[EDA] practice by air polution of seoul data

"데이터 과학의 80%는 데이터 클리닝에 소비되고, 나머지 20%는 데이터 클리닝하는 시간을 불평하는데 쓰인다."
- Kaggle 창립자

그만큼 데이터 전처리가 중요하다는 의미죠!!

0. Before EDA

- Learning Pandas library
- knowing DataStructure
- Learning Visualization

is very helpful your studying ^^

1. Library & Data Load

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
df = pd.read_csv('/content/drive/MyDrive/한국분석/air_pollution_in_seoul/AirPollutionSeoul/Measurement_summary.csv')
```

2. Data

2.1 Data 구조

[참고] Pandas

- Series & DataFrame
- loc & iloc
- groupby & sortby
- concat
- drop

```
# find row, column
df.shape
```

```
(647511, 11)
```

```
# look DataFrame
df.head()
```

	Measurement date	Station code	Address	Latitude	Longitude	SO2	NO2	O3	CO	PM10	PM2.5
0	2017-01-01 00:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.059	0.002	1.2	73.0	57.0
1	2017-01-01 01:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ...	37.572016	127.005008	0.004	0.058	0.002	1.2	71.0	59.0
2	2017-01-01 02:00	101	19, Jong-ro 35ga-gil, Jongno-gu, Seoul, Republ	37.572016	127.005008	0.004	0.056	0.002	1.2	70.0	59.0

```
# look detail
df.describe()
```

	Station code	Latitude	Longitude	SO2	NO2	O3	CO	PM10	
count	647511.000000	647511.000000	647511.000000	647511.000000	647511.000000	647511.000000	647511.000000	647511.000000	647511.0
mean	113.000221	37.553484	126.989340	-0.001795	0.022519	0.017979	0.509197	43.708051	25.4
std	7.211315	0.053273	0.078790	0.078832	0.115153	0.099308	0.405319	71.137342	43.9
min	101.000000	37.452357	126.835151	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.0
25%	107.000000	37.517528	126.927102	0.003000	0.016000	0.008000	0.300000	22.000000	11.0
50%	113.000000	37.544962	127.004850	0.004000	0.025000	0.021000	0.500000	35.000000	19.0
75%	119.000000	37.584848	127.047470	0.005000	0.038000	0.034000	0.600000	53.000000	31.0
max	125.000000	37.658774	127.136792	3.736000	38.445000	33.600000	71.700000	3586.000000	6256.0

```
# dytype
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  float64
 10  PM2.5             647511 non-null  float64
dtypes: float64(8), int64(1), object(2)
memory usage: 54.3+ MB
```

```
# find NA, NULL, NAN
df.isnull().sum()
```

	0
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

2.2 column

- Latitude & Longitude
- S02(이산화 황)
- N02(이산화 질소)
- 03(오존)
- CO(일산화 탄소)
- PM10(미세먼지)
- PM2.5(초 미세먼지)

```
# 각 column별 value
df.nunique()

# station code, Adress, Latitude, Longtitude 가 25개의 다른 지역에서 조사함을 보여줌
```

	0
Measurement date	25906
Station code	25
Address	25
Latitude	25
Longitude	25
SO2	186
NO2	132
O3	253
CO	172
PM10	551
PM2.5	333

dtype: int64

3. Extracting Data

- 3.1 시간별 미세먼지 농도 변화 확인
- 3.2 미세먼지에 영향을 미치는 변인 확인
- 3.3 관측소 위치 Longtitude & Latitude

3.1 시간별 미세먼지 농도 변화 확인

```
from datetime import datetime

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

df.hour.head(24)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3790         try:
-> 3791             return self._engine.get_loc(casted_key)
   3792         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Measurement date'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                          2 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3796             ):
   3797                 raise InvalidIndexError(key)
-> 3798             raise KeyError(key) from err
   3799         except TypeError:
   3800             # If we have a listlike key, _check_indexing_error will raise

KeyError: 'Measurement date'
```

다음 단계: 오류 설명

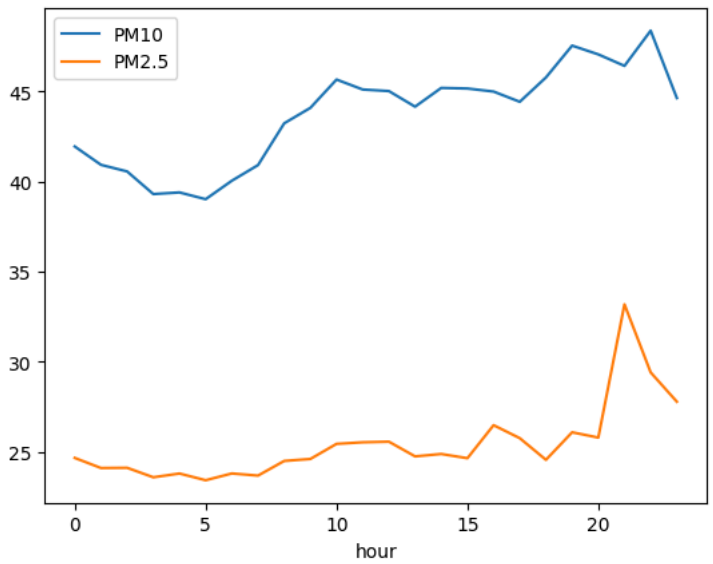
```
data = df.groupby('hour', as_index=False).agg({'SO2':'mean', 'NO2':'mean', 'O3':'mean', 'CO':'mean', 'PM10':'mean', 'PM2.5':'mean'})
data
```

	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

다음 단계: data변수로 코드 생성 | 추천 차트 보기 | New interactive sheet

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

<Axes: xlabel='hour'>



출근시간 9:00, 퇴근시간 18:00 부터 증가

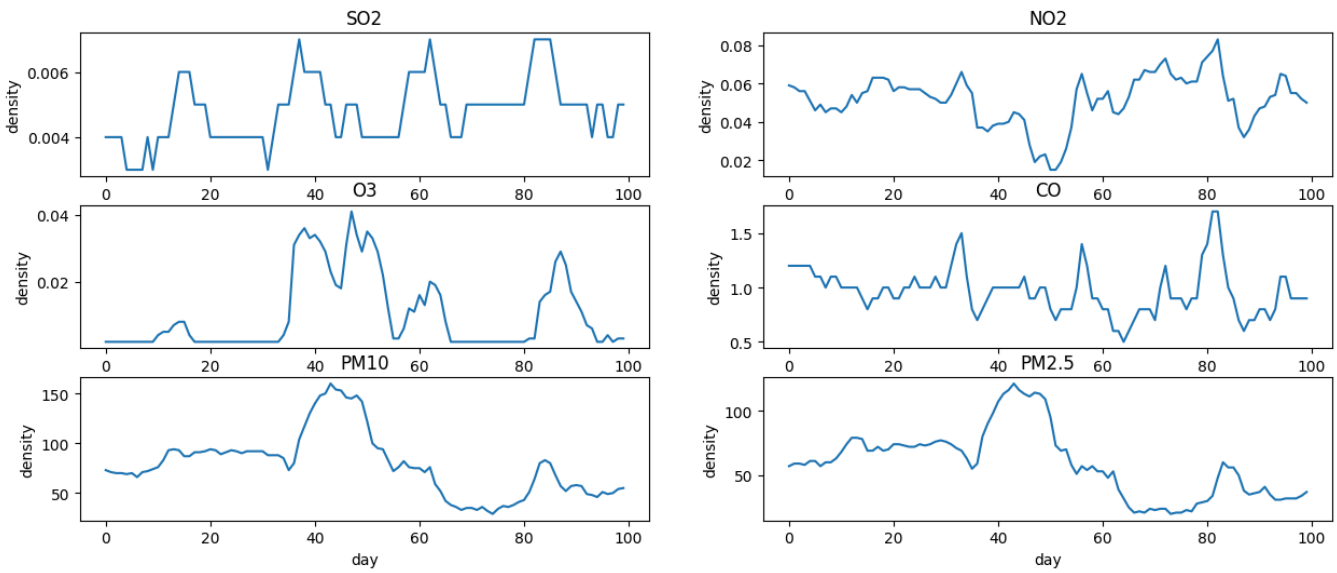
- 요인은 미세먼지를 수집하는 장치의 위치에 따라 다름!

3.2 미세먼지에 영향을 미치는 변인 확인

```
# 데이터 시각화
plt.figure(figsize=(15, 10))

for i in range(4, 10):
    y = df.iloc[:, i]
    plt.subplot(5, 2, i-3)
    plt.title(y.name)
    plt.xlabel('day')
    plt.ylabel('density')
    plt.plot(y[:100])

# O3, !NO2 와 관계가 있음!
```



3.3 관측소 위치 Longtitude & Latitude

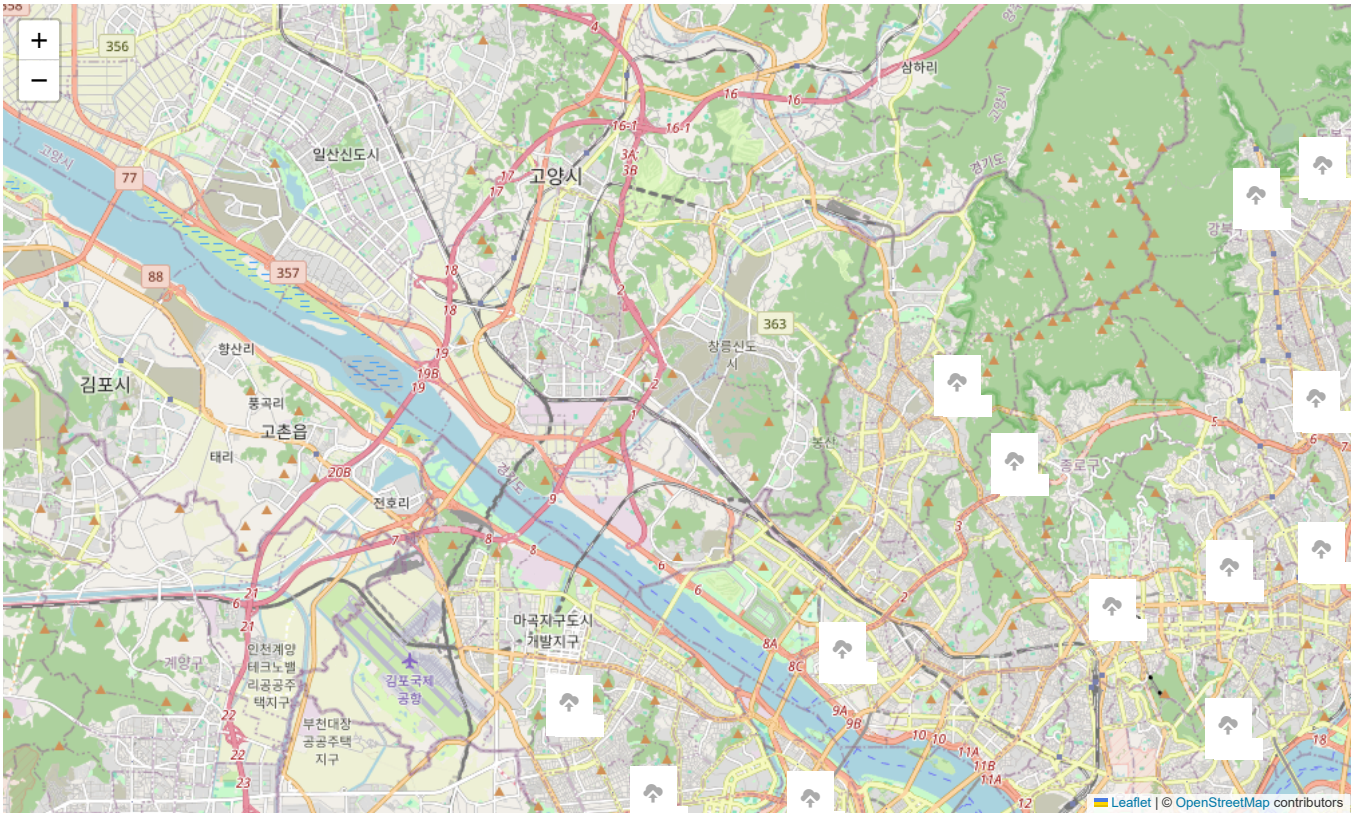
```
# 위도 경도 DataFrame
location = df.groupby('Station code')['PM10'].agg([np.mean])
location['Latitude'] = df['Latitude'].unique() # 절대 이렇게 코드짜면 안되요!
location['Longitude'] = df['Longitude'].unique()
location.head()
```

```
<ipython-input-17-72f461a0b840>:2: FutureWarning: The provided callable <function mean at 0x7ea1ea393d90> is currently using SeriesGroupBy.mean.
  location = df.groupby('Station code')['PM10'].agg([np.mean])
```

Station code	mean	Latitude	Longitude
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

다음 단계: location변수로 코드 생성 | 추천 차트 보기 | New interactive sheet

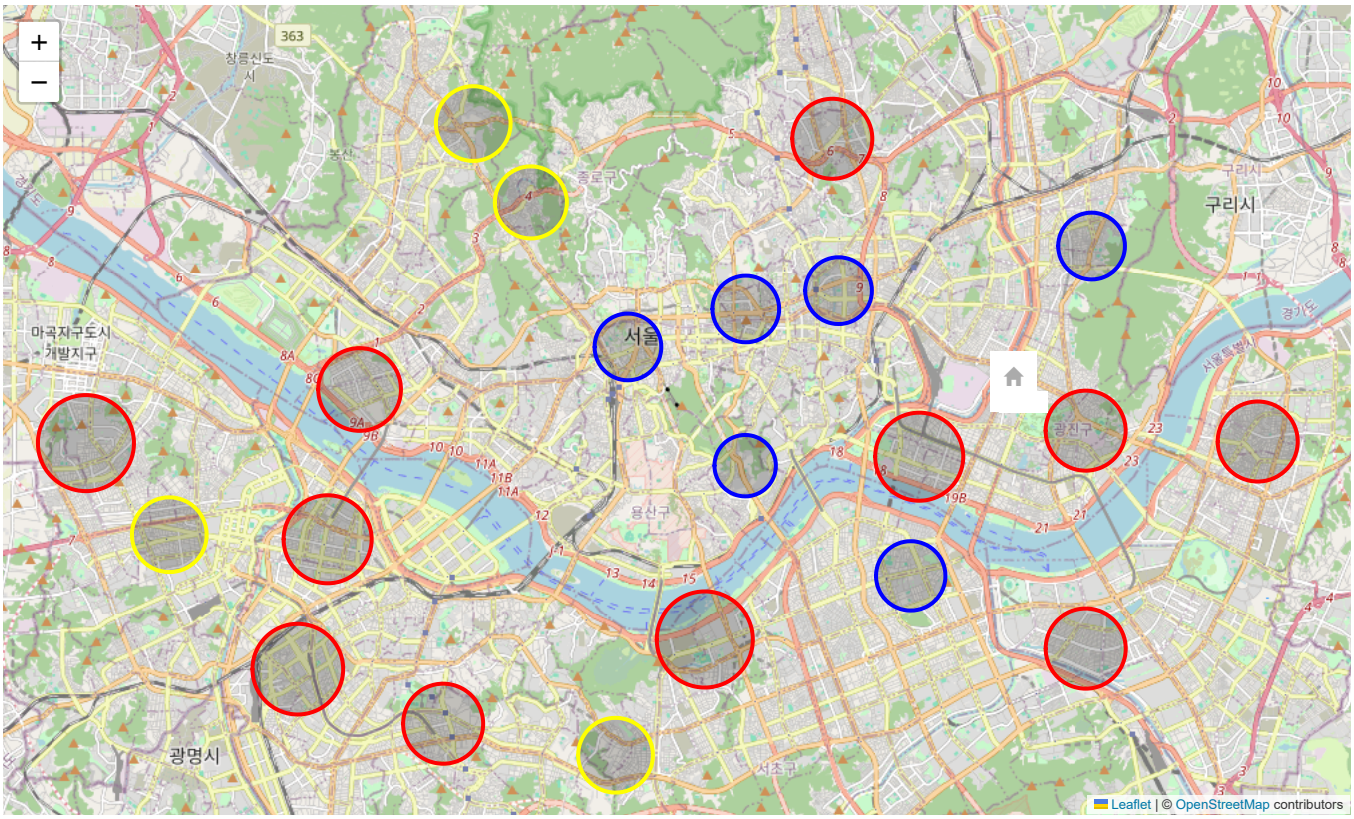
```
import folium
seoul = folium.Map(location=[37.55138077230307, 126.98712254969668], zoom_start=12)
for i in range(len(location)):
    marker = folium.Marker([location.iloc[i,1], location.iloc[i,2]], icon=folium.Icon(popup=str(location.index[i]), color='blue', icon='glyphicon gly
folium.Marker([37.55195608145124, 127.07362532752212], icon=folium.Icon(popup='Sejoing Univ', color='red', icon='glyphicon glyphicon-home')).add_to(s
seoul
```



```
import folium
from folium.plugins import MarkerCluster
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]*20, color=color_select(location.iloc[i,0]),fill_col
# Marker / Sejong Univ.
folium.Marker([37.55195608145124, 127.07362532752212], icon=folium.Icon(popup='Sejoing Univ.', color='red', icon='glyphicon glyphicon-home')).add_to(
seoul
```



코딩을 시작하거나 AI로 코드를 생성하세요.

코딩을 시작하거나 AI로 코드를 생성하세요.