사물인터넷 기초 D조 시각자료

한국가스공사 수요예측 모델 개발

B611156 이은
B711106 안지수
B811038 김은영
B995107 이상명

목차

프로젝트 주제 설명

가스공급량 수요예측 모델개발 from.DACON

프로젝트 주제 설명

한국가스공사의 시간단위 공급량 내부데이터와 기상정보 및 가스외 발전량 등 외부데이터를 포함한 데이터셋을 구축하여 90일 한도 일간 공급량을 예측하는 인공지능 모델을 개발합니다.

누구나 획득 가능한 외부 데이터로서 기상정보 및 가스 외 발전량 등 외부데이터를 포함한 데이터사용이 권장됩니다.

데이터 수집 & 전처리

데이터 변환 & CSV 추출

(2013 ~ 2018년도)

데이터 변환 및 CSV추출

내부데이터

연월일	시간	구분	공급량
2013-01-01	1	A	2497.129
2013-01-01	2	A	2363.265
2013-01-01	3	A	2258.505
2013-01-01	4	A	2243.969
2013-01-01	5	A	2344.105
2013-01-01	6	A	2390.961
2013-01-01	7	A	2378.457
2013-01-01	8	A	2518.921
2013-01-01	9	A	2706.481
2013-01-01	10	A	2832.057
2013-01-01	11	A	2895.185
2013-01-01	12	A	2689.361
2013-01-01	13	A	2425.537
2013-01-01	14	A	2254.289
2013-01-01	15	A	2153.361
2013-01-01	16	A	2126.969
2013-01-01	17	A	2210.481
2013-01-01	18	A	2546.873
2013-01-01	19	A	2886.097
2013-01-01	20	A	2863.009
2018-12-31	4	H	475.35
2018-12-31	5	H	518.521
2018-12-31	6	H	528.382
2018-12-31	7	H	594.463
2018-12-31	8	H	699.816
2018-12-31	9	H	742.313
2018-12-31	10	H	713.533
2018-12-31	11	H	694.308
2018-12-31	12	H	609.857
2018-12-31	13	H	575.594
2018-12-31	14	H	551.823
2018-12-31	15	H	525.488
2018-12-31	16	H	518.009
2018-12-31	17	H	542.36
2018-12-31	18	H	603.138
2018-12-31	19	H	678.975
2018-12-31	20	H	681.033
2018-12-31	21	H	669.961
2018-12-31	22	H	657.941
2018-12-31	23	H	610.953
2018-12-31	24	H	560.896

데이터 설명

Dacon에서 제공받은 데이터

<구성>

열 개수 : 4

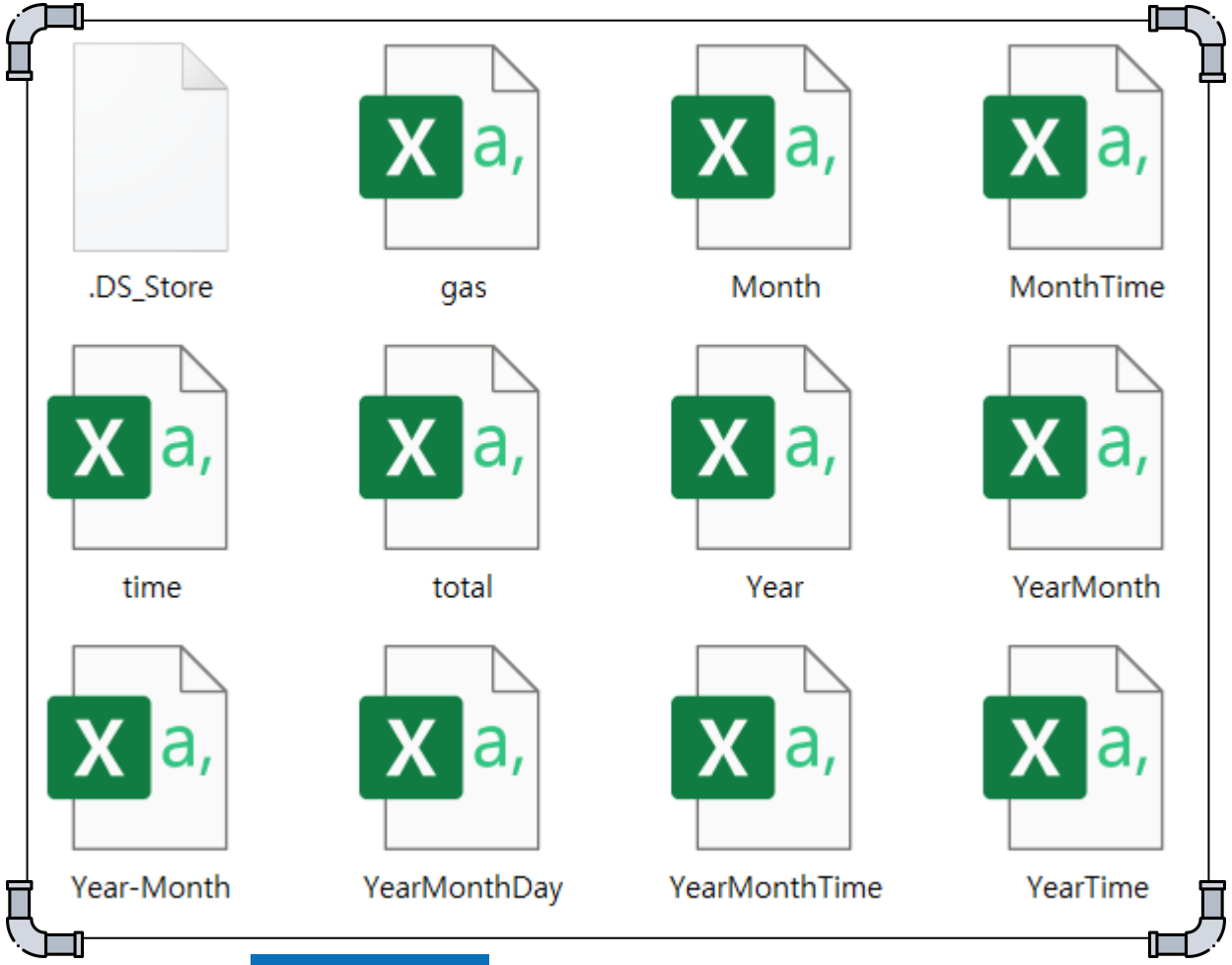
연월일 / 시간 / 구분(공급사) / 공급량

행 개수 : 368,087

시간별(1시간)

데이터 변환 및 CSV추출

내부데이터



데이터 출처 : DATA 공공데이터포털 .GO.KR

자료설명

*데이터 변환 및 csv추출

1. total.csv : 7개 공급사의 공급량을 연-월-일 별로 전부 합친 데이터
2. Year.csv : 연도별 가스공급량
3. Month.csv : 월별 가스공급량
4. time.csv : 시간별 가스공급량
5. YearMonthDay.csv : 연도-월-일별 가스공급량
6. YearMonthTime.csv : 월-시간별 가스공급량
7. YearMonth.csv : 연도-월별 가스공급량
8. YearTime.csv : 연도-시간별 가스공급량
9. MonthTime.csv : 월-시간별 가스공급량

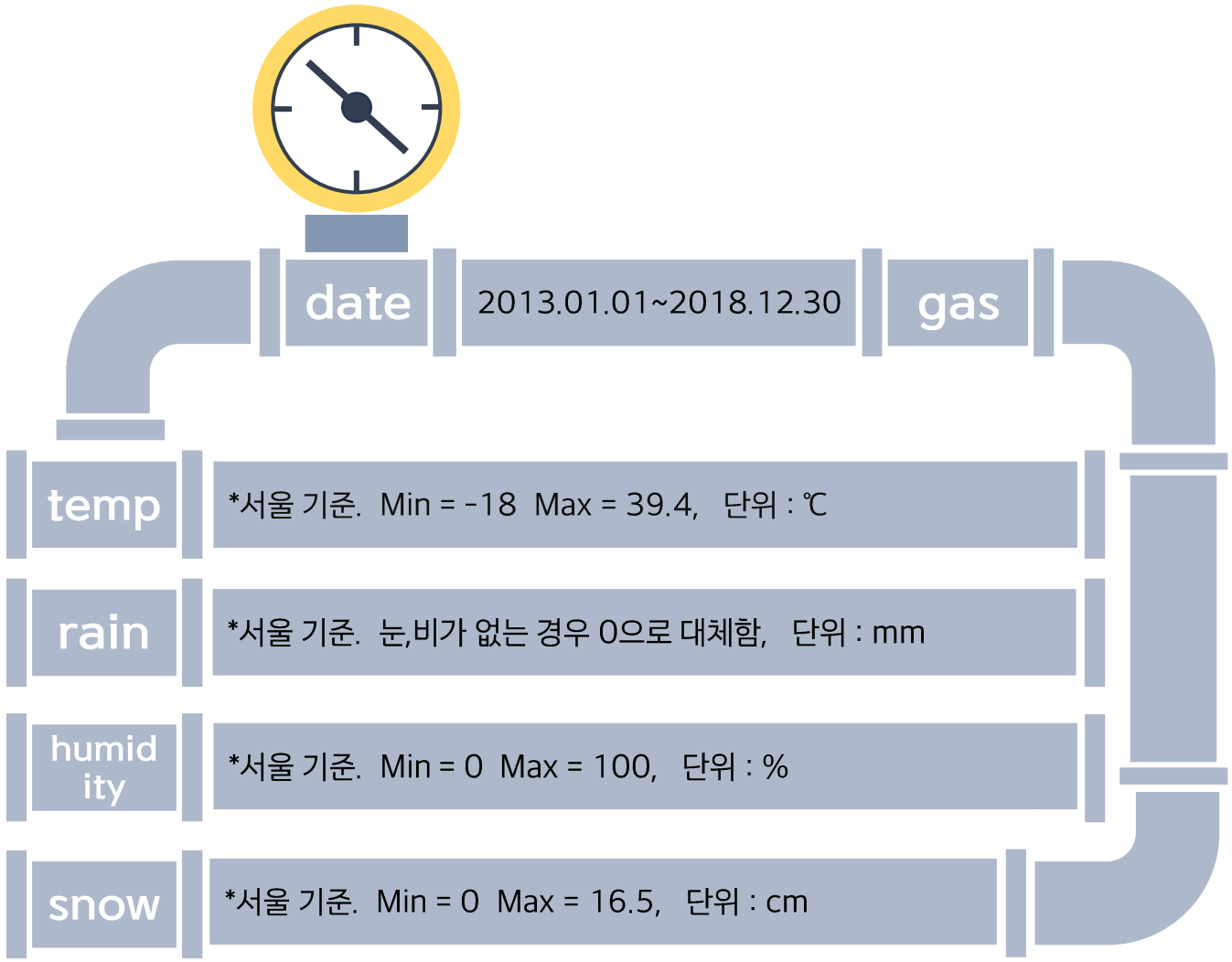
데이터 변환 및 CSV추출

내부데이터 + 외부데이터

	A	B	C	D	E	F
1	date	temp	rain	humidity	snow	gas
2	2013-01-01 1:00	-8.5	0	57	6.4	13723.08
3	2013-01-01 2:00	-8.4	0	60	6.4	12919.01
4	2013-01-01 3:00	-8.1	0	58	6.4	12174.49
5	2013-01-01 4:00	-8.2	0	58	6.4	12146.66
6	2013-01-01 5:00	-8.2	0	61	6.4	12667.67
7	2013-01-01 6:00	-8.6	0.2	81	6.9	12656.67
8	2013-01-01 7:00	-8.3	0	85	8	12820.95
9	2013-01-01 8:00	-7.9	0	84	9.2	13722.24
10	2013-01-01 9:00	-7	1.2	81	9.2	15027.62
11	2013-01-01 10:00	-5.5	0	74	9.2	15665.92
12	2013-01-01 11:00	-4.2	0	71	9.1	15761.5
13	2013-01-01 12:00	-4.2	0.7	83	9.9	14532.59
14	2013-01-01 13:00	-1.9	0	71	9.7	13150.74
15	2013-01-01 14:00	-0.6	0	65	9.6	12269.45
16	2013-01-01 15:00	-0.5	0	56	9.5	11753.28
17	2013-01-01 16:00	-0.6	0	55	9.4	11642.43
18	2013-01-01 17:00	-1.1	0	56	9.4	12346.47
19	2013-01-01 18:00	-1.7	0	59	9.4	14142.39
20	2013-01-01 19:00	-1.6	0	60	9.4	15871.61
21	2013-01-01 20:00	-2.5	0	84	9.7	15711.25
22	2013-01-01 21:00	-2.8	0.7	87	10.3	15326.83

total

데이터 출처: 기상청 기상자료개방포털



데이터 시각화

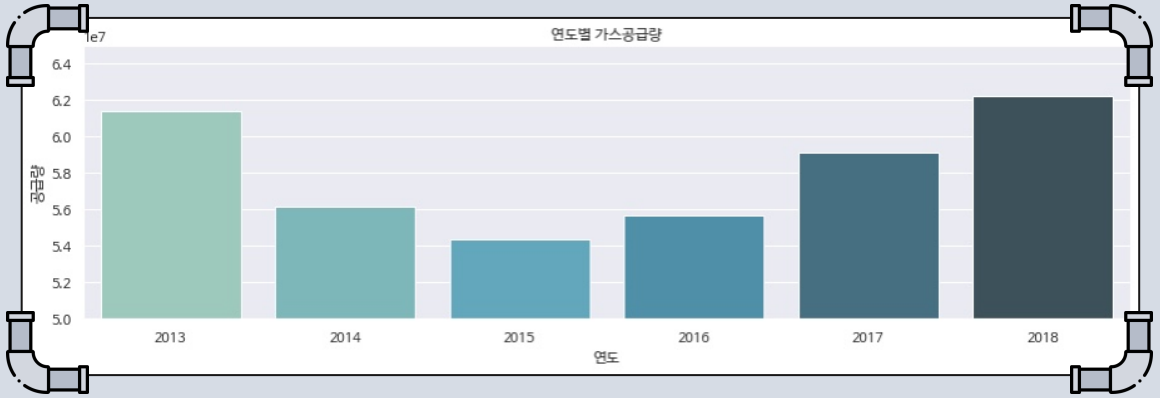
연도별 & 월별 & 시간별 데이터 분석

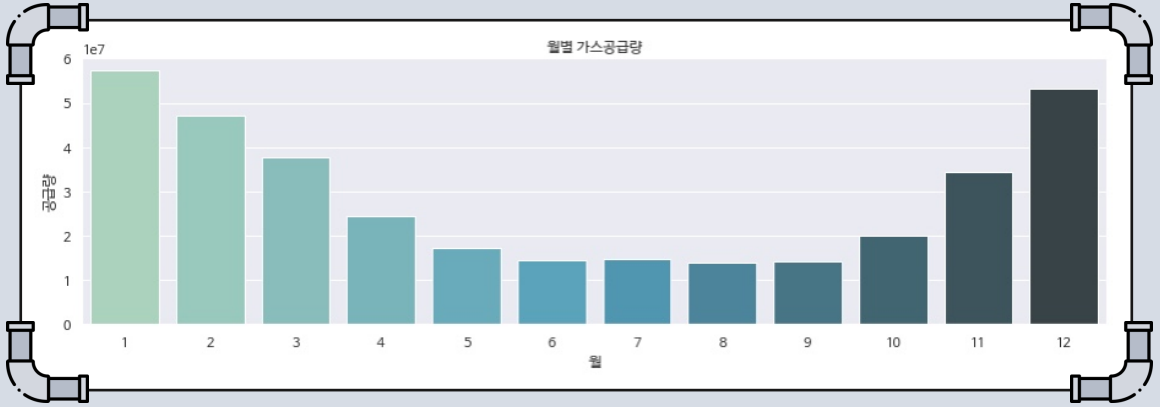
(2013 ~ 2018년도)

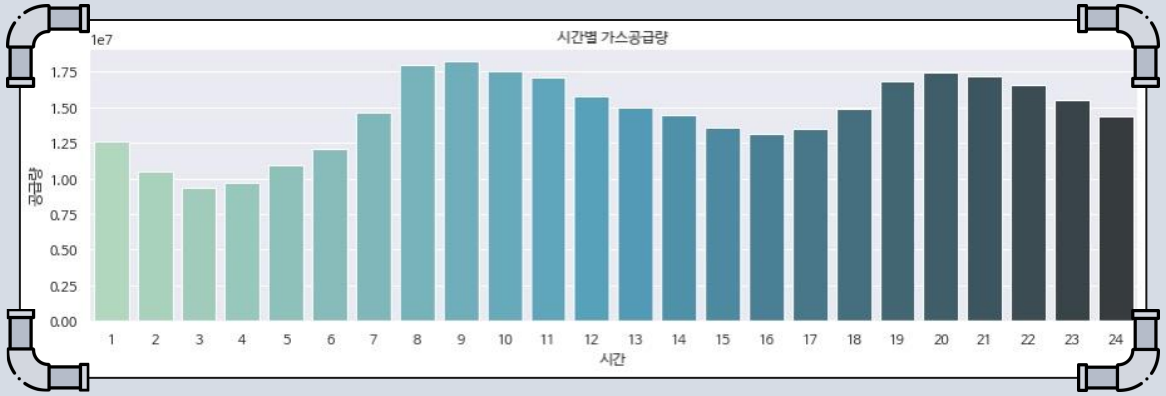
Barplot, Lineplot, Regplot, Heatmap

가스공급량 Barplot

연도별 & 월별 & 시간별 분석







연도별 가스공급량

X 축 - 연도(2013 ~ 2018)
y 축 - 공급량 (기준 : ton)
공급량이 많은 순서는
2018 > 2013 > 2017 > · · · >2015

월별 가스공급량

X 축 - 월(1~12월)
y 축 - 공급량 (기준 : ton)
<결과>Max = 1월 / min = 6월
<해석>기온이 높은 달(6~9월)에 공급량이 가장 낮고,
기온이 낮은 달(11~2월)에 공급량이 가장 높다.

시간별 가스공급량

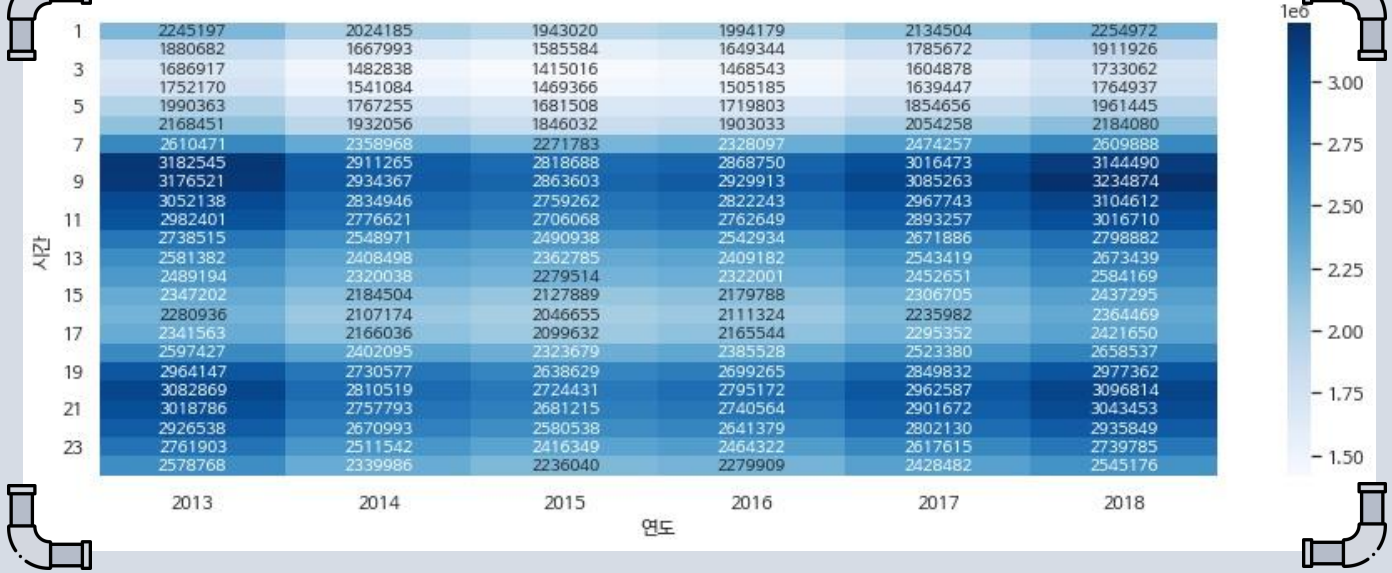
X 축 - 시간(24시)
y 축 - 공급량 (기준 : ton)
<결과> Max = 9시 / min = 3시
<해석> 활동율이 적은 새벽시간 & 가장 따뜻한 낮시간의 공급량이 가장 낮음

가스공급량 heatmap

연도 & 월 분석

연도 & 월별 가스공급량

X 축 - 연도(2013 ~ 2018)
y 축 - 월(1~12)
〈결과〉 Max = 2013.01 / min = 2015.08
〈해석〉 11~2월에 공급량이 높고, 5~9월에 공급량이 낮은 경향.

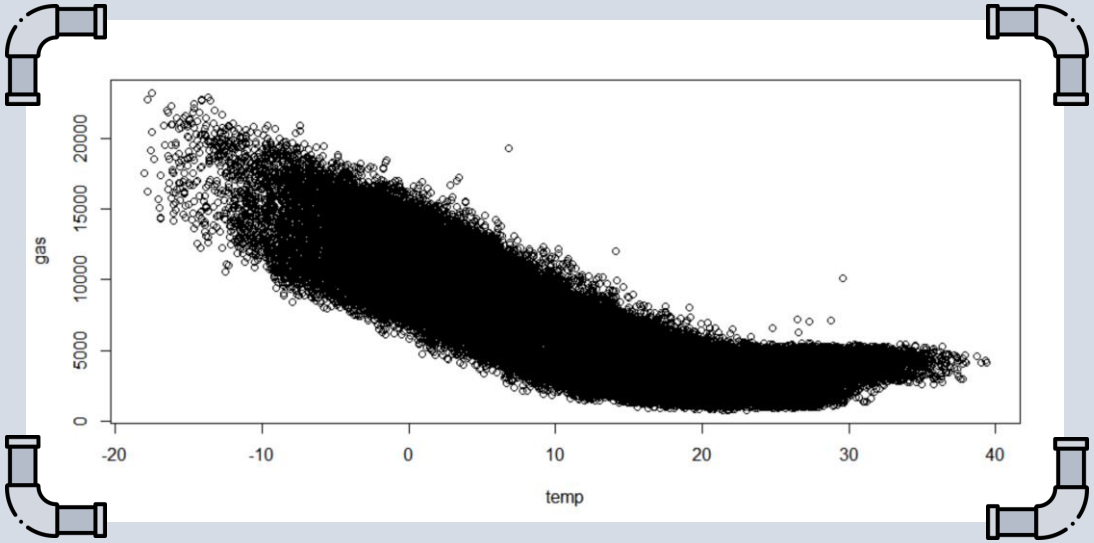


연도 & 시간별 가스공급량

X 축 - 연도(2013 ~ 2018)
y 축 - 월(1~12)
〈결과〉 Max = 2018, 9시 / min = 2015, 4시
〈해석〉 9시, 20시에 공급량이 높고, 새벽시간에 공급량이 낮은 경향.

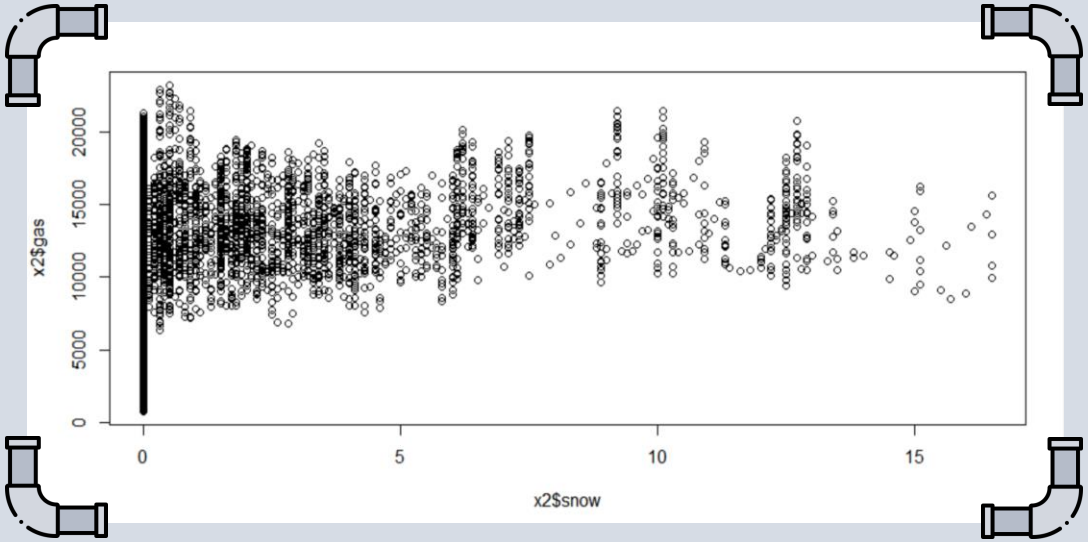
가스공급량

기온&공급량 상관관계, 적설&공급량 상관관계 분석



기온 & 공급량 상관관계

X 축 - 기온
y 축 - 가스 공급량
R-studio 이용
〈결과〉 -0.84821917
〈해석〉 강한 음의 상관관계

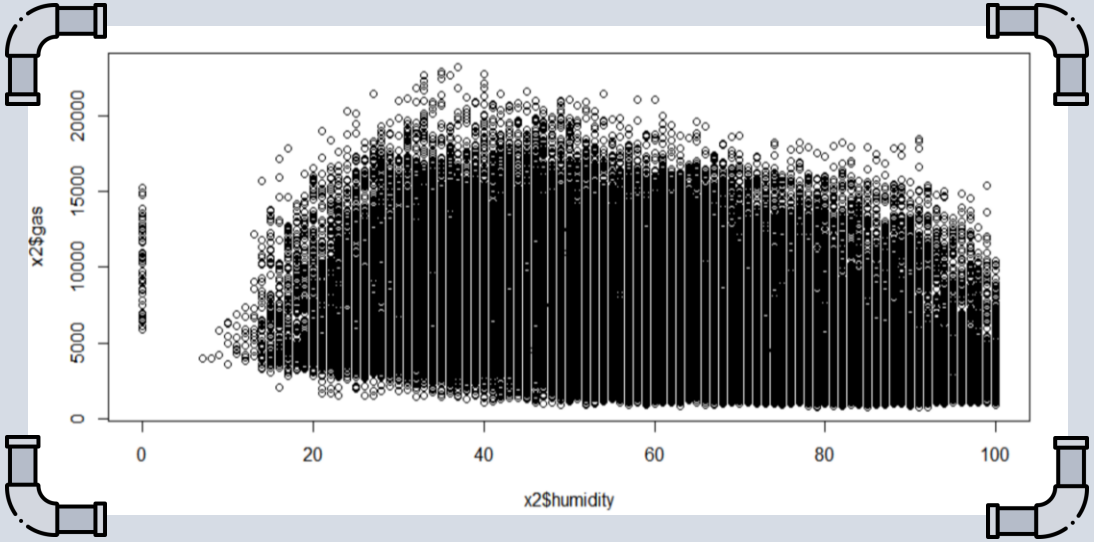


적설 & 공급량 상관관계

X 축 - 적설
y 축 - 가스 공급량
R-studio 이용
〈결과〉 0.2790566281
〈해석〉 약한 양의 상관관계

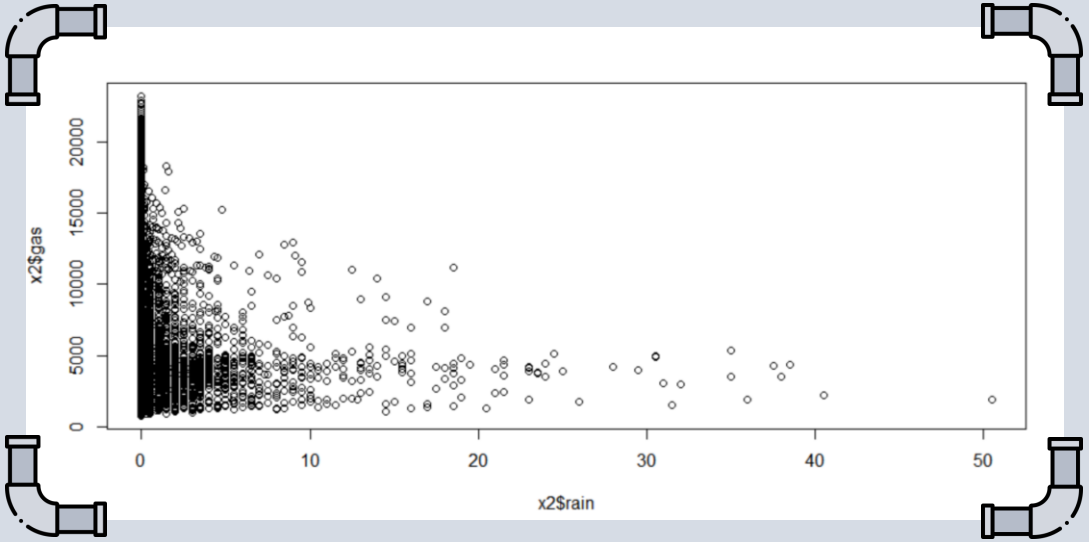
가스공급량

습도&공급량 상관관계, 강수&공급량 상관관계 분석



습도 & 공급량 상관관계

X 축 - 습도
y 축 - 가스 공급량
R-studio 이용
〈결과〉 -0.2913962138
〈해석〉 약한 음의 상관관계



강수 & 공급량 상관관계

X 축 - 강수
y 축 - 가스 공급량
R-studio 이용
〈결과〉 -0.061632864
〈해석〉 상관관계 없음

가스공급량 모델링 예측

- Xgboost를 이용한 시계열 분석

수요 예측 모델

Information explanation

■ GBM : 초기 예측을 하고 예측에 대한 loss function에 미분으로 gradient를 구하고, 값을 전달하여 오차를 줄 이는 학습 방식으로 학습을 마치고 최종적으로 초기 예측값과 각 결과값의 학습률의 곱으로 최종예측 But, 속도가 느리고 과적합의 문제가 존재하는 알고리즘

■ XGBoost: Gradient Boosting 알고리즘을 분산환경에서도 실행할 수 있도록 구현한 라이브러리로 여러 개의

Decision Tree를 조합해서 사용하는 Ensemble 알고리즘

■ XGBoost 장점 :

1.병렬처리를 통한 빠른 학습 -> GBM의 단점으로 지적되는 느린 속도를 XGBoost는 병렬처리를 통해 극복

2. 유연한 learning system 과 다양한 파라미터를 이용한 확장성 -> 목적과 평가기준을 Setting 할 수 있으며, 트리를 Split하는 과정에서 미리 셋팅해 둔 값 까지만 Split한다.

3. Overfitting 방지를 위한 설계 -> 매 Iteration 마다 Cross-Validation을 수행하게 해준다. 따라서 최적의 Boosting iteration을 알 수 있다.

수요 예측 모델

Model parameter explanation

■ n_estimators : 빌드하고 싶은 tree 수

■ min_child_weight: 관측에서 요구되는 최소 가중치의 합으로 over fitting과 under fitting의 관측에서 요구되는 가중치의 합

■ max_depth: 트리의 최대 깊이

■ subsample: 각 트리마다의 관측 데이터의 샘플링 비율

■ colsample_bytree: 각 트리마다의 feature 샘플링 비율

■ 평가지표인 NMAE - Normalized Mean Absolute Error로 표준화된 평균 절대 오차를 뜻함

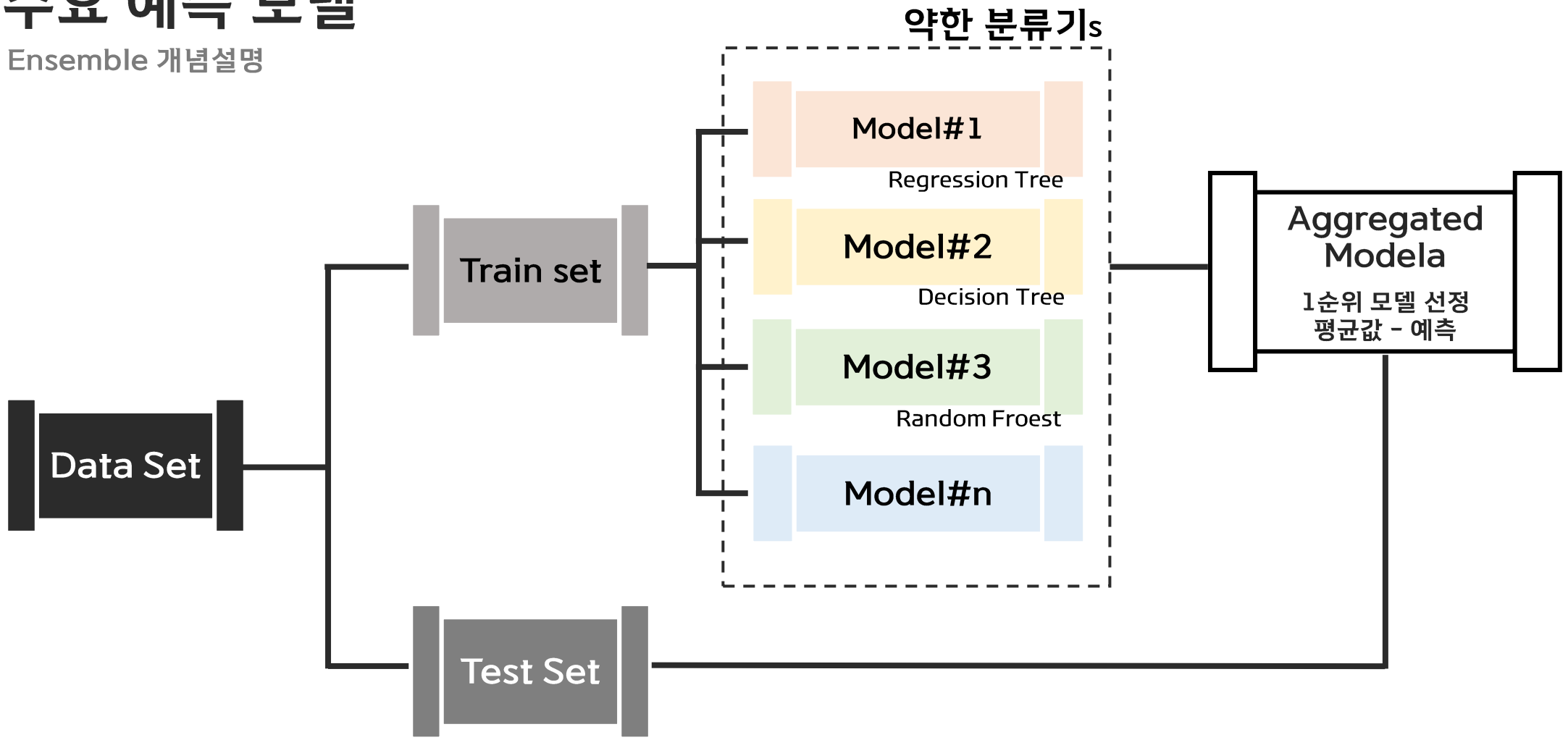
```
# nmae 정의
def nmae(true, pred):
    score = np.mean((np.abs(true-pred))/true)
    return score
```

```
# nmae값은 실제값보다 작게 추정할 때와 높게 예측할 때의 예측평가가 같음을 확인
## xgboost의 기본 objective function으로 훈련
print("실제값이 100일 때 50으로 underestimate할 때의 nmae : {}".format(nmae(100, 50)))
print("실제값이 100일 때 150으로 overestimate할 때의 nmae : {}".format(nmae(100, 150)))
```

```
실제값이 100일 때 50으로 underestimate할 때의 nmae : 0.5
실제값이 100일 때 150으로 overestimate할 때의 nmae : 0.5
```

수요 예측 모델

Ensemble 개념설명



파이썬 코드(함수) 설명

- Xgboost를 이용한 시계열 분석

수요 예측 모델 - XGBoost

파이썬 코드 & 코드(함수) 설명

```
1 import pandas as pd
2 import sys
3 import tqdm as tq
4 import seaborn as sns
5 import sklearn as skl
6 import numpy as np
7 import matplotlib.pyplot as plt
8 from tqdm import tqdm
9 import sktime
10 from sktime.forecasting.model_selection import temporal_train_test_split
11 from sktime.utils.plotting import plot_series
12 from xgboost import XGBRegressor
13 import xgboost as xgb
14 Total = pd.read_csv('/content/Koreagas.csv',encoding='cp949')
```

필요한 모듈 import

수요 예측 모델 - XGBoost

파이썬 코드 & 코드(함수) 설명

```
[60]  1 Total.tail()
```

	연월일	구분	공급량	시간
368083	2018-12-31	H	681.033	20
368084	2018-12-31	H	669.961	21
368085	2018-12-31	H	657.941	22
368086	2018-12-31	H	610.953	23
368087	2018-12-31	H	560.896	24

```
[61]  1 Total.describe()
```

	공급량	시간
count	368088.000000	368088.000000
mean	948.100037	12.500000
std	927.211578	6.922196
min	1.378000	1.000000
25%	221.973000	6.750000
50%	637.014000	12.500000
75%	1398.919000	18.250000
max	11593.617000	24.000000

데이터 설명

열 : 4

행 : 368,087

평균 공급량 = 약948

표준편차 = 927

최소 = 1.37

최대 = 11593.62

수요 예측 모델 - XGBoost

파이썬 코드 & 코드(함수) 설명

알파벳으로 되어있는 구분을 분석을 위해
수치형으로 변환, 변수 명들 또한 변환

```
[60]  1 Total.tail()
```

	연월일	구분	공급량	시간
368083	2018-12-31	6	681.033	20
368084	2018-12-31	6	669.961	21
368085	2018-12-31	6	657.941	22
368086	2018-12-31	6	610.953	23
368087	2018-12-31	6	560.896	24

```
[61]  1 Total.describe()
```

	공급량	시간
count	368088.000000	368088.000000
mean	948.100037	12.500000
std	927.211578	6.922196
min	1.378000	1.000000
25%	221.973000	6.750000
50%	637.014000	12.500000
75%	1398.919000	18.250000
max	11593.617000	24.000000

데이터 설명

열 : 4

행 : 368087

평균 공급량 = 약948

표준편차 = 927

최소 = 1.37

최대 = 11593.62

수요 예측 모델 - XGBoost

파이썬 코드 & 코드(함수) 설명

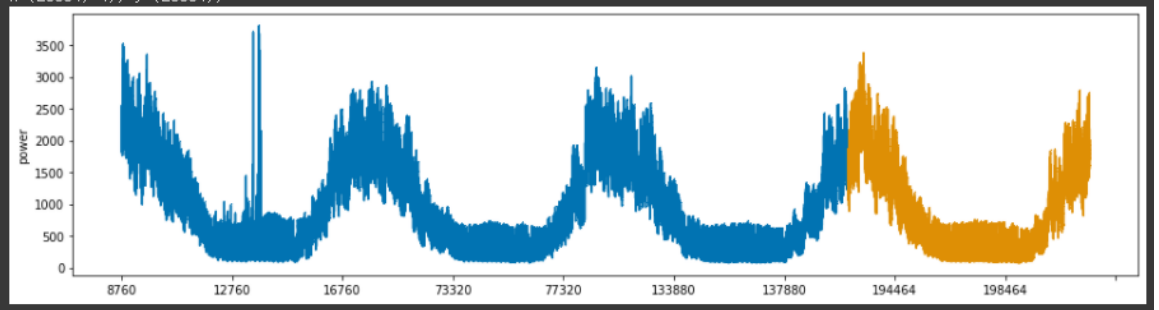
```
1 # train test 분할
2 train = Total.loc[Total.index < 245450,:]
3 test  = Total.loc[Total.index >= 245450,:]
4 print(train.shape, test.shape)
```

```
(245450, 7) (122638, 7)
```

```
[100] 1 # 시계열 train valid 분할 -> 1년을 test로 잡음
      2 y_train, y_valid, x_train, x_valid = temporal_train_test_split(y = y, X = x, test_size = 8760)
```

```
[101] 1 # train test 시각화
      2 print('train data shape#nx:{}, y:{}'.format(x_train.shape, y_train.shape))
      3
      4 plot_series(y_train, y_valid, markers=[',' , ','])
      5 plt.show()
      6 tqdm.pandas()
```

```
train data shape
x:(26304, 4), y:(26304,)
```



Train test 분할

분석을 위해 train test 분할 2년치가 test이다.

Train 데이터 모델 학습을 위해 train, vallid로 분할 1년치

수요 예측 모델 - XGBoost

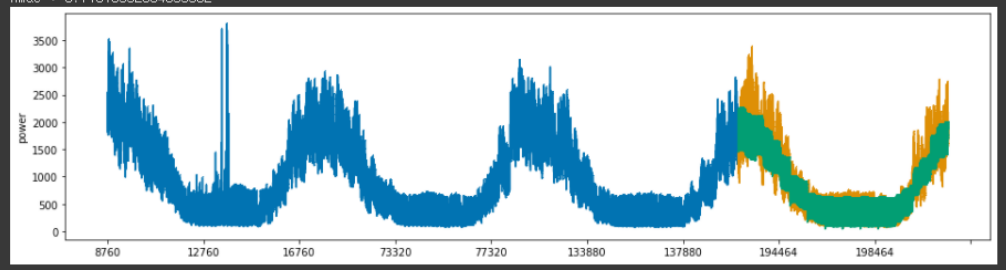
파이썬 코드 & 코드(함수) 설명

```
[111] 1 ## 파라미터 수정 전
      2 xgb_params = pd.read_csv('/content/hyperparameter_xgb.csv')
      3
      4 xgb_reg = XGBRegressor(n_estimators = 10000, objective ='reg:squarederror',eta = xgb_params.iloc[10,1], min_child_weight = xgb_params.iloc[10,2],
      5                        max_depth = xgb_params.iloc[10,3], colsample_bytree = xgb_params.iloc[10,4],
      6                        subsample = xgb_params.iloc[10,5], seed=0)
      7
      8 xgb_reg.fit(x_train, y_train, eval_set=[(x_train, y_train), (x_valid, y_valid)],
      9         early_stopping_rounds=300,
     10        verbose=False)

XGBRegressor(colsample_bytree=0.8, eta=0.01, n_estimators=10000,
             objective='reg:squarederror', seed=0, subsample=0.8)
```

```
[112] 1 ## 주황색이 실제 가스공급량, 초록색이 예측값
      2 pred = xgb_reg.predict(x_valid)
      3 pred = pd.Series(pred)
      4 pred.index = np.arange(y_valid.index[0], y_valid.index[-1]+1)
      5 plot_series(y_train, y_valid, pd.Series(pred), markers=[',' , ',', ','])
      6
      7 print('best iterations: {}'.format(xgb_reg.best_iteration))
      8 print('nmae : {}'.format(nmae(y_valid, pred)))

best iterations: 81
nmae : 0.17873552034359502
```



코드 설명

발전소 1을 기준으로,
기본 파라미터로 모델을 실행

(Make Scorer 함수 ➔ 작을수록 좋게 설정함)

<결과> Best Iteration = 81
NMAE값 = 0.1787

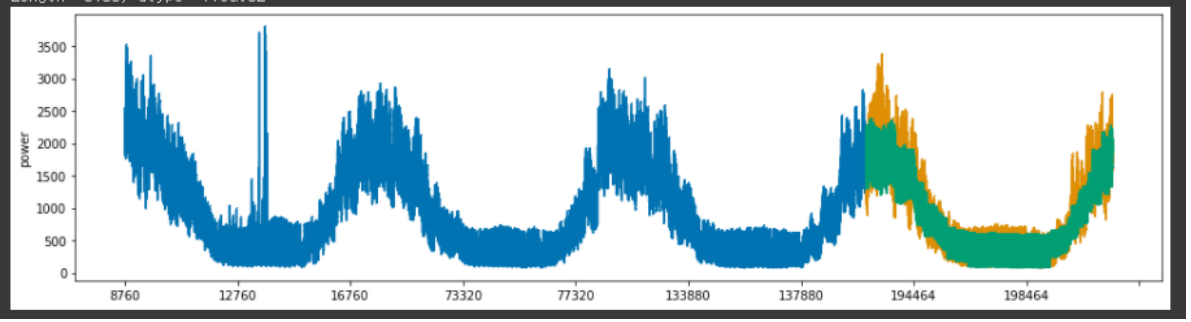
<해석> 최적의 반복 학습 횟수는 81회,
NMAE값은 0.1787로 측정

수요 예측 모델 - XGBoost

파이썬 코드 & 코드(함수) 설명

```
[81]  1 prndex = np.arange(y_valid.index[0], y_valid.index[-1]+1)
      2 plot_series(y_train, y_valid, pd.Series(pred), markers=[',' , ',', ','])
      3
      4 print('best iterations: {}'.format(xgb_reg.best_iteration))
      5 print('nmae : {}'.format(nmae(y_valid, pred)))
      6 pred = pd.Series(pred)
      7 pred

best iterations: 49
nmae : 0.15578044469959362
192768    1606.314941
192769    1357.977417
192770    1289.824951
192771    1318.659058
192772    1476.585327
             ...
201523    2045.326416
201524    2045.326416
201525    2038.978638
201526    1956.527710
201527    1880.114868
Length: 8760, dtype: float32
```



하이퍼 파라미터 수정후
-by. Grid Search CV

〈결과〉 Best Iteration = 49
NMAE값 = 0.1557

〈해석〉 최적의 반복횟수가
81 ➔ 49로 줄었고,
NMAE값이
0.1787 ➔ 0.1557로 줄음.

수요 예측 모델 - XGBoost

파이썬 코드 & 코드(함수) 설명

```
preds = np.array([])
for i in tqdm(range(6)):

    pred_df = pd.DataFrame()   # 시드별 예측값을 담을 data frame

    for seed in [0,1,2,3,4,5]: # 각 시드별 예측
        y_train = train.loc[train.company == i+1, 'power']
        x_train, x_test = train.loc[train.company == i+1, ].iloc[:, 3:], test.loc[test.company == i+1, ].iloc[:,1:]
        x_test = x_test[x_train.columns]

        xgb = XGBRegressor(seed = seed, n_estimators = 10000,objective ='reg:squarederror', eta = 0.01,
                           min_child_weight = 6, max_depth =6,
                           colsample_bytree=0.8, subsample= 0.8)



        xgb.fit(x_train, y_train)
        y_pred = xgb.predict(x_test)
        pred_df.loc[:,seed] = y_pred   # 각 시드별 예측 담기

    pred = pred_df.mean(axis=1)        # (i+1)번째 발전소의 예측 =  (i+1)번째 발전소의 각 시드별 예측 평균값
    preds = np.append(preds, pred)
```

```
100%|██████████| 6/6 [2:02:09<00:00, 1221.55s/it]
```

Seed Ensemble

- 시드별로 예측값이 조금씩 바뀌었다

➔시드의 영향을 제거하기 위해 6개의 seed(0~5)별로 훈련, 예측하여 6개의 예측값의 평균으로 계산

수요 예측 모델 - XGBoost

파이썬 코드 & 코드(함수) 설명 & 시각화

```
preds = pd.Series(preds)

fig, ax = plt.subplots(60, 1, figsize=(100,200), sharex = True)
ax = ax.flatten()
for i in range(6):
    train_y = train.loc[train.company == i+1, 'power'].reset_index(drop = True)
    test_y = preds[i*2160:(i+1)*2160]
    ax[i].scatter(np.arange(35064) , train.loc[train.company == i+1, 'power'])
    ax[i].scatter(np.arange(35064, 35064+2160) , test_y)
    ax[i].tick_params(axis='both', which='major', labelsize=6)
    ax[i].tick_params(axis='both', which='minor', labelsize=4)
plt.show()
```

시각화

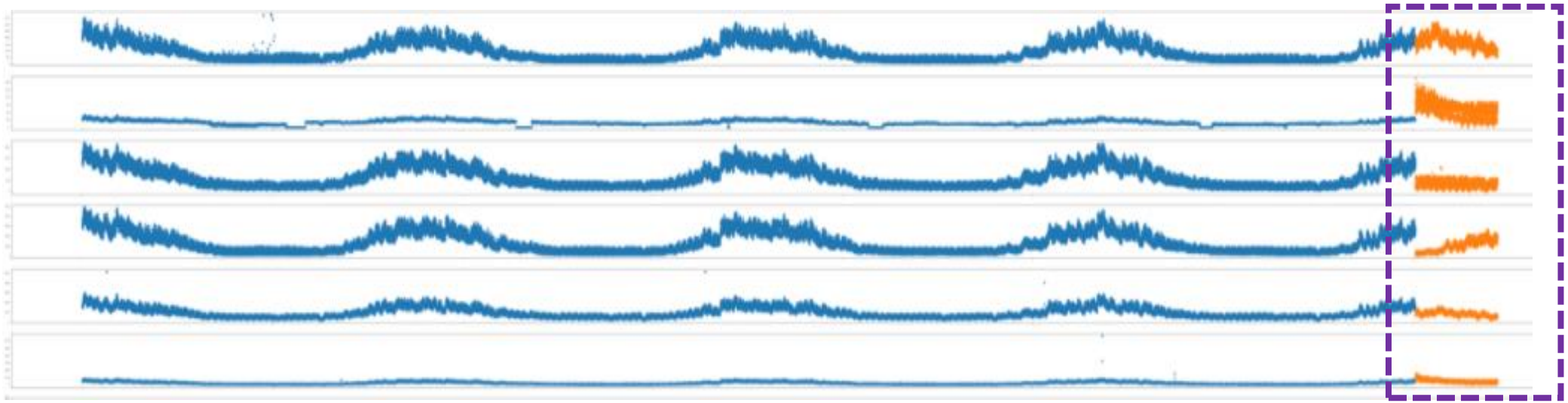
예측은 90일 * 24시간인 2160을 대입함.

<결과> 1번 발전소의 예측결과 좋음.

1번 발전소의 예측 plot

수요 예측 모델 - XGBoost

한계점



가스공급량 모델링 예측

- 최소제곱법을 이용한 시계열 분석(더미변수, 삼각함수)

수요 예측 모델(2) - 최소제곱법

더미변수 - 파이썬 코드 & 코드(함수) 설명

```
gas['time'] = range(1,len(gas)+1) ## time 변수 생성
y = gas['공급량']
season = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
     'Jul', 'Aug', 'Set', 'Oct', 'Nov']
temp_data = dict()
period = 12

for i, s in enumerate(season):
    temp = []
    for time in gas['time']:
        if time%period == i+1:
            temp_val = 1
        else:
            temp_val = 0
        temp.append(temp_val)
    temp_data[s] = temp

temp_df = pd.DataFrame(temp_data)
temp_df.index =
pd.date_range(start_date,end_date,freq='m')
X = temp_df
X = sm.add_constant(X)

# 파라미터 추정
X_tX_inv = np.linalg.inv(X.T.dot(X))
bs = X_tX_inv.dot(X.T.dot(y.values)) ## estimated
parameters
fitted_val = X.dot(bs) ## fitted values
```

```
# 예측값 및 예측구간 구하기
future = list(range(1,13))
n = len(gas)
alpha = 0.05
t_val = t.ppf(1-alpha/2,df=n-len(bs))

predict_vals = []
upper_limit = []
lower_limit = []
s2 = np.sum(np.square(y-fitted_val))/(n-len(bs))
for l in future:
    if l%period == 0:
        predict_val = bs[0]
    else:
        predict_val = bs[l%period]+bs[0]
    x = np.zeros(period)
    x[0] = 1
    if l%period != 0:
        x[l%period] = 1
    x = np.expand_dims(x,axis=1)
    variance_factor =
np.sqrt(1+x.T.dot(X_tX_inv.dot(x)))
    limit = t_val*np.sqrt(s2)*variance_factor[0][0]
    predict_vals.append(predict_val)
    upper_limit.append(predict_val+limit)
    lower_limit.append(predict_val-limit)
strt_date = '2019-01-01'0
```

더미변수 모델링

월별 더미변수 생성

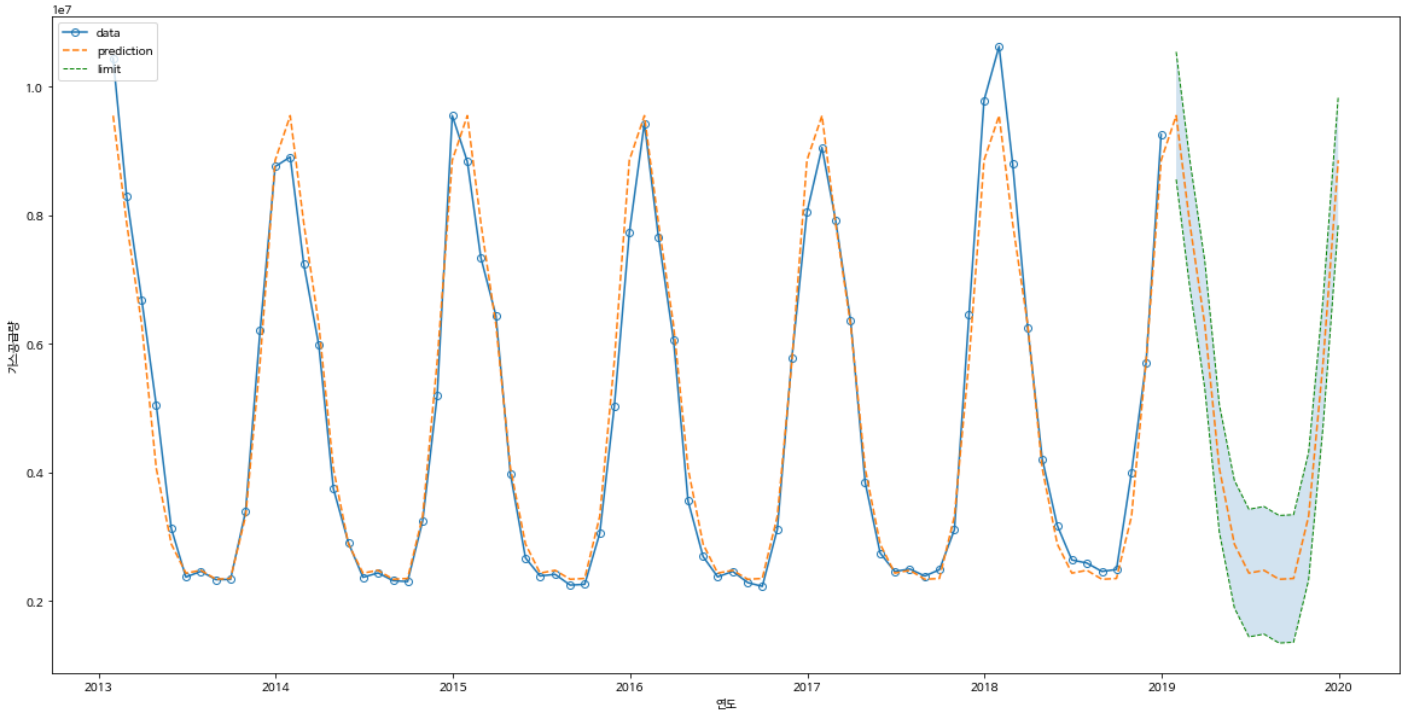
파라미터 추정

2013년 1월 ~ 2018년 12월

의 적합값 추정

수요 예측 모델(2) - 최소제곱법

더미변수 모델 - 시각화



더미변수 모델 시각화

초록색 점선 = 최대 최소의 한계
파란색 너비 : 예측값

수요 예측 모델(1)

삼각함수 - 파이썬 코드 & 코드(함수) 설명

```
gas['time'] = range(1,len(gas)+1) ## time 변수
생성
f = 12
sin = []
cos = []
for time in gas['time']:
    angle = 2*np.pi*time/f

    sin.append(round(np.sin(angle),5))
    cos.append(round(np.cos(angle),5))

temp_data = {
    'sin' : sin,
    'cos' : cos
}
temp_df = pd.DataFrame(temp_data)
temp_df.index =
pd.date_range(start_date,end_date,freq='m')
X = sm.add_constant(temp_df)

# 최소 제곱 추정량과 적합값 구하기
X_tX_inv = np.linalg.inv(X.T.dot(X))
bs = X_tX_inv.dot(X.T.dot(y.values)) ## estimated
parameters
fitted_val = X.dot(bs) ## fitted values
```

```
print(bs)

future = list(range(1,13))
n = len(gas)
alpha = 0.05

t_val = t.ppf(1-alpha/2,df=n-len(bs))

predict_vals = []
upper_limit = []
lower_limit = []
s2 = np.sum(np.square(y-fitted_val))/(n-len(bs))
for l in future:
    sin_val = round(np.sin(2*np.pi*l/f),5)
    cos_val = round(np.cos(2*np.pi*l/f),5)
    x = np.array([1,sin_val,cos_val])
    x = np.expand_dims(x,axis=1)
    variance_factor =
np.sqrt(1+x.T.dot(X_tX_inv.dot(x)))
    limit =
t_val*np.sqrt(s2)*variance_factor[0][0]
    predict_val = x.T.dot(bs)[0]
    predict_vals.append(predict_val)
    upper_limit.append(predict_val+limit)
    lower_limit.append(predict_val-limit)
```

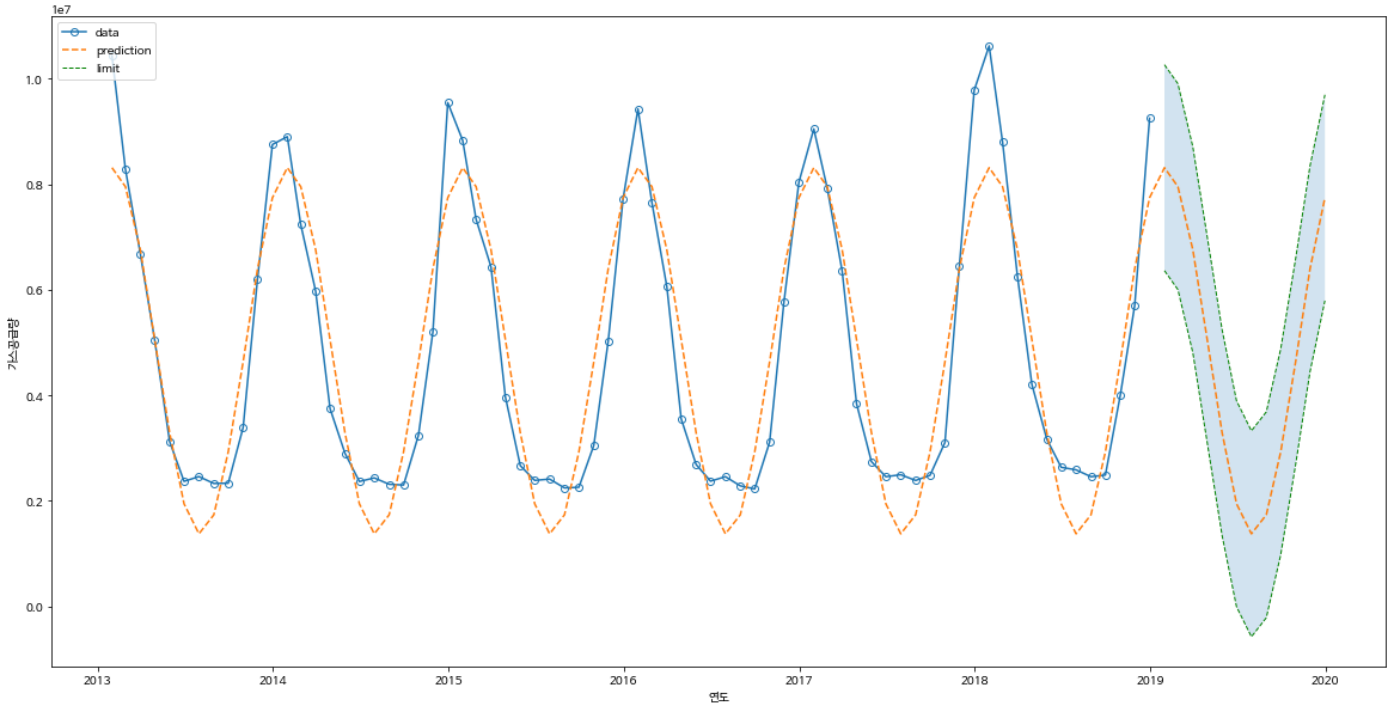
삼각함수 모델링

최소 제곱 추정량&적합값 구하기

2019년 1월~ 12월의 예측값 & 예측구간 구하기

수요 예측 모델(2) - 최소제곱법

삼각함수 모델 - 시각화



삼각함수 모델 시각화

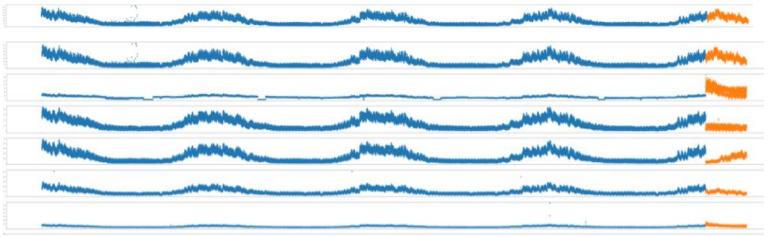
초록색 점선 = 최대 최소의 한계
파란색 너비 : 예측값

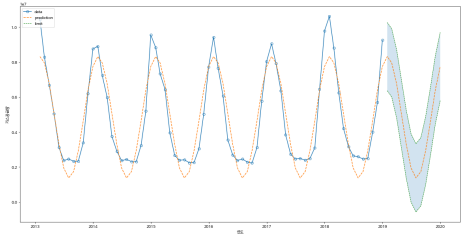
결론 및 한계점

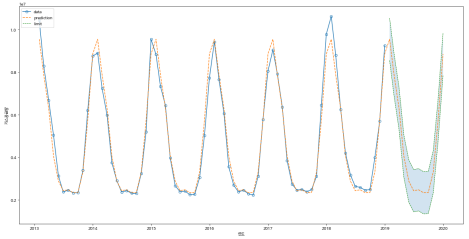
결론 & 한계점

1. 공급량과 기후간의 상관관계를 분석해본 결과, 기온과 강한 음의 상관관계를 갖고있음을 확인함.
2. 기온이 높은 6~9월에 가스 공급량이 낮았고, 기온이 낮은 11~2월에 가스 공급량이 높았던 것으로 보아, 가스공급량은 계절성을 띄는 것을 확인함.
3. XGBoost와 최소제곱법을 통해 모델을 예측한 결과, 아래와 같은 결과를 도출해낼 수 있었음.

<한계> 여러 외부데이터를 변수로 설정하여 패턴을 분석하였으나 특별히 의미 있다는 결론을 내리지는 못하였음.








감사합니다.