전력 수요예측 시뮬레이션

20200720

Contents

분석배경

LS산전 전력 관련 issue

LS산전 전력·자동화 제품, 독일 iF 디자인 어워드 수상

연합뉴스 | 2020.02.18. | 네이버뉴스

LS산전 수상 제품은 차세대 **전력** 솔루션 수솔 스마트 배선용차단기(Susol Smart MCCB)와 자동화 핵심 제품 iXP2(HMI·Human-Machine Interface)다. 수솔 스마트 배선용차단기는 공장이나 빌딩 내부 각종 장비와 주요 설비에서...

- **LS산전**, ㅞ iF 디자인 어워드서 본… ZDNet Korea | 2020.02.18. | 네이버뉴스
- **LS산전 전력**·자동화 제품, 독일 iF… 국민일보 | 2020.02.18. | 네이버뉴스
- **LS산전** 차단기 ㅞ 3대 디자인상 수… 파이낸셜뉴스 | 2020.02.18. | 네이버뉴스
- **LS산전**, '세계 3대 디자인상' 독일 i… 뉴스1 | 2020.02.18. | 네이버뉴스

관련뉴스 28건 전체보기 ›

LS산전-한전 **전력**硏, 세계 최대 '직류 자립섬' 조성

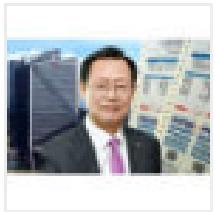
헤럴드경제 | 11면1단 | 2019.08.19. | 네이버뉴스

한전 **전력**연구원과 **LS산전**은 서거차도에 기존의 디젤발전기를 대신해 200kW급 태양광, 100kW급 풍력발전, 1.5MWh급 에너지저장장치(ESS) 등 직류 전기를 생산하고 저장하는 신재생에너지 전원(電源)을 구축했다. 이와 함께...

- KEPRI·**LS산전**, 세계 최대 '직류 에… 조선비즈 pick | 2019.08.19. | 네이버뉴스
- **LS산전**-한전 **전력**연구원, 세계 …
- **LS산전** '직류 전기섬' 불밝혔다

관련뉴스 4건 전체보기 ›

관련도순 최신순 오래된순 검색결과 자동고침 시작



한국**전력**, 그린뉴딜에 전기요금체계 개편 명분 얻어, 관건은 시기조율

비즈니스포스트 | 1일 전

전기요금이 연료비와 연동되지 않은 현 제도 아래서 기존 원자력이나 석탄 발전방식보다 **전력** 도매가격(**SMP**)이 높은 신재생에너지를 늘리면 한국**전력**의 원가 부담은 더욱 커질 수밖에 없다. 한국**전력**은...

#그린뉴딜

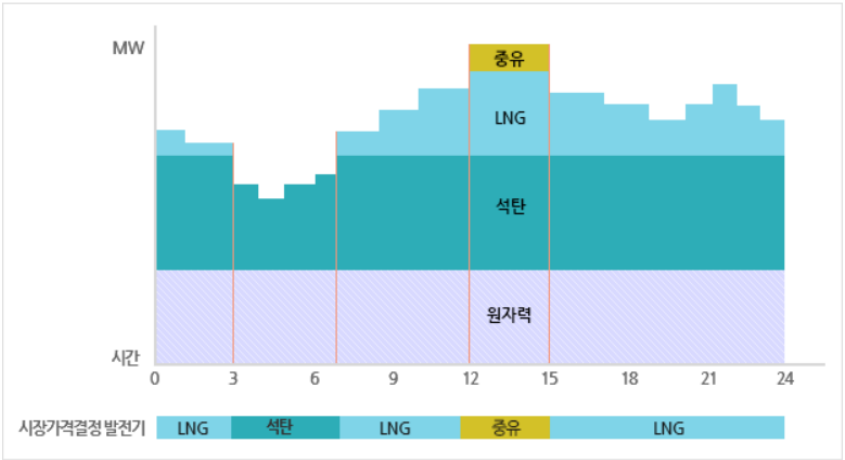
#SMP(도매가)

#코로나

01 분석배경

전력 수요예측과
정전 방지를 위한 예비전력 확보방안 분석

기상 · 전력 · SMP



지역별 날씨데이터

기온/ 강수량

내용 입력란

섬 / 육지

한계가격

시장가격 결정을 위한 "발전계획 프로그램"은 공급입찰에 참여한 발전기의 비용 최소화 원칙에 따라 발전기 가동여부와 발전출력을 결정하게 되는데,
이 중 가장 높은 발전비용의 발전기를 한계가격 결정 발전기(MARGINAL PLANT)로 처리하고
이 한계가격(SMP : SYSTEM MARGINAL PRICE)을 그 시간대의 시장가격으로 결정

분석 목표

DACON
DATA TO VALUE

한국전력연구원

2020.05.25 ~ 2020.06.21 의 전력수요 및 SMP 예측

7일 후를 예측하는 모델을 예로 들자면,

" 18년 5월 1일부터 5월 31일까지의 데이터 = 6월 7일의 평균 SMP 값,
18년 5월 2일부터 6월 1일 까지의 데이터 = 6월 8일의 평균 SMP 값,
...
20년 5월 1일부터 6월 1일까지의 데이터 = 20년 6월 7일의 평균 SMP값 "

분석 브레인스토밍

고려해 볼 종속변수 및 조건

LNG 가격
유류 가격
지역정보 – 육지? 제주도?
신재생 에너지 거래량/ 가격
날씨 지표

데이터전처리 및 EDA

01 DATA 셋 소개

해결해야 하는 문제

한국전력연구원

- 일주일 후 28일간의 일자별 전력수급실적(전력수요량) 및 SMP 최대, 최소, 가중평균 예측

데이터 설명

target_v2.csv

- 데이터 기간: 2018.02.01 ~ 2020.05.18 - 일자별 전력수급실적 및 SMP 최대, 최소, 가중평균 데이터
-

weather.csv

- 시간별 기상 데이터 (0시 ~ 23시 단위)
- QCFlag 종류: 0(정상), 1(오류), 9(결측)

sample_submission.csv

- 제출

01 DATA 셋 소개

```
In [4]: #날짜마다 smp, supply 량을 알 수 있음
target
```

Out [4]:

	date	smp_max	smp_min	smp_mean	supply
0	2018-02-01	150.65	116.84	132.71	87.47
1	2018-02-02	163.86	116.84	134.19	86.64
2	2018-02-03	164.07	116.85	131.39	88.28
3	2018-02-04	171.00	115.76	131.89	86.14
4	2018-02-05	170.34	123.89	137.96	90.63
...	...	...	...	...	...
833	2020-05-14	193.28	66.78	100.46	62.70
834	2020-05-15	198.23	61.81	102.38	64.91
835	2020-05-16	220.91	88.50	121.19	61.75
836	2020-05-17	207.75	65.78	116.82	61.55
837	2020-05-18	113.31	66.86	98.98	63.91

838 rows × 5 columns

target.csv
- **일자별 전력수급 실적 및 SMP 최대, 최소, 가중평균 데이터**

DATA 셋 소개

weather.csv
- 지역별, 시간별 기상 데이터 (0시 ~ 23시 단위)
- QCFlag 종류: 0(정상) 1(오류), 9(결측)

기온 강수량

In [6]:
```
weather
```

Out [6]:

	area	datetime	temp	temp_QCFlag	prec	prec_QCFlag	ws	ws_QCFlag	wd	wd_QCFlag	...	vis	sfctype	wc
0	184	2018-02-01 01:00	4.7	0.0	NaN	NaN	3.6	0.0	20.0	0.0	...	1950.0	NaN	
1	184	2018-02-01 02:00	4.8	0.0	NaN	NaN	2.6	0.0	360.0	0.0	...	1865.0	NaN	
2	184	2018-02-01 03:00	4.8	0.0	NaN	NaN	4.6	0.0	20.0	0.0	...	1855.0	NaN	
3	184	2018-02-01 04:00	4.5	0.0	NaN	NaN	5.7	0.0	20.0	0.0	...	1425.0	NaN	
4	184	2018-02-01 05:00	4.5	0.0	NaN	9.0	4.4	0.0	20.0	0.0	...	1043.0	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
746825	893	2020-05-18 19:00	16.4	NaN	0.5	NaN	4.1	NaN	302.9	NaN	...	NaN	NaN	
746826	893	2020-05-18 20:00	15.8	NaN	0.0	NaN	3.1	NaN	311.6	NaN	...	NaN	NaN	
746827	893	2020-05-18 21:00	15.1	NaN	0.0	NaN	0.0	NaN	0.0	NaN	...	NaN	NaN	
746828	893	2020-05-18 22:00	14.6	NaN	0.0	NaN	0.2	NaN	0.0	NaN	...	NaN	NaN	
746829	893	2020-05-18 23:00	13.6	NaN	0.0	NaN	1.2	NaN	221.4	NaN	...	NaN	NaN	

746830 rows × 37 columns

In [147]:
```
import time
oil_price_du = get_oil_price('OIL_DU')
oil_price_wti = get_oil_price('OIL_CL')
oil_price_brent = get_oil_price('OIL_BRT')
```

```
[2020/07/19 11:57:38] 데이터 수집을 시작합니다. (code: OIL_DU)
[2020/07/19 11:57:45] 데이터 수집을 종료합니다. (code: OIL_DU, 수집시간: 7초, 데이터수: 577개)
[2020/07/19 11:57:45] 데이터 수집을 시작합니다. (code: OIL_CL)
[2020/07/19 11:57:54] 데이터 수집을 종료합니다. (code: OIL_CL, 수집시간: 8초, 데이터수: 578개)
[2020/07/19 11:57:54] 데이터 수집을 시작합니다. (code: OIL_BRT)
[2020/07/19 11:58:03] 데이터 수집을 종료합니다. (code: OIL_BRT, 수집시간: 8초, 데이터수: 588개)
```

In [148]:
```
oil_list = []
oil_list.append(oil_price_du)
oil_list.append(oil_price_wti)
oil_list.append(oil_price_brent)
code_list = ['OIL_DU','OIL_CL','OIL_BRT']

for i, code in enumerate(code_list):
    oil_list[i].drop(['code'],axis=1,inplace=True)
    oil_list[i].columns = ['date',str(code)+'_price']
```

In [149]:
```
start = '2018-02-01'
end = '2020-05-18'
daily_price_oil = pd.DataFrame(columns = ['date'])
date_range = pd.date_range(start,end,freq = 'D')
daily_price_oil['date'] = date_range
```

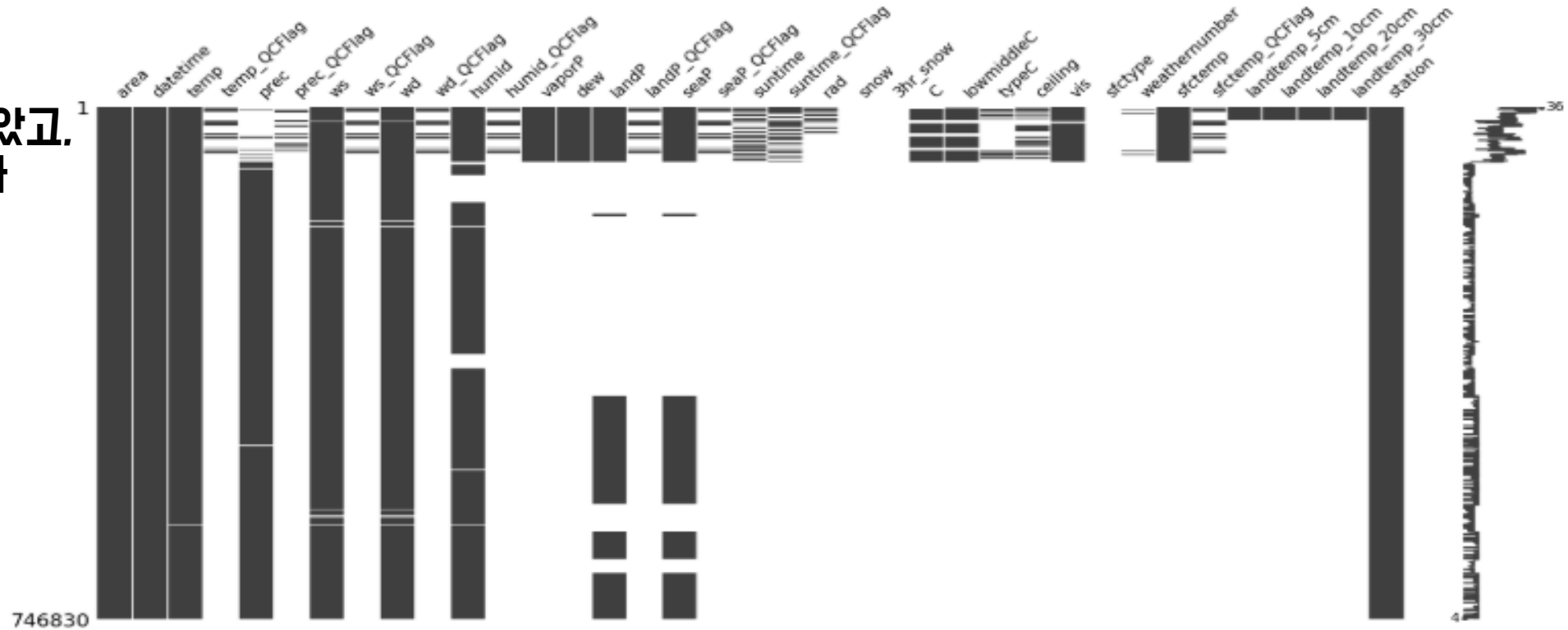
In [150]:
```
for d in oil_list:
    daily_price_oil = daily_price_oil.merge(d,how='outer')
```

In [151]:
```
daily_price_oil['mean'] = round(daily_price_oil.mean(axis=1),2)
daily_price_oil = daily_price_oil.loc[:,['date','mean']]
daily_price_oil.columns = ['date','oil_price']
```

전처리 -weather

In [5]:
```
# 결측값 확인 missingno 라이브러리를 통해서( 하얀색이 많을 수록 결측치가 없는거임.)
import missingno as msno
msno.matrix(weather)
```

Out [5]: <matplotlib.axes._subplots.AxesSubplot at 0x2c23b267cc8>



전체적으로 결측값이 많았고.
그 중 `온도` 라는 변수가
가장 결측치가 없었음
-> 일자 별 통합

온도가 결측치가 제일 없음. 다른 결측치가 없는 날씨 요소는 매일 측정되지 않는 요소

분석 목표

DACON
DATA TO VALUE

한국전력연구원

2020.05.25 ~ 2020.06.21 의 전력수요 및 SMP 예측

7일 후를 예측하는 모델을 예로 들자면,

" 18년 5월 1일부터 5월 31일까지의 데이터 = 6월 7일의 평균 SMP 값,
18년 5월 2일부터 6월 1일 까지의 데이터 = 6월 8일의 평균 SMP 값,
...
20년 5월 1일부터 6월 1일까지의 데이터 = 20년 6월 7일의 평균 SMP값 "

전처리

```
In [74]: # 결측치는 1시간 후 온도로 처리
         hourly_temperature['temperature'][hourly_temperature['temperature'].isna()] = hourly_temperature.shift(-1)['te
```

```
In [160]: hourly_temperature['temperature'].isnull().sum()
Out[160]: 0
```

```
In [122]: #전체 값을 평균값으로, target data가 날짜별로 되어있으니 평균값으로 concat
          weather['date'] = weather['datetime'].apply(lambda x : x[:10])
          weather_mean = weather.groupby('date').mean().reset_index(drop = True)
```

일자별로 통합(평균) 결측치-1시간 후 데이터로 (838)

```
In [124]: weather_mean
Out[124]:
```

	area	temp	temp_QCFlag	prec	prec_QCFlag	ws	ws_QCFlag	wd	wd_QCFlag	humi
0	714.270270	1.271798	0.0	0.003444	8.100000	3.523141	0.0	175.879137	0.0	67.42345
1	714.270270	0.419820	0.0	0.000000	9.000000	3.852709	0.0	216.583973	0.0	57.45858
2	711.137615	-1.787600	0.0	0.289295	7.024390	5.602110	0.0	269.011723	0.0	76.50167
3	714.270270	-3.873874	0.0	0.270012	6.483871	5.975114	0.0	273.509932	0.0	79.00012
4	711.048276	-2.729229	0.0	0.243742	6.656250	5.134080	0.0	269.770637	0.0	77.27865
...	...	...	...	...	...	...	...	...	...	...
833	719.000000	16.911513	NaN	0.000000	NaN	3.341164	NaN	146.815038	NaN	54.06273
834	719.000000	18.643202	NaN	1.988746	9.000000	3.897588	NaN	173.301864	NaN	92.25381
835	718.531353	17.191510	NaN	0.168819	9.000000	1.884323	NaN	199.394625	NaN	91.19844
836	719.000000	16.407675	NaN	0.000000	9.000000	1.934122	NaN	162.449099	NaN	80.91214
837	719.000000	17.885307	0.0	0.303318	9.000000	3.222185	NaN	169.713401	NaN	91.48267

838 rows × 34 columns

데이터 추가 수집-1

DATA 공공데이터포털 .GO.KR

일자별 통합, 날짜만 있음
예측의 정확성 높이기 위해
국경일/ 공휴일 정보
- 일자에 대한 정보
- 공공데이터 포털 api

```
In [135]:
# 열 이름 변경
names = target.columns.tolist()
names[names.index('name_x')] = 'holidays'
target.columns = names
# 공휴일 특성 이진수로 변경
target['holidays'].fillna(0, inplace = True)
```

```
In [140]:
target
```

Out[140]:

	date	smp_max	smp_min	smp_mean	supply	year	month	day	dayofweek	holidays	na
0	2018-02-01	150.65	116.84	132.71	87.47	2018	2	1	3	0	
1	2018-02-02	163.86	116.84	134.19	86.64	2018	2	2	4	0	
2	2018-02-03	164.07	116.85	131.39	88.28	2018	2	3	5	0	
3	2018-02-04	171.00	115.76	131.89	86.14	2018	2	4	6	0	
4	2018-02-05	170.34	123.89	137.96	90.63	2018	2	5	0	0	
...	...	...	...	...	...	...	...	...	...	...	...
833	2020-05-14	193.28	66.78	100.46	62.70	2020	5	14	3	0	
834	2020-05-15	198.23	61.81	102.38	64.91	2020	5	15	4	0	
835	2020-05-16	220.91	88.50	121.19	61.75	2020	5	16	5	0	
836	2020-05-17	207.75	65.78	116.82	61.55	2020	5	17	6	0	
837	2020-05-18	113.31	66.86	98.98	63.91	2020	5	18	0	0	

EDA

기온이 가장 상관관계가 높음

In [126]:
```
# 날씨 지표와 상관관계 분석
#기온이 가장 상관관계가 높음을 알 수 있다.
pd.concat([target,weather_mean], axis = 1).corr().loc['smp_max':'supply','area':].abs().mean().sort_values(asc
```

Out[126]:

landtemp_30cm	0.323563
landtemp_5cm	0.300492
landtemp_20cm	0.293111
landtemp_10cm	0.292360
temp	0.291700
sfctemp	0.283049
dew	0.269303
vaporP	0.246189
landP	0.223619
seaP	0.182458
wd	0.177165
3hr_snow	0.167618
ws	0.157772
snow	0.156866

전처리

지역의 특성 반영하기 위해서 ASOS 지역으로 한정
+
기온 데이터만 남겨놓고 CONCAT

In [156]:
```
weather = weather[weather['station'] == 'ASOS']
#제주도 고산 성산 서귀포라고 함
```

In [157]:
```
#weathed  데이타 전처리 - temp만 남겨놓는다
weather['datetime'] = pd.to_datetime(weather['datetime'])
weather = weather.loc[:,'area':'temp']
```

In [158]:
```
#area별로 리스트에묶어야함
weather
```

Out[158]:

	area	datetime	temp
0	184	2018-02-01 01:00:00	4.7
1	184	2018-02-01 02:00:00	4.8
2	184	2018-02-01 03:00:00	4.8
3	184	2018-02-01 04:00:00	4.5
4	184	2018-02-01 05:00:00	4.5
...	...	...	...
80427	189	2020-05-18 19:00:00	19.0
80428	189	2020-05-18 20:00:00	19.8
80429	189	2020-05-18 21:00:00	18.9
80430	189	2020-05-18 22:00:00	18.1
80431	189	2020-05-18 23:00:00	17.7

최종 DATASET

In [82]:
```
#날짜별로 smp, temp, 최대최소평균
target
```

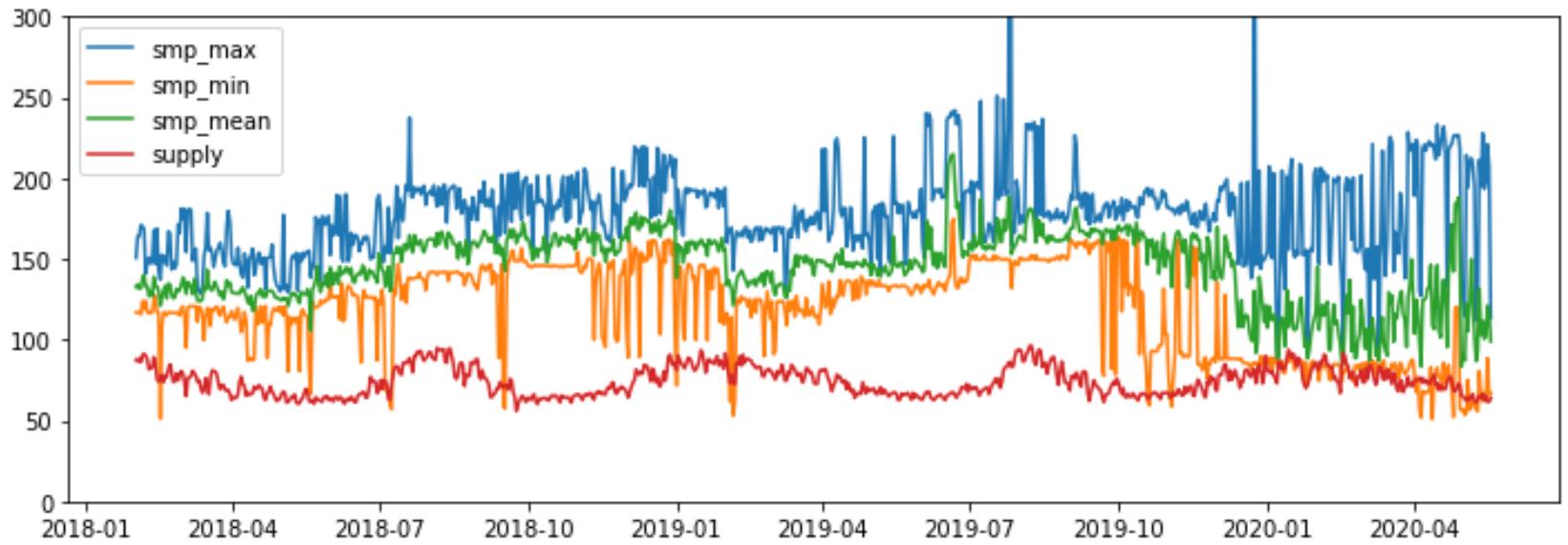
Out [82]:

	date	smp_max	smp_min	smp_mean	supply	year	month	day	dayofweek	temp_max	temp_min	temp_mean
0	2018-02-01	150.65	116.84	132.71	87.47	2018	2	1	3	5.95	3.30	4.298077
1	2018-02-02	163.86	116.84	134.19	86.64	2018	2	2	4	5.25	2.45	3.576923
2	2018-02-03	164.07	116.85	131.39	88.28	2018	2	3	5	4.00	-0.50	1.457692
3	2018-02-04	171.00	115.76	131.89	86.14	2018	2	4	6	0.30	-2.05	-1.000000
4	2018-02-05	170.34	123.89	137.96	90.63	2018	2	5	0	2.05	-1.70	0.015385
...	...	...	...	...	...	...	...	...	...	...	...	...
833	2020-05-14	193.28	66.78	100.46	62.70	2020	5	14	3	NaN	NaN	NaN
834	2020-05-15	198.23	61.81	102.38	64.91	2020	5	15	4	NaN	NaN	NaN
835	2020-05-16	220.91	88.50	121.19	61.75	2020	5	16	5	NaN	NaN	NaN
836	2020-05-17	207.75	65.78	116.82	61.55	2020	5	17	6	NaN	NaN	NaN
837	2020-05-18	113.31	66.86	98.98	63.91	2020	5	18	0	NaN	NaN	NaN

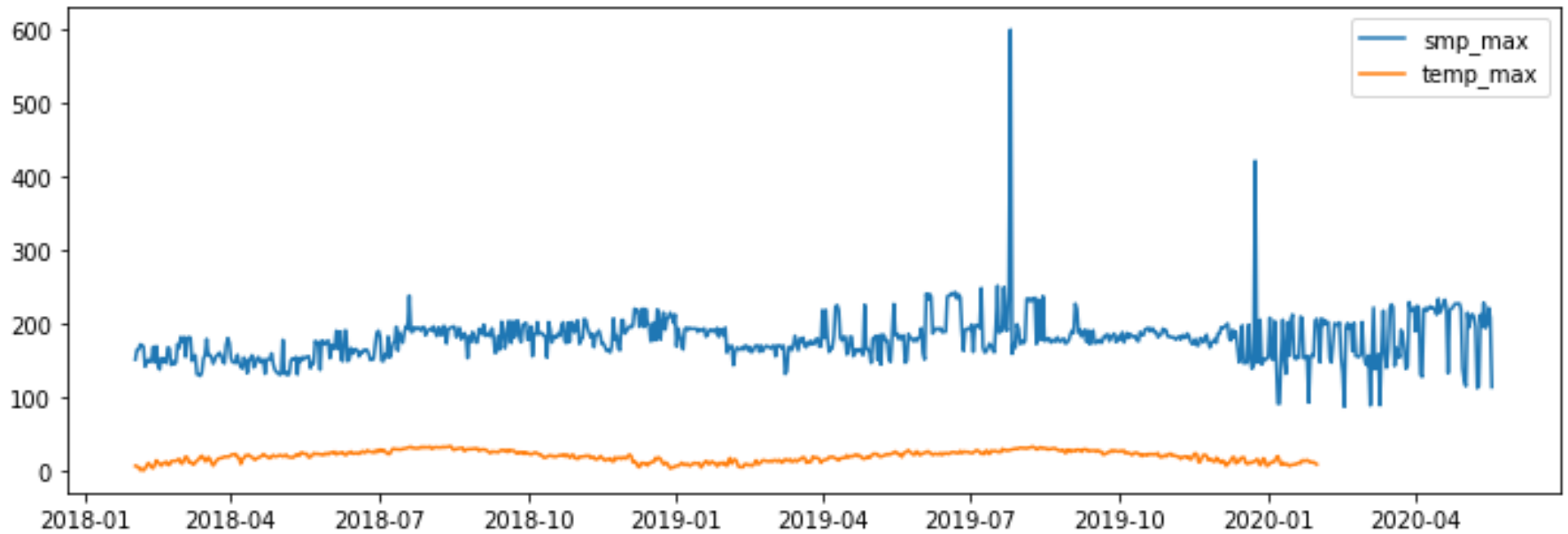
838 rows × 12 columns

EDA

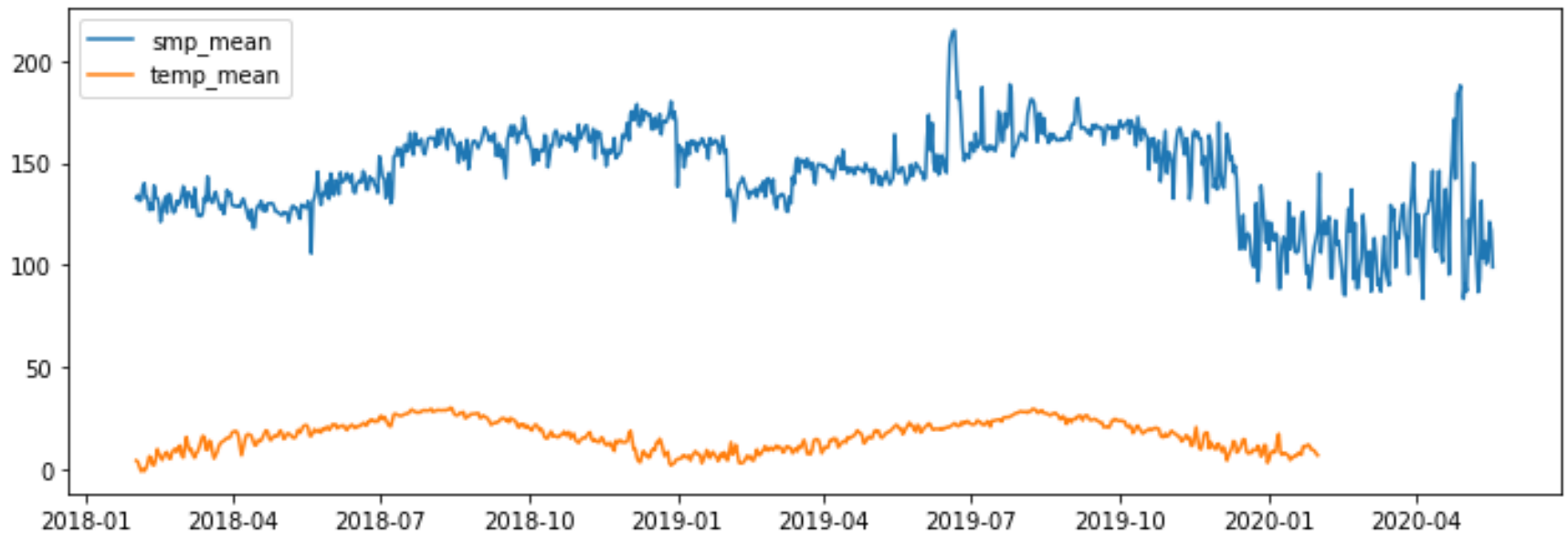
In [83]:
```
#날짜별 시각화
plt.rcParams['figure.figsize'] = [12, 4]
plt.plot(target.loc[:, 'date'], target.loc[:, 'smp_max'], label='smp_max')
plt.plot(target.loc[:, 'date'], target.loc[:, 'smp_min'], label='smp_min')
plt.plot(target.loc[:, 'date'], target.loc[:, 'smp_mean'], label='smp_mean')
plt.plot(target.loc[:, 'date'], target.loc[:, 'supply'], label='supply')
plt.ylim(0,300)
plt.legend()
plt.show()
```



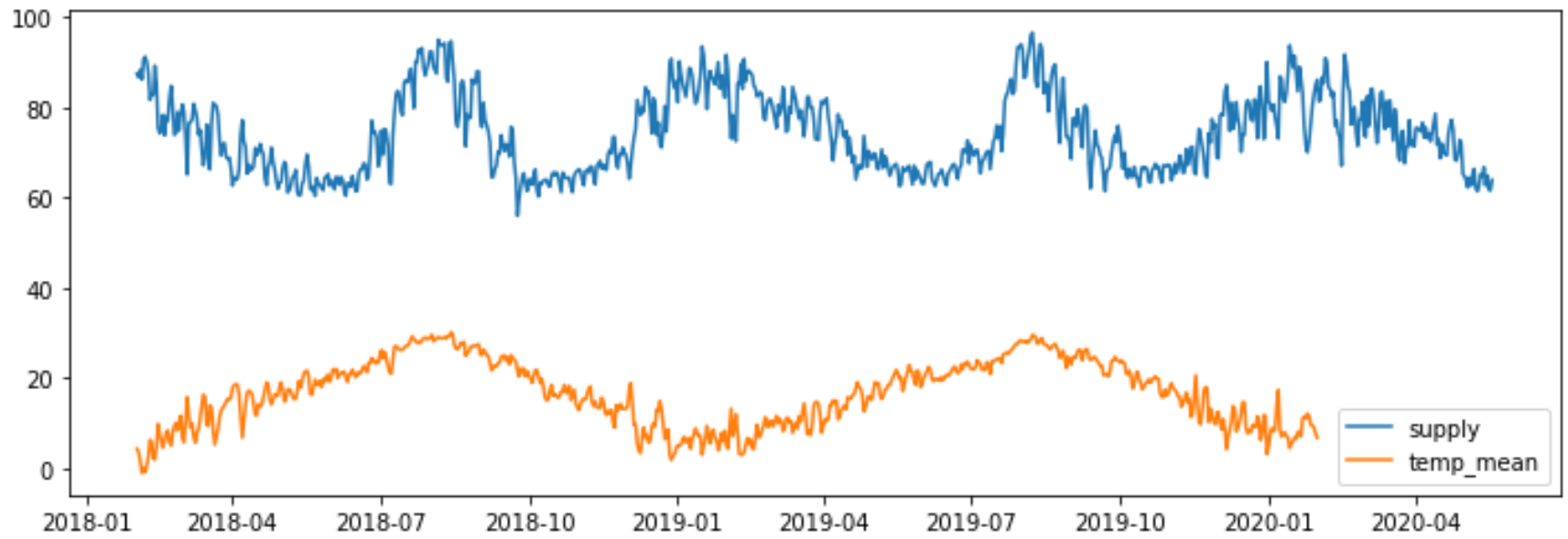
In [86]:
```
plt.rcParams['figure.figsize'] = [12, 4]
plt.plot(target.loc[:, 'date'], target.loc[:, 'smp_max'], label='smp_max')
plt.plot(target.loc[:, 'date'], target.loc[:, 'temp_max'], label='temp_max')
plt.legend()
plt.show()
```



In [85]:
```
plt.rcParams['figure.figsize'] = [12, 4]
plt.plot(target.loc[:, 'date'], target.loc[:, 'smp_mean'], label='smp_mean')
plt.plot(target.loc[:, 'date'], target.loc[:, 'temp_mean'], label='temp_mean')
plt.legend()
plt.show()
```

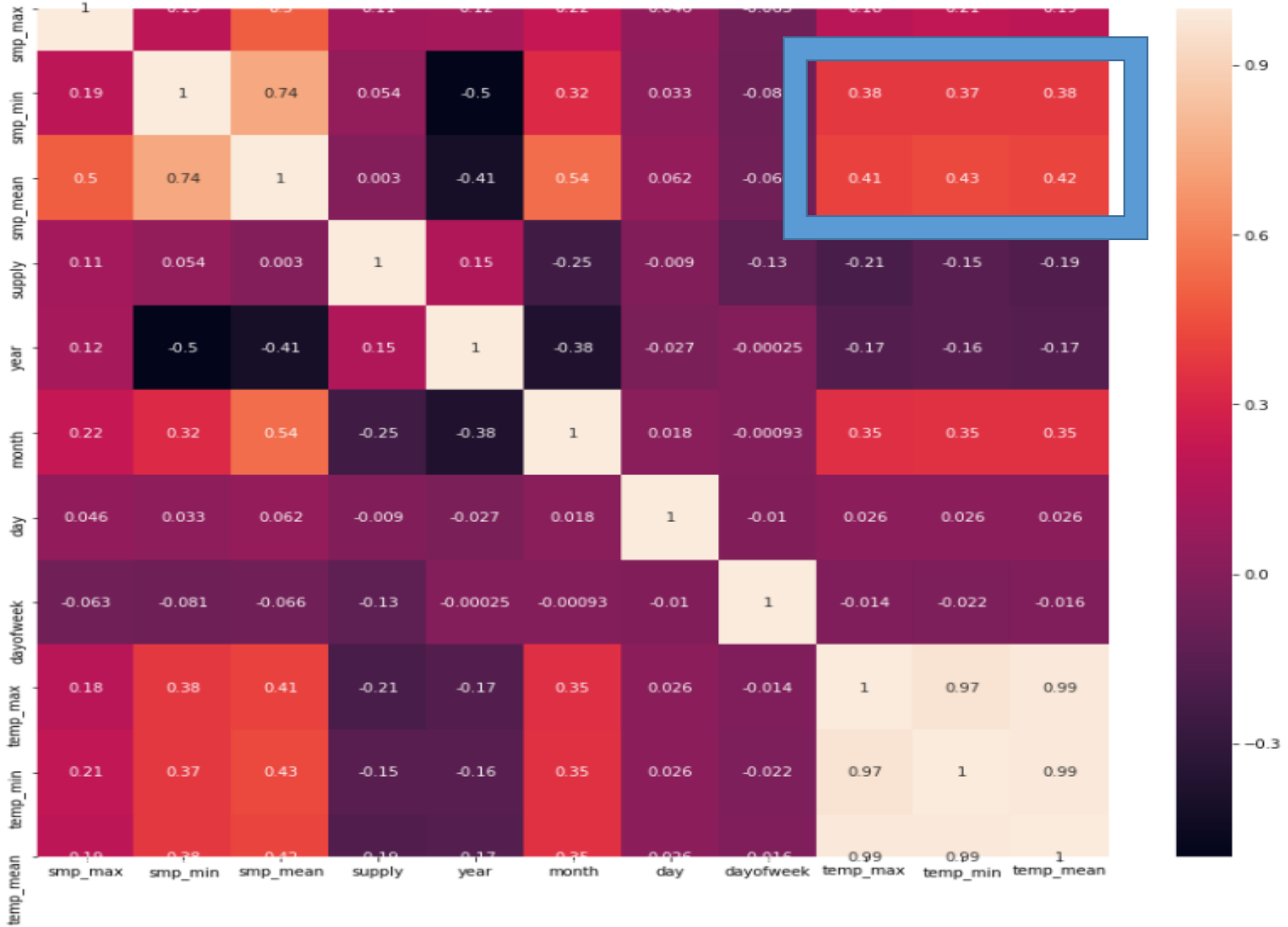


In [84]:
```
#온도와 시각화 - 월별로 추세가 다른것으로 보임 ( 상관관계가 있는 달과 아닌 달이 있음)
plt.rcParams['figure.figsize'] = [12, 4]
plt.plot(target.loc[:, 'date'], target.loc[:, 'supply'], label='supply')
plt.plot(target.loc[:, 'date'], target.loc[:, 'temp_mean'], label='temp_mean')
plt.legend()
plt.show()
```



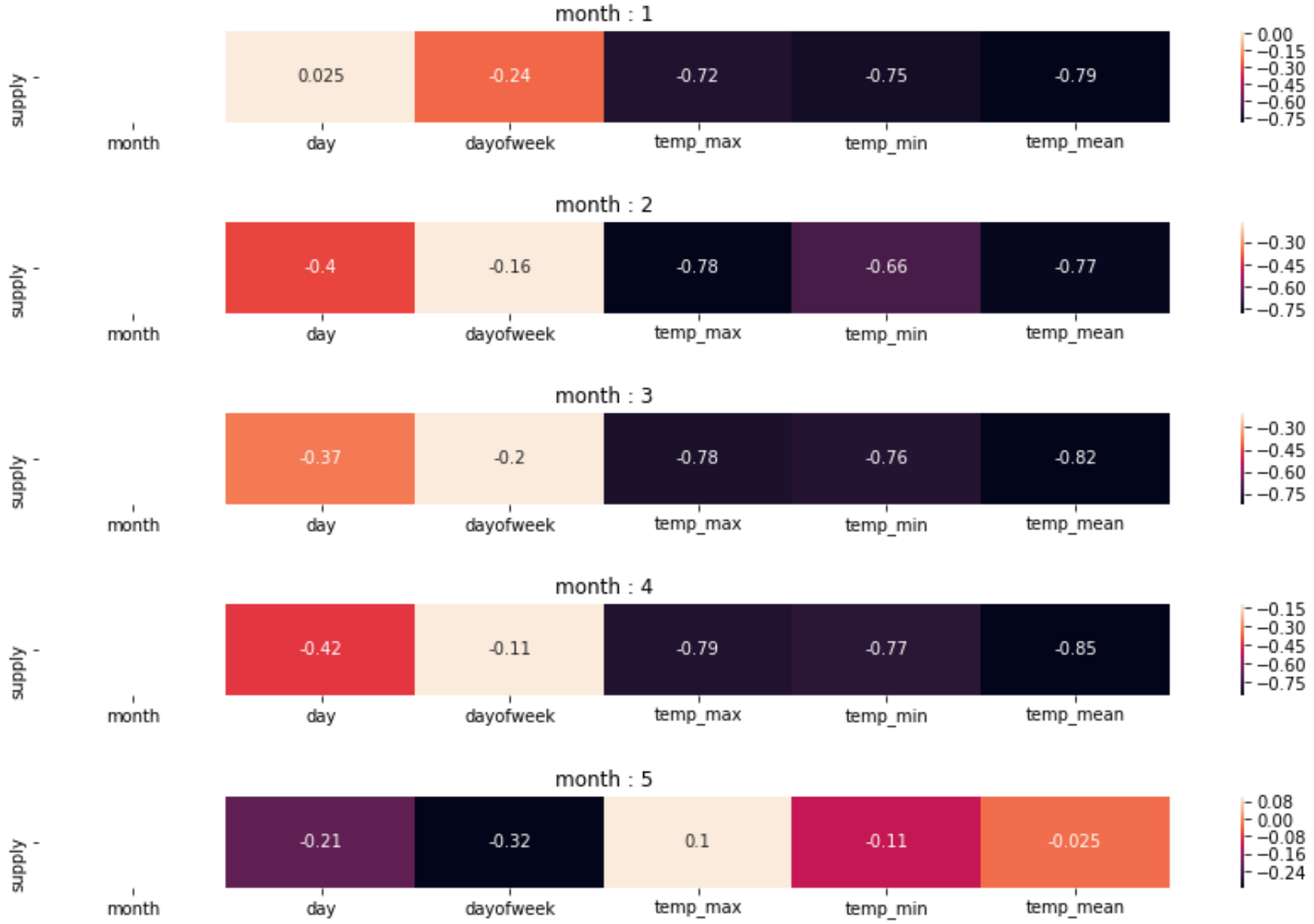
In [31]:
```
plt.rcParams['figure.figsize'] = [15, 13]
sns.heatmap(target.corr(), annot=True)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x1d70bd3d0c8>



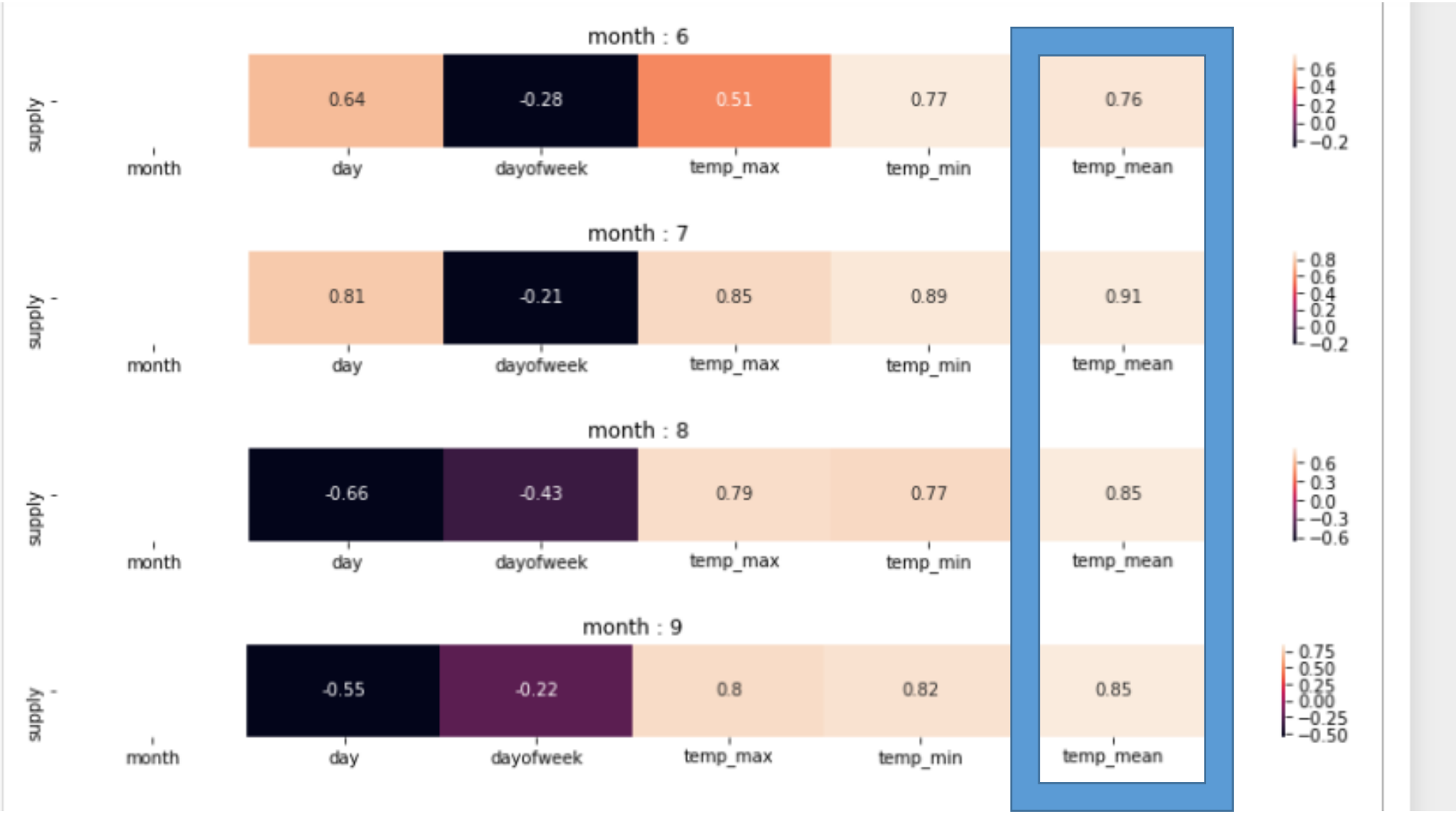
Supply

1~5월
-> 낮은
상관관계



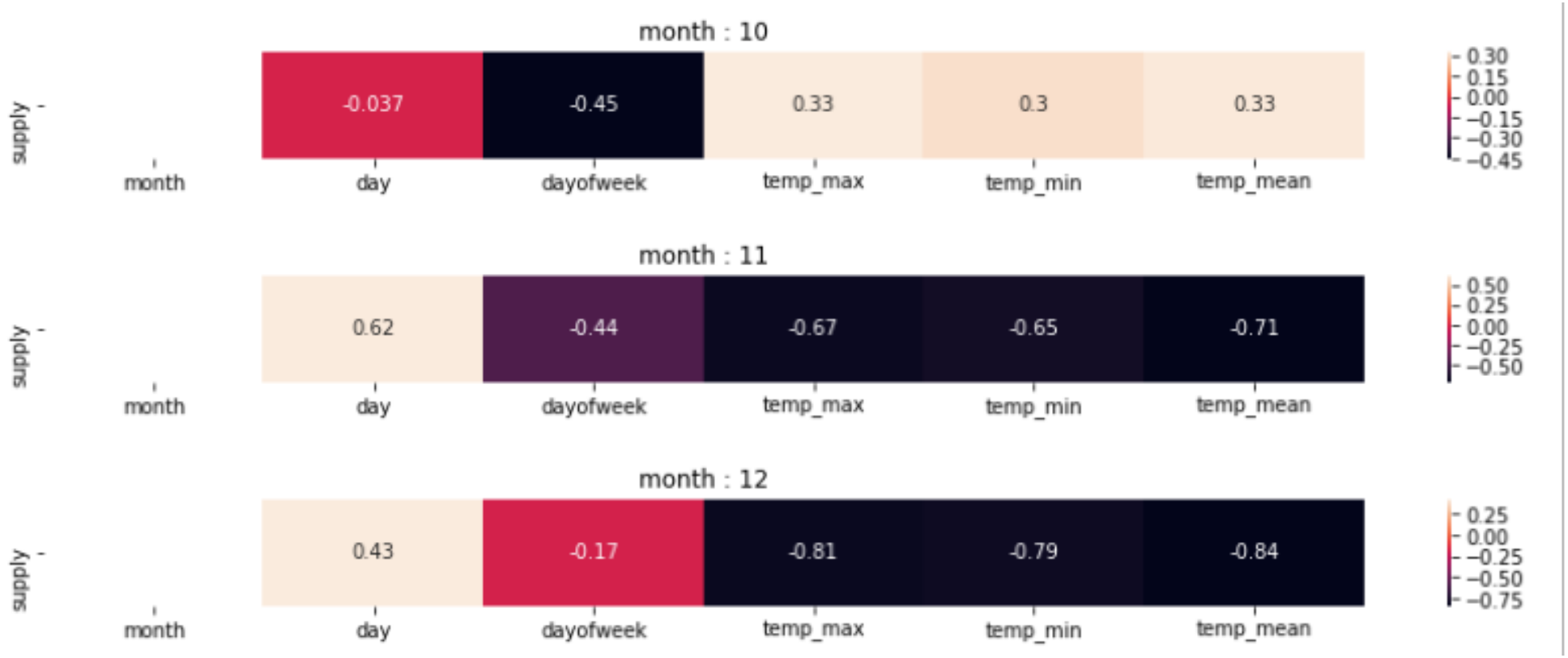
Supply

6~9월
-> 높은
상관관계



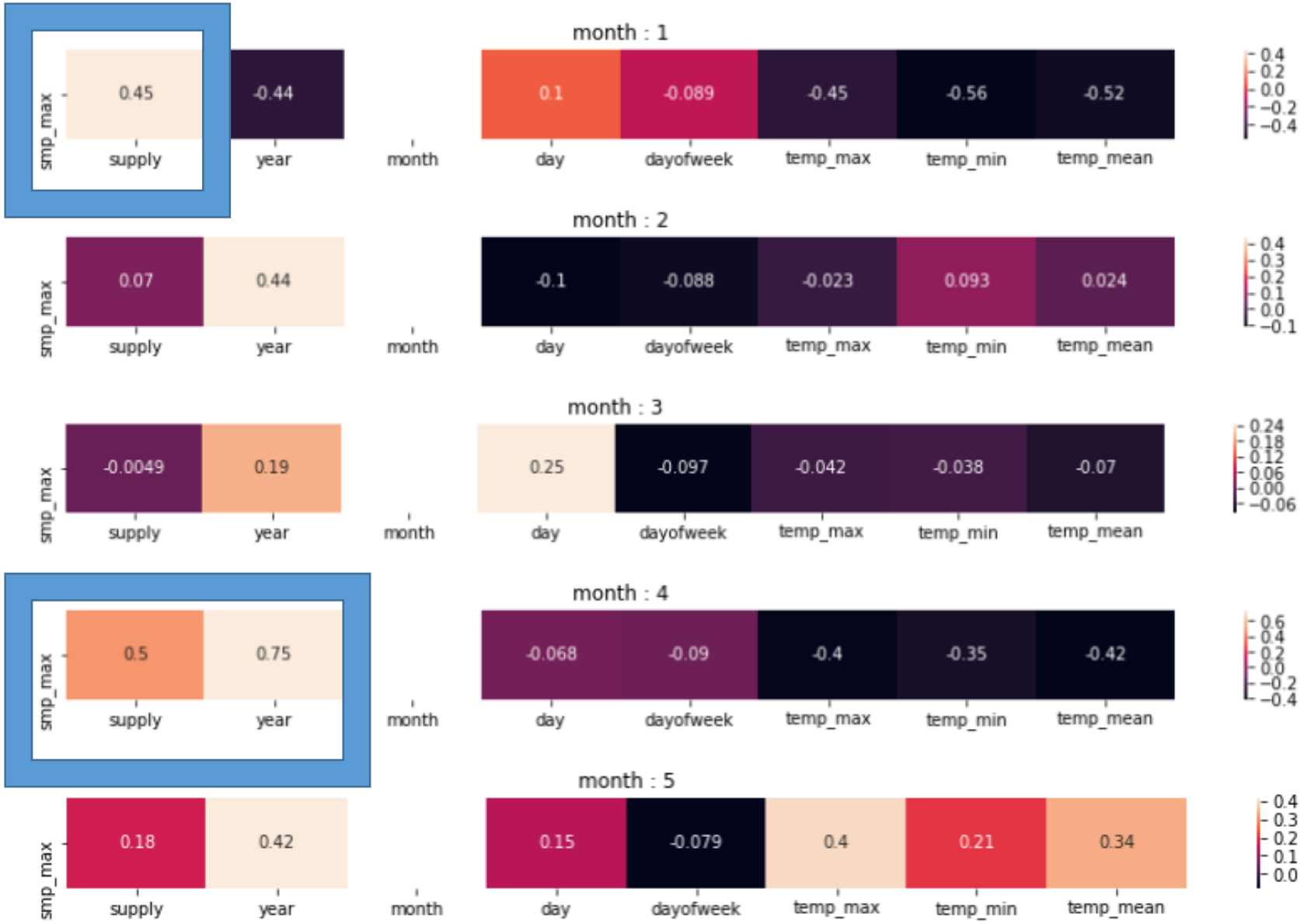
Supply

10~12월
-> 낮은
상관관계

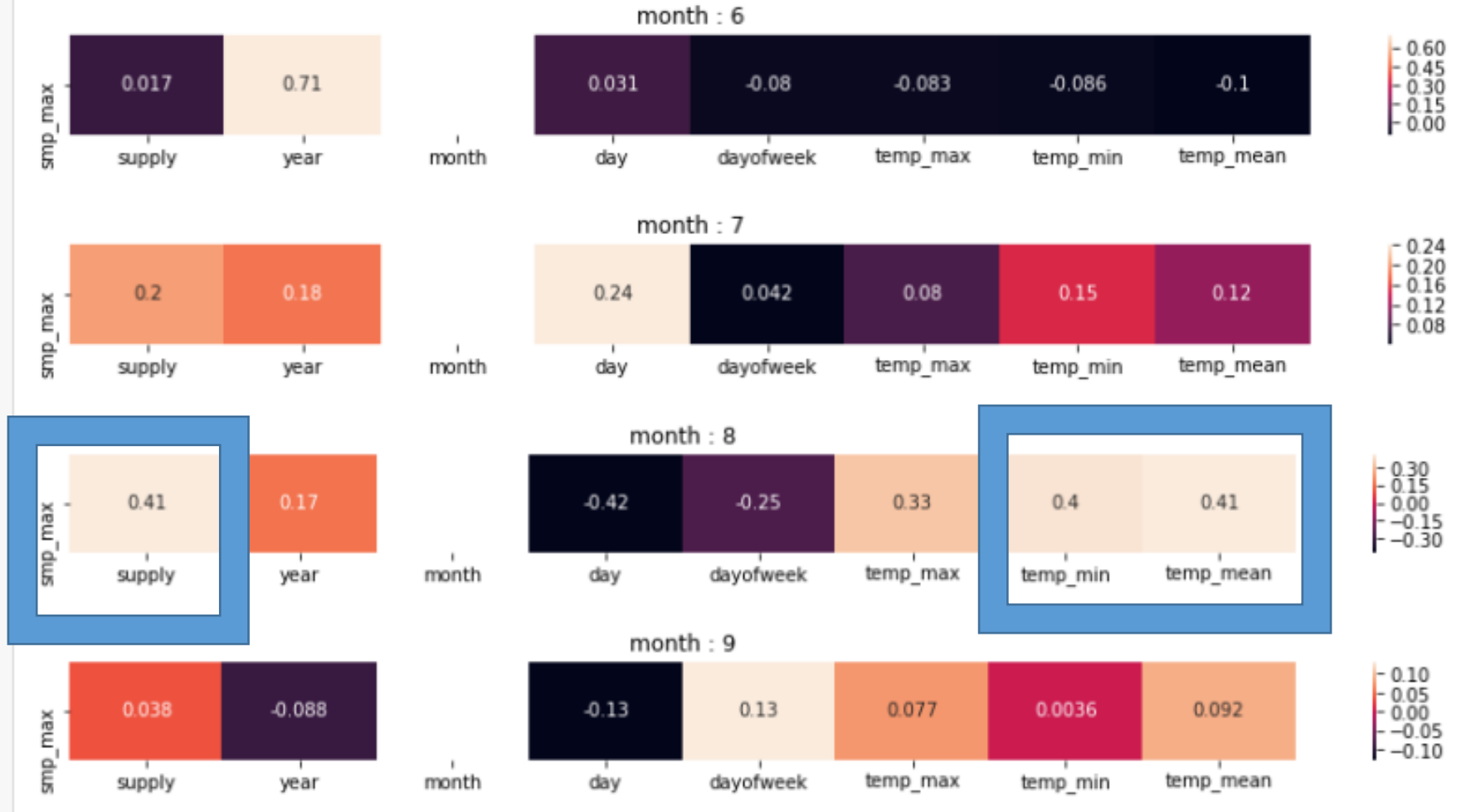


smp_max

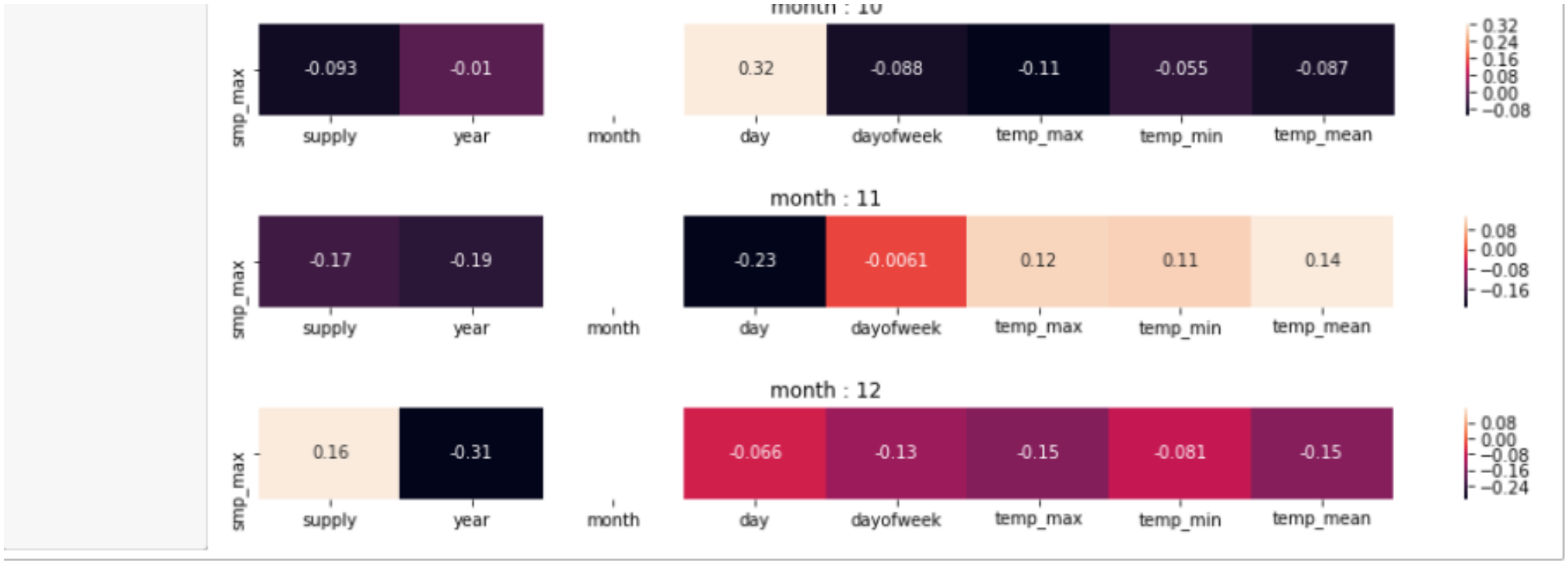
기온과는
직접적 관계가
없는 것 같지만,
supply와
관련이 있음



smp_max


month : 6
smp_max
0.017
0.71
supply
year
month
0.031
-0.08
-0.083
-0.086
-0.1
day
dayofweek
temp_max
temp_min
temp_mean
0.60
0.45
0.30
0.15
0.00
month : 7
smp_max
0.2
0.18
supply
year
month
0.24
0.042
0.08
0.15
0.12
day
dayofweek
temp_max
temp_min
temp_mean
0.24
0.20
0.16
0.12
0.08
month : 8
smp_max
0.41
0.17
supply
year
month
-0.42
-0.25
0.33
0.4
0.41
day
dayofweek
temp_max
temp_min
temp_mean
0.30
0.15
0.00
−0.15
−0.30
month : 9
smp_max
0.038
-0.088
supply
year
month
-0.13
0.13
0.077
0.0036
0.092
day
dayofweek
temp_max
temp_min
temp_mean
0.10
0.05
0.00
−0.05
−0.10

smp_max


month : 10
smp_max
-0.093
-0.01
0.32
-0.088
-0.11
-0.055
-0.087
supply
year
month
day
dayofweek
temp_max
temp_min
temp_mean
0.32
0.24
0.16
0.08
0.00
−0.08
month : 11
smp_max
-0.17
-0.19
-0.23
-0.0061
0.12
0.11
0.14
supply
year
month
day
dayofweek
temp_max
temp_min
temp_mean
0.08
0.00
−0.08
−0.16
month : 12
smp_max
0.16
-0.31
-0.066
-0.13
-0.15
-0.081
-0.15
supply
year
month
day
dayofweek
temp_max
temp_min
temp_mean
0.08
0.00
−0.08
−0.16
−0.24

01 추가 DATA 셋 소개

종류	사용 여부	이유	자료처
LNG 가격	O	-	네이버 금융
유류 가격	O	-	네이버 금융
제주도 연료 발전기 현황	X	2018년까지 정보	전력통계정보시스템
육지 - 제주 송전선(HVDC) 실시간 송전량	X	못찾음	-
신재생 에너지 거래량	X	태양광 20년 2월까지	전력통계정보시스템

추가 데이터 추가 수집-유가

In [135]:
```
# 열 이름 변경
names = target.columns.tolist()
names[names.index('name_x')] = 'holidays'
target.columns = names
# 공휴일 특성 이진수로 변경
target['holidays'].fillna(0, inplace = True)
```

In [140]:
```
target
```

Out[140]:

	date	smp_max	smp_min	smp_mean	supply	year	month	day	dayofweek	holidays	na
0	2018-02-01	150.65	116.84	132.71	87.47	2018	2	1	3	0	
1	2018-02-02	163.86	116.84	134.19	86.64	2018	2	2	4	0	
2	2018-02-03	164.07	116.85	131.39	88.28	2018	2	3	5	0	
3	2018-02-04	171.00	115.76	131.89	86.14	2018	2	4	6	0	
4	2018-02-05	170.34	123.89	137.96	90.63	2018	2	5	0	0	
...	...	...	...	...	...	...	...	...	...	...	...
833	2020-05-14	193.28	66.78	100.46	62.70	2020	5	14	3	0	
834	2020-05-15	198.23	61.81	102.38	64.91	2020	5	15	4	0	
835	2020-05-16	220.91	88.50	121.19	61.75	2020	5	16	5	0	
836	2020-05-17	207.75	65.78	116.82	61.55	2020	5	17	6	0	
837	2020-05-18	113.31	66.86	98.98	63.91	2020	5	18	0	0	

모델생성

모델생성

In [163]:
```
def create_model(train, val):
    params = {
        'metric': 'mae',
        'seed':7777
        }

    model = lgb.train(params, d_train, 1000, d_val, verbose_eval=1000, early_stopping_rounds=100)

    plt.rcParams['figure.figsize'] = [6, 4]
    plt.plot(np.array(y_val), '.-', label='y_val')
    plt.plot(model.predict(x_val), '.-', label='y_pred')
    plt.title(str(future)+'days later')
    plt.legend()
    plt.show()

    return model
```

모델생성- 과거 30일 데이터(년, 월, 일, 주말, supply, 공휴일, 기온, 유가)로 7~21일 이후 'supply' 예측모델 15개

In [164]:
```
def trans(dataset, start_index, end_index, past, future, x_columns, y_columns):
    dataset.index = range(dataset.shape[0])
    data = []
    labels = []

    start_index = start_index + past

    if end_index is None:
        end_index = dataset.shape[0]

    for i in range(start_index, end_index-future):
        indices = np.array(dataset.loc[i-past:i, x_columns])
        data.append(indices)

        labels.append(np.array(dataset.loc[i+future, y_columns]))

    data = np.array(data)
    data = data.reshape(data.shape[0], -1)
    labels = np.array(labels)
    labels = labels.reshape(-1)

    return data, labels
```

In [165]:
```
past = 59# 최근 30일 정보를 이용하여 n일 후를 예측
```

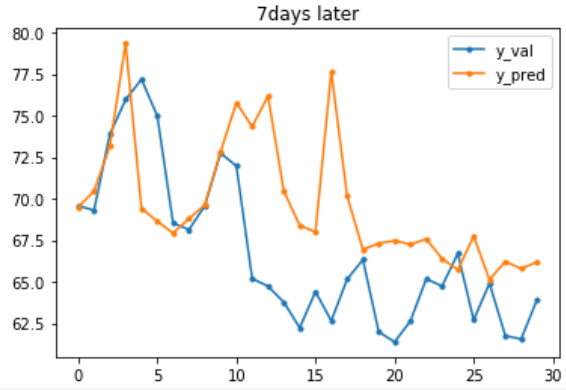
모델생성- 과거 30일 데이터(년, 월, 일, 주말, supply, 공휴일, 기온, 유가)로 7~21일 이후 'supply' 예측모델 15개

In [165]:
```
past = 59# 최근 30일 정보를 이용하여 n일 후를 예측
```

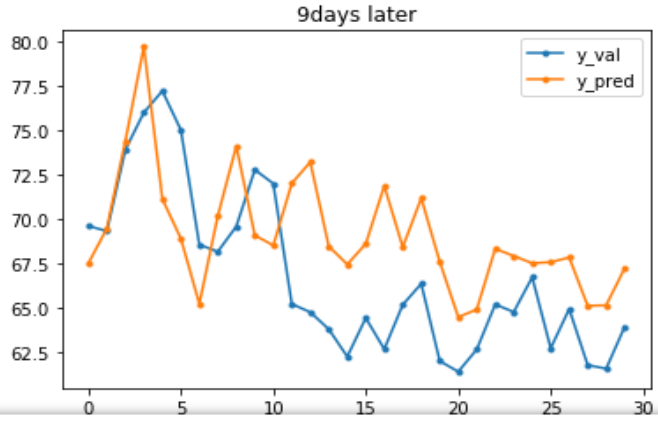
In [166]:
```
x_columns = ['year','month','day','dayofweek','supply', 'holidays','temp_max', 'temp_min', 'temp_mean','oil_pr
y_columns = ['supply']
supply_models = {}

# 7일~34일 후를 예측하는 각각의 모델 구축
for future in range(7, 35):
    train_split = target.shape[0]-past-future-30 # 마지막 30일을 validation set으로 사용
    x_train, y_train = trans(target, 0, train_split, past, future, x_columns, y_columns)
    x_val, y_val = trans(target, train_split, None, past, future, x_columns, y_columns)

    d_train = lgb.Dataset(x_train, y_train)
    d_val = lgb.Dataset(x_val, y_val)

    supply_models[future] = create_model(d_train, d_val)
    print('================================================================================')
```

모델생성- 과거 30일 데이터(년, 월, 일, 주말, supply, 공휴일, 기온, 유가)로 7~21일 이후 'supply' 예측모델 15개

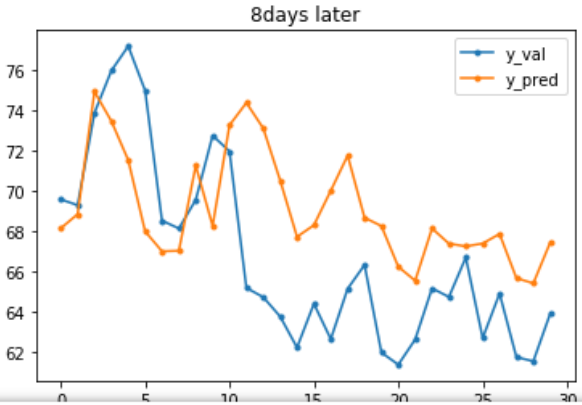
```
Training until validation scores don't improve for 100 rounds
Early stopping, best iteration is:
[40]    valid_0's l1: 3.98991
```




```
Training until validation scores don't improve for 100 rounds
Early stopping, best iteration is:
[62]    valid_0's l1: 3.93944
```




```
Training until validation scores don't improve for 100 rounds
Early stopping, best iteration is:
[95]    valid_0's l1: 3.91513
```



과거 7일 데이터(년, 월, 일, 주말, supply, 공휴일, 기온, 유가)로 7~34일 이후의 '유가' 예측모델 28개

In [51]:
```
past = 6
x_columns = ['year','month','day','dayofweek','supply', 'holidays','temp_max', 'temp_min', 'temp_mean','oil_price']
y_columns = ['oil_price']
oil_price_models = {}

# 7일~ 34일 후를 예측하는 각각의 모델 구축
for future in range(7,35):
    train_split = target.shape[0]-past-future-30 #마지막 30일은 validation set
    x_train, y_train = trans(target,0,train_split,past,future,x_columns,y_columns)
    x_val, y_val = trans(target,train_split,None,past,future,x_columns,y_columns)

    d_train = lgb.Dataset(x_train, y_train)
    d_val = lgb.Dataset(x_val, y_val)

    oil_price_models[future] = create_model(d_train, d_val)
    print('============================================================')
```

```
Training until validation scores don't improve for 100 rounds
Early stopping, best iteration is:
[168]   valid_0's l1: 5.58011
```

7days later

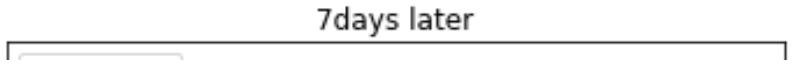
과거 30일 데이터(년, 월, 일, 기온)로 7~34일 이후의 '기온'을 예측모델 28개

```
============================================================
```

In [52]:
```
x_columns = ['year','month','day','temp_max','temp_min','temp_mean']
```

In [55]:
```
past = 29
y_columns = ['temp_max']
temp_max_models = {}

for future in range(7,35):
    train_split = target.shape[0]-past-future-30
    x_train, y_train = trans(target, 0, train_split, past, future, x_columns, y_columns)
    x_val, y_val = trans(target, train_split, None, past, future, x_columns, y_columns)

    d_train = lgb.Dataset(x_train, y_train)
    d_val = lgb.Dataset(x_val, y_val)

    temp_max_models[future] = create_model(d_train, d_val)
    print('============================================================')
```

```
Training until validation scores don't improve for 100 rounds
Early stopping, best iteration is:
[6]     valid_0's l1: 1.93806
```



예측한 supply, 유가, 기온 + 공휴일, 년, 월, 일 주말 데이터를 모델에 넣고 평균냄 -〉〉smp 예측

예측된 supply와 날씨데이터를 가지고 smp예측하기

In [67]:
```
def create_model(x_data, y_data, k=5):
    models = []

    k_fold = KFold(n_splits=k, shuffle=True, random_state=77)

    for train_idx, val_idx in k_fold.split(x_data):
        x_train, y_train = x_data.iloc[train_idx], y_data.iloc[train_idx]
        x_val, y_val = x_data.iloc[val_idx], y_data.iloc[val_idx]

        d_train = lgb.Dataset(x_train, y_train)
        d_val = lgb.Dataset(x_val, y_val)

        params = {
            'metric':'mse',
            'seed':777
        }

        model = lgb.train(params, d_train, 1000, d_val, verbose_eval=1000, early_stopping_rounds = 100)

        plt.rcParams['figure.figsize'] = [12,4]
        plt.plot(np.array(y_val),'.-',label='y_val')
        plt.plot(model.predict(x_val),'.-',label='y_pred')
        plt.legend()
        plt.show()
        models.append(model)

    return models
```

훈련 데이터를 랜덤으로 섞고 KFold 진행 (k=5), 5개의 모델 생성

In [70]:
```
# 5번 진행된 것을 평균냄
x_test = test.loc[:,['supply', 'year', 'month', 'day', 'dayofweek', 'holidays', 'temp_max', 'temp_min', 'temp_mean','oil_pric
e']]
for label in ['smp_min', 'smp_max', 'smp_mean']:
    preds = []
    for i in range(5):
        preds.append(smp_models[label][i].predict(x_test))
    pred = sum(preds)/len(preds)
    test[label] = pred
```

In [71]:
```
submission.reset_index(drop=True, inplace=True)
```

In [72]:
```
submission.loc[:, ['smp_min', 'smp_max', 'smp_mean', 'supply']] = test.loc[:,['smp_min', 'smp_max', 'smp_mean', 'supply']]
```

In [73]:
```
submission = pd.concat([submission_top_half, submission], axis = 0)
submission.reset_index(drop=True, inplace=True)
```

In [74]:
```
submission
```

Out[74]:

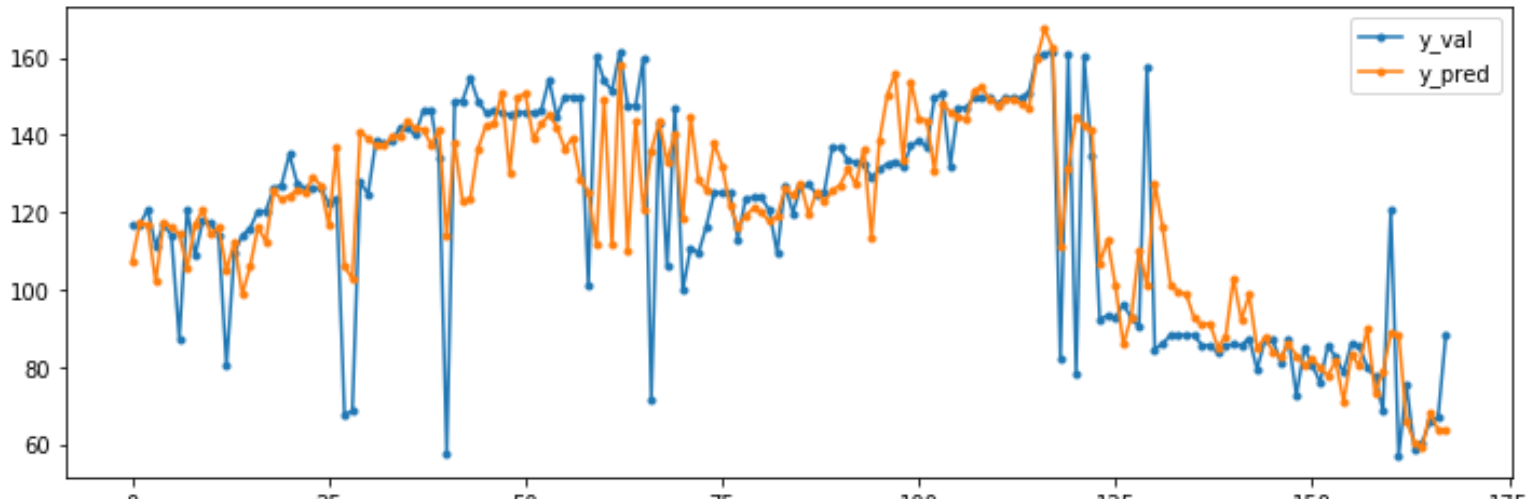
	date	smp_max	smp_min	smp_mean	supply
0	2020-02-07	201.690000	84.280000	123.620000	84.370000
1	2020-02-08	152.360000	77.420000	93.720000	84.110000
2	2020-02-09	146.770000	74.880000	93.660000	82.410000
3	2020-02-10	173.090000	82.820000	113.480000	83.300000
4	2020-02-11	199.750000	84.910000	121.860000	75.870000
5	2020-02-12	197.670000	84.450000	109.970000	77.110000
6	2020-02-13	198.270000	85.020000	111.640000	74.240000
7	2020-02-14	201.010000	84.300000	103.990000	73.260000

```
    return models
```

In [68]:
```
x_train = target.loc[:,['supply', 'year', 'month', 'day', 'dayofweek', 'holidays','temp_max', 'temp_min', 'temp_mean','oil_pr
ice']]
y_train = target.loc[:, ['smp_min', 'smp_max', 'smp_mean' ]]
```

In [69]:
```
# train과 test를 5개로 나누어서 5번 진행
smp_models = {}
for label in y_train.columns:
    print('train column : ', label)
    smp_models[label] = create_model(x_train, y_train[label])
    print('==================================================')
```

```
train column :  smp_min
Training until validation scores don't improve for 100 rounds
Early stopping, best iteration is:
[67]    valid_0's l2: 281.31
```



결론

datetime 전처리,
목표에 맞게

예측값을 바탕으로 예측을 하는 것 ?
기온예측 -〉supply -〉 smp 예측

예측 모델 평균 -〉 예측

결론

30일

LightGBM	0.7607	961.77	1431.12

60일모델로 확장

	R-squre value	MAE	RMSE
LightGBM	0.5400	1475.08	2004.23

23등

23 DSK NN 1.95819 19 2달 전

감사합니다