

KT AIVLE School

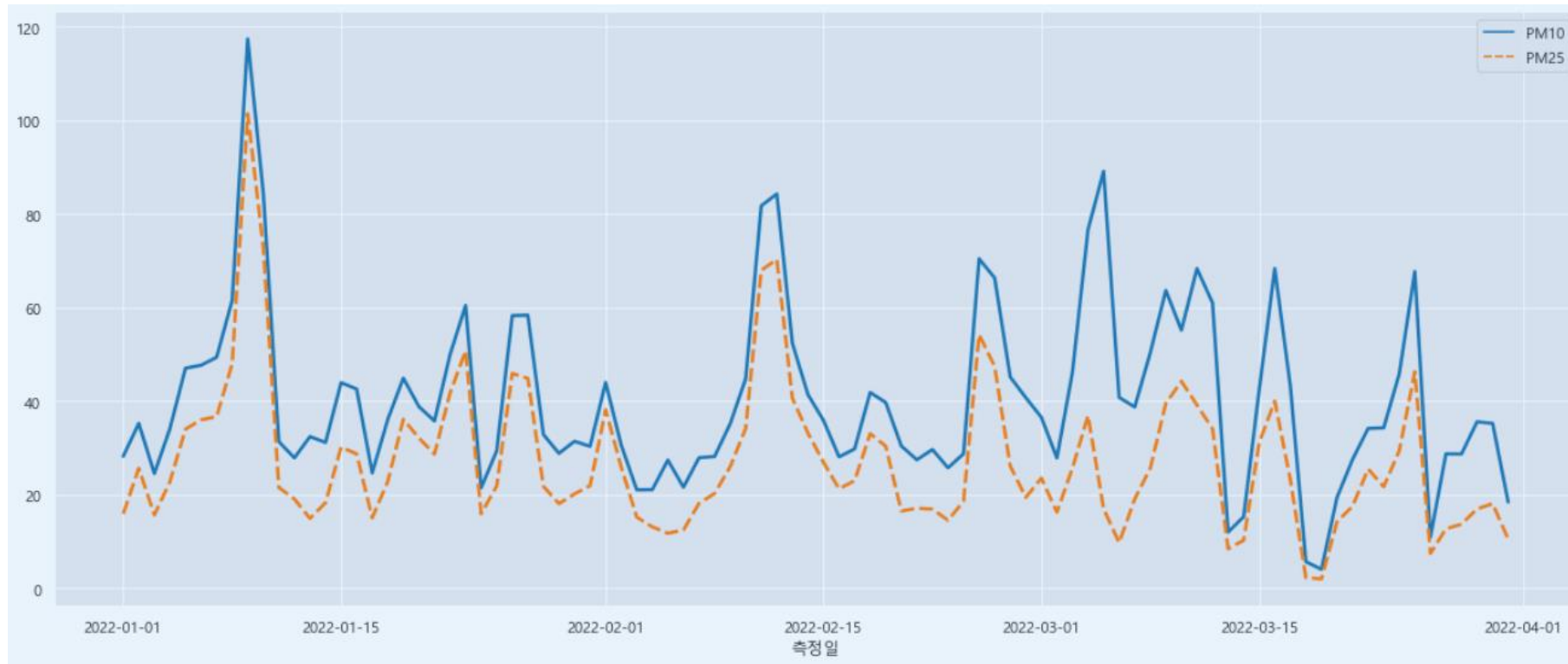
공공데이터 기반 미세먼지 농도 예측

AI 08반 23조

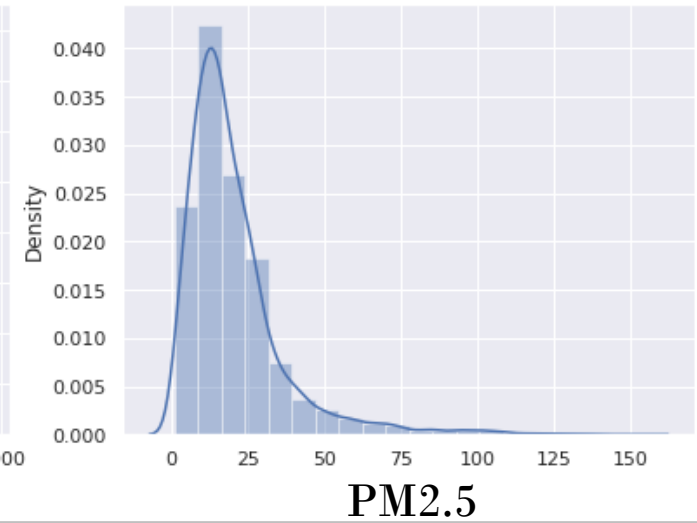
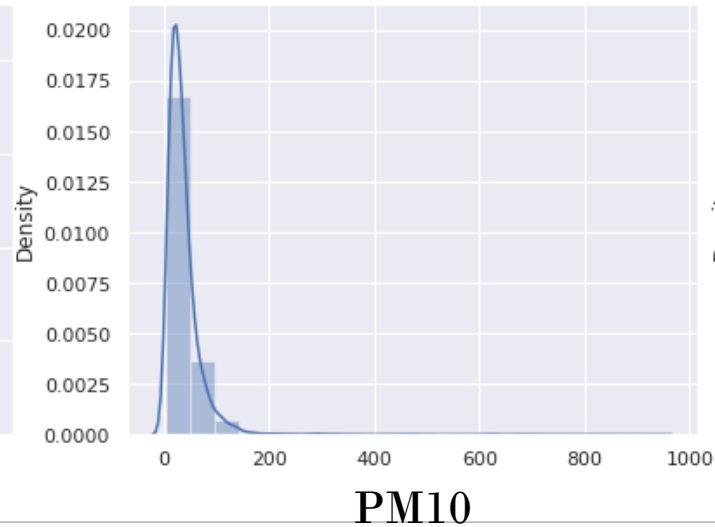
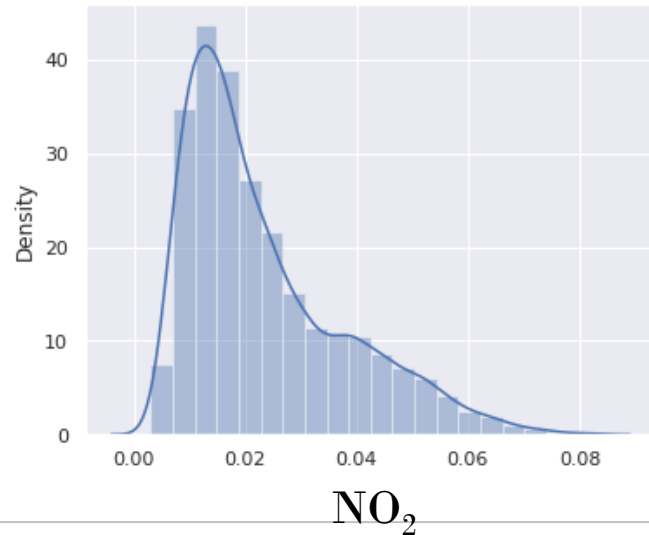
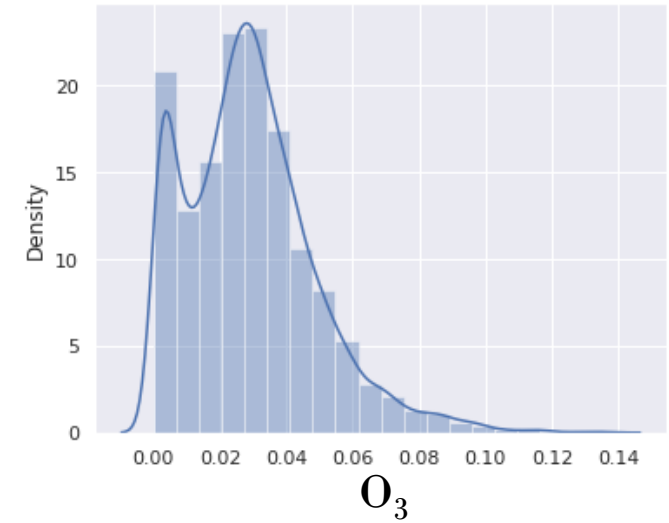
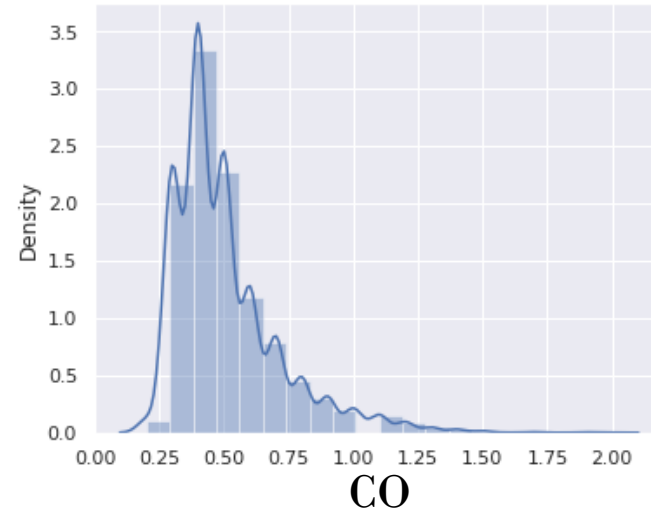
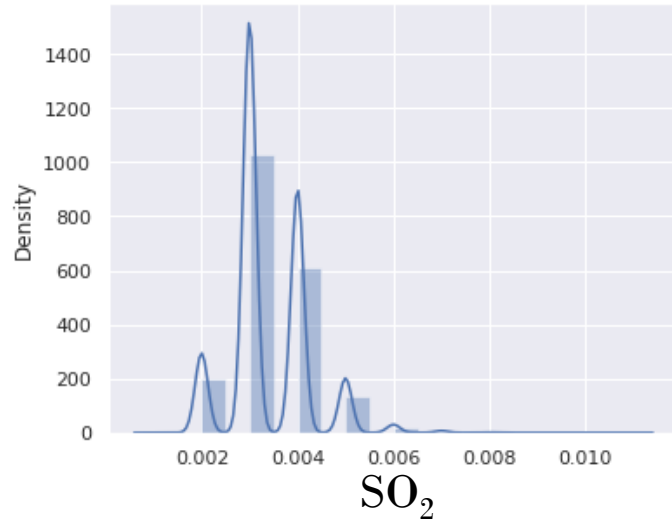
변수 분석

미세먼지, 초미세먼지 변화

✓ 2022년 미세먼지, 초미세먼지 변화



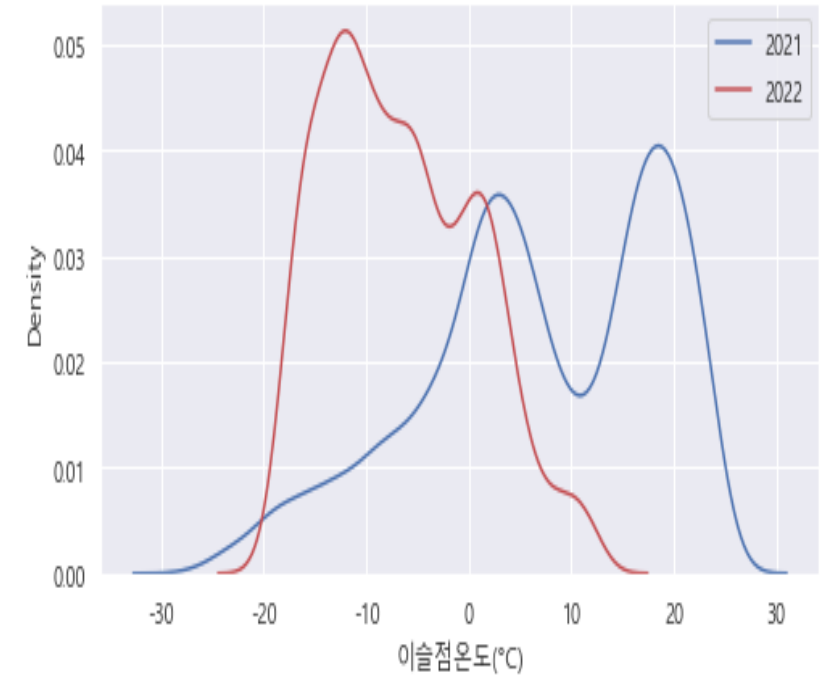
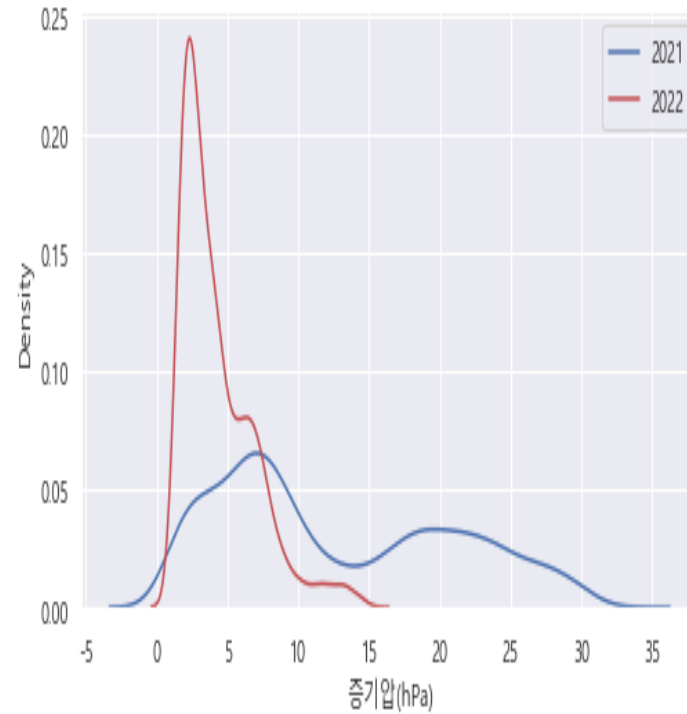
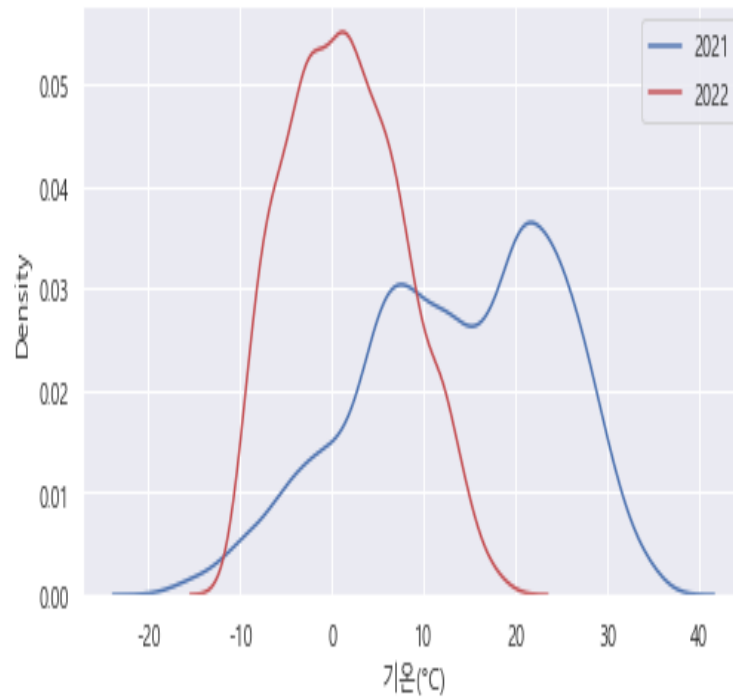
Air 데이터의 분포



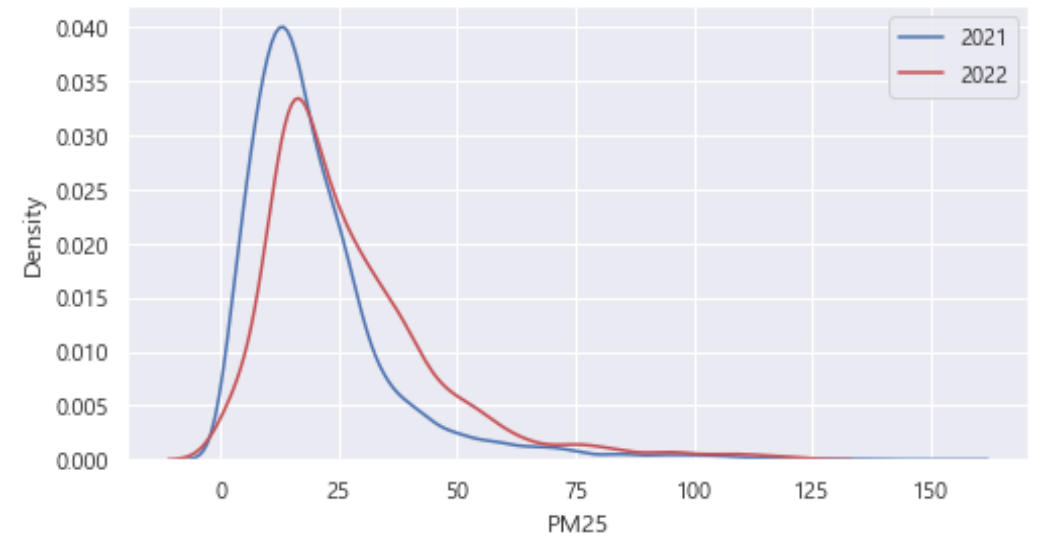
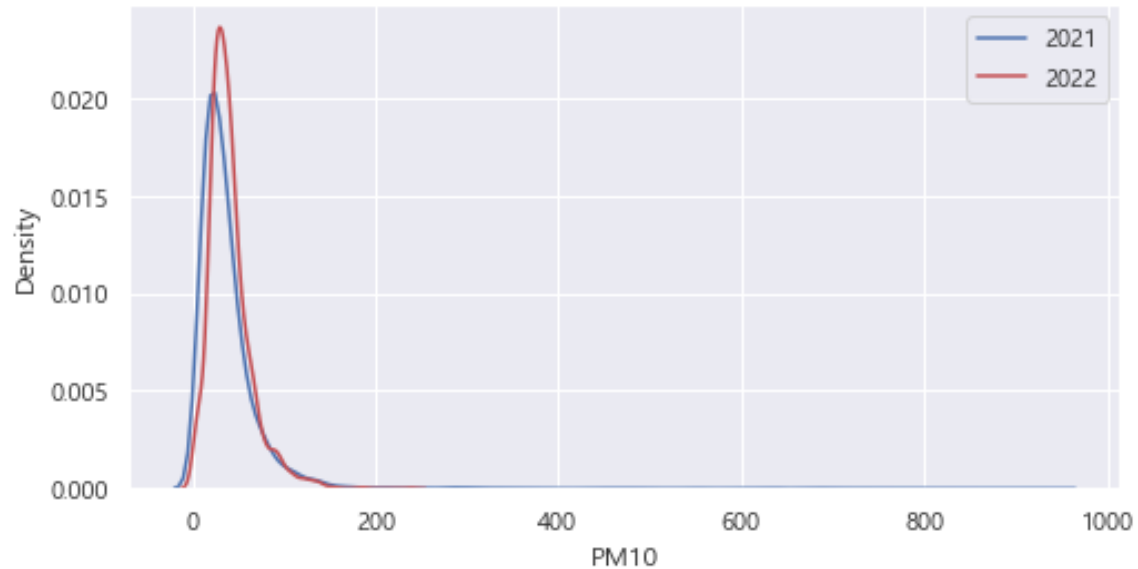
Weather 데이터의 분포 (21, 22년 비교)

2021 data : 2020년 전체 데이터 수집

2022 data : 2022년 1~3월(겨울) 데이터 수집



데이터의 분포



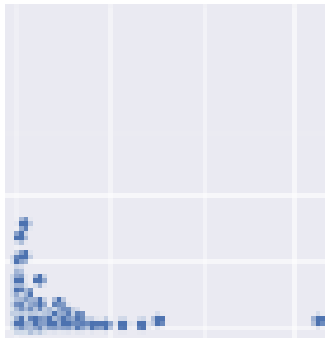
2021, 2022 수집한 데이터의 계절이 다름
온도, 기압에서 분포 차이를 보임
But 미세먼지와 초미세먼지 분포는 큰 차이를 보이지 않음 -> 영향?

상관이 높아 보이는 변수 & 낮아 보이는 변수

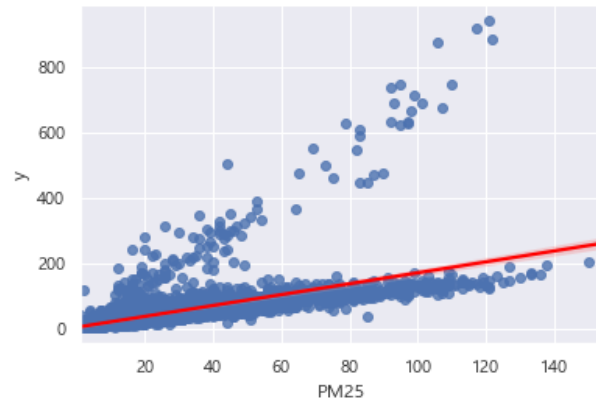
X: features
Y: PM10

상관이 높아 보이는 요소의 분포

강수량



PM2.5

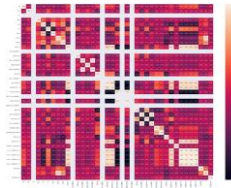


높은 PM10
강수가 없거나 낮은 날에
많이 분포

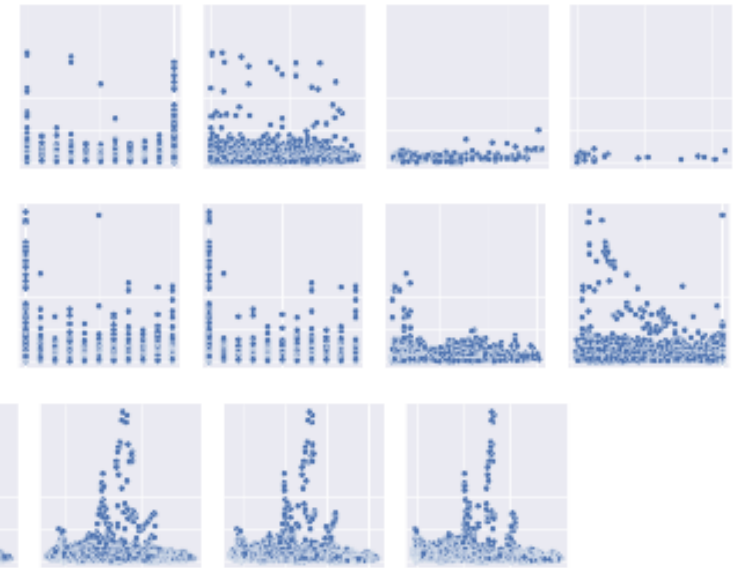
낮은 PM10
강수량 높은 날에 많이 분포

PM2.5(초미세먼지 농도)
→ 강한 양의 상관관계

상관이 낮아 보이는 요소의 분포



Heatmap에서 밝은 색
→ 연관이 낮을 것



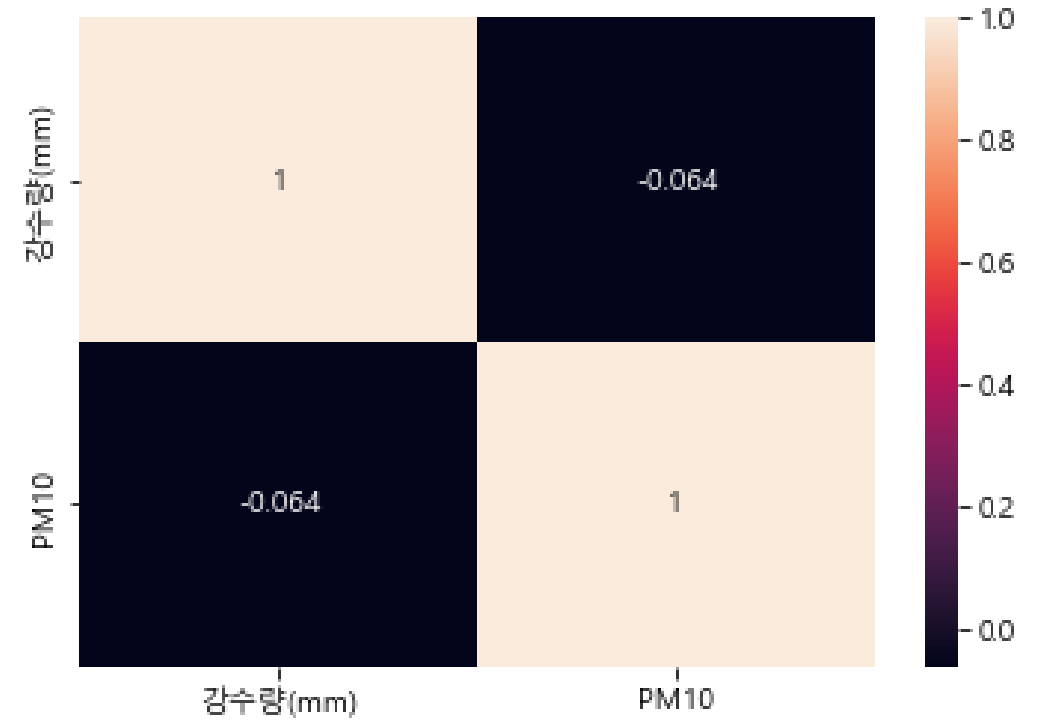
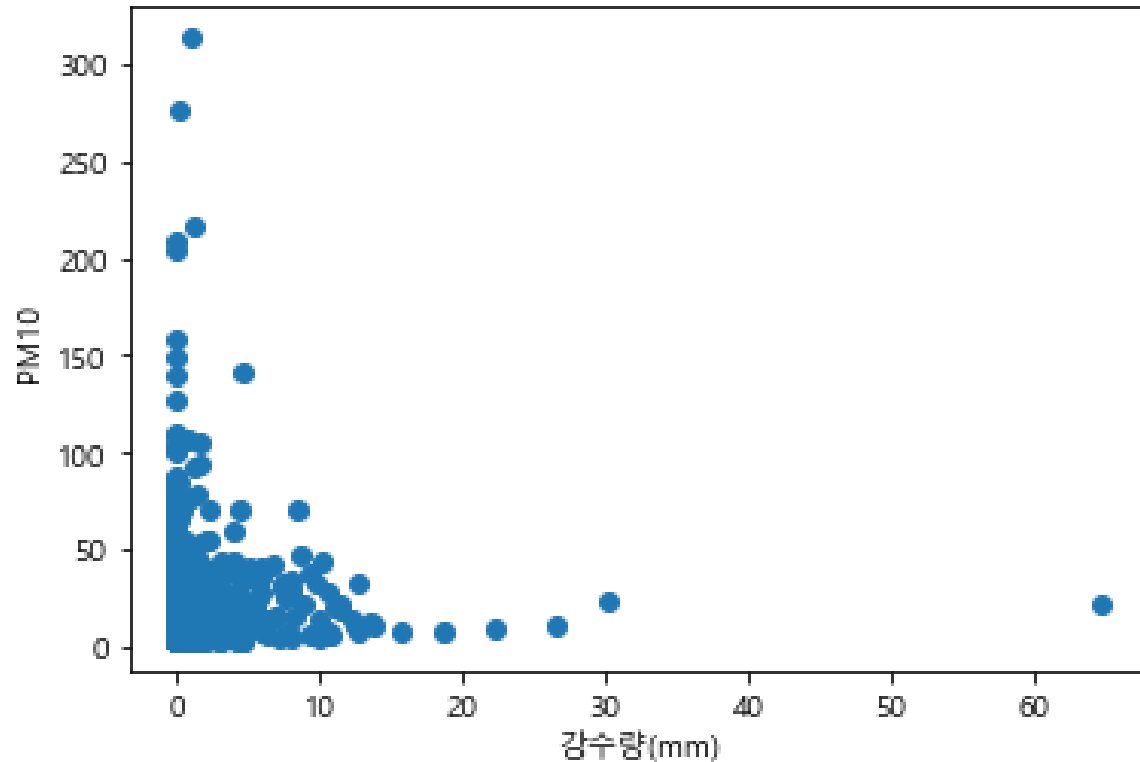
물리화학적 특성으로 인한 원래의 분포와 일치 > 관련이 낮을 것
예) 온도 > 평균 기온을 mean으로 해서 정규 분포를 이룰 것이라고 예상 가능

일조, 기압, 운량, 지중온도, 풍속, 풍향, ...

변수 분석 내용 정리

✓ 변수 1 : 강수량

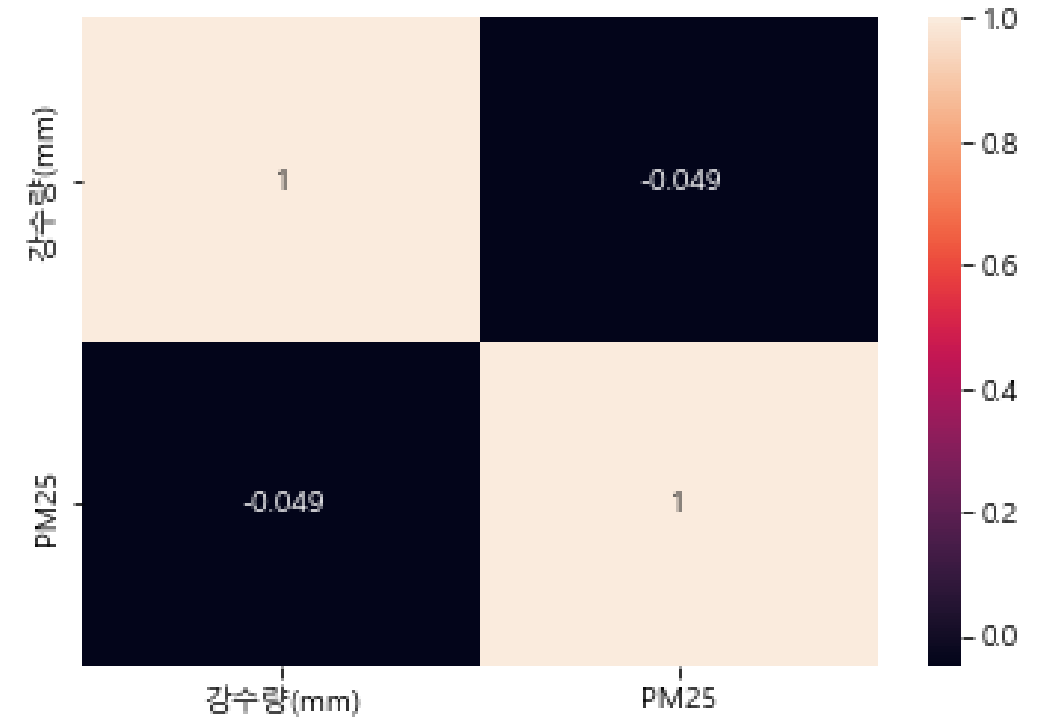
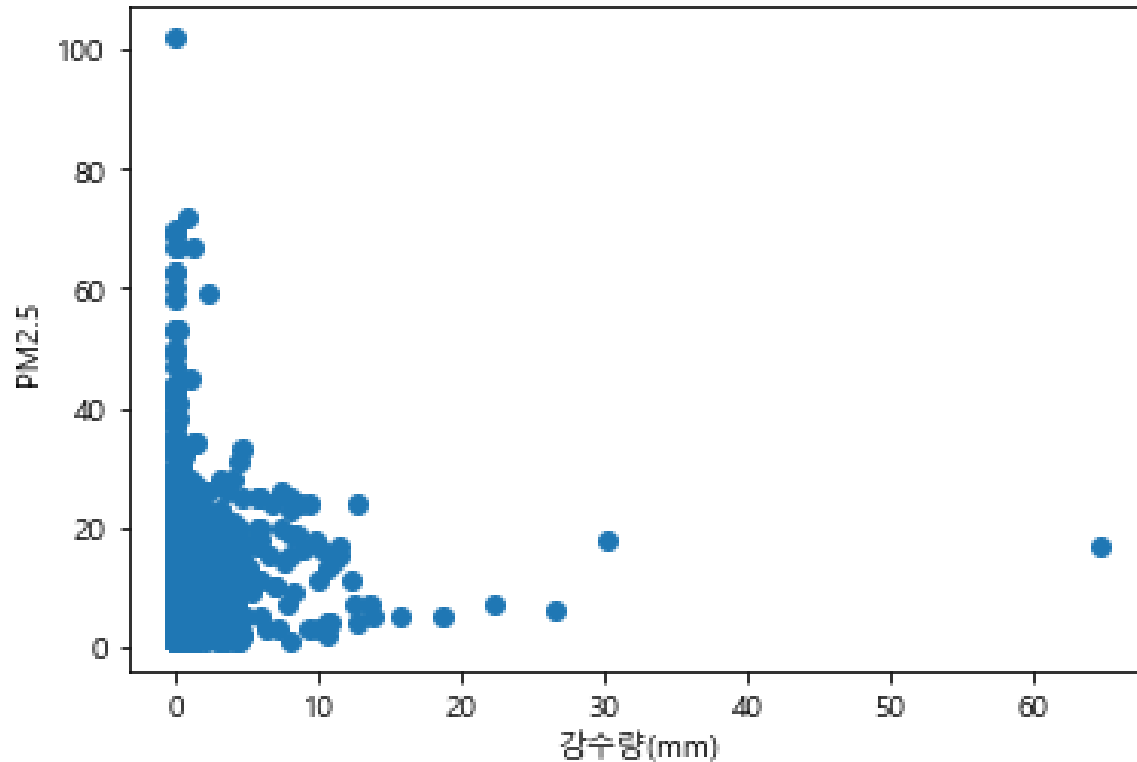
✓ 강수량과 PM10의 상관계수 : -0.064



변수 분석 내용 정리

✓ 변수 1 : 강수량

✓ 강수량과 PM2.5의 상관계수 : -0.049

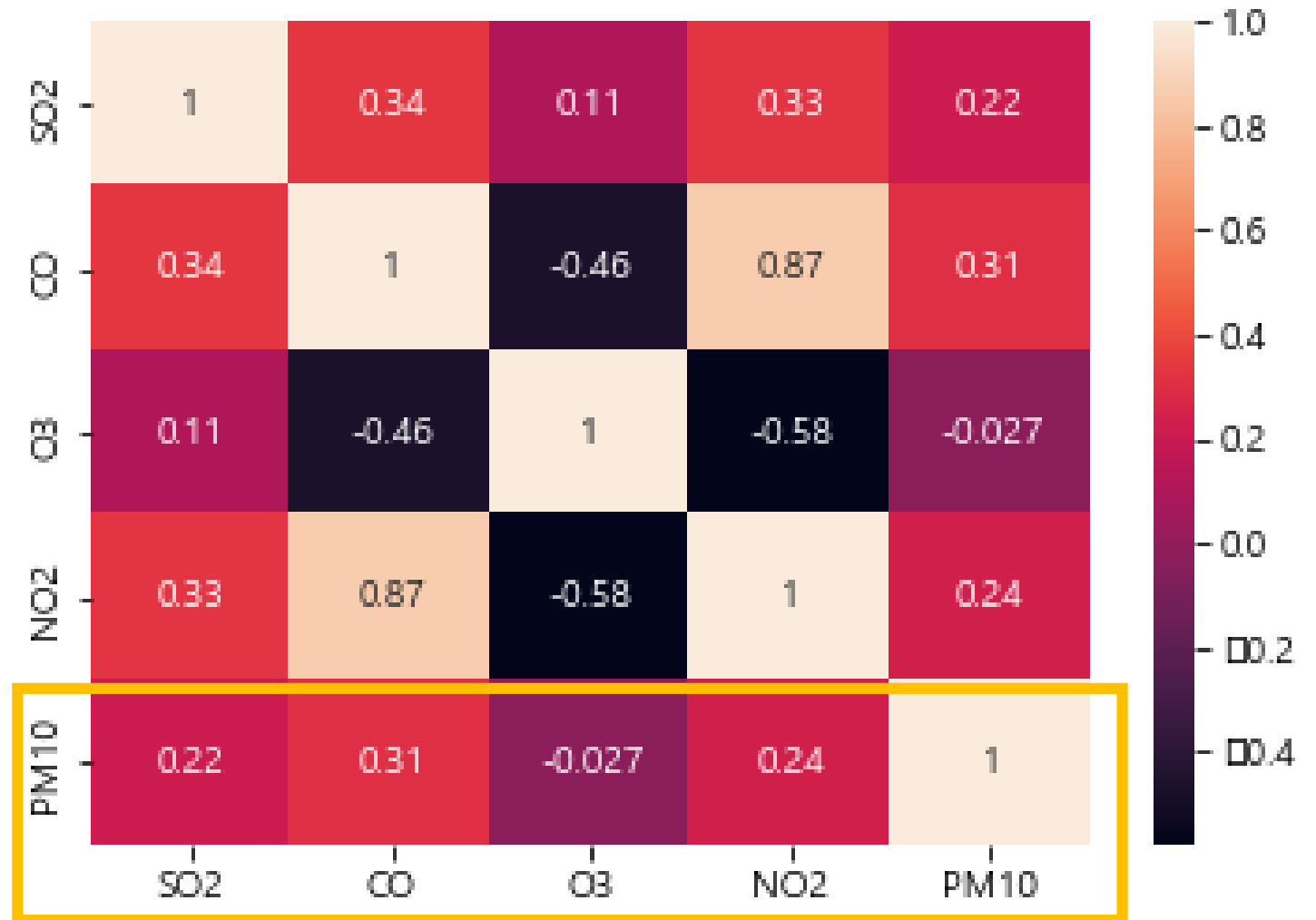


변수 2: air 데이터

✓ 변수 2

- ✓ SO2
- ✓ CO
- ✓ O3
- ✓ NO2

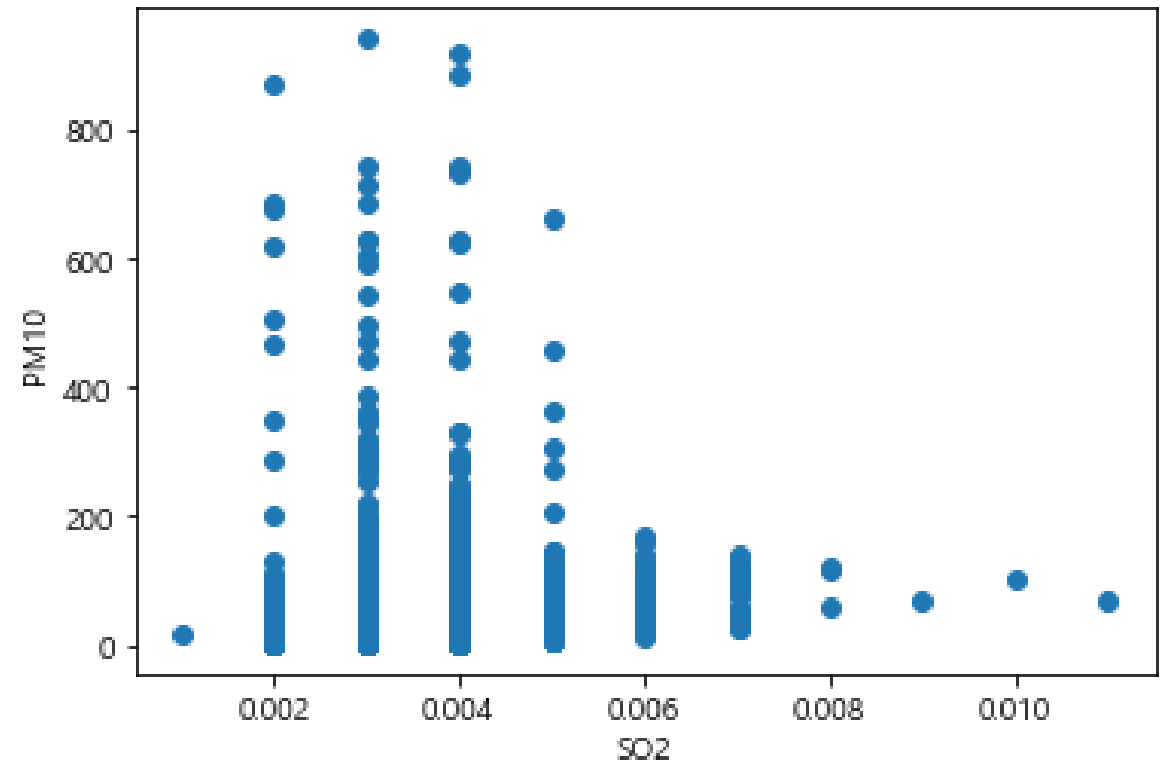
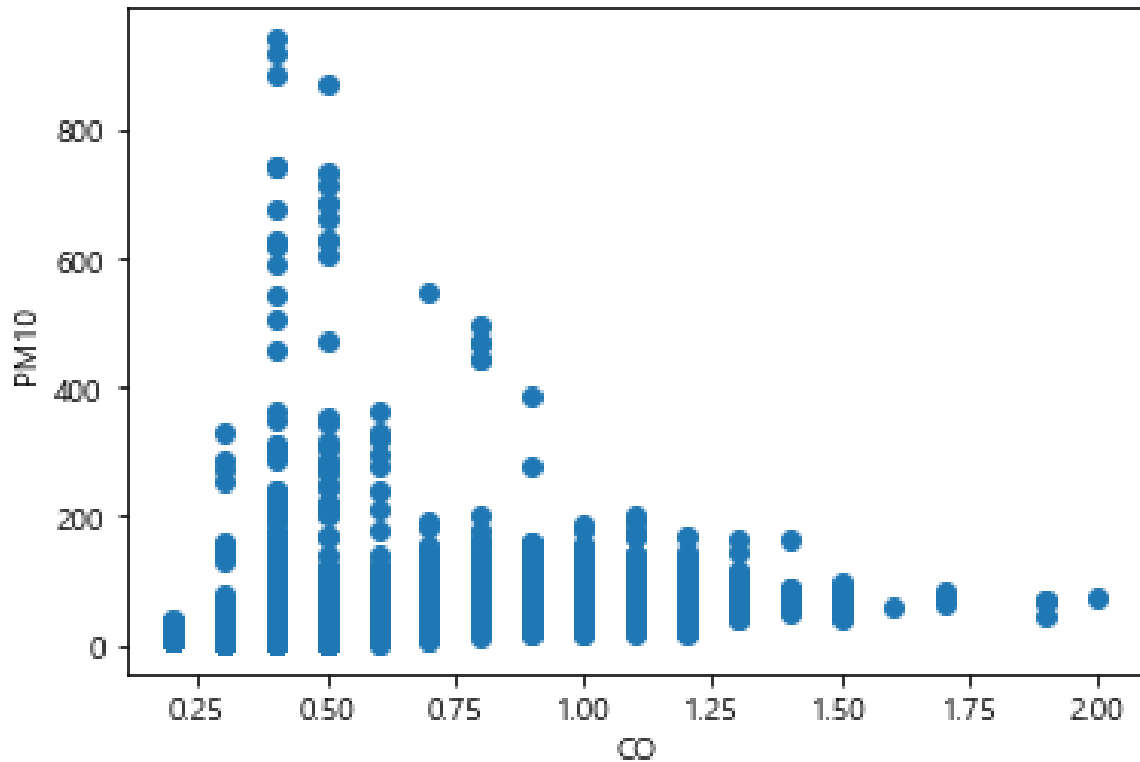
=> 전부 낮은 상관계수



변수 분석 내용 정리

✓ 변수 2 : CO, SO2

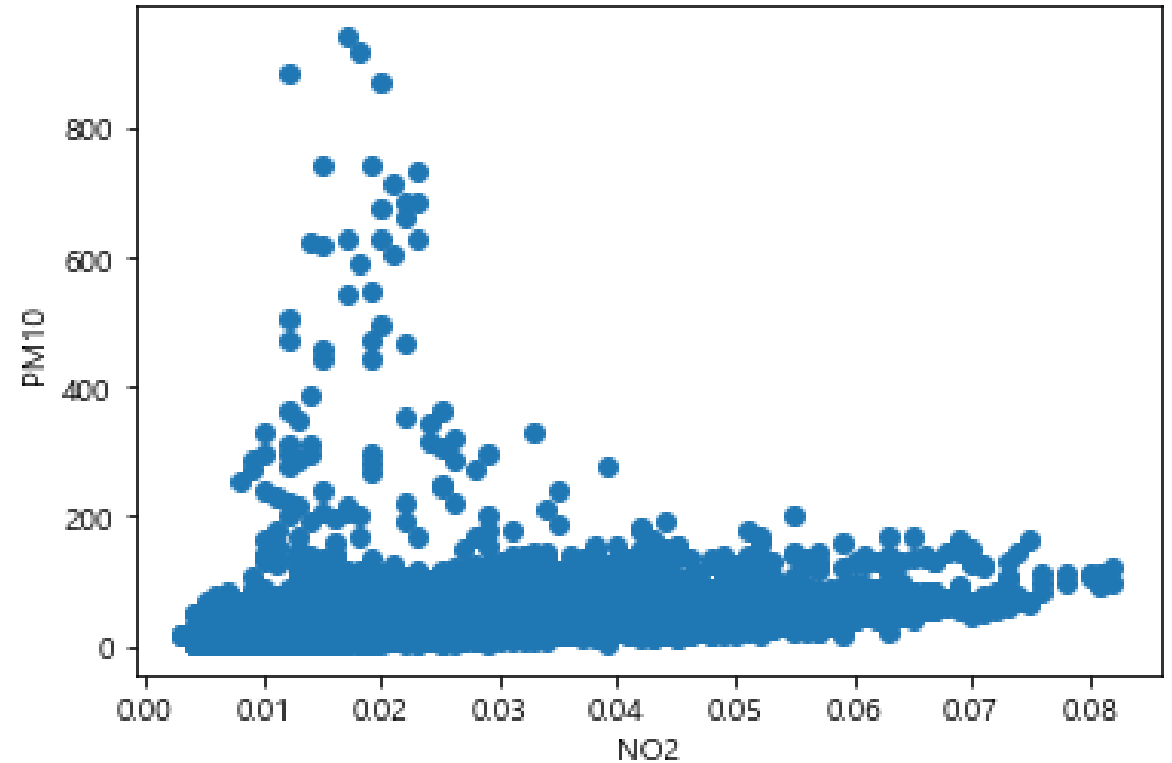
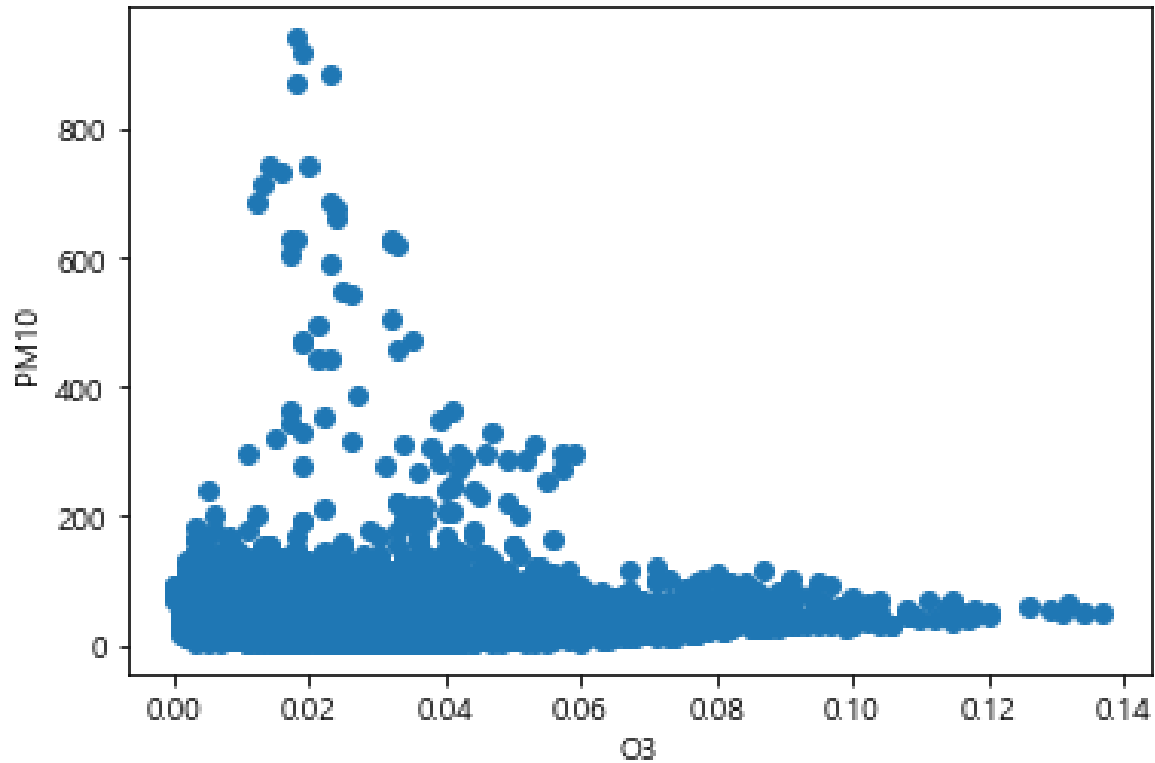
Feature가 상대적으로 낮은 농도일 때, PM10은 높은 관측값을 보임



변수 분석 내용 정리

✓ 변수 2 : O3, NO2

Feature가 상대적으로 낮은 농도일 때, PM10은 높은 관측값을 보임

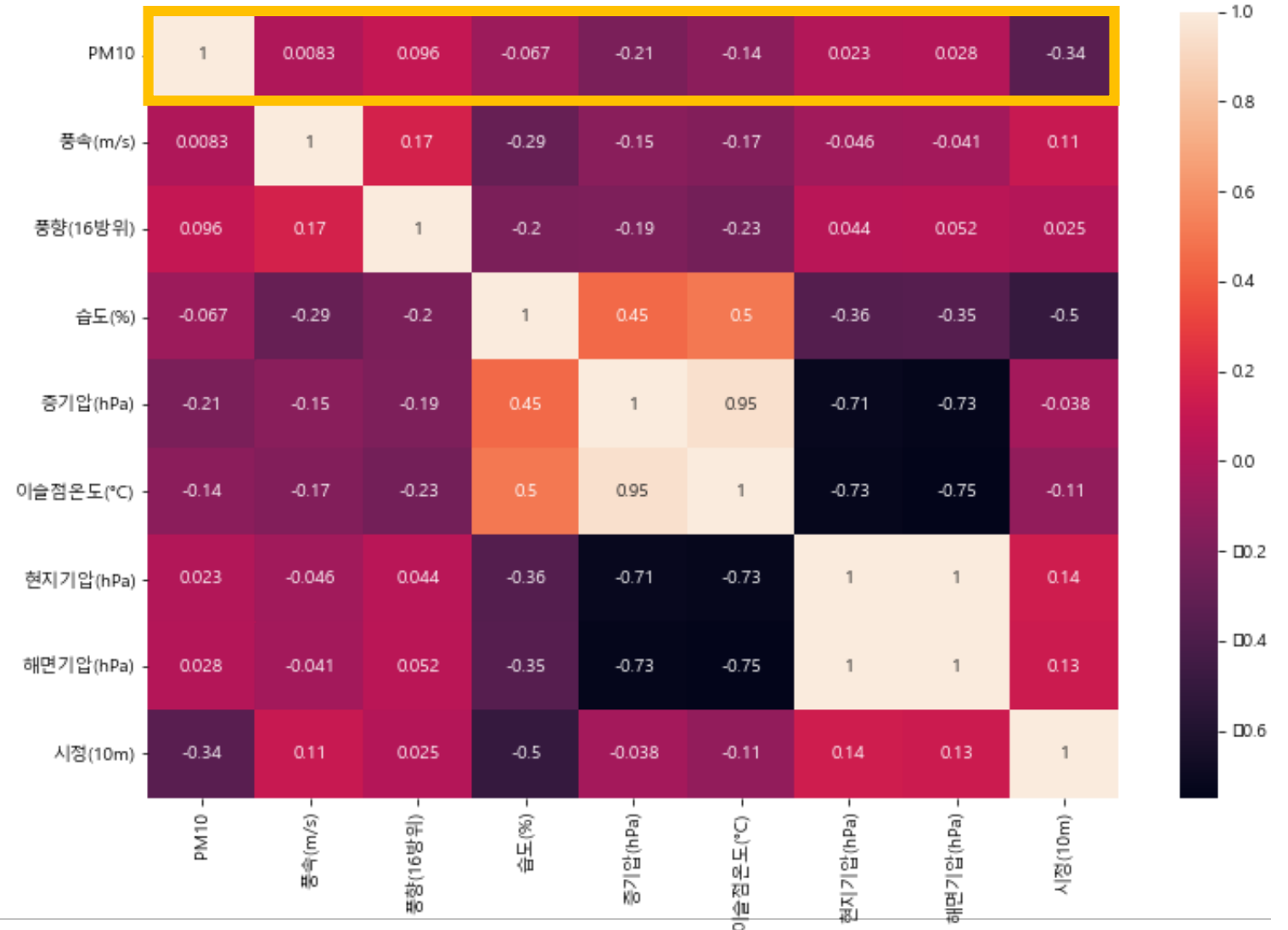


변수 3: weather 데이터

✓ 변수 3

- ✓ 풍속(m/s)
- ✓ 풍향(16방위)
- ✓ 습도(%)
- ✓ 증기압(hPa)
- ✓ 이슬점 온도(°C)
- ✓ 현지기압(hPa)
- ✓ 해면기압(hPa)
- ✓ 시정(10m)

=> 낮은 상관계수



결측치 제거

Air data : 데이터에서 채우기

✓ “에어코리아” 데이터를 사용하여 해당 결측치 보완

통계정보

대기환경 월간/연간 보고서 +

최종확정 측정자료 조회 -

국외대기 오염현황 +

최종확정 측정자료 조회

지역명검색
서울특별시 중로구 동로35가길 15(호계동)
검색

선택 측정소명 측정소 주소 거리 측정방

데이터 구분
시간
일간

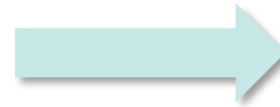
조회기간
2021
년
01
월
검색
그래프보기
엑셀

측정자료(수치)

"일평균" 자료는 국립환경과학원의 최종확정자료(시간)을 00-24시까지 산술하여 일평균 자료입니다.

날짜 (년.월.일)
PM₁₀ (μg/m³)
PM_{2.5} (μg/m³)
오존 (ppm)
이산화질소 (ppm)
일산화탄소 (ppm)
아황산가스 (ppm)

2021-01-01	30	18	0.016	0.028	0.5	0.003
2021-01-02	36	12	0.026	0.014	0.4	0.003
2021-01-03	41	14	0.022	0.019	0.4	0.003
2021-01-04	44	24	0.007	0.042	0.6	0.003
2021-01-05	28	16	0.025	0.017	0.4	0.003
2021-01-06	29	14	0.018	0.025	0.5	0.003
2021-01-07	36	13	0.024	0.01	0.4	0.003
2021-01-08	23	12	0.025	0.011	0.4	0.003
2021-01-09	27	16	0.022	0.015	0.5	0.004
2021-01-10	40	28	0.006	0.04	0.7	0.004



	A	B	C	D	E	F	G
측정일시	PM ₁₀	PM _{2.5}	O3	NO2	CO	SO2	
09-01-01	5	3	0.028	0.010	0.4	0.003	
09-01-02	3	3	0.031	0.007	0.3	0.003	
09-01-03	3	1	0.031	0.006	0.3	0.003	
09-01-04	3	1	0.029	0.007	0.3	0.003	
09-01-05	3	1	0.030	0.007	0.3	0.003	
09-01-06	3	1	0.027	0.010	0.3	0.003	
09-01-07	3	1	0.023	0.015	0.4	0.003	
09-01-08	3	1	0.025	0.016	0.4	0.002	
09-01-09	3	1	0.024	0.018	0.4	0.003	
09-01-10	4	1	0.029	0.016	0.4	0.003	
09-01-11	3	1	0.033	0.014	0.4	0.003	
09-01-12	3	1	0.035	0.013	0.4	0.003	
09-01-13	3	1	0.037	0.011	0.4	0.003	
09-01-14	3	1	0.038	0.010	0.4	0.003	
09-01-15	3	1	0.036	0.012	0.4	0.003	
09-01-16	3	1	0.033	0.014	0.4	0.003	
09-01-17	5	2	0.032	0.013	0.4	0.003	
09-01-18	6	2	0.028	0.015	0.4	0.003	
09-01-19	5	1	0.024	0.016	0.4	0.003	
09-01-20	3	1	0.025	0.016	0.4	0.003	
09-01-21	4	3	0.025	0.015	0.4	0.002	
09-01-22	4	1	0.027	0.014	0.4	0.003	
09-01-23	5	2	0.030	0.012	0.3	0.003	
09-01-24	3	1	0.028	0.013	0.3	0.002	
09-02-01	7	3	0.029	0.011	0.3	0.003	
09-02-02	5	1	0.030	0.010	0.3	0.003	
09-02-03	3	2	0.033	0.008	0.3	0.002	
09-02-04	5	2	0.033	0.009	0.3	0.002	
09-02-05	6	4	0.031	0.009	0.3	0.002	
09-02-06	6	1	0.024	0.015	0.3	0.003	
09-02-07	7	6	0.020	0.019	0.4	0.003	
09-02-08	11	4	0.022	0.021	0.4	0.002	
09-02-09	9	2	0.029	0.020	0.4	0.003	
09-02-10	10	3	0.035	0.015	0.4	0.003	
09-02-11	10	6	0.037	0.015	0.4	0.003	
09-02-12	12	4	0.039	0.013	0.4	0.003	
09-02-13	10	2	0.044	0.012	0.4	0.003	
09-02-14	11	5	0.049	0.013	0.4	0.003	

KT AIVLE School

15

데이터에서 채우기

✓ IterativeImputer / KNNImputer 결측치 처리

```
1 from sklearn.impute import KNNImputer
```

```
✓ ✓ ✓ ✓ 0.3s
```

```
1 from sklearn.experimental import enable_iterative_imputer
```

```
2 from sklearn.impute import IterativeImputer
```

```
✓ ✓ ✓ ✓ 0.7s
```

```
iter_imputer = IterativeImputer(max_iter = 10, random_state = 0)  
data_21_iter[num_col] = iter_imputer.fit_transform(train_data[num_col])
```

```
knn_imputer = IterativeImputer(max_iter = 10, random_state = 0)  
data_21_iter[num_col] = knn_imputer.fit_transform(train_data[num_col])
```



month	0
day	0
hour	0
SO2	0
CO	0
O3	0
NO2	0
PM10	0
PM25	0
강수량(mm)	0
풍속(m/s)	0
풍향(16방위)	0
습도(%)	0
증기압(hPa)	0
이슬점온도(°C)	0
현지기압(hPa)	0
해면기압(hPa)	0
시정(10m)	0
dtype:	int64

데이터 전처리

강수량

1. 0으로 채우기
2. knn imputer
3. iterative imputer

2021 데이터를 train, 2022 데이터를 test로 사용

- 2021 데이터를 train, val
2022 데이터를 test 로 사용

최종 결측치 처리방법

Air : air data 가져와서 채우기
Weather : weather KNN Imputer



가장 성능이 좋았음

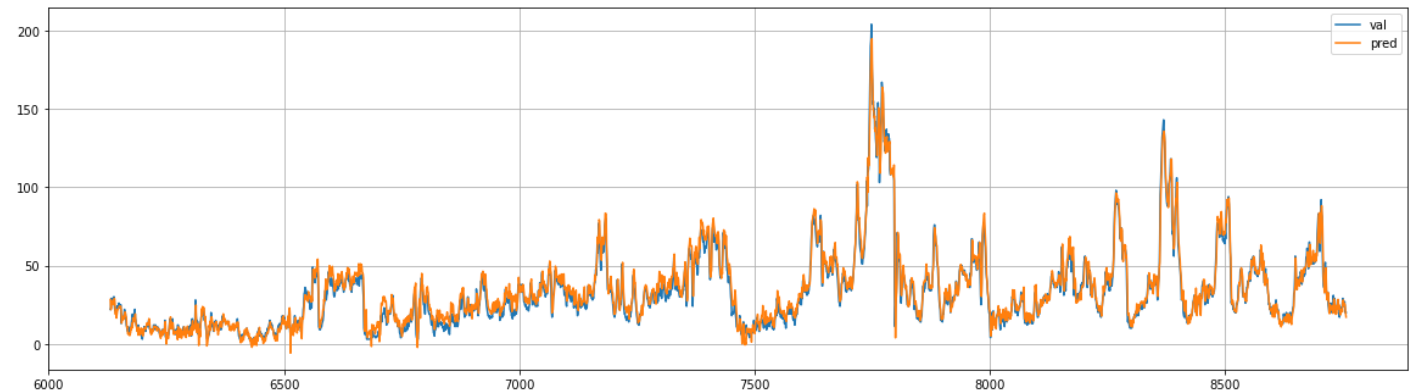
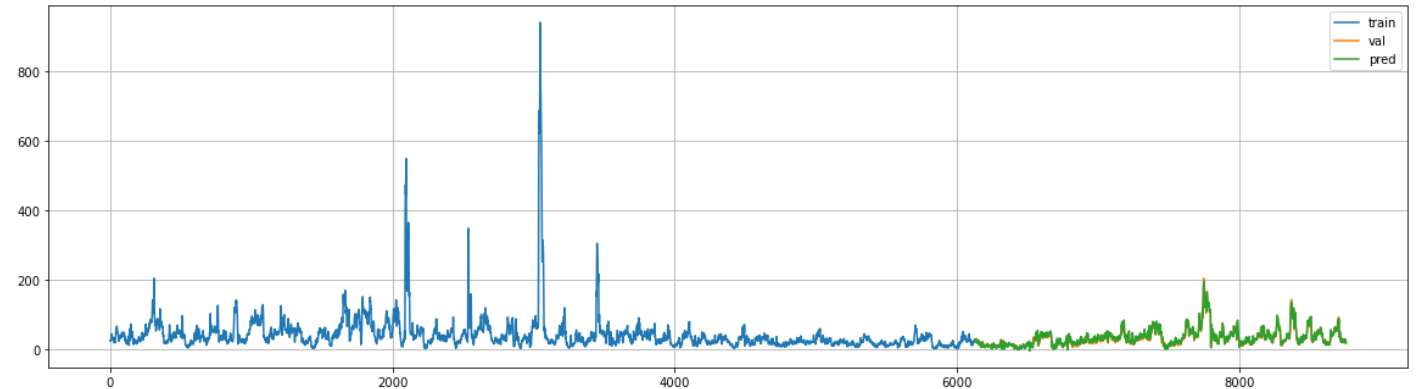
AI 모델링

Linear Regression

- 선형회귀 모델링

```
# 아래에 실습코드를 작성하세요.  
model_lr = LinearRegression()  
model_lr.fit(x_train, y_train)
```

model	Linear
R2Score	0.954252
RMSE	5.424289
MAE	3.967838
MAPE	0.187151

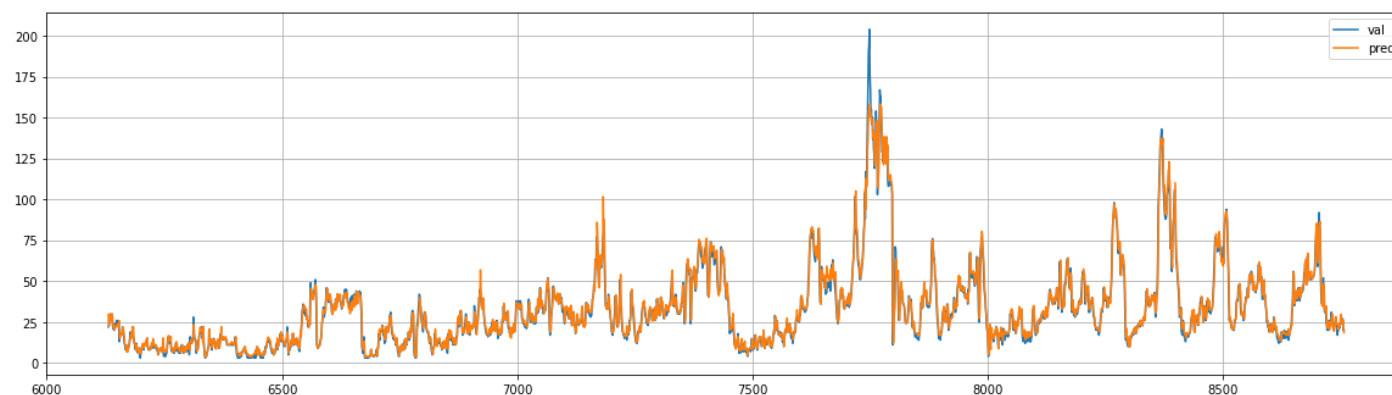
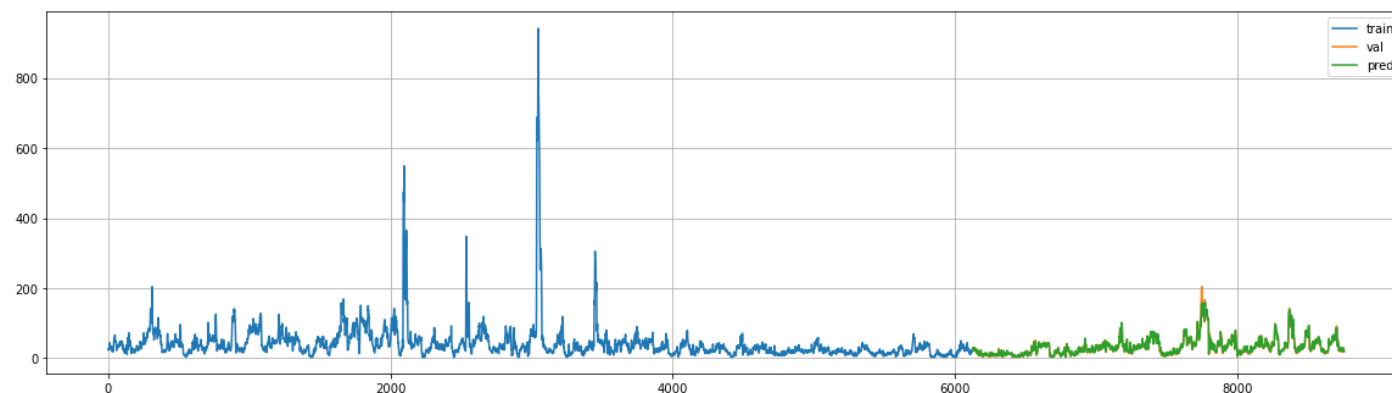


RandomForest Regressor

- RandomForest 모델링

```
# 아래에 실습코드를 작성하세요.  
model_rf = RandomForestRegressor()  
model_rf.fit(x_train, y_train)
```

model	Random Forest
R2Score	0.95721
RMSE	5.24597
MAE	3.393903
MAPE	0.138865

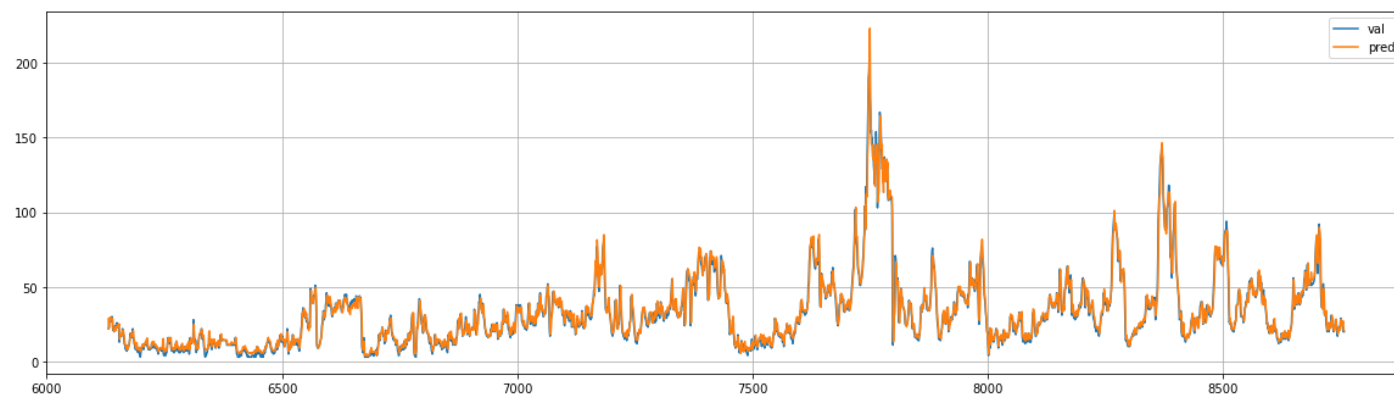
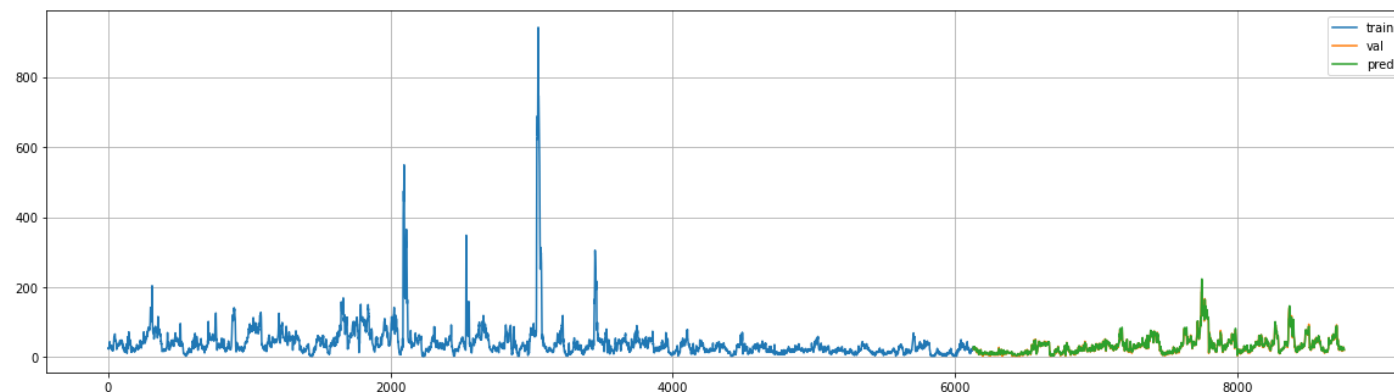


Gradient Boosting Regressor

- Gradient Boosting 모델링

```
# 아래에 실습코드를 작성하세요.  
model_gb = GradientBoostingRegressor()  
model_gb.fit(x_train, y_train)
```

model	Gradient Boosting
R2Score	0.962173
RMSE	4.932366
MAE	3.259528
MAPE	0.143858

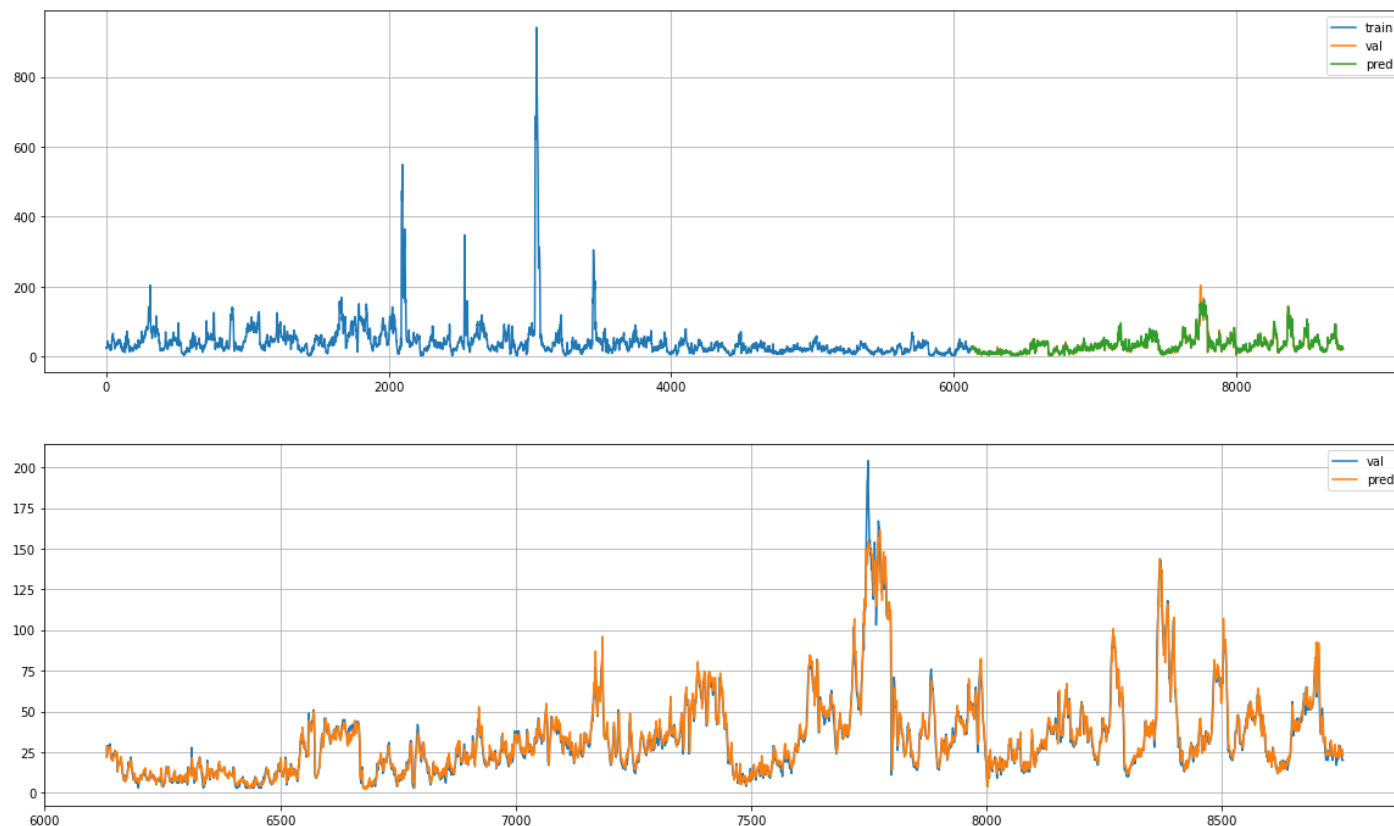


XGB Regressor

- XGBoosting 모델링

```
# 아래에 실습코드를 작성하세요.  
model_xgb = XGBRegressor()  
model_xgb.fit(x_train, y_train)
```

model	XGB
R2Score	0.957417
RMSE	5.233269
MAE	3.411547
MAPE	0.141425



Stacking Regressor

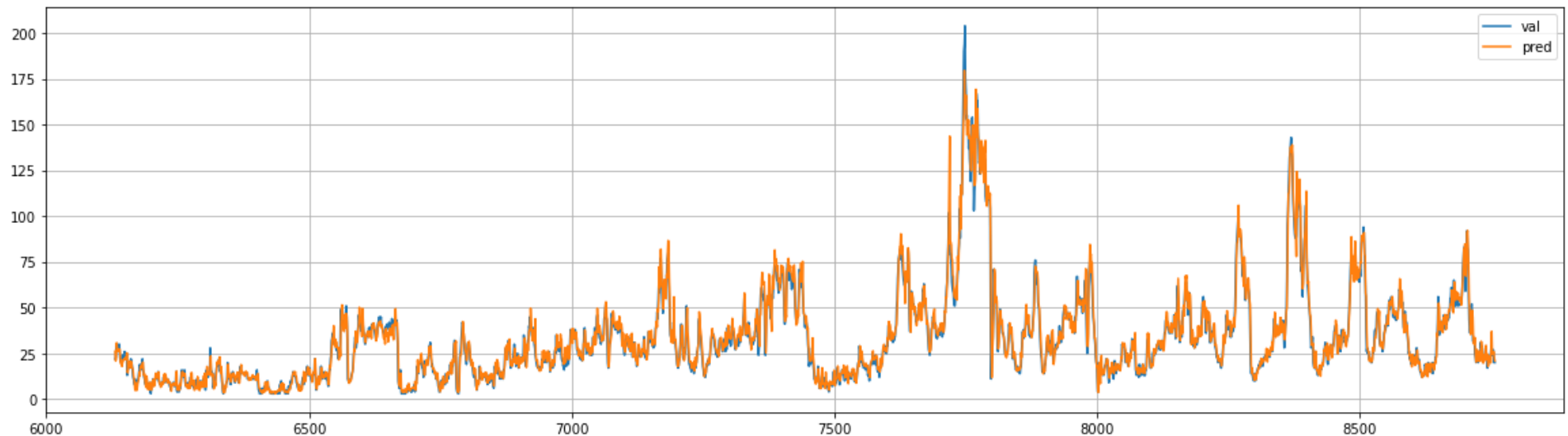
- Stacking 모델링
- final_estimator = RF

```
from sklearn.pipeline import make_pipeline

estimators = [('rf', RandomForestRegressor()),
              ('xgb', XGBRegressor()),
              ('gb', GradientBoostingRegressor()),
              ('linear', LinearRegression())]

model_stacking = StackingRegressor(estimators=estimators,
                                   final_estimator=RandomForestRegressor())
```

model	Stacking
R2Score	0.957768
RMSE	5.211659
MAE	3.454414
MAPE	0.138953



최종 성능

결론

[결측치 처리가 보여주는 성능]

	model	R2Score	RMSE	MAE	MAPE
0	Linear	0.891064	7.825481	4.745637	0.150392
1	Random Forest	0.899079	7.532105	4.209543	0.131770
2	Gradient Boosting	0.905728	7.279772	4.062354	0.133741
3	XGB	0.879926	8.215802	4.526941	0.139474
4	Stacking	0.879397	8.233877	4.416504	0.130406

[최종 test 성능]

Best R2Score : 0.905728

- R2 score : Gradient Boosting
- RMSE : Gradient Boosting
- MAE : Gradient Boosting
- MAPE : Stacking

결측치 결론

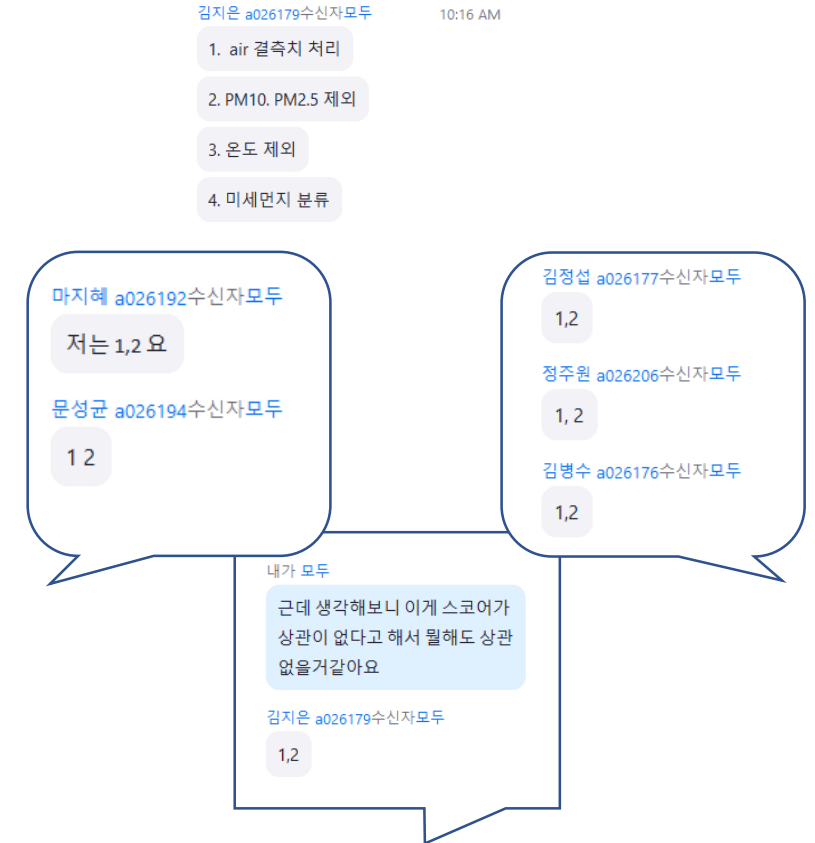
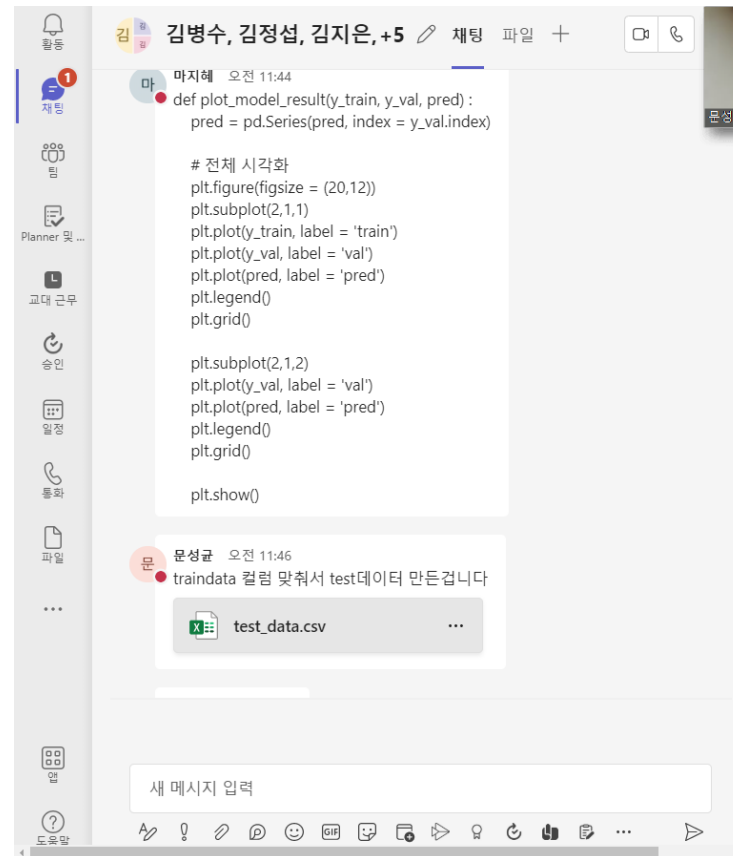
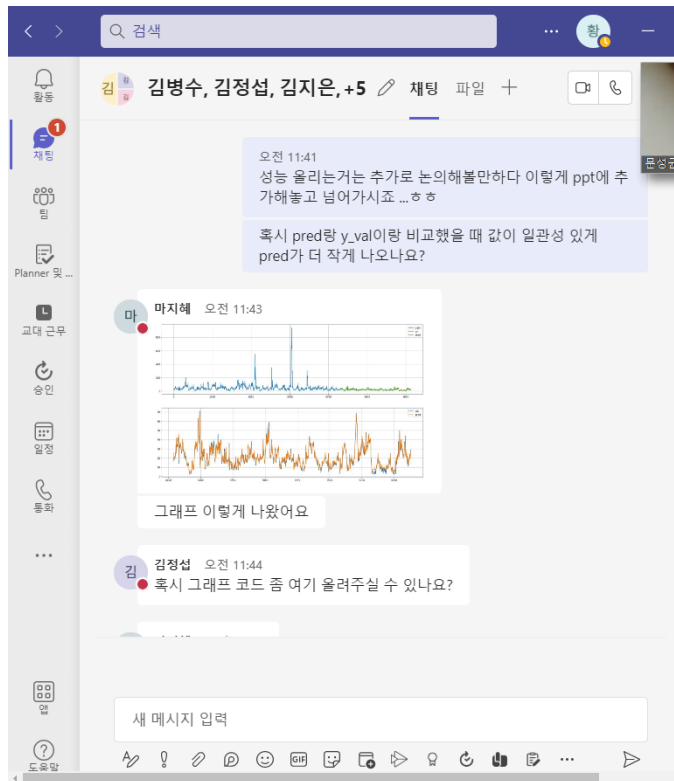
```
[93]: print(xgb_m1.best_params_, xgb_m1.best_score_)  
{'learning_rate': 0.05, 'n_estimators': 200} 0.8836356843625934  
  
[94]: new_XGB1 = XGBRegressor(learning_rate = 0.05, n_estimators=200 )  
new_XGB1.fit(x_train, y_train)  
p_xgb1 = new_XGB1.predict(x_val)  
print('XGB r2score : ' + str(r2_score(y_val, p_xgb1)))  
print('XGB RMSE : ' + str(mean_squared_error(y_val, p_xgb1, squared = False)))  
  
XGB r2score : 0.9621046055904187  
XGB RMSE : 4.936913181647892
```

[최종 test 성능]

AIR 결측치제거 : 직접 채우기
XGB Regressor 모델링을 사용했을 때
성능이 가장 좋음

조별 토론 과정

Teams 채팅 이용 & Zoom을 통한 토의



kt

 AIVLE

 AIVLE
Let's make it possible