001360	0.020498	0.022907	0.500881	45.108788	25.520666
12	12	-0.001286	0.018103	0.028270	0.482626	45.022000	25.553704
13	13	-0.001212	0.016675	0.031950	0.470215	44.155556	24.737778
14	14	-0.001377	0.016059	0.034195	0.459807	45.197106	24.870575
15	15	-0.001595	0.016763	0.034715	0.438539	45.164152	24.636552
16	16	-0.001363	0.018260	0.032921	0.446770	44.994926	26.470481
17	17	-0.001417	0.021212	0.029132	0.455848	44.426963	25.748778
18	18	-0.001553	0.024476	0.023835	0.481205	45.778767	24.546031
19	19	-0.001558	0.026721	0.019516	0.507660	47.546279	26.075595
20	20	-0.001681	0.027166	0.016499	0.522186	47.059484	25.781436
21	21	-0.001718	0.027119	0.014502	0.528154	46.420701	33.179113
22	22	-0.001829	0.027196	0.012546	0.531633	48.375496	29.401208
23	23	-0.004239	0.024149	0.009250	0.526959	44.636799	27.775472

groupby함수에서 **agg**함수는 집계함수이다. 여러 개의 함수를 여러 개의 칼럼에 적용할 수 있는 그룹 연산 방법이다.

5.2 데이터 시각화(시간별 미세먼지 농도)

```
[17]: # 미세먼지 농도변화 Hour  
data.plot(x='hour', y=['PM10', 'PM2.5'])
```

```
[17]: <AxesSubplot:xlabel='hour'>
```



PM10은 약 22시경에 농도가 제일 높고 PM2.5는 약 21시경에 농도가 제일 높은 것을 볼 수 있다.

5.2 데이터 시각화(시간별 미세먼지 농도)

```
[19]: #세종대 주변 미세먼지 농도
df_sj = pd.read_csv('/kaggle/input/airpollution/air_seoul.csv')
## 데이터 다시 가져오기
df_sj.head()
```

```
[19]:
```

	Measurement date	Station code	Address	Latitude	Longitude	SO2	NO2	O3	CO	PM10	PM2.5
0	2021-01-01 0:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.059	0.002	1.2	73	57
1	2021-01-01 1:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.058	0.002	1.2	71	59
2	2021-01-01 2:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.056	0.002	1.2	70	59
3	2021-01-01 3:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.056	0.002	1.2	70	58
4	2021-01-01 4:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.003	0.051	0.002	1.2	69	61

```
[22]: condition = (df_sj.date == '2019-04-03')
df_birth = df_sj[condition]
df_birth.head()
## 특정 시간 기준으로 추출
```

```
[22]:
```

	Station code	Address	Latitude	Longitude	SO2	NO2	O3	CO	PM10	PM2.5	date	time
19505	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.003	0.026	0.035	0.4	30	18	2019-04-03	0:00
19506	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.003	0.026	0.033	0.5	29	17	2019-04-03	1:00
19507	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.003	0.025	0.035	0.5	31	21	2019-04-03	2:00
19508	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.003	0.022	0.036	0.4	27	15	2019-04-03	3:00
19509	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.003	0.029	0.028	0.5	27	15	2019-04-03	4:00

5.2 데이터 시각화(시간별 미세먼지 농도)

```
[23]:
cheak = df_birth['Address'].unique()
cheak
## 세종대 주변 위치 찾기

[23]: array(['19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republic of Korea',
        '15, Deoksugung-gil, Jung-gu, Seoul, Republic of Korea',
        '136, Hannam-daero, Yongsan-gu, Seoul, Republic of Korea',
        '215, Jinheung-ro, Eunpyeong-gu, Seoul, Republic of Korea',
        '32, Segeomjeong-ro 4-gil, Seodaemun-gu, Seoul, Republic of Korea',
        '10, Poeun-ro 6-gil, Mapo-gu, Seoul, Republic of Korea',
        '18, Ttukseom-ro 3-gil, Seongdong-gu, Seoul, Republic of Korea',
        '571, Gwangnaru-ro, Gwangjin-gu, Seoul, Republic of Korea',
        '43, Cheonho-daero 13-gil, Dongdaemun-gu, Seoul, Republic of Korea',
        '369, Yongmasan-ro, Jungnang-gu, Seoul, Republic of Korea',
        '70, Samyang-ro 2-gil, Seongbuk-gu, Seoul, Republic of Korea',
        '49, Samyang-ro 139-gil, Gangbuk-gu, Seoul, Republic of Korea',
        '34, Sirubong-ro 2-gil, Dobong-gu, Seoul, Republic of Korea',
        '17, Sanggye-ro 23-gil, Nowon-gu, Seoul, Republic of Korea',
        '56, Jungang-ro 52-gil, Yangcheon-gu, Seoul, Republic of Korea',
        '71, Gangseo-ro 45da-gil, Gangseo-gu, Seoul, Republic of Korea',
        '45, Gamasan-ro 27-gil, Guro-gu, Seoul, Republic of Korea',
        '20, Geumha-ro 21-gil, Geumcheon-gu, Seoul, Republic of Korea',
        '11, Yangsan-ro 23-gil, Yeongdeungpo-gu, Seoul, Republic of Korea',
        '6, Sadang-ro 16a-gil, Dongjak-gu, Seoul, Republic of Korea',
        '14, Sillindong-gil, Gwanak-gu, Seoul, Republic of Korea',
        '16, Sinbanpo-ro 15-gil, Seocho-gu, Seoul, Republic of Korea',
        '426, Hakdong-ro, Gangnam-gu, Seoul, Republic of Korea',
        '236, Baekjegobun-ro, Songpa-gu, Seoul, Republic of Korea',
        '59, Gucheonmyeon-ro 42-gil, Gangdong-gu, Seoul, Republic of Korea'],
      dtype=object)
```

```
[24]:
condition = (df_birth.Address == '571, Gwangnaru-ro, Gwangjin-gu, Seoul, Republic of Korea')
df_add = df_birth[condition]
df_add.head()
## 평균구 기준으로 데이터 추출
```

```
[24]:
```

	Station code	Address	Latitude	Longitude	SO2	NO2	O3	CO	PM10	PM2.5	date	time
200801	108	571, Gwangnaru-ro, Gwangjin-gu, Seoul, Republi...	37.54718	127.092493	0.004	0.029	0.027	0.6	31	24	2019-04-03	0:00
200802	108	571, Gwangnaru-ro, Gwangjin-gu, Seoul, Republi...	37.54718	127.092493	0.004	0.026	0.029	0.6	31	18	2019-04-03	1:00
200803	108	571, Gwangnaru-ro, Gwangjin-gu, Seoul, Republi...	37.54718	127.092493	0.004	0.021	0.035	0.6	26	18	2019-04-03	2:00
200804	108	571, Gwangnaru-ro, Gwangjin-gu, Seoul, Republi...	37.54718	127.092493	0.004	0.025	0.028	0.6	28	21	2019-04-03	3:00
200805	108	571, Gwangnaru-ro, Gwangjin-gu, Seoul, Republi...	37.54718	127.092493	0.004	0.043	0.004	0.7	34	24	2019-04-03	4:00

```
[25]:
df_add = df_add.loc[:,['SO2', 'NO2', 'O3', 'CO', 'PM10', 'PM2.5', 'time']]
df_add.head()
## 원하는 컬럼들로만 데이터프레임 다시 만들기
```

```
[25]:
```

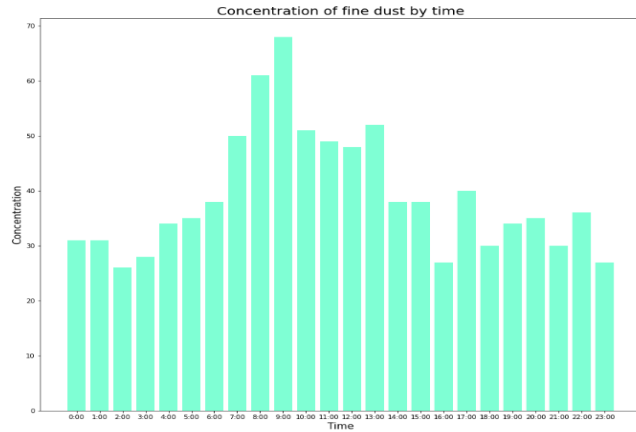
	SO2	NO2	O3	CO	PM10	PM2.5	time
200801	0.004	0.029	0.027	0.6	31	24	0:00
200802	0.004	0.026	0.029	0.6	31	18	1:00
200803	0.004	0.021	0.035	0.6	26	18	2:00
200804	0.004	0.025	0.028	0.6	28	21	3:00
200805	0.004	0.043	0.004	0.7	34	24	4:00

-> 원하는 컬럼들로만 df를 다시 만든다.
(SO2, NO2, O3, CO, PM10, PM2.5, time)

5.2 데이터 시각화(시간별 미세먼지 농도)

```
[27]: plt.figure(figsize = (15,13))
plt.bar(X_sj,V_sj,color = 'aquamarine')
plt.title('Concentration of fine dust by time',fontsize = 20)
plt.xlabel('Time',fontsize=15)
plt.ylabel('Concentration',fontsize = 15)
plt.show()
```

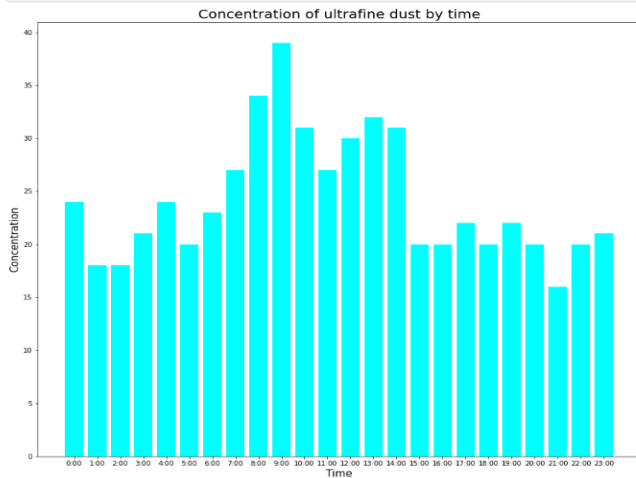
세종대 주변 시간별 미세먼지 그래프



-> 시간별 미세먼지 농도

```
[28]: plt.figure(figsize = (15,13))
plt.bar(X_sj,V2_sj,color = 'cyan')
plt.title('Concentration of ultrafine dust by time',fontsize = 20)
plt.xlabel('Time',fontsize=15)
plt.ylabel('Concentration',fontsize = 15)
plt.show()
```

세종대 주변 시간별 초미세먼지 시간별 그래프



-> 시간별 초미세먼지 농도

5.2 데이터 시각화(지역별 미세먼지 농도)

[28]:

```
#지역별 미세먼지 농도
'''SO2 배출이 높은 정보 10개 출력'''
SO2 = df.sort_values(by = ['SO2'], ascending=False)
SO2.head(10)
```

[28]:

	Station code	Address	Latitude	Longitude	SO2	NO2	O3	CO	PM10	PM2.5	hour
424709	117	45 Gamasan-ro 27-gil Guro-gu Seoul Republi...	37.498498	126.889692	3.736	38.445	12.455	0.4	35	17	9
424686	117	45 Gamasan-ro 27-gil Guro-gu Seoul Republi...	37.498498	126.889692	2.700	20.100	33.600	0.3	8	1	10
424685	117	45 Gamasan-ro 27-gil Guro-gu Seoul Republi...	37.498498	126.889692	2.700	30.700	23.400	0.4	5	6	9
424710	117	45 Gamasan-ro 27-gil Guro-gu Seoul Republi...	37.498498	126.889692	1.330	12.805	6.320	0.5	34	15	10
15589	101	19 Jong-ro 35ga-gil Jongno-gu Seoul Republ...	37.572016	127.005008	0.406	0.044	0.003	40.0	22	12	13
644605	125	59 Gucheonmyeon-ro 42-gil Gangdong-gu Seoul...	37.544982	127.136792	0.378	-1.000	0.002	36.7	14	1	14
15565	101	19 Jong-ro 35ga-gil Jongno-gu Seoul Republ...	37.572016	127.005008	0.372	0.030	0.030	38.4	15	4	13
589026	123	426 Hakdong-ro Gangnam-gu Seoul Republic o...	37.517528	127.047470	0.370	0.002	0.013	7.1	48	30	11
537080	121	14 Sillindong-gil Gwanak-gu Seoul Republic...	37.487355	126.927102	0.365	0.001	0.012	36.8	57	42	13
589001	123	426 Hakdong-ro Gangnam-gu Seoul Republic o...	37.517528	127.047470	0.350	0.005	0.008	5.2	36	27	10

[30]:

```
SO2_Address = df.groupby('Address').agg({'SO2' : 'median'}).sort_values('SO2',ascending=False).reset_index()
# Address를 기준으로 그룹화하여 SO2 값의 중앙값을 출력
# reset_index -> 인덱스 리셋(단순한 경우 인덱스로 세팅)
print(SO2_Address)
```

```
Address SO2
0 369, Yongmasan-ro, Jungnang-gu, Seoul, Republi... 0.006
1 71, Gangseo-ro 45da-gil, Gangseo-gu, Seoul, Re... 0.005
2 45, Gamasan-ro 27-gil, Guro-gu, Seoul, Republi... 0.005
3 43, Cheonho-daero 13-gil, Dongdaemun-gu, Seoul... 0.005
4 426, Hakdong-ro, Gangnam-gu, Seoul, Republic o... 0.005
5 226, Beekjeogun-ro, Songpa-gu, Seoul, Republi... 0.004
6 59, Gucheonmyeon-ro 42-gil, Gangdong-gu, Seoul... 0.004
7 571, Gwangnaru-ro, Gwangjin-gu, Seoul, Republi... 0.004
8 56, Jungang-ro 52-gil, Yangcheon-gu, Seoul, Re... 0.004
9 34, Sirubong-ro 2-gil, Dobong-gu, Seoul, Repub... 0.004
10 11, Yangsan-ro 23-gil, Yeongdeungpo-gu, Seoul... 0.004
11 10, Poeun-ro 6-gil, Mapo-gu, Seoul, Republic o... 0.004
12 215, Jinheung-ro, Eunpyeong-gu, Seoul, Republi... 0.004
13 20, Geunha-ro 21-gil, Geuncheon-gu, Seoul, Rep... 0.004
14 19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Repub... 0.004
15 18, Itukseom-ro 3-gil, Seongdong-gu, Seoul, Re... 0.004
16 17, Sanggye-ro 23-gil, Nowon-gu, Seoul, Repub... 0.004
17 16, Sinbanpo-ro 15-gil, Seocho-gu, Seoul, Repu... 0.004
18 14, Sillindong-gil, Gwanak-gu, Seoul, Republic... 0.004
19 32, Segeonjeong-ro 4-gil, Seodaemun-gu, Seoul... 0.004
20 49, Sanyang-ro 139-gil, Gangbuk-gu, Seoul, Rep... 0.003
21 15, Deoksugung-gil, Jung-gu, Seoul, Republic o... 0.003
22 136, Hannam-daero, Yongsan-gu, Seoul, Republic... 0.003
23 6, Sadang-ro 16a-gil, Dongjak-gu, Seoul, Repub... 0.003
24 70, Sanyang-ro 2-gil, Seongbuk-gu, Seoul, Repu... 0.003
```

[33]:

```
# 상위 10개 데이터만 저장
SO2 = SO2_Address.sort_values('SO2',ascending=False).head(10)
```

위와 같은 방법으로 NO2, O3, CO, PM10, PM2.5도 상위 10개 데이터를 저장해준다.

5.2 데이터 시각화(지역별 미세먼지 농도)

[48]:

```
plt.figure(figsize=(12,35))

plt.subplot(6,1,1)
sns.barplot(y="Address", x="SO2", data = SO2_Address.head(10))

plt.subplot(6,1,2)
sns.barplot(y="Address", x="NO2", data = NO2_Address.head(10))

plt.subplot(6,1,3)
sns.barplot(y="Address", x="O3", data = O3_Address.head(10))

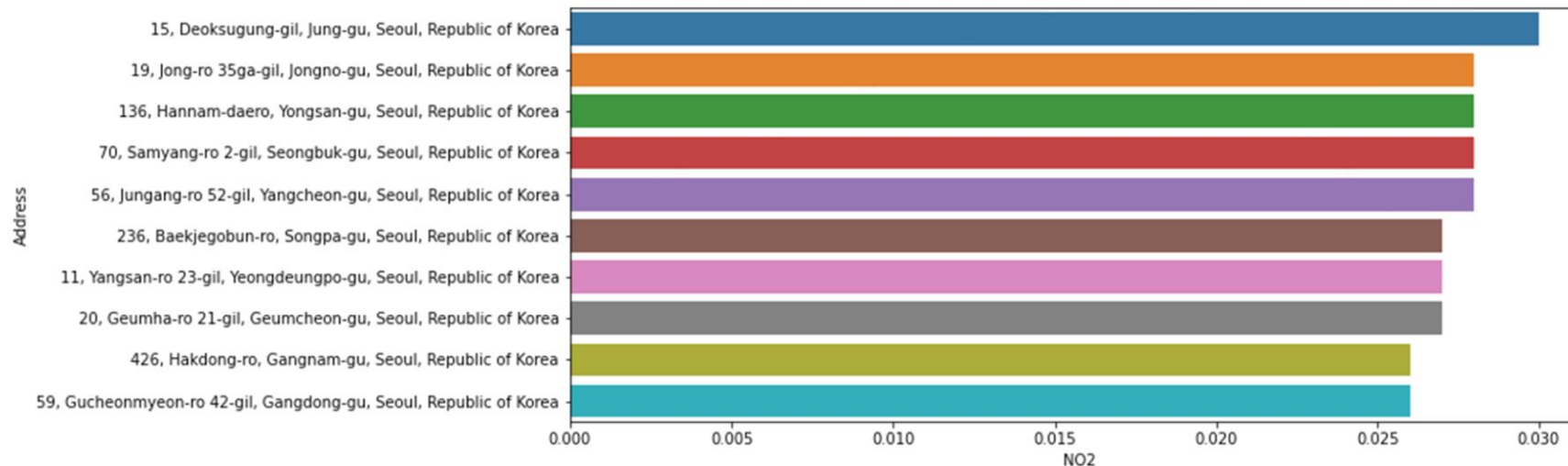
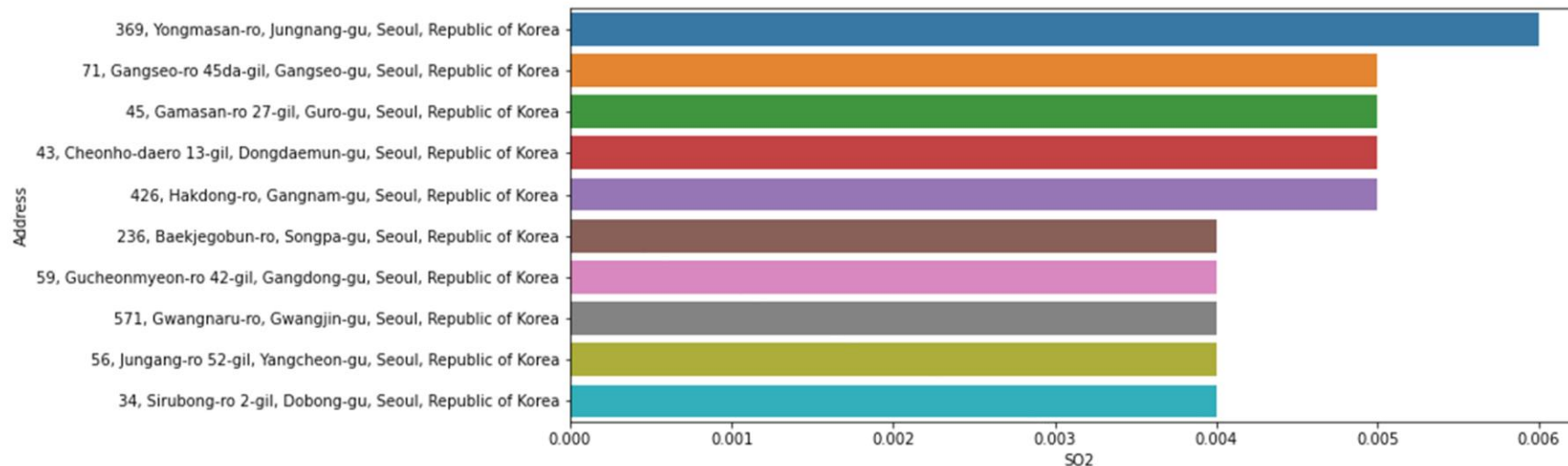
plt.subplot(6,1,4)
sns.barplot(y="Address", x="CO", data = CO_Address.head(10))

plt.subplot(6,1,5)
sns.barplot(y="Address", x="PM10", data = PM10_Address.head(10))

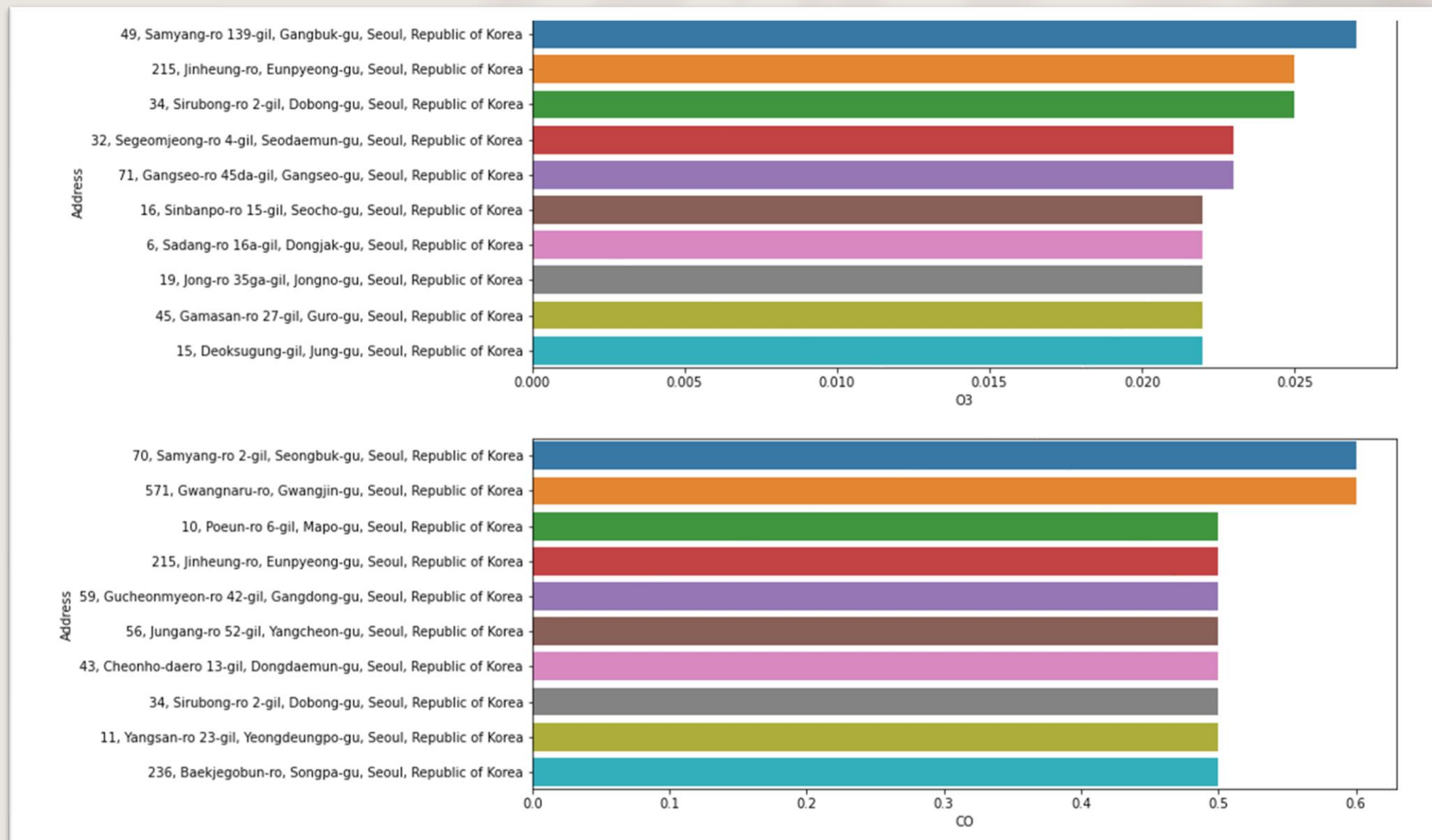
plt.subplot(6,1,6)
sns.barplot(y="Address", x="PM2.5", data = PM2_5_Address.head(10))
```

5.2 데이터 시각화(지역별 미세먼지 농도)

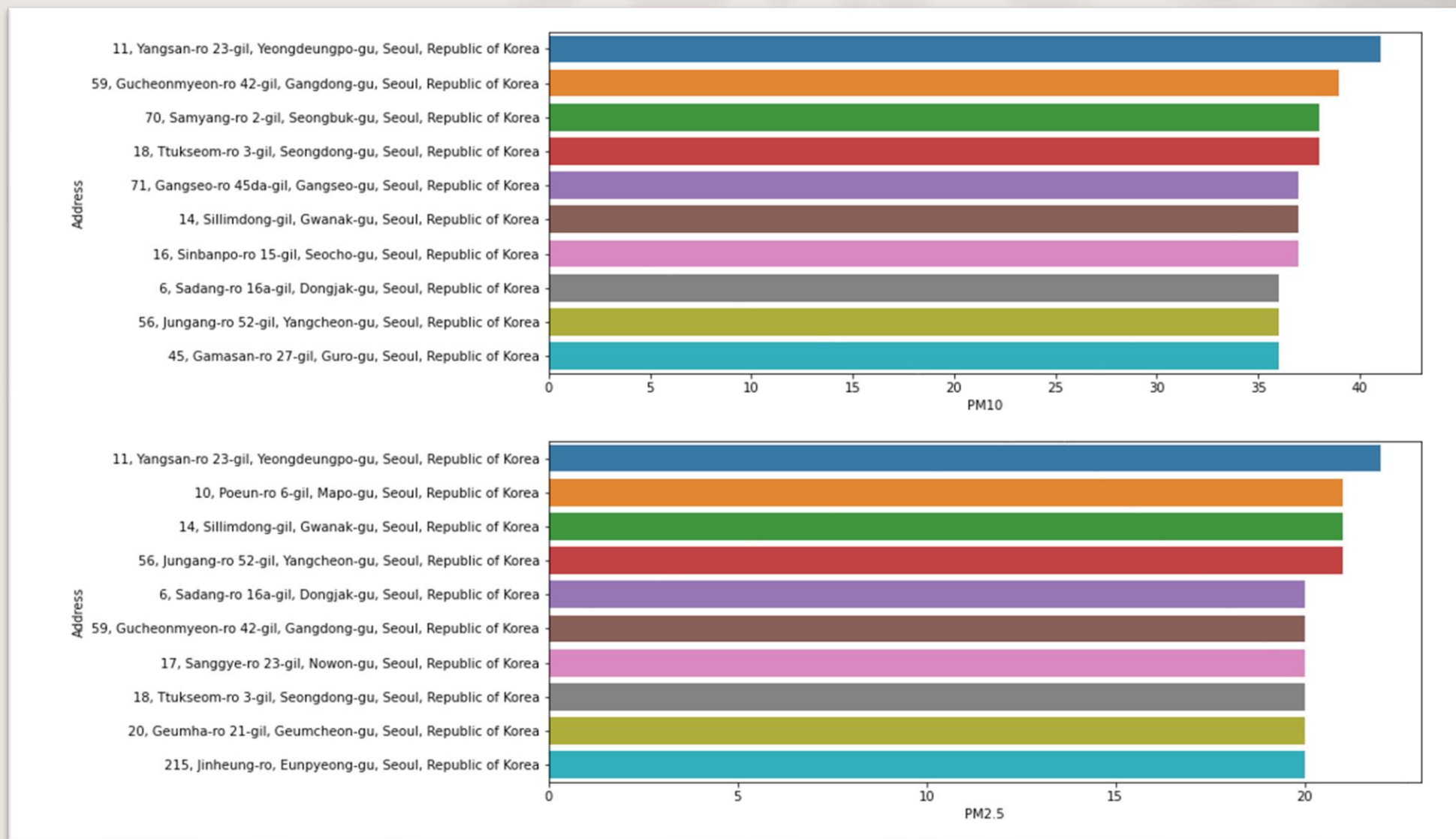
[48]: <AxesSubplot:xlabel='PM2.5', ylabel='Address'>



5.2 데이터 시각화(지역별 미세먼지 농도)



5.2 데이터 시각화(지역별 미세먼지 농도)



5.3 데이터 상관관계

```
[53]: df_air = df_0.corr()  
df_air
```

```
[53]:
```

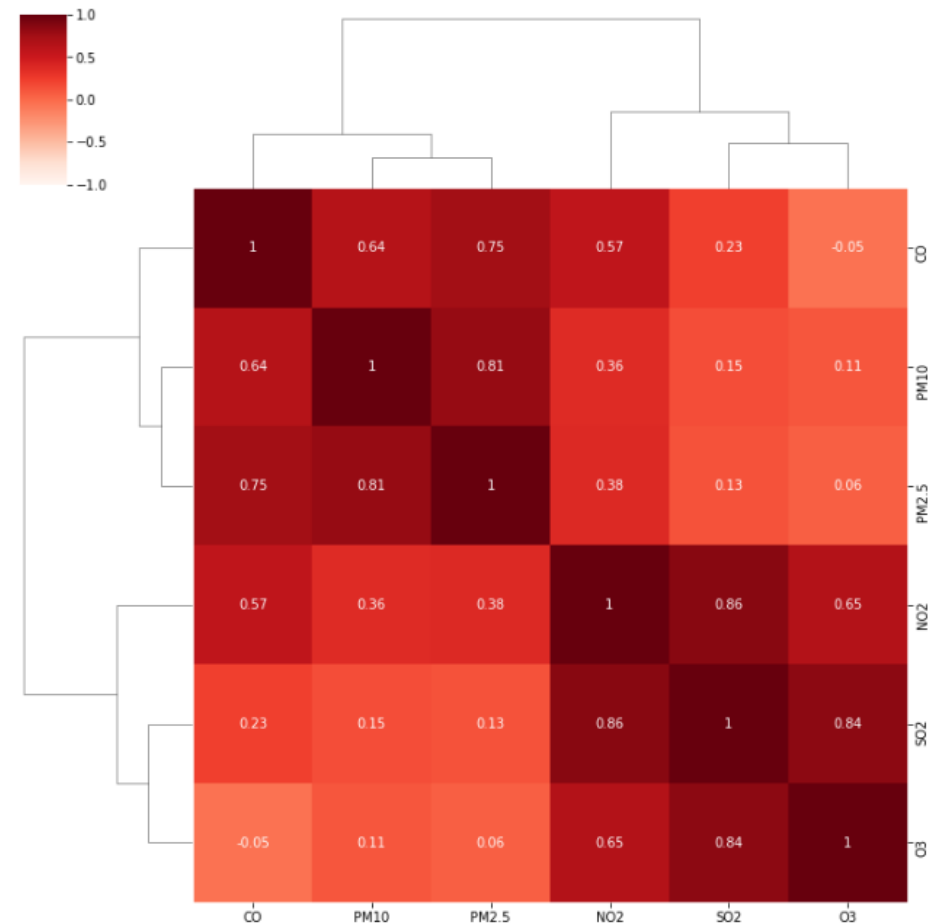
	SO2	NO2	O3	CO	PM10	PM2.5
SO2	1.000000	0.861220	0.844340	0.233561	0.151871	0.125292
NO2	0.861220	1.000000	0.648878	0.573746	0.363326	0.382629
O3	0.844340	0.648878	1.000000	-0.050166	0.106191	0.059617
CO	0.233561	0.573746	-0.050166	1.000000	0.641362	0.752211
PM10	0.151871	0.363326	0.106191	0.641362	1.000000	0.807827
PM2.5	0.125292	0.382629	0.059617	0.752211	0.807827	1.000000

PM10(미세먼지) 농도가 높을 때,
PM2.5(초미세먼지)의 농도도 높은
상관관계를 가진다는 것을 알 수 있다.

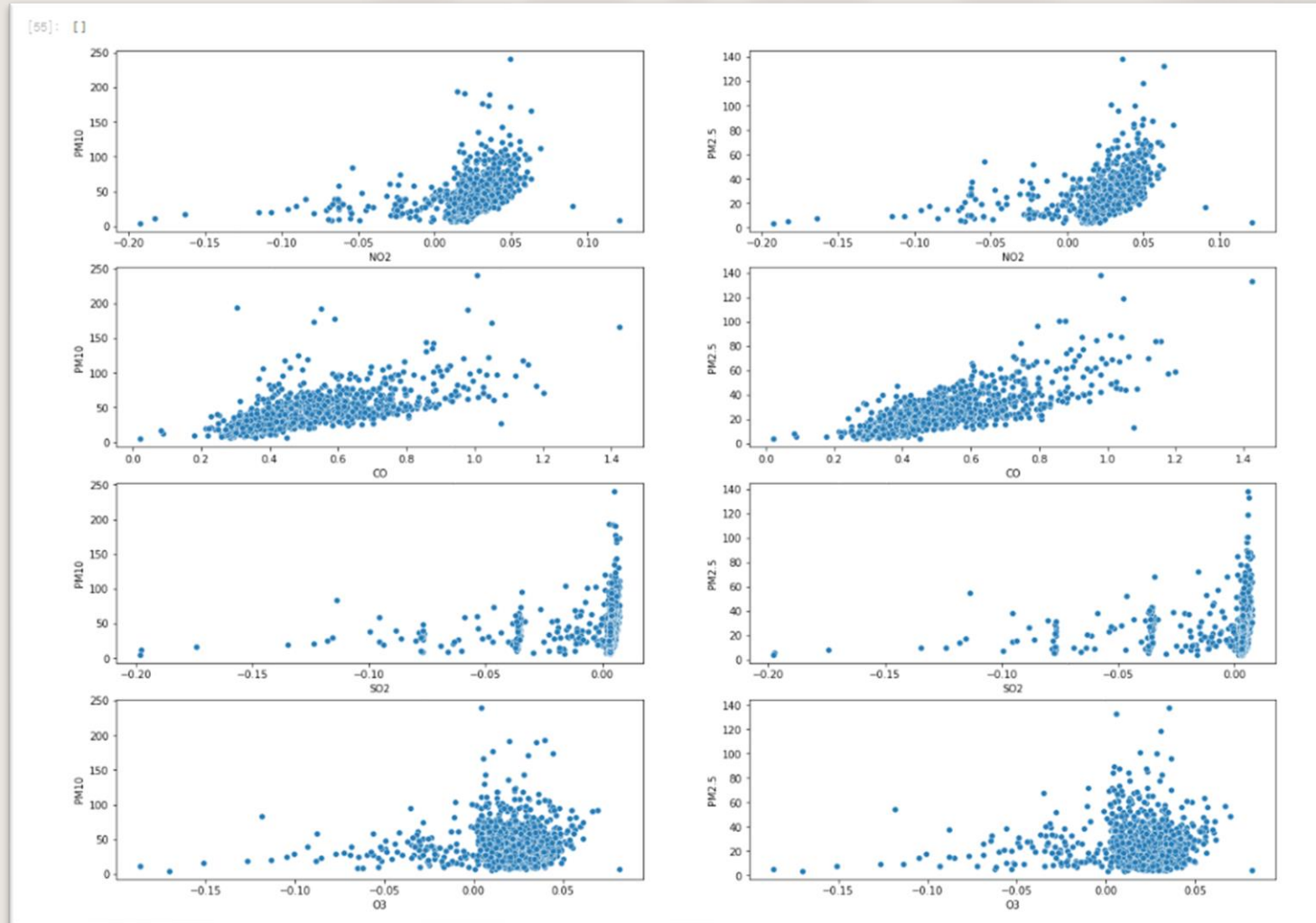


```
[54]: sns.clustermap(df_air,  
                    annot = True,  
                    cmap = 'Reds',  
                    vmin = -1, vmax = 1  
                    )
```

```
[54]: <seaborn.matrix.ClusterGrid at 0x7f540a0e3390>
```

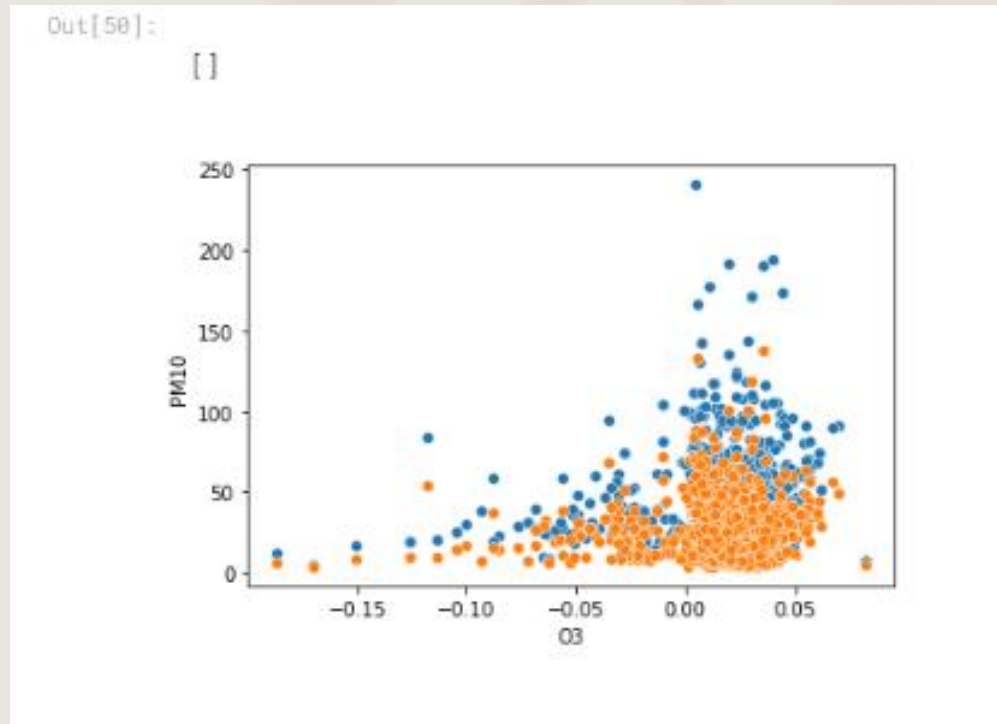


6. 테스트 결과



산점도를 통해 NO₂, CO의 값이 미세먼지, 초미세먼지의 농도와 상관관계가 높음을 확인할 수 있다. 따라서 대기오염은 NO₂, CO의 영향을 받고, 그 중에서 CO는 대기오염에 많은 영향을 끼치는 것을 알 수 있다.

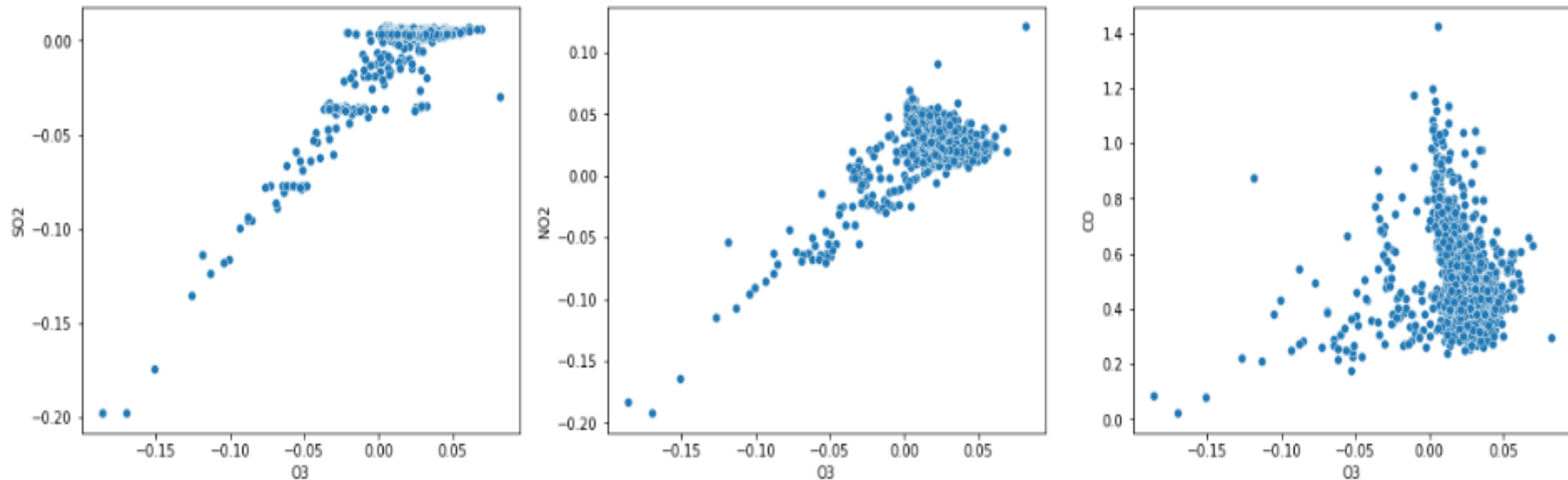
6. 테스트 결과



O3는 농도에 따른 대기오염의 영향이 적은 것으로 보여진다.

6. 테스트 결과

[57]: []



O3는 SO2, NO2와 높은 상관관계를 보인다.
즉, O3는 대기오염에 직접적인 영향을 주는 것처럼 보이지만, SO2, NO2에 영향을 주어 간접적으로 대기오염에 영향을 미치는 것으로 판단된다.

8. 소감

환경 오염 중에서도 대기 오염은 현대 사회에서도 여러 매체에서 언급되고 있습니다. 전에는 특정 물질들이 대기 오염에 무조건 영향을 끼친다고 생각했지만 이번 과제를 진행하며 데이터도 분석해보고 각 상관관계를 통해 모든 물질들이 대기 오염에 직접적인 영향을 끼치지 않는다는 점을 알게 되었습니다.

이번 과제에서는 서울 일부 지역으로만 한정지어서 데이터를 분석한 점이 아쉬웠습니다. 전처리나 데이터를 분석하고 학습할 때 다양한 모델들에 대한 지식을 좀 더 쌓아서 나중에는 더 넓은 지역에 데이터들을 분석을 해봐야겠다고 느끼게 되었습니다.

감사합니다.