구내식당 식수 인원 예측

송현근

목차

1. 분석 배경

지금 전세계는 세계사를 통틀어 제일 빠른 속도로 발전해 나가는 상황이다. 인류는 인류의 편의성을 위해 무분별한 개발을 일삼아 수십년 전부터 여러 학자들로부터 경고메시지를 들어왔다. 그래서 최근에는 여러 국가들이 자연을 지키는데에도 신경을 쏟고 있다. 이를 뒷받침 할 수 있는 제일 유명한 사례가 바로 오존층이다. 우리는 예전부터 남극에 큰 오존층 구멍이 있다는 것을 여러 뉴스들을 통해 들어왔다. 이는 매우 극단적인 '구멍'이라는 단어를 사용해 사람들에게 충격을 가져다 주었고 이는 곧 정부들이 움직이는 계기가 되었다. 그 결과로 국제적으로 오존 파괴물질의 생산을 제한하는 몬트리올 의정서를 체결했다. 이 효과로 인해 최근 오존층이 회복되어 가고 있다는 여러 연구가 시행되었다. 이 사례가 이미 망가진 자연, 기후도 각국 정부들이 합심하여 협력하면 다시 정상으로 되돌릴 수 있다는 것을 보여주는 사례가 되었다.

2. 분석 목적

지금 세계적으로 문제가 되고 있는 다른 것 중 하나는 식량문제다. 해당 프로젝트는 한국토지주택공사(이하 LH) 구내식당 식수 인원을 예측하여 잔반 발생량을 줄이고자 하는 목적이다. 이는 작게 보자면 LH의 예산절감에 그칠 수 있지만, 이를 시작으로 여러 기업들에서 보다 정밀한 식수 인원 예측을 통해 기업의 입장에서는 잔반량이 줄어 잔반처리비용 절감으로 이익을 볼 수 있고 더 나아가 전체적으로 본다면 식량낭비가 줄어드는 효과를 볼 수 있다.

3. 분석 대상

LH에서는 날짜, 요일, 본사 정원수, 본사 휴가자수, 본사 출장자수, 시간외 근무 명령서 승인건수, 현 본사 소속 재택 근무자수, 조식메뉴, 중식메뉴 ,석식메뉴, 중식계, 석식계 데이터를 제공하였다. 이 주어진 데이터들을 통해 가설설정, 가설검정을 하고 더 나은 예측모델을 통해 식수 인원을 예측하고자 한다. 또한 기상청에서 해당하는 날짜의 날씨 데이터를 추가하였다.

4. 가설 설정

강수 여부에 따라 식수 인원에 차이가 있는지, 기온에 따라 식수 인원에 차이가 있는지, 요일에 따라 식수 인원에 차이가 있는지, 또 시간외 근무자가 많고 적어짐에 따라 저녁 식수 인원에 차이가 있는지 검정하고자 한다.

5. 데이터 전처리

우선 LH 본사의 소재지가 경상남도 진주시인걸로 파악하여 기상자료개방포털에서 영향이 있을 것이라 보여지는 기온, 강수량, 적설량을 시간별로 데이터 수집하였다. 그리고 점심과 저녁식사 시간에 맞는 12시와 18시 데이터만 추린다음 LH에서 제공한 데이터와 merge 하였다. 그랬더니

LH에서 제공한 데이터에는 주말이나 휴일엔 데이터가 없어 NaN행이 생겨나 해당 행들을 제거하였다.

0	2016-02-01	0.0	2601.0	50.0	150.0	238.0	0.0	모닝롤/찐빵 우유/두유/주스 계란후라이 호두죽/쌀밥 (쌀:국내산) 된장찌개 쥐...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 오징어찌개 쇠불고기 (쇠고기:호주산) 계란찜 ...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 육개장 자반고등어구이 두부조림 건파래무침 ...	1039.0	331.0	2401.0	2016.0	2.0	1.0	5.0	2016-02-01 12:00:00	3.0
1	2016-02-02	1.0	2601.0	50.0	173.0	319.0	0.0	모닝롤/단호박샌드 우유/두유/주스 계란후라이 팥죽/쌀밥 (쌀:국내산) 호박젓국찌...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 김치찌개 가자미튀김 모둠소세지구이 마늘쫑무...	콩나물밥*양념장 (쌀,현미흑미:국내산) 어묵국 유산슬 (쇠고기:호주산) 아삭고추무...	867.0	560.0	2378.0	2016.0	2.0	2.0	5.0	2016-02-02 12:00:00	1.7
2	2016-02-03	2.0	2601.0	56.0	180.0	111.0	0.0	모닝롤/베이글 우유/두유/주스 계란후라이 표고버섯죽/쌀밥 (쌀:국내산) 콩나물국...	카레덮밥 (쌀,현미흑미:국내산) 팽이장국 치킨핑거 (닭고기:국내산) 쫄면야채무침 ...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 청국장찌개 황태양념구이 (황태:러시아산) 고기...	1017.0	573.0	2365.0	2016.0	2.0	3.0	5.0	2016-02-03 12:00:00	3.2
3	2016-02-04	3.0	2601.0	104.0	220.0	355.0	0.0	모닝롤/토마토샌드 우유/두유/주스 계란후라이 닭죽/쌀밥 (쌀,닭:국내산) 근대국...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 쇠고기무국 주꾸미볶음 부추전 시금치나물 ...	미니김밥*겨자장 (쌀,현미흑미:국내산) 우동 멕시칸샐러드 군고구마 무피클 포...	978.0	525.0	2277.0	2016.0	2.0	4.0	5.0	2016-02-04 12:00:00	3.7
4	2016-02-05	4.0	2601.0	278.0	181.0	34.0	0.0	모닝롤/와플 우유/두유/주스 계란후라이 쇠고기죽/쌀밥 (쌀:국내산) 재첩국 방...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 떡국 돈육씨앗강정 (돼지고기:국내산) 우엉잡채...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 차돌박이찌개 (쇠고기:호주산) 닭갈비 (닭고기:...	925.0	330.0	2142.0	2016.0	2.0	5.0	5.0	2016-02-05 12:00:00	6.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1816	2021-01-10	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2021-01-10 12:00:00	-1.6
1817	2021-01-16	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2021-01-16 12:00:00	6.8
1818	2021-01-17	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2021-01-17 12:00:00	1.1
1819	2021-01-23	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2021-01-23 12:00:00	10.1

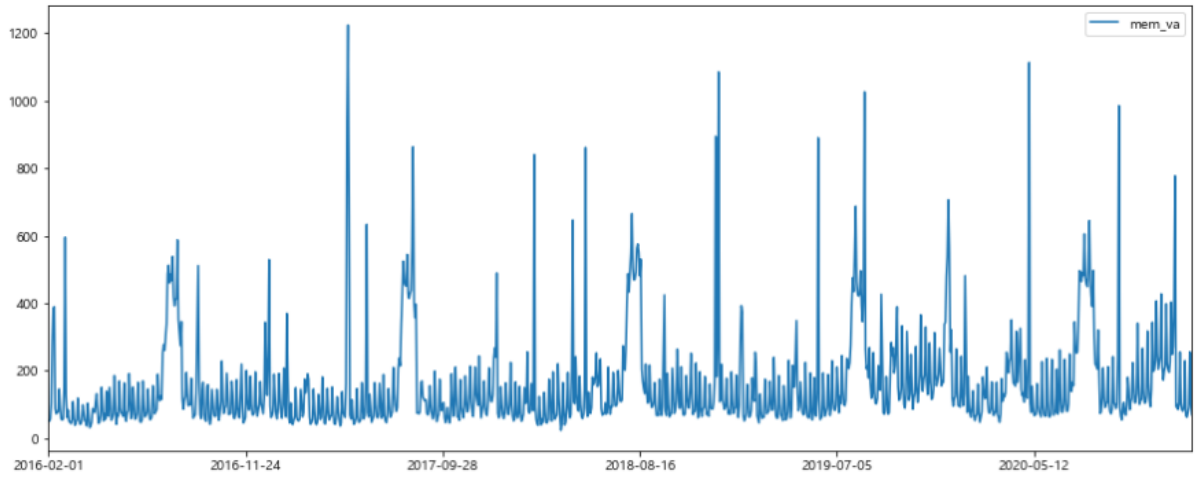
그 후 column 이름을 'date', 'week', 'mem_tot', 'mem_va', 'mem_bt', 'mem_ot', 'mem_hj', 'breakfast', 'lunch', 'dinner', 'lunch_num', 'dinner_num', 'lunch_tem', 'lunch_rain', 'lunch_snow', 'dinner_tem', 'dinner_rain','dinner_snow' 로 바꾸었다.

그리고 본사 정원수(mem_tot)에서 본사 휴가자수(mem_va), 본사 출장자수(mem_bt), 현 본사 소속 재택 근무자수(mem_hj)를 뺀 본사 식수 가능자수(mem_poss)변수를 추가하였다. 또한 date에서 month, day 변수를 따로 추출해서 추가하였다.

추가로 보다 정확한 예측을 위해 휴일 전날(금요일포함)(holi_be), 휴일 다음날(월요일포함)(holi_af), 여름휴가(7/27 ~ 8/17)(summer1), 연말(12/22 ~ 12/31)(year_end) 변수를 추가하였다. 해당 변수들의 날짜 구분 기준은 휴가자 데이터가 많아지는 시점을 기준으로 하였다.

```
train.plot('date', 'mem_va', figsize = (15, 6))
```

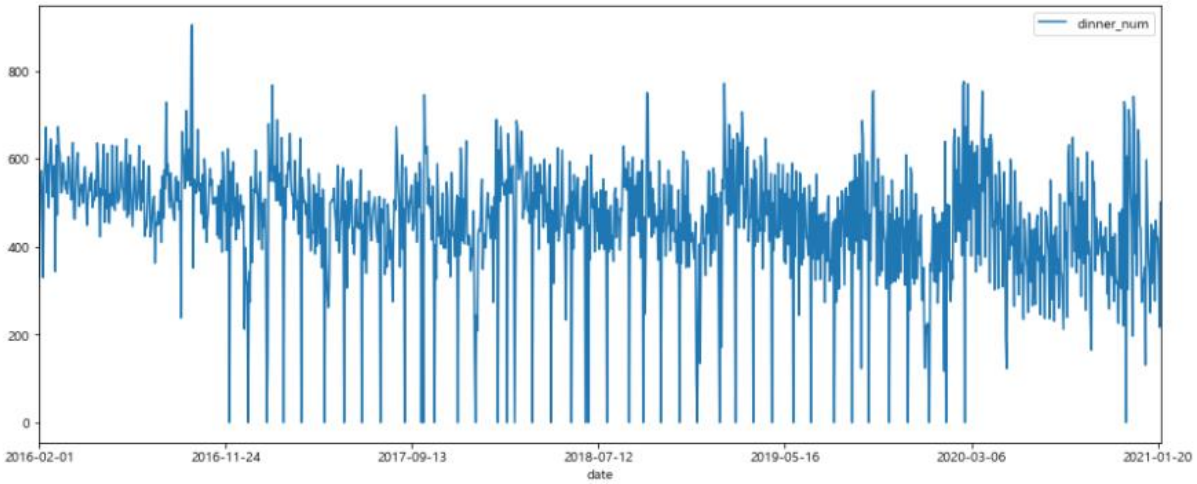
```
<matplotlib.axes._subplots.AxesSubplot at 0x21bfbef85c8>
```



그 후 결측값을 찾아 보았는데 적설량 데이터가 딱 하루 있어서 적설량은 제거하였다. 그리고 하루 점심기온이 누락되어 전날 점심기온으로 대체하였다. 또한 석식계가 일정 주기로 0이 되는날을 살펴보니 매달 마지막 수요일이 자기계발의 날로 지정되어 석식이 제공되지 않는 것을 발견하였다. 하여 석식계가 0인 날짜들을 제거하였다.

```
train.plot('date', 'dinner_num', figsize = (15, 6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x21bfc339488>
```



그리고 알아보고자 하는 가설에는 조식, 중식, 석식 메뉴 데이터가 쓰이지 않아 제거하였다.

	일자	요일	본사정원수	본사휴가자수	본사출장자수	본사시간외근무명령서승인건수	현본사소속재택근무자수	조식메뉴	중식메뉴	석식메뉴	중식계	석식계
0	2016-02-01	월	2601	50	150	238	0.0	모닝롤/찐빵 우유/두유/주스 계란후라이 호두죽/쌀밥 (쌀:국내산) 된장찌개 쥐...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 오징어찌개 쇠불고기 (쇠고기:호주산) 계란찜 ...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 육개장 자반고등어구이 두부조림 건파래무침 ...	1039.0	331.0
1	2016-02-02	화	2601	50	173	319	0.0	모닝롤/단호박샌드 우유/두유/주스 계란후라이 팥죽/쌀밥 (쌀:국내산) 호박젓국찌...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 김치찌개 가자미튀김 모둠소세지구이 마늘쫑무...	콩나물밥*양념장 (쌀,현미흑미:국내산) 어묵국 유산슬 (쇠고기:호주산) 아삭고추무...	867.0	560.0
2	2016-02-03	수	2601	56	180	111	0.0	모닝롤/베이글 우유/두유/주스 계란후라이 표고버섯죽/쌀밥 (쌀:국내산) 콩나물국...	카레덮밥 (쌀,현미흑미:국내산) 팽이장국 치킨핑거 (닭고기:국내산) 풋연야채무침 ...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 청국장찌개 황태양념구이 (황태:러시아산) 고기...	1017.0	573.0
3	2016-02-04	목	2601	104	220	355	0.0	모닝롤/토마토샌드 우유/두유/주스 계란후라이 닭죽/쌀밥 (쌀,닭:국내산) 근대국...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 쇠고기무국 주꾸미볶음 부추전 시금치나물 ...	미니김밥*겨자장 (쌀,현미흑미:국내산) 우동 멕시칸샐러드 군고구마 무피클 포...	978.0	525.0
4	2016-02-05	금	2601	278	181	34	0.0	모닝롤/와플 우유/두유/주스 계란후라이 쇠고기죽/쌀밥 (쌀:국내산) 재첩국 방...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 떡국 돈육씨앗강정 (돼지고기:국내산) 우엉잡채...	쌀밥/잡곡밥 (쌀,현미흑미:국내산) 차돌박이찌개 (쇠고기:호주산) 닭갈비 (닭고기:...	925.0	330.0

===> 초기 데이터

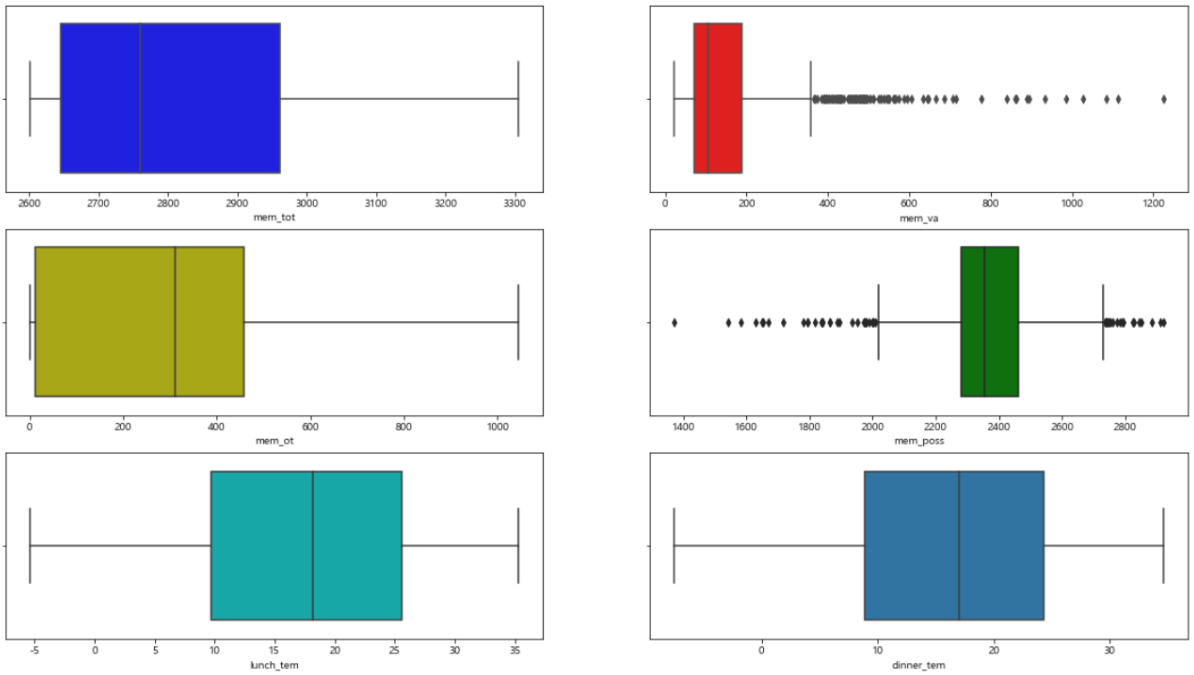
```
train.head()
```

	date	week	mem_tot	mem_va	mem_bt	mem_ot	mem_hj	lunch_num	dinner_num	lunch_tem	...	dinner_rain	mem_poss	month	day	lunch_ratio	dinner_ratio	holi_be	holi_af	summer1	year_end
0	2016-02-01	월	2601.0	50.0	150.0	238.0	0.0	1039.0	331.0	3.0	...	0.0	2401.0	2.0	1.0	0.432736	0.137859	0.0	1.0	0.0	0.0
1	2016-02-02	화	2601.0	50.0	173.0	319.0	0.0	867.0	560.0	1.7	...	0.0	2378.0	2.0	2.0	0.364592	0.235492	0.0	0.0	0.0	0.0
2	2016-02-03	수	2601.0	56.0	180.0	111.0	0.0	1017.0	573.0	3.2	...	0.0	2365.0	2.0	3.0	0.430021	0.242283	0.0	0.0	0.0	0.0
3	2016-02-04	목	2601.0	104.0	220.0	355.0	0.0	978.0	525.0	3.7	...	0.0	2277.0	2.0	4.0	0.429513	0.230567	0.0	0.0	0.0	0.0
4	2016-02-05	금	2601.0	278.0	181.0	34.0	0.0	925.0	330.0	6.3	...	0.0	2142.0	2.0	5.0	0.431839	0.154062	1.0	0.0	0.0	0.0

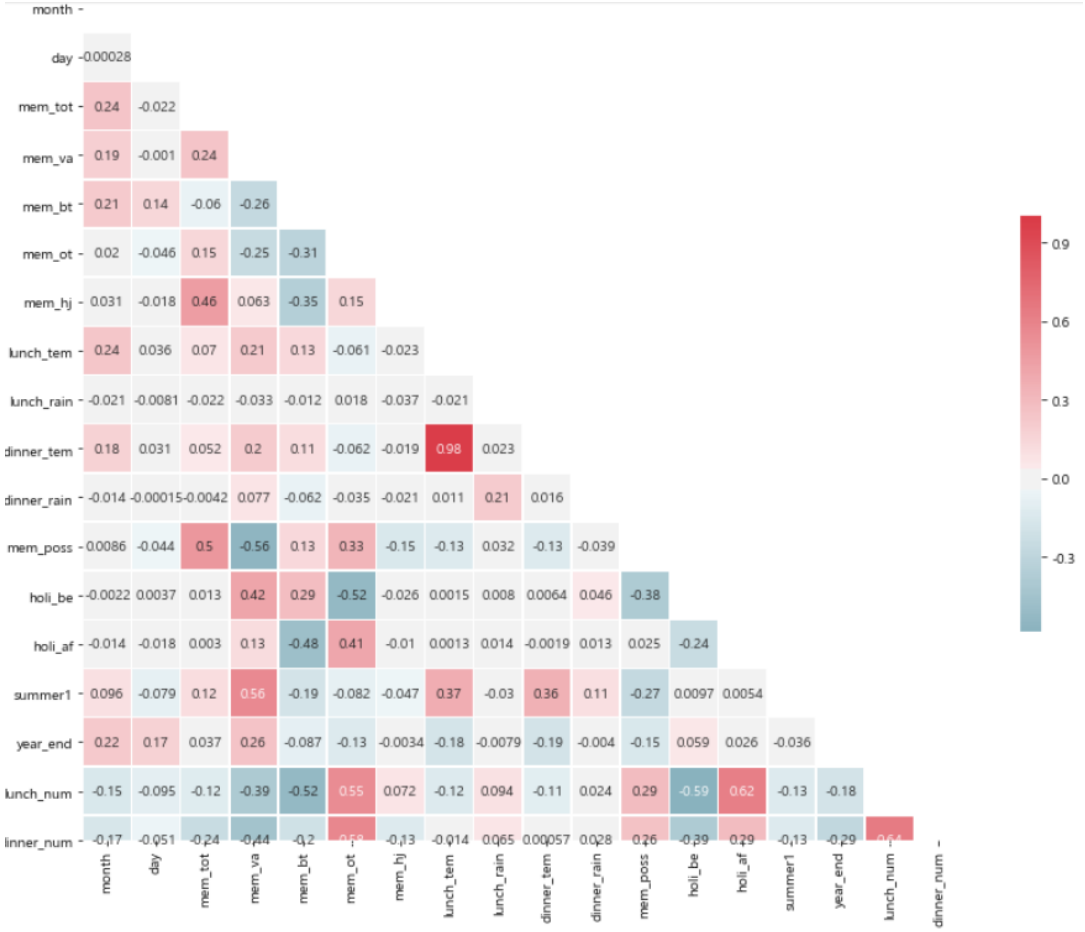
===> 데이터 전처리 후 최종 데이터

6. 탐색적 데이터 분석(EDA)

본격적인 가설검정에 들어가기에 앞서 데이터들을 살펴보았다.



boxplot을 통해 이상치를 처리하고자 하였는데 휴가자가 특정 날짜에만 몰리다 보니깐 이상치가 많이 나오게 된 것으로 보여진다. 하지만 이 또한 휴가자변수의 특성으로 보여져 앞서 휴일관련 변수들을 추가하여 이를 보정하는 역할로 사용하였다.



상관계수를 통해 알 수 있는 점은 점심 기온과 저녁 기온은 1에 가까운 관계를 갖고 있지만 이는 점심 식수 인원 예측과 저녁 식수 인원 예측에 각각 따로 쓰일 예정이라 무관하고 앞서 말한 휴가자 변수와 여름휴가 변수가 다른 것들 중에 제일 높은 관계가 있다고 보여진다.

이제 앞서 설정한 가설검정을 해보겠다.

7. 가설검정

강수 여부에 따라 식수 인원에 차이가 있는지 알아보기 위해 먼저 점심에 비가 온 날, 비가 오지 않은 날, 저녁에 비가 온 날, 비가 오지 않은 날로 데이터를 구분하였다.

```
# 귀무가설 : 두집단 분포 동일하다
stats.levene(yeslunch_rain['lunch_num'],nolunch_rain['lunch_num'])
```

```
LeveneResult(statistic=0.24672672614554716, pvalue=0.6194835774860699)
```

p-value > 0.05이므로 귀무가설을 기각하지 못한다.

두 집단의 분포는 동일하다.

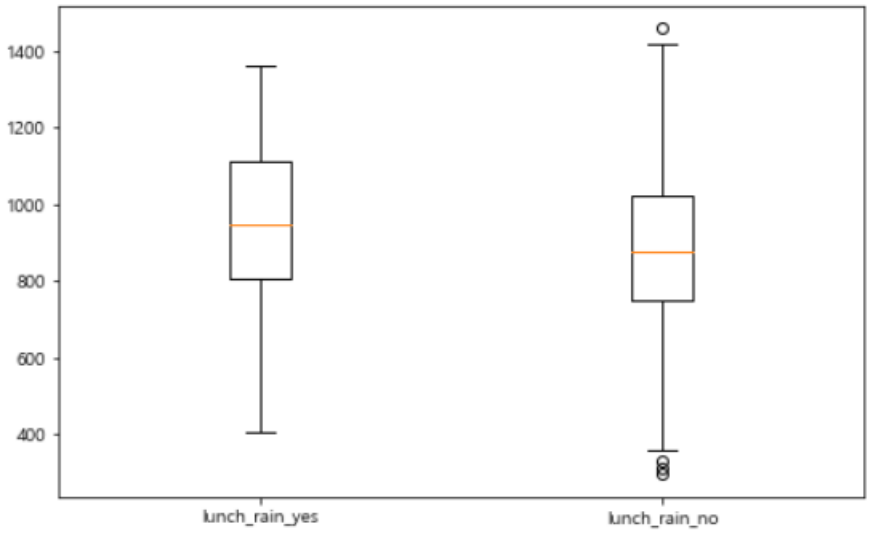
```
ttest_ind(yeslunch_rain['lunch_num'], nolunch_rain['lunch_num'], equal_var=True)
```

```
Ttest_indResult(statistic=2.940148215143178, pvalue=0.003345645144892463)
```

p-value < 0.05이므로 귀무가설을 기각할 수 있다.

두 집단의 점심 식수 인원은 차이가 있다고 볼 수 있다.

===> 점심에 강수여부에 따라 점심 식수인원에 유의미한 차이가 있다.



마찬가지로 저녁도 실시

```
stats.levene(yesdinner_rain['dinner_num'],nodinner_rain['dinner_num'])
```

```
LeveneResult(statistic=0.6186033773883779, pvalue=0.4317275355428386)
```

p-value > 0.05이므로 귀무가설을 기각하지 못한다.

두 집단의 분포는 동일하다.

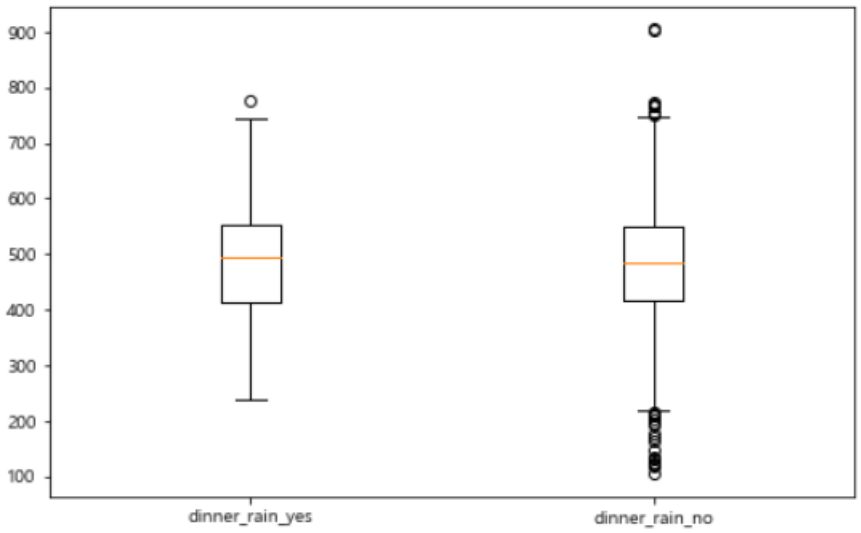
```
ttest_ind(yesdinner_rain['dinner_num'], nodinner_rain['dinner_num'], equal_var=True)
```

```
Ttest_indResult(statistic=0.24314469394312666, pvalue=0.8079363339956721)
```

p-value > 0.05이므로 귀무가설을 기각할 수 없다.

두 집단의 저녁 식수 인원은 차이가 있다고 볼 수 없다.

===> 저녁에 강수여부에 따라 저녁 식수인원에 유의미한 차이가 없다.



다음 가설인 **기온에 따라 식수 인원에 차이가 있는지** 알아보겠다.

먼저 기온별로 집단을 3개로 나누었다. 기준은 1사분위수, 3사분위수로 하였다.

```
train.lunch_tem.describe()
```

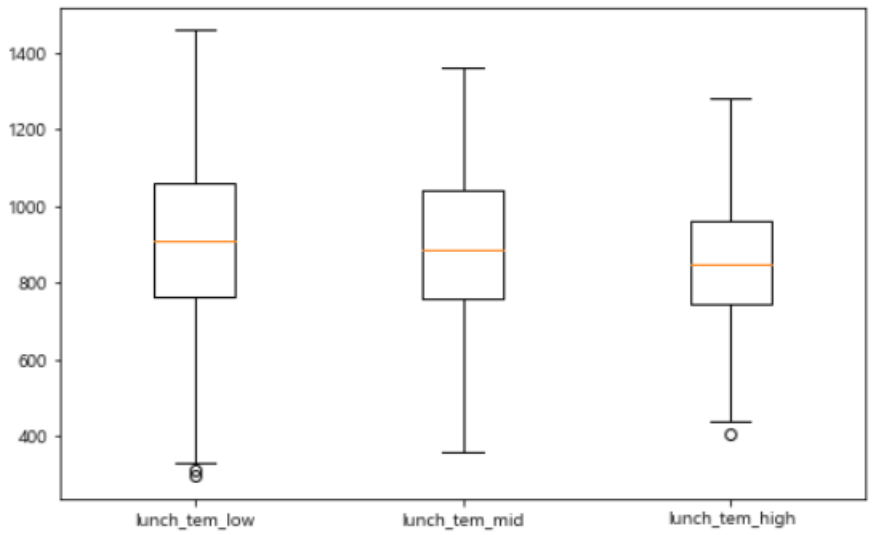
```
count    1161.000000
mean       17.635659
std         9.173468
min        -5.400000
25%         9.700000
50%        18.200000
75%        25.600000
max        35.300000
Name: lunch_tem, dtype: float64
```

```
train_l_l = train[train['lunch_tem'] <= 9.7]
train_m_l = train[(train['lunch_tem'] > 9.7) & (train['lunch_tem'] <= 25.6)]
train_h_l = train[train['lunch_tem'] > 25.6]
```

```
tem_l_df = [train_l_l, train_m_l, train_h_l]
tem_l_tx = ['train_l_l', 'train_m_l', 'train_h_l']
print('=' * 40)
for i in range(3):
    print(tem_l_tx[i])
    for j in range(3):
        _, pval = stats.levene(tem_l_df[i]['lunch_num'], tem_l_df[j]['lunch_num'])
        if pval > 0.05:
            _2, pval_t = ttest_ind(tem_l_df[i]['lunch_num'], tem_l_df[j]['lunch_num'], equal_var = True)
            print(pval_t,'    vs', tem_l_tx[j])
        else :
            _2, pval_t = ttest_ind(tem_l_df[i]['lunch_num'], tem_l_df[j]['lunch_num'], equal_var = False)
            print(pval_t,'    vs', tem_l_tx[j])
    print('=' * 40)
```

```
========================================
train_l_l
1.0     vs train_l_l
0.1763692267566472     vs train_m_l
0.0007144996006216482     vs train_h_l
========================================
train_m_l
0.1763692267566472     vs train_l_l
1.0     vs train_m_l
0.008433089107484708     vs train_h_l
========================================
train_h_l
0.0007144996006216482     vs train_l_l
0.008433089107484708     vs train_m_l
1.0     vs train_h_l
========================================
```

각각 집단을 비교하기 위해 위 코드를 작성했다. 결과를 보면 점심에 기온이 낮을 때와 중간일때 점심 식수 인원은 유의미한 차이가 없지만 나머지 집단끼리는 유의미한 차이가 있다.



저녁도 같은 방법으로 시행하였다.

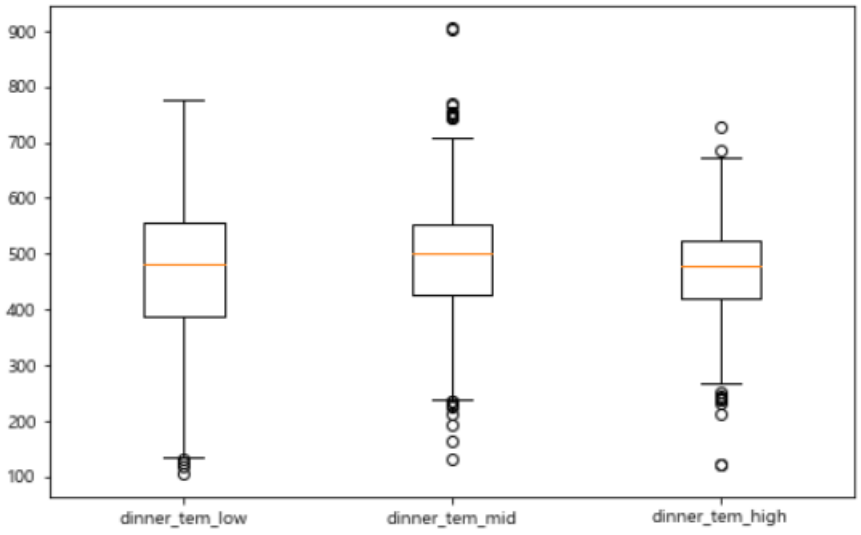
```
train.dinner_tem.describe()
```

```
count    1161.000000
mean       16.529027
std         9.198115
min        -7.500000
25%         8.900000
50%        17.000000
75%        24.300000
max        34.600000
Name: dinner_tem, dtype: float64
```

```
train_l_d = train[train['dinner_tem'] <= 8.9]
train_m_d = train[(train['dinner_tem'] > 8.9) & (train['dinner_tem'] <= 24.3)]
train_h_d = train[train['dinner_tem'] > 24.3]
```

```
========================================
train_l_d
1.0     vs train_l_d
0.016707412975336112     vs train_m_d
0.935898712766036     vs train_h_d
========================================
train_m_d
0.016707412975336112     vs train_l_d
1.0     vs train_m_d
0.001999427890081429     vs train_h_d
========================================
train_h_d
0.935898712766036     vs train_l_d
0.001999427890081429     vs train_m_d
1.0     vs train_h_d
========================================
```

결과를 보면 저녁에 기온이 낮을 때와 높을때 저녁 식수 인원은 유의미한 차이가 없지만 나머지 집단끼리는 유의미한 차이가 있다.



다음 가설인 **요일에 따라 식수 인원에 차이가 있는지** 알아보겠다.

먼저 점심 식수 인원에 차이가 있는지 알아보겠다.

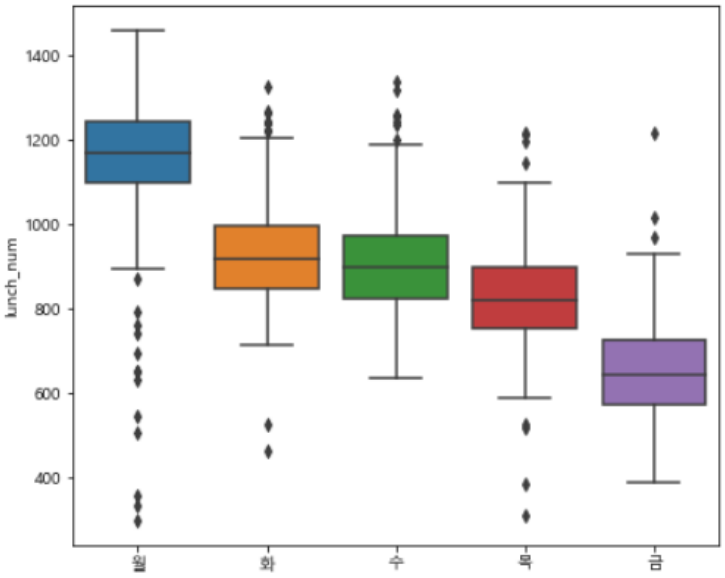
```
train_mon = train.loc[train['week'] == '월']
train_tue = train.loc[train['week'] == '화']
train_wed = train.loc[train['week'] == '수']
train_thu = train.loc[train['week'] == '목']
train_fri = train.loc[train['week'] == '금']
```

```
week_df = [train_mon, train_tue, train_wed, train_thu, train_fri]
week_tx = ['train_mon', 'train_tue', 'train_wed', 'train_thu', 'train_fri']
print('=' * 40)
for i in range(5):
    print(week_tx[i])
    for j in range(5):
        _, pval = stats.levene(week_df[i]['lunch_num'], week_df[j]['lunch_num'])
        if pval > 0.05:
            _2, pval_t = ttest_ind(week_df[i]['lunch_num'], week_df[j]['lunch_num'], equal_var = True)
            print(pval_t,'    vs', week_tx[j])
        else :
            _2, pval_t = ttest_ind(week_df[i]['lunch_num'], week_df[j]['lunch_num'], equal_var = False)
            print(pval_t,'    vs', week_tx[j])
    print('=' * 40)
```

요일별로 집단을 나누어 t-test를 시행하기 위해 위 코드를 작성하였다.

```
========================================
train_mon
1.0     vs train_mon
7.460344657903165e-47     vs train_tue
3.138241722033116e-46     vs train_wed
1.360769834405051e-82     vs train_thu
6.201291395979666e-139     vs train_fri
========================================
train_tue
7.460344657903165e-47     vs train_mon
1.0     vs train_tue
0.13853103300217712     vs train_wed
1.1689702494479013e-18     vs train_thu
8.540662591390006e-86     vs train_fri
========================================
train_wed
3.138241722033116e-46     vs train_mon
0.13853103300217712     vs train_tue
1.0     vs train_wed
4.7957244870189515e-12     vs train_thu
2.1986296403397977e-69     vs train_fri
========================================
train_thu
1.360769834405051e-82     vs train_mon
1.1689702494479013e-18     vs train_tue
4.7957244870189515e-12     vs train_wed
1.0     vs train_thu
1.5760907265634135e-41     vs train_fri
========================================
train_fri
6.201291395979666e-139     vs train_mon
8.540662591390006e-86     vs train_tue
2.1986296403397977e-69     vs train_wed
1.5760907265634135e-41     vs train_thu
1.0     vs train_fri
========================================
```

화요일과 수요일만 점심 식수 인원에 유의미한 차이가 없고 나머지 요일끼리는 유의미한 차이가 있다.

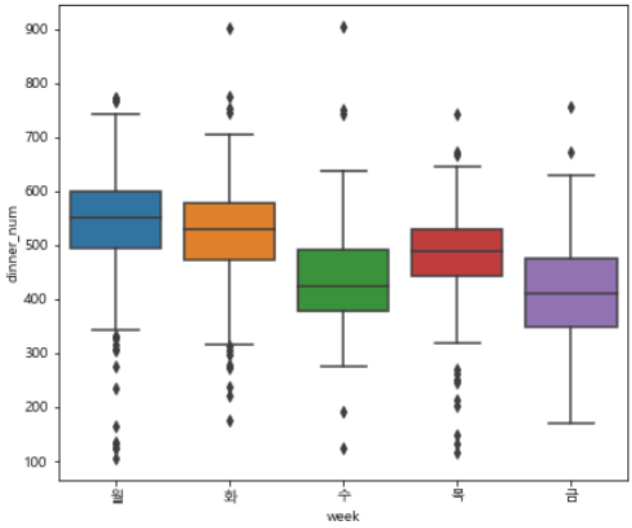


월요일 점심 식수 인원이 다른 요일에 비해 많고 금요일 점심 식수 인원이 다른 요일에 비해 적은 것을 알 수 있다.

다음은 저녁 식수 인원에 대해 시행하겠다.

```
========================================
train_mon
1.0     vs train_mon
0.07854221031324399     vs train_tue
8.909818362151205e-23     vs train_wed
1.835465537685554e-10     vs train_thu
2.029104126138287e-36     vs train_fri
========================================
train_tue
0.07854221031324399     vs train_mon
1.0     vs train_tue
4.116724508931756e-19     vs train_wed
7.192441494033345e-07     vs train_thu
2.3025923660791158e-32     vs train_fri
========================================
train_wed
8.909818362151205e-23     vs train_mon
4.116724508931756e-19     vs train_tue
1.0     vs train_wed
5.175386943689183e-07     vs train_thu
0.002171886803901302     vs train_fri
========================================
train_thu
1.835465537685554e-10     vs train_mon
7.192441494033345e-07     vs train_tue
5.175386943689183e-07     vs train_wed
1.0     vs train_thu
1.774528396002984e-16     vs train_fri
========================================
train_fri
2.029104126138287e-36     vs train_mon
2.3025923660791158e-32     vs train_tue
0.002171886803901302     vs train_wed
1.774528396002984e-16     vs train_thu
1.0     vs train_fri
========================================
```

월요일과 화요일만 저녁 식수 인원에 유의미한 차이가 없고 나머지 요일끼리는 유의미한 차이가 있다. 하지만 월요일과 화요일도 pvalue가 0.05와 미미한 차이이다.



마지막으로 **시간외 근무자가 많은 날과 적은 날이 저녁 식수 인원에 차이가 있는지** 알아보겠다.

```
train['mem_ot'].describe()
```

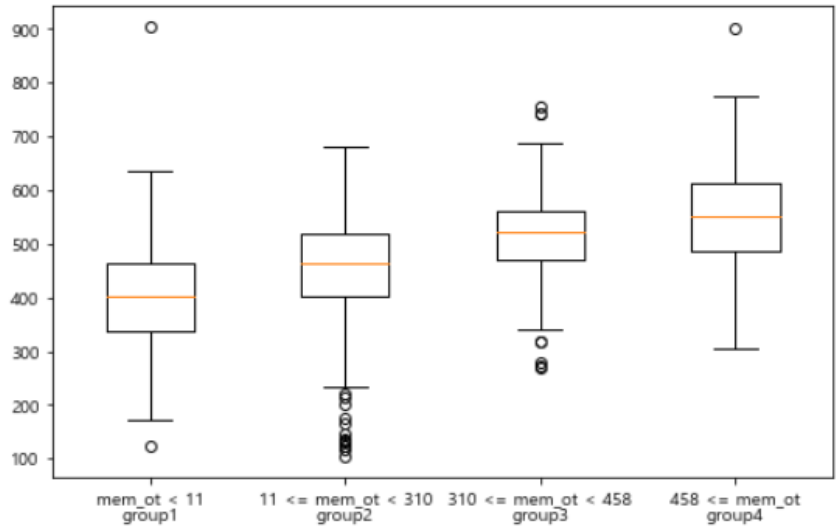
```
count    1161.000000
mean      284.371232
std       245.045800
min         0.000000
25%        11.000000
50%       310.000000
75%       458.000000
max      1044.000000
Name: mem_ot, dtype: float64
```

```
train_ot1 = train[train['mem_ot'] < 11]
train_ot2 = train[(train['mem_ot'] >= 11) & (train['mem_ot'] < 310)]
train_ot3 = train[(train['mem_ot'] >= 310) & (train['mem_ot'] < 458)]
train_ot4 = train[train['mem_ot'] >= 458]
```

시간외 근무자는 1사분위수, 2사분위수, 3사분위수를 기준으로 4가지 집단으로 구분하였다.

```
======================================
group1
1.0     vs group1
9.78058113341322e-11     vs group2
2.2137106993412654e-50     vs group3
3.41521999161864e-64     vs group4
======================================
group2
9.78058113341322e-11     vs group1
1.0     vs group2
2.6219817819803154e-16     vs group3
3.6537336819460045e-29     vs group4
======================================
group3
2.2137106993412654e-50     vs group1
2.6219817819803154e-16     vs group2
1.0     vs group3
1.2537514764110131e-06     vs group4
======================================
group4
3.41521999161864e-64     vs group1
3.6537336819460045e-29     vs group2
1.2537514764110131e-06     vs group3
1.0     vs group4
======================================
```

결과를 보면 모든 pvalue가 매우 작아 모든 집단끼리 유의미한 차이가 있다.



시간외 근무자가 많아질수록 저녁 식수 인원이 많아진다고 보여진다.

8. 가설검정 결과

가설	가설검정 결과	
강수 여부에 따라 식수 인원에 차이가 있다	점심	점심에 강수여부에 따라 점심 식수 인원에 유의미한 차이가 있다.
	저녁	저녁에 강수여부에 따라 저녁 식수 인원에 유의미한 차이가 없다.
가설인 기온에 따라 식수 인원에 차이가 있다	점심	점심에 기온이 낮을 때와 중간일때 점심 식수 인원은 유의미한 차이가 없지만 나머지 집단끼리는 유의미한 차이가 있다.
	저녁	저녁에 기온이 낮을 때와 높을때 저녁 식수 인원은 유의미한 차이가 없지만 나머지 집단끼리는 유의미한 차이가 있다.
요일에 따라 식수 인원에 차이가 있다	점심	화요일과 수요일만 점심 식수 인원에 유의미한 차이가 없고 나머지 요일끼리는 유의미한 차이가 있다.
	저녁	월요일과 화요일만 저녁 식수 인원에 유의미한 차이가 없고 나머지 요일끼리는 유의미한 차이가 있다. 하지만 월요일과 화요일도 pvalue가 0.05와 미미한 차이이다.
시간외 근무자가 많은 날과 적은 날이 저녁 식수 인원에 차이가 있다	모든 집단끼리 유의미한 차이가 있다.	

9. 모델링

다음으로 모델을 이용해 수요예측을 해보았다.

```
feature = ['month','day','week','mem_poss','mem_ot','mem_hj','lunch_tem','lunch_rain',
           'dinner_tem','dinner_rain','holi_be','holi_af','summer1','year_end']
```

```
target = ['lunch_num','dinner_num']
```

```
train_model = train[feature+target]
```

```
x_l = train_model[['month', 'day', 'mem_poss', 'mem_ot', 'lunch_tem','lunch_rain',
                   'holi_be','holi_af','summer1','year_end']]
y_l = train_model['lunch_num']

x_d = train_model[['month', 'day', 'mem_poss', 'mem_ot', 'dinner_tem','dinner_rain',
                   'holi_be','holi_af','summer1','year_end']]
y_d = train_model['dinner_num']
```

```
x_l_train, x_l_test, y_l_train, y_l_test = train_test_split(x_l, y_l, test_size=0.3, shuffle=True, random_state=1004)
x_d_train, x_d_test, y_d_train, y_d_test = train_test_split(x_d, y_d, test_size=0.3, shuffle=True, random_state=1004)
```

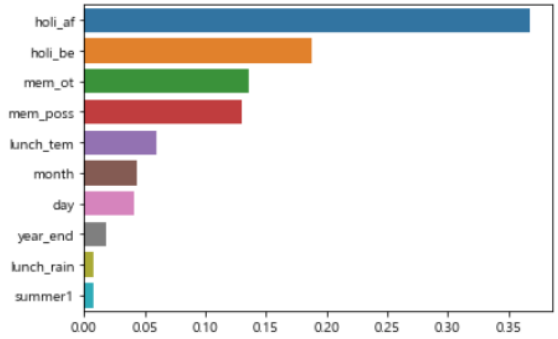
먼저 모델에 사용할 데이터프레임을 만들고 train set과 test set으로 분리했다.

RandomForestRegressor

```
from sklearn.ensemble import RandomForestRegressor
rf_l = RandomForestRegressor()
rf_l.fit(x_l_train, y_l_train)
rf_l.score(x_l_test, y_l_test)
```

```
0.714183157691914
```

```
ft_importance_values = rf_l.feature_importances_
ft_series = pd.Series(ft_importance_values, index = x_l.columns)
ft_top = ft_series.sort_values(ascending=False)
plt.figure()
g = sns.barplot(x = ft_top, y = ft_top.index)
g
plt.show()
```

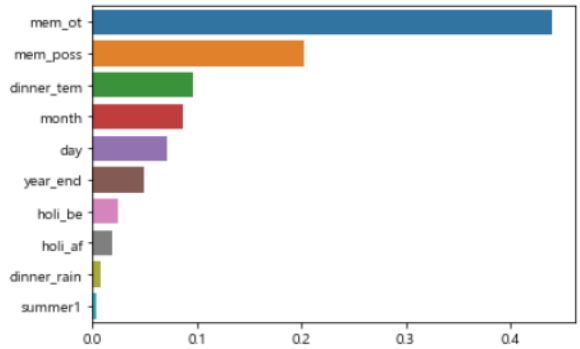


점심score는 0.71418, 점심 예측 모델에서 변수 중요도는 휴일 전날과 휴일 다음날인지가 중요하다.

```
rf_d = RandomForestRegressor()
rf_d.fit(x_d_train, y_d_train)
rf_d.score(x_d_test, y_d_test)
```

```
0.499673943890816
```

```
ft_importance_values = rf_d.feature_importances_
ft_series = pd.Series(ft_importance_values, index = x_d.columns)
ft_top = ft_series.sort_values(ascending=False)
plt.figure()
g = sns.barplot(x = ft_top, y = ft_top.index)
g
plt.show()
```



저녁score는 0.5, 저녁 예측 모델에서 변수 중요도는 시간외 근무자가 중요하다.

점심에 비해 저녁 식수 인원 예측력이 낮다.

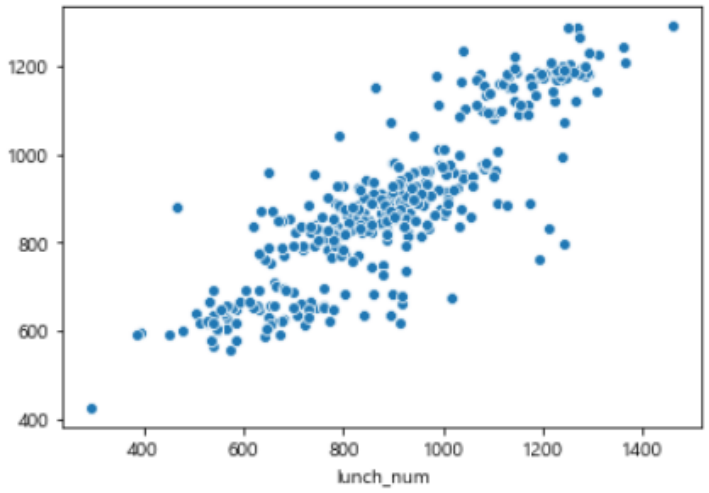
XGBoost

```
xg_l = XGBRegressor()
xg_l.fit(x_l_train, y_l_train)
```

•••

```
sns.scatterplot(y_l_test, xg_l.predict(x_l_test))
print('점심 식수인원 예측 score : ',xg_l.score(x_l_test, y_l_test))
```

```
점심 식수인원 예측 score :  0.7447773997576094
```

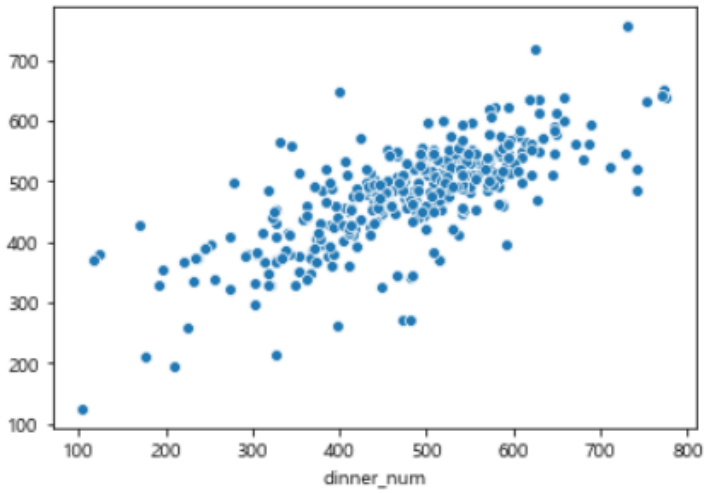



```
xg_d = XGBRegressor()
xg_d.fit(x_d_train, y_d_train)
```

•••

```
sns.scatterplot(y_d_test, xg_d.predict(x_d_test))
print('저녁 식수인원 예측 score : ',xg_d.score(x_d_test, y_d_test))
```

```
저녁 식수인원 예측 score :  0.5462858717622264
```



RandomForest와 마찬가지로 점심에 비해 저녁 식수 인원 예측력이 낮다.

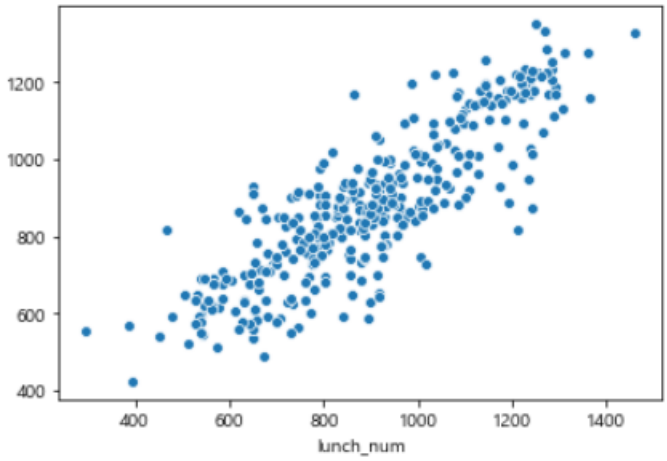
LGBM

```
lg_l = lgb.LGBMRegressor()
lg_l.fit(x_l_train, y_l_train)
```

•••

```
sns.scatterplot(y_l_test, lg_l.predict(x_l_test))
print('점심 식수인원 예측 score : ',lg_l.score(x_l_test, y_l_test))
```

```
점심 식수인원 예측 score :  0.7202868530727615
```

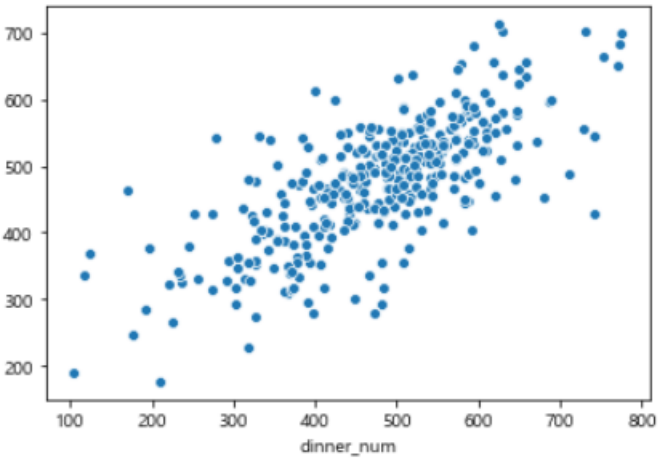



```
lg_d = lgb.LGBMRegressor()
lg_d.fit(x_d_train, y_d_train)
```

•••

```
sns.scatterplot(y_d_test, lg_d.predict(x_d_test))
print('저녁 식수인원 예측 score : ',lg_d.score(x_d_test, y_d_test))
```

```
저녁 식수인원 예측 score :  0.5187265861971204
```

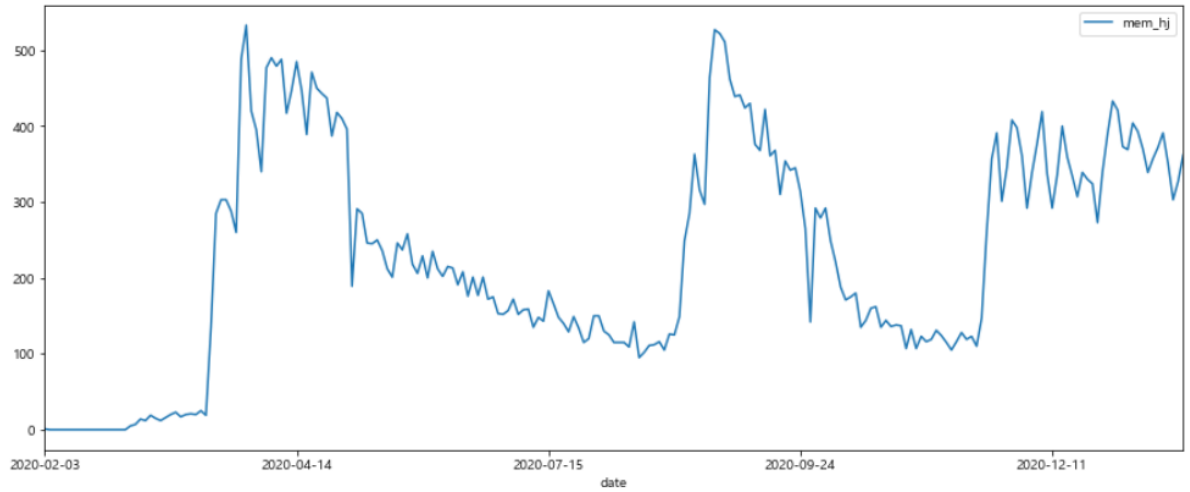


앞선 두 모델들과 마찬가지로 점심에 비해 저녁 식수 인원 예측력이 낮다.

왜 모든 모델들이 점심보다 저녁예측력이 낮은지 생각을 해보다 한가지를 발견했다.

```
train[train['date'] > '2020-02-01'].plot('date', 'mem_hj', figsize = (15, 6))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x21bf732a1c8>
```



위 그래프는 재택근무자의 시계열 그래프이다. 재택근무자가 늘어나는 시기를 보면 2020년 2월경 신천지 코로나시기이다. 그렇기 때문에 코로나 이전과 이후를 나누어 보았다.

```
before_corona = train[train['date'] < '2020-03-01']
after_corona = train[train['date'] > '2020-03-01']
```

그 후 다시 모델을 적용시켰다.

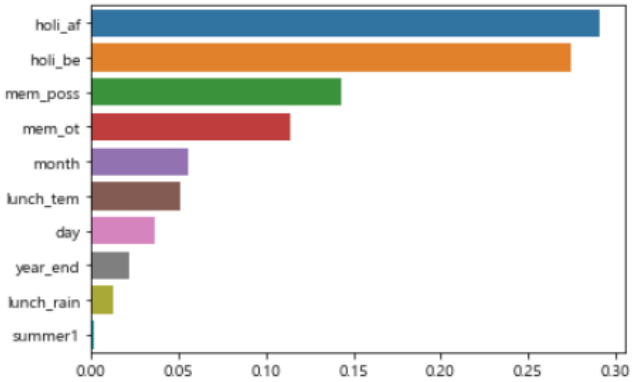
RandomForestRegressor

코로나 이전

```
rf_bc_l = RandomForestRegressor()
rf_bc_l.fit(x_bc_l_train, y_bc_l_train)
rf_bc_l.score(x_bc_l_test, y_bc_l_test)
```

```
0.7270559293223549
```

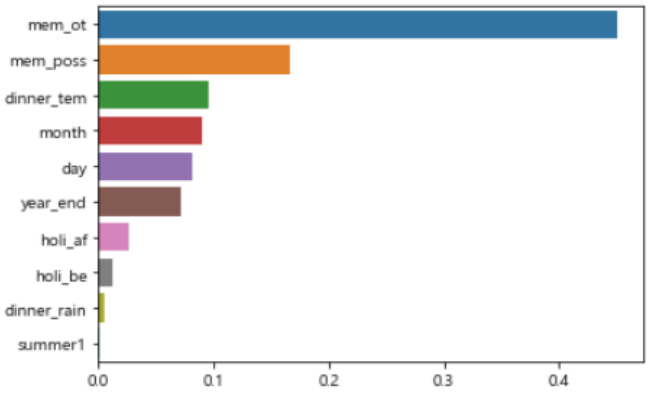
```
ft_importance_values = rf_bc_l.feature_importances_
ft_series = pd.Series(ft_importance_values, index = x_bc_l.columns)
ft_top = ft_series.sort_values(ascending=False)
plt.figure()
g = sns.barplot(x = ft_top, y = ft_top.index)
g
plt.show()
```




```
rf_bc_d = RandomForestRegressor()
rf_bc_d.fit(x_bc_d_train, y_bc_d_train)
rf_bc_d.score(x_bc_d_test, y_bc_d_test)
```

```
0.4650135510298954
```

```
ft_importance_values = rf_bc_d.feature_importances_
ft_series = pd.Series(ft_importance_values, index = x_bc_d.columns)
ft_top = ft_series.sort_values(ascending=False)
plt.figure()
g = sns.barplot(x = ft_top, y = ft_top.index)
g
plt.show()
```

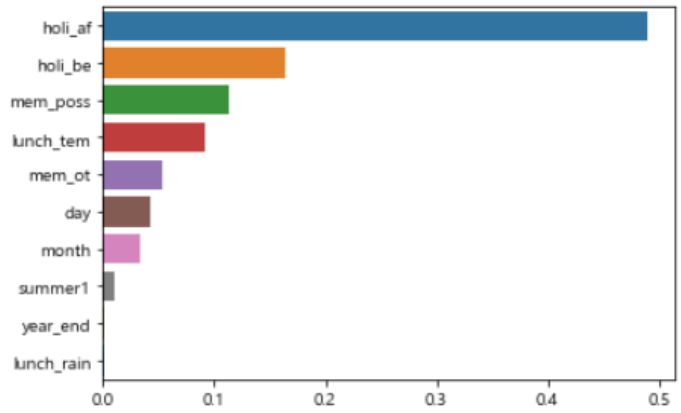


코로나 이후

```
rf_ac_l = RandomForestRegressor()
rf_ac_l.fit(x_ac_l_train, y_ac_l_train)
rf_ac_l.score(x_ac_l_test, y_ac_l_test)
```

```
0.5065103240089295
```

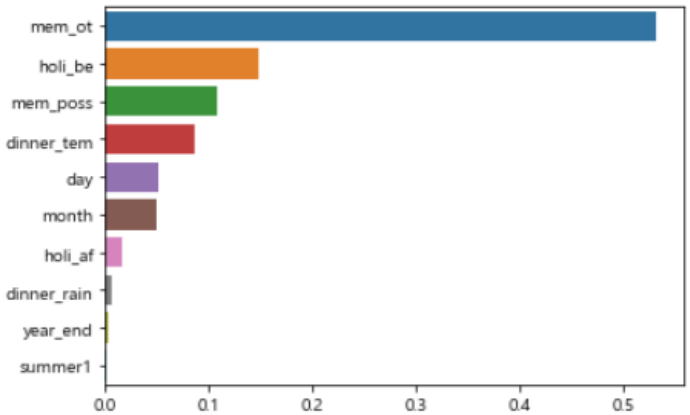
```
ft_importance_values = rf_ac_l.feature_importances_
ft_series = pd.Series(ft_importance_values, index = x_ac_l.columns)
ft_top = ft_series.sort_values(ascending=False)
plt.figure()
g = sns.barplot(x = ft_top, y = ft_top.index)
g
plt.show()
```




```
rf_ac_d = RandomForestRegressor()
rf_ac_d.fit(x_ac_d_train, y_ac_d_train)
rf_ac_d.score(x_ac_d_test, y_ac_d_test)
```

```
0.7547481018485069
```

```
ft_importance_values = rf_ac_d.feature_importances_
ft_series = pd.Series(ft_importance_values, index = x_ac_d.columns)
ft_top = ft_series.sort_values(ascending=False)
plt.figure()
g = sns.barplot(x = ft_top, y = ft_top.index)
g
plt.show()
```



XGBoost

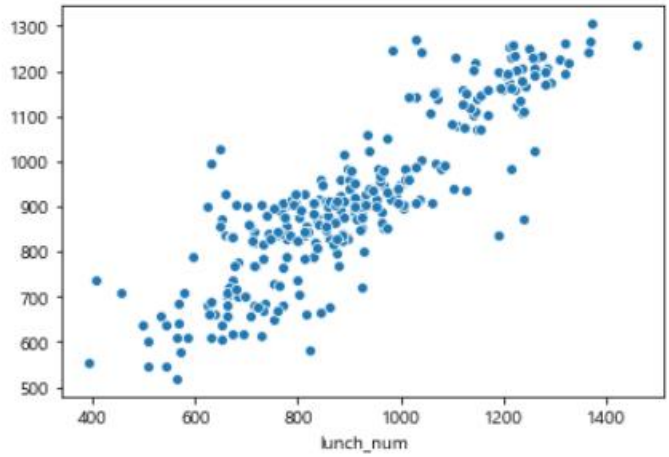
코로나 이전

```
xg_bc_l = XGBRegressor()
xg_bc_l.fit(x_bc_l_train, y_bc_l_train)
```

•••

```
sns.scatterplot(y_bc_l_test, xg_bc_l.predict(x_bc_l_test))
print('코로나 이전 점심 식수인원 예측 score : ',xg_bc_l.score(x_bc_l_test, y_bc_l_test))
```

```
코로나 이전 점심 식수인원 예측 score :  0.7545933443756944
```

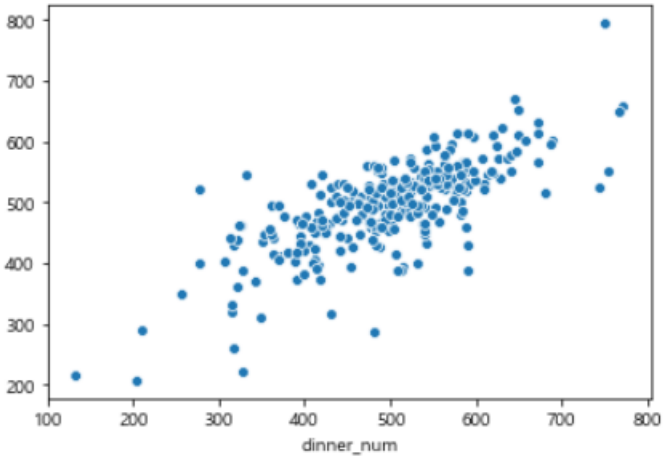



```
xg_bc_d = XGBRegressor()
xg_bc_d.fit(x_bc_d_train, y_bc_d_train)
```

•••

```
sns.scatterplot(y_bc_d_test, xg_bc_d.predict(x_bc_d_test))
print('코로나 이전 저녁 식수인원 예측 score : ',xg_bc_d.score(x_bc_d_test, y_bc_d_test))
```

```
코로나 이전 저녁 식수인원 예측 score :  0.5509644717971985
```



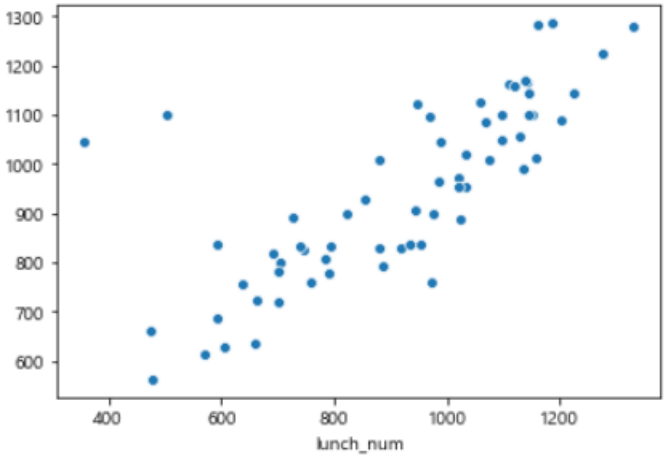
코로나 이후

```
xg_ac_l = XGBRegressor()
xg_ac_l.fit(x_ac_l_train, y_ac_l_train)
```

•••

```
sns.scatterplot(y_ac_l_test, xg_ac_l.predict(x_ac_l_test))
print('코로나 이후 점심 식수인원 예측 score : ',xg_ac_l.score(x_ac_l_test, y_ac_l_test))
```

```
코로나 이후 점심 식수인원 예측 score :  0.5740874145466672
```

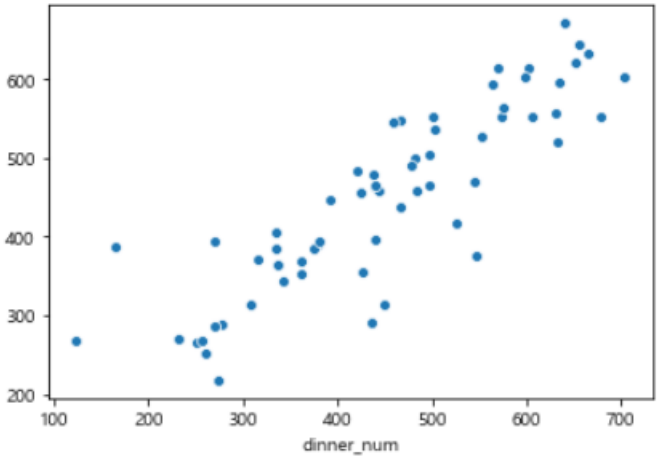



```
xg_ac_d = XGBRegressor()
xg_ac_d.fit(x_ac_d_train, y_ac_d_train)
```

•••

```
sns.scatterplot(y_ac_d_test, xg_ac_d.predict(x_ac_d_test))
print('코로나 이후 저녁 식수인원 예측 score : ',xg_ac_d.score(x_ac_d_test, y_ac_d_test))
```

```
코로나 이후 저녁 식수인원 예측 score :  0.7557081191168687
```



LGBM

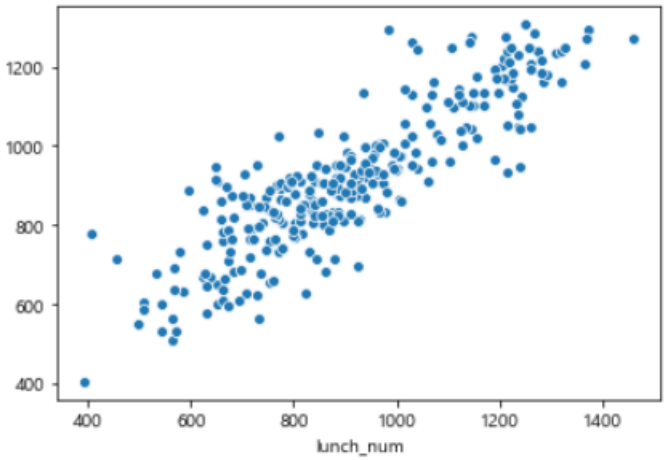
코로나 이전

```
lg_bc_l = lgb.LGBMRegressor()
lg_bc_l.fit(x_bc_l_train, y_bc_l_train)
```

•••

```
sns.scatterplot(y_bc_l_test, lg_bc_l.predict(x_bc_l_test))
print('코로나 이전 점심 식수인원 예측 score : ',lg_bc_l.score(x_bc_l_test, y_bc_l_test))
```

```
코로나 이전 점심 식수인원 예측 score :  0.7530076395271359
```

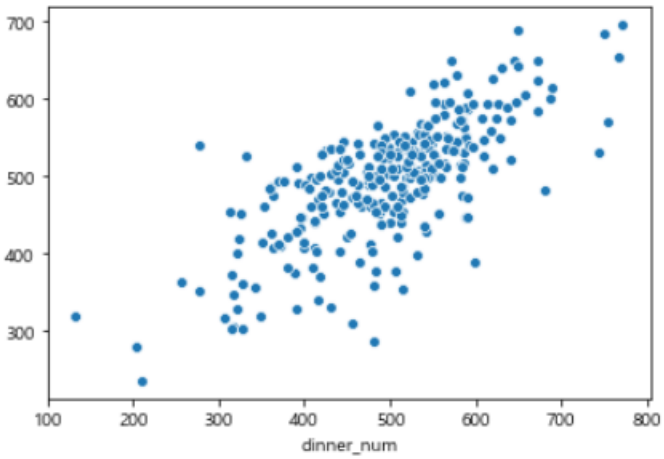



```
lg_bc_d = lgb.LGBMRegressor()
lg_bc_d.fit(x_bc_d_train, y_bc_d_train)
```

•••

```
sns.scatterplot(y_bc_d_test, lg_bc_d.predict(x_bc_d_test))
print('코로나 이전 저녁 식수인원 예측 score : ',lg_bc_d.score(x_bc_d_test, y_bc_d_test))
```

```
코로나 이전 저녁 식수인원 예측 score :  0.5271551532704286
```



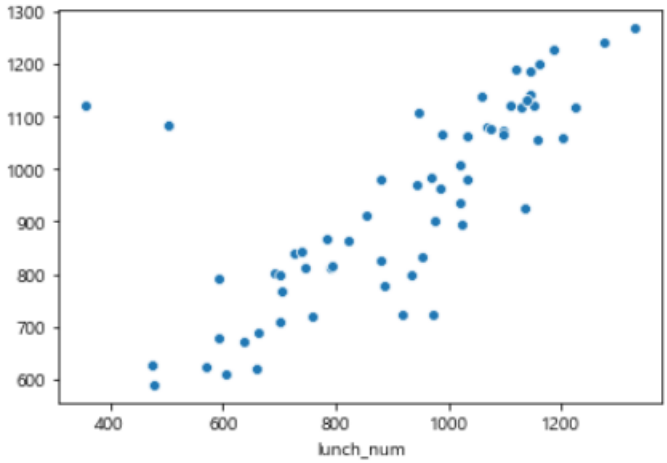
코로나 이후

```
lg_ac_l = lgb.LGBMRegressor()
lg_ac_l.fit(x_ac_l_train, y_ac_l_train)
```

•••

```
sns.scatterplot(y_ac_l_test, lg_ac_l.predict(x_ac_l_test))
print('코로나 이후 점심 식수인원 예측 score : ',lg_ac_l.score(x_ac_l_test, y_ac_l_test))
```

```
코로나 이후 점심 식수인원 예측 score :  0.5595293746116121
```

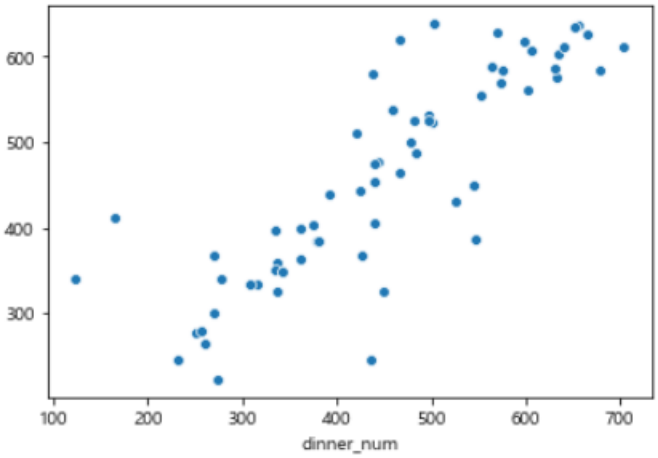



```
lg_ac_d = lgb.LGBMRegressor()
lg_ac_d.fit(x_ac_d_train, y_ac_d_train)
```

•••

```
sns.scatterplot(y_ac_d_test, lg_ac_d.predict(x_ac_d_test))
print('코로나 이후 저녁 식수인원 예측 score : ',lg_ac_d.score(x_ac_d_test, y_ac_d_test))
```

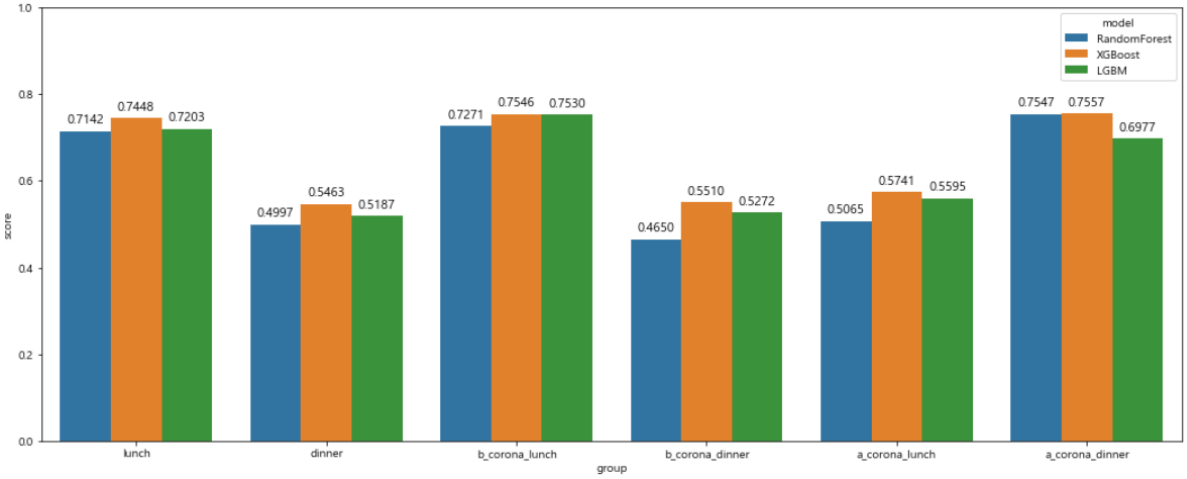
```
코로나 이후 저녁 식수인원 예측 score :  0.69770460752862
```



위 모델의 예측 score를 정리해보았다.

```
score = pd.DataFrame(columns = ['group','score','model'])
```

```
score.loc[0] = ['lunch',rf_l.score(x_l_test, y_l_test), 'RandomForest']
score.loc[1] = ['dinner',rf_d.score(x_d_test, y_d_test), 'RandomForest']
score.loc[2] = ['b_corona_lunch',rf_bc_l.score(x_bc_l_test, y_bc_l_test), 'RandomForest']
score.loc[3] = ['b_corona_dinner',rf_bc_d.score(x_bc_d_test, y_bc_d_test), 'RandomForest']
score.loc[4] = ['a_corona_lunch',rf_ac_l.score(x_ac_l_test, y_ac_l_test), 'RandomForest']
score.loc[5] = ['a_corona_dinner',rf_ac_d.score(x_ac_d_test, y_ac_d_test), 'RandomForest']
score.loc[6] = ['lunch',xg_l.score(x_l_test, y_l_test), 'XGBoost']
score.loc[7] = ['dinner',xg_d.score(x_d_test, y_d_test), 'XGBoost']
score.loc[8] = ['b_corona_lunch',xg_bc_l.score(x_bc_l_test, y_bc_l_test), 'XGBoost']
score.loc[9] = ['b_corona_dinner',xg_bc_d.score(x_bc_d_test, y_bc_d_test), 'XGBoost']
score.loc[10] = ['a_corona_lunch',xg_ac_l.score(x_ac_l_test, y_ac_l_test), 'XGBoost']
score.loc[11] = ['a_corona_dinner',xg_ac_d.score(x_ac_d_test, y_ac_d_test), 'XGBoost']
score.loc[12] = ['lunch',lg_l.score(x_l_test, y_l_test), 'LGBM']
score.loc[13] = ['dinner',lg_d.score(x_d_test, y_d_test), 'LGBM']
score.loc[14] = ['b_corona_lunch',lg_bc_l.score(x_bc_l_test, y_bc_l_test), 'LGBM']
score.loc[15] = ['b_corona_dinner',lg_bc_d.score(x_bc_d_test, y_bc_d_test), 'LGBM']
score.loc[16] = ['a_corona_lunch',lg_ac_l.score(x_ac_l_test, y_ac_l_test), 'LGBM']
score.loc[17] = ['a_corona_dinner',lg_ac_d.score(x_ac_d_test, y_ac_d_test), 'LGBM']
```



모든 집단에서 XGBoost 모델의 예측력이 높은 것을 알 수 있다.

또한 전체와 코로나 이전일때 점심 식수 인원 예측력이 저녁 식수 인원 예측력보다 높은데 코로나 이후 시기에는 점심 식수 인원 예측력보다 저녁 식수 인원 예측력이 높다.

프로젝트를 진행하면서 아쉬웠던 점은 모델링에 대한 지식 부족으로 인해 예측력이 많이 낮은 것이다. 이는 GRIDSearch를 통해 parameter tuning을 거치면 더 높은 예측력을 가진 모델을 만들 수 있으리라 예상된다. 그리고 해당 프로젝트에서는 메뉴 데이터를 삭제하였지만 형태소분석을 통해 메뉴를 구분하면 보다 정확한 분석이 이루어질 것이라 생각된다.