머신러닝 기반 구내식당 식수 인원 예측

2021.10.06

[1조] 구자호
고준수 김병훈
김연주 김진혁

목차

01

프로젝트 배경

- ✓ 프로젝트 주제
- ✓ 프로젝트 선정 이유

프로젝트 배경

프로젝트 주제

주 제

✓ 한국토지주택공사(LH) 구내식당 식수 인원 예측

목 적

✓ 시설물 관리, 행사 주최, 매장 운영 등 사회 각종 분야에서 이용 인원 예측은 중요한 화두

✓ 요인 분석을 통해서 이용 인원을 합리적인 방법으로 예측할 수 있다면 불확실한 상황에서 발생하는 문제들을 회피하고, 선제적인 대응을 가능하게 함

✓ 한국토지주택공사(LH) 구내식당 식수 인원 데이터를 활용하여 인원 수를 예측해보면서 여러 데이터 사이의 상관관계를 찾아보고, 회귀 분석 기술을 익히는 것이 목적

방 법

✓ 회귀 분석을 이용한 탐색적 자료분석, 시각화를 통해 현황을 파악하고 예측하고자 함

프로젝트 배경

프로젝트 주제 선정 이유

담당자의 직관적인 경험에 의존하여 식수인원 예측 → 음식물 폐기물 발생 → 음식물 폐기물로 인한 환경문제와 비용손실

만약, 식수 인원을 예측할 수 있다면??

머신러닝을 이용한 식수 인원 예측 → 불확실한 식수 인원 대응 가능 → 음식물 폐기물 양 감소 → 환경문제와 비용손실 감소

02

프로젝트
팀 구성 및 역할

- ✓ 팀 구성 소개
- ✓ 역할 분담

프로젝트 팀 구성 및 역할

취업을 원한다 일을 원한다! 우리의 이름은 1쥐!!

팀장
구 자 호

데이터 수집
데이터 전처리
모델 평가

팀원
김 진 혁

데이터 수집
데이터 전처리
모델 평가

팀원
고 준 수

데이터 수집
PT제작

팀원
김 병 훈

데이터 수집
PT 발표

팀원
김 연 주

데이터 수집
데이터 시각화

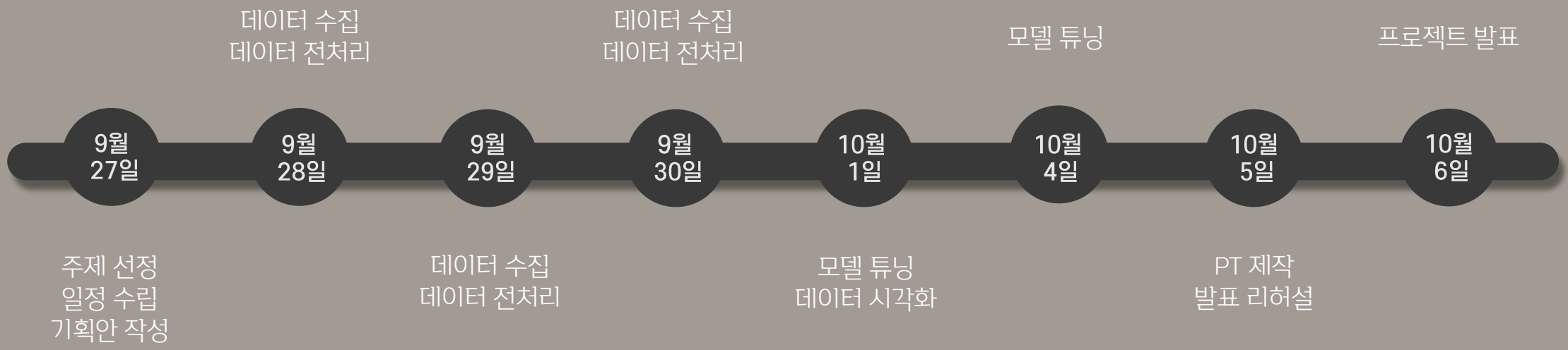
03

프로젝트 수행 절차 및 방법

- ✓ 수행 절차
- ✓ 데이터 수집
- ✓ 데이터 전처리
- ✓ 데이터 분석 및 시각화
- ✓ 예측 모델 선택
- ✓ 예측 모델 튜닝
- ✓ 인원 수 예측
- ✓ 최종 결과

프로젝트 수행 절차 및 방법

수행 절차



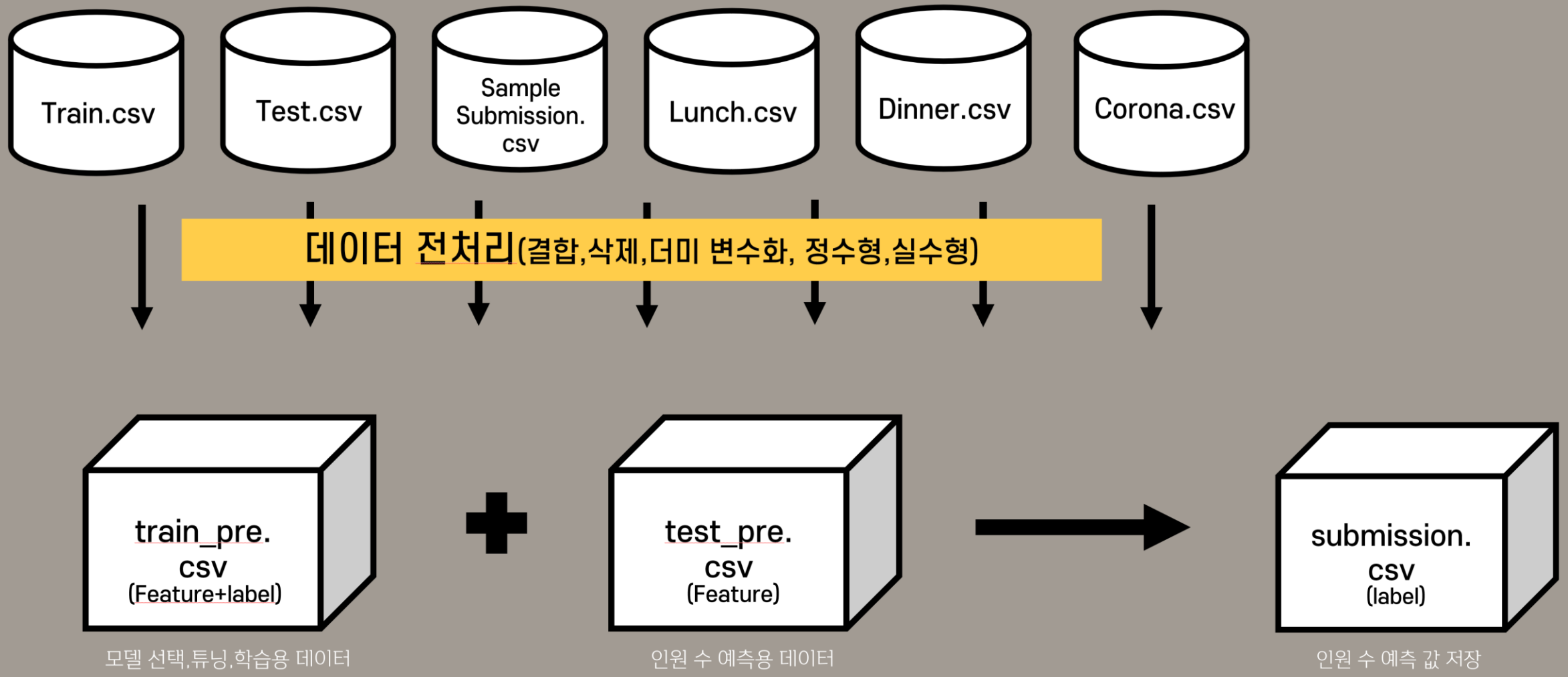
프로젝트 수행 절차 및 방법

데이터 수집 과정

	LH 구내식당 데이터	휴일	기상청	코로나 신규 확진자	코로나 거리 두기 단계
항목	일자 요일 본사정원수 본사휴가자수 본사출장자수 시간외근무명령서승인건수 현본사소속재택근무자수 조식메뉴 중식메뉴 석식메뉴 중식계 석식계	휴일 휴일전날	시간 기온 강수량 풍속 습도 적설량 지면온도	코로나신규확진자수	거리두기단계
활용 방법		기본 제공된 데이터와 네이버 캘린더를 참고하여 휴일 데이터를 생성하고, 생성된 데이터를 이용하여 휴일 하루 전 근무일을 뜻하는 휴일전날 데이터 생성	공공데이터 사이트에서 진주시의 시간대별 기상 데이터를 통해 식사 시간에 영향을 미치는 날씨 데이터 생성	진주시청 홈페이지에서 코로나 관련 보도자료를 모두 취합하여 일자별 진주시의 코로나 신규 확진자 데이터 생성	진주시청 거리두기 보도자료와 기사 라이브러리를 활용하여 일자별 진주시의 코로나 거리 두기 단계 데이터 생성
출처	DACON DATA TO VALUE	LH 한국토지주택공사	기상청	진주시	진주시

프로젝트 수행 절차 및 방법

데이터 전처리 (1) 과정



프로젝트 수행 절차 및 방법

데이터 전처리 (2) 기상청1

✓ 1. API 호출 및 데이터프레임 생성

```
[ ]  1 encoding_k = '5hKnz%2FSOwgTelwTnXx36H%2BKOB9ObVv9LZ5KPCTZiYY0hYEubhGfovemBVL884JkVyGPNNj5aD%2BYPJ2k3cKFkvg%
     2 decoding_k = '5hKnz/SOwgTelwTnXx36H+KOB9ObVv9LZ5KPCTZiYY0hYEubhGfovemBVL884JkVyGPNNj5aD+YPJ2k3cKFkvg=='
```

```
[ ]  1 start_date = '20160201' # train 데이터의 시작 일자
     2 end_date = '20210126' # train 데이터의 마감 일자
     3 page_no = '1'
     4 num_rows = '999' # 최대 1000개 이상을 요청할수는 없음..
     5 location = '192' # 진주 위치코드
```

```
[ ]  1 # 기상청_지상(종관, ASOS) 시간자료 조회서비스 (https://www.data.go.kr/data/15057210/openapi.do)
     2 baseurl = 'http://apis.data.go.kr/1360000/AsosHourlyInfoService/getWthrDataList?serviceKey=' + encoding_k
     3 url1 = '&numOfRows=' + num_rows
     4 url2 = '&pageNo=' + page_no
     5 url3 = '&dataCd=ASOS&dateCd=HR'
     6 url4 = '&stnIds=' + location # 위치 코드
     7 url5 = '&endDt=' + end_date
     8 url6 = '&endHh=' + '19' #
     9 url7 = '&startHh='+ '11' # 1시간 단위로 체크
    10 url8 = '&startDt=' + start_date
    11
    12 url = baseurl + url1 + url2 + url3 + url4 + url5 + url6 + url7 + url8
    13 print(url)
```

```
 1 # - tm : 시간
 2 # - ta : 기온
 3 # - rn : 강수량
 4 # - ws : 풍속
 5 # - hm : 습도
 6 # - dsnw : 적설
 7 # - ts : 지면온도
 8
 9 data = soup.find_all('item')
10 con_lst=[]
11
12 for item in data:
13     time = item.find('tm').text # 시간
14     temper = item.find('ta').text # 기온
15     rain = item.find('rn').text # 강수량
16     wind = item.find('ws').text # 풍속
17     hm = item.find('hm').text # 습도
18     snow = item.find('dsnw').text # 적설
19     tem_s = item.find('ts').text # 지면온도
20     con_lst.append({'날짜':time, '기온':temper, '습도':hm,
21                     '강수량':rain, '적설량':snow, '풍속':wind})
22 con_lst
```

```
 1 xml = requests.get(url)
 2 soup = bs(xml.text, 'html.parser')
```

```
4 df = pd.DataFrame(con_lst, columns = ['날짜','기온','습도','강수량','적설량','풍속'])
5
```

기상청 API 참조

프로젝트 수행 절차 및 방법

데이터 전처리 (3) 기상청2

✓ 2. 결측치 처리 및 체감온도 불쾌지수 생성

```
1 df.강수량.unique()
```

```
array(['', '0.0', '3.6', '4.0', '9.4', '15.8', '20.4', '4.1', '5.5',
       '1.6', '0.1', '0.5', '4.5', '3.0', '0.7', '0.9', '2.1', '0.2'
       '1.8', '1.7', '2.2', '20.3', '5.4', '2.6', '6.1', '16.2', '16
       '1.3', '0.6', '2.7', '6.5', '0.3'], dtype=object)
```

```
[ ] 1 df.적설량.unique()
```

```
array([''], dtype=object)
```

```
[ ] 1 # '' 빈 값을 nan값으로 바꾼후 결측치 제거
    2 df.강수량.replace('', np.nan, inplace=True)
    3 df.적설량.replace('', np.nan, inplace=True)
```

```
[ ] 1 df['강수량'] = df['강수량'].fillna(0)
    2 df['적설량'] = df['적설량'].fillna(0)
```

```
1 numeric_list = ['기온', '습도', '강수량', '적설량', '풍속']
2
3 for x in numeric_list:
4   try:
5     df[x] = df[x].astype(float)
6   except:
7     print(x)
```

Ta : 건구온도 (℃)

RH : 상대습도 (소수단위)

V : 풍속

불쾌지수 = 9/5 x Ta - 0.55 (1-RH)(9/5 x Ta-26)+32

체감온도(℃)=13.12 + 0.6215 × Ta - 11.37 x V^0.16 + 0.3965 x V^0.16 x Ta

```
[ ] 1 # DI = 9/5Ta-0.55(1-RH)(9/5Ta-26)+32
    2 df['불쾌지수'] = (9/5) * df['기온'] - 0.55 * (1-df['습도']/100) * (((9/5) * df['기온']) - 26) + 32
```

```
[ ] 1 # 체감온도(℃)=13.12+0.6215×T-11.37V^0.16+0.3965V^0.16×T
    2 df['체감온도'] = np.round(13.12 + 0.6215 * df['기온'] - \
    3                           11.37 * df['풍속'] ** 0.16 + 0.3965 * df['풍속'] ** 0.16 * df['기온'], 1)
```

```
1 df.head()
```

	날짜	기온	습도	강수량	적설량	풍속	불쾌지수	체감온도
0	2016-02-01 11:00	1.8	27.0	0.0	0.0	1.3	44	3.1
1	2016-02-01 12:00	3.0	24.0	0.0	0.0	1.9	46	3.7
2	2016-02-01 13:00	4.2	22.0	0.0	0.0	1.9	47	5.0
3	2016-02-01 14:00	4.6	20.0	0.0	0.0	2.6	48	4.9
4	2016-02-01 15:00	4.2	20.0	0.0	0.0	3.5	47	3.9

기상청 API 참조

프로젝트 수행 절차 및 방법

✓ 3. 시간데이터를 활용해 점심, 저녁 시간대로 구분

```
1 df['시간'] = df['날짜'].apply(lambda x : x.split(' ')[1])
2 df['날짜'] = df['날짜'].apply(lambda x : x.split(' ')[0])
```

```
[ ] 1 df.head()
```

	날짜	기온	습도	강수량	적설량	풍속	불쾌지수	체감온도	시간
0	2016-02-01	1.8	27.0	0.0	0.0	1.3	44	3.1	11:00
1	2016-02-01	3.0	24.0	0.0	0.0	1.9	46	3.7	12:00
2	2016-02-01	4.2	22.0	0.0	0.0	1.9	47	5.0	13:00

```
[ ] 1 df['날짜'] = pd.to_datetime(df['날짜'], format = '%Y-%m-%d')
2
3 # 계산의 편의성을 위해 시간데이터를 00~23으로 변환
4 df['시간'] = df['시간'].apply(lambda x: int(x.split(':')[0]))
```

```
[28] 1 df.head()
```

	날짜	기온	습도	강수량	적설량	풍속	불쾌지수	체감온도	시간
0	2016-02-01	1.8	27.0	0.0	0.0	1.3	44	3.1	11
1	2016-02-01	3.0	24.0	0.0	0.0	1.9	46	3.7	12
2	2016-02-01	4.2	22.0	0.0	0.0	1.9	47	5.0	13
3	2016-02-01	4.6	20.0	0.0	0.0	2.6	48	4.9	14
4	2016-02-01	4.2	20.0	0.0	0.0	3.5	47	3.9	15

```
[ ] 1 lun_df = df[(df['시간'] > 10) & (df['시간'] < 14)] # 점심 데이터
```

```
1 eve_df = df[(df['시간'] > 16) & (df['시간'] < 20)] # 저녁 데이터
```

```
[ ] 1 lun_df.head()
```

	날짜	기온	습도	강수량	적설량	풍속	불쾌지수	체감온도	시간
0	2016-02-01	1.8	27.0	0.0	0.0	1.3	44	3.1	11
1	2016-02-01	3.0	24.0	0.0	0.0	1.9	46	3.7	12
2	2016-02-01	4.2	22.0	0.0	0.0	1.9	47	5.0	13
24	2016-02-02	0.3	29.0	0.0	0.0	1.1	42	1.9	11
25	2016-02-02	1.7	24.0	0.0	0.0	1.6	44	2.6	12

```
[ ] 1 eve_df.head()
```

	날짜	기온	습도	강수량	적설량	풍속	불쾌지수	체감온도	시간
6	2016-02-01	3.0	25.0	0.0	0.0	2.2	45	3.4	17
7	2016-02-01	0.9	31.0	0.0	0.0	1.5	42	1.9	18
8	2016-02-01	-0.7	39.0	0.0	0.0	1.0	39	1.0	19
30	2016-02-02	5.2	19.0	0.0	0.0	0.9	48	7.2	17
31	2016-02-02	2.4	28.0	0.0	0.0	1.3	44	3.7	18

기상청 API 참조

프로젝트 수행 절차 및 방법

데이터 전처리 (5) 기상청4

✓ 4. 앞서 처리된 데이터를 가지고 더미변수화 진행

```python
# sensible temperature 체감온도
# discomfort index 불쾌지수
# df를 변수로 받아서 3시간 단위로 주어지는 불쾌지수와 체감온도의 평균계산
# 강수량과 적설량을 판단해 우산이 필요한지 판단
# 새로운 DataFrame으로 만들어서 반환하는 함수

def make_st_di_um(df):
  temp_list=[]
  # 순서i 와 idx를 따로 받아서 iloc를 해서 호출해야 정상적으로 불러지고,
  # 날짜 데이터를 가져올때는 idx
  for i, idx in enumerate(lun_df.index):
    if idx % 3 == 0: # 첫 시작 idx를 찾는다
      temp = lun_df.iloc[i:i+3] # 3개의 row를 가져와 계산

      umb = 0 # 우산유무

      for r in temp['강수량'].values:
        if r > 0.0:
          umb = 1

      if umb == 0:
        for s in temp['적설량'].values:
          if s > 0.0:
            umb = 1

      tem_mean = temp['체감온도'].mean()
      di_mean = temp['불쾌지수'].mean()

      # temp['날짜'][idx] 로 호출해야 정상적으로 그 묶음의 날짜를 가져올 수 있음
      temp_list.append({'날짜': temp['날짜'][idx], '체감온도': np.round(tem_mean,1), '불쾌지수': int(di_mean), '우산': umb})
  new_df = pd.DataFrame(temp_list, columns = ['날짜', '체감온도', '불쾌지수', '우산'])
  return new_df
```

```python
def make_discomfort(df):
  df['쾌적'] = df['불쾌지수'].apply(lambda x: 1 if  (x < 75) else 0)
  df['불쾌'] = df['불쾌지수'].apply(lambda x: 1 if (75 <= x) else 0)

  return df
```

```python
def make_temper(df):
  # df['폭염'] = df['체감온도'].apply(lambda x: 1 if x >= 33.0 else 0)
  # df['한파'] = df['체감온도'].apply(lambda x: 1 if x < (-10.0) # 한파에 대해서
  df['저온'] = df['체감온도'].apply(lambda x: 1 if x < 0.0 else 0)
  df['중온'] = df['체감온도'].apply(lambda x: 1 if (0.0 <= x) & (x < 20.0) else 0)
  df['고온'] = df['체감온도'].apply(lambda x: 1 if 20.0 < x else 0)
  return df
```

```python
new_lun_df.head(600).tail()
```

	날짜	체감온도	불쾌지수	우산	쾌적	불쾌	저온	중온	고온
595	2017-09-18	28.3	72	0	1	0	0	0	1
596	2017-09-19	28.4	73	0	1	0	0	0	1
597	2017-09-20	26.5	70	0	1	0	0	0	1
598	2017-09-21	25.9	69	0	1	0	0	0	1
599	2017-09-22	26.4	70	0	1	0	0	0	1

```python
lunch.to_csv(filepath+'lunch.csv', index=False)
dinner.to_csv(filepath+'dinner.csv', index=False)
```

기상청 API 참조

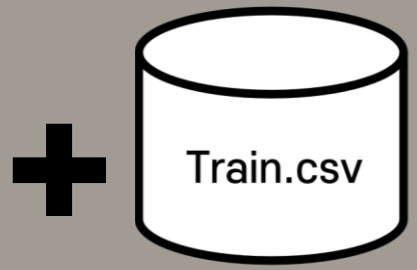
프로젝트 수행 절차 및 방법

데이터 전처리 (6) 기상청5

✓ 5. 데이터 병합을 위해 .csv파일로 만들어 저장

- 일자 : 연-월-일 (2016-02-01 ~ 2021-04-09)
- 체감온도 : 해당 일자의 11~13시 평균 체감온도
- 불쾌지수 : 75미만 쾌적, 75이상 불쾌
- 우산 : 해당 일자의 외출시 우산필요여부

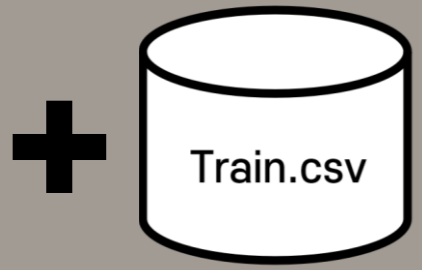
	일자	우산	쾌적	불쾌	저온	중온	고온
0	2016-02-01	0	1	0	0	1	0
1	2016-02-02	0	1	0	0	1	0
2	2016-02-03	0	1	0	0	1	0
3	2016-02-04	0	1	0	0	1	0
4	2016-02-05	0	1	0	0	1	0



	일자	요일	본사정원수	본사휴가자수	본사출장자수	본사시간
0	2016-02-01	월	2601	50	150	
1	2016-02-02	화	2601	50	173	
2	2016-02-03	수	2601	56	180	
3	2016-02-04	목	2601	104	220	
4	2016-02-05	금	2601	278	181	

- 일자 : 연-월-일 (2016-02-01 ~ 2021-04-09)
- 체감온도 : 해당 일자의 17~19시 평균 체감온도
- 불쾌지수 : 75미만 쾌적, 75이상 불쾌
- 우산 : 해당 일자의 외출시 우산필요여부

	일자	우산	쾌적	불쾌	저온	중온	고온
0	2016-02-01	0	1	0	0	1	0
1	2016-02-02	0	1	0	0	1	0
2	2016-02-03	0	1	0	0	1	0
3	2016-02-04	0	1	0	0	1	0
4	2016-02-05	0	1	0	0	1	0



	일자	요일	본사정원수	본사휴가자수	본사출장자수	본사시간
0	2016-02-01	월	2601	50	150	
1	2016-02-02	화	2601	50	173	
2	2016-02-03	수	2601	56	180	
3	2016-02-04	목	2601	104	220	
4	2016-02-05	금	2601	278	181	

기상청 API 참조

프로젝트 수행 절차 및 방법

데이터 전처리 (7) 코로나

corona.csv

corona.csv

- 일자 : 연-월-일 (2020-02-28 ~ 2021-04-09)
- 코로나신규확진자 : 해당 일자의 경상남도 진주시청 브리핑 시간 15:30 이전 신규 확진자 수

1 .[일자]를 기준으로 train.csv + corona.csv

```
train = pd.merge(train, corona, left_on = '일자', right_on = '일자', how = 'left')
test = pd.merge(test, corona, left_on = '일자', right_on = '일자', how = 'left')
```

2. 경상남도 진주시 코로나 첫 확진자는

2020-02-28에 발생하였으므로 그 이전 일자는 결측치

```
train = train.fillna(0)
test = test.fillna(0)
```

level.csv

- 일자 : 연-월-일 (2016-02-01 ~ 2021-04-09)
- 2016-02-01 ~ 2020-01-26 : 평시
- 2020-01-27 ~ 2020-11-25 : 1단계
- 2020-11-26 ~ 2021-01-10 : 2단계 - 진주시 이통장 연수 집단 감염
- 2021-01-11 ~ 2021-01-24 : 2.5단계 - 진주시 교회 집단 감염
- 2021-01-25 ~ 2021-03-11 : 1.5단계
- 2021-03-12 ~ 2021-04-09 : 2단계 - 진주시 목욕탕 집단 감염

1 .[일자]를 기준으로 train.csv + level.csv

```
train = pd.merge(train, level, left_on = '일자', right_on = '일자', how = 'left')
test = pd.merge(test, level, left_on = '일자', right_on = '일자', how = 'left')
```

진주시청 홈페이지 코로나 관련 보도자료 참조

프로젝트 수행 절차 및 방법

데이터 전처리 (8) 휴일1

```
[2016-02-08 월, 2016-02-09 화, 2016-02-10 수, 2016-03-01 화, 2016-04-13 수,
 2016-05-05 목, 2016-05-06 금, 2016-06-06 월, 2016-08-15 월, 2016-09-14 수,
 2016-09-15 목, 2016-09-16 금, 2016-10-03 월, 2017-01-27 금, 2017-01-30 월,
 2017-03-01 수, 2017-05-01 월, 2017-05-03 수, 2017-05-05 금, 2017-05-09 화,
 2017-06-06 화, 2017-08-15 화, 2017-10-09 월, 2017-12-25 월, 2018-01-01 월,
 2018-02-15 목, 2018-02-16 금, 2018-03-01 목, 2018-05-01 화, 2018-05-07 월,
 2018-05-22 화, 2018-06-06 수, 2018-06-13 수, 2018-08-15 수, 2018-10-09 화,
 2018-12-25 화, 2019-01-01 화, 2019-02-04 월, 2019-02-05 화, 2019-02-06 수,
 2019-03-01 금, 2019-05-01 수, 2019-05-06 월, 2019-06-06 목, 2019-08-15 목,
 2019-09-12 목, 2019-09-13 금, 2019-10-01 화, 2019-10-03 목, 2019-10-09 수,
 2019-12-25 수, 2020-01-01 수, 2020-01-24 금, 2020-01-27 월, 2020-04-15 수,
 2020-04-30 목, 2020-05-01 금, 2020-05-05 화, 2020-06-15 월, 2020-06-16 화,
 2020-06-17 수, 2020-06-18 목, 2020-06-19 금, 2020-06-22 월, 2020-06-23 화,
 2020-06-24 수, 2020-06-25 목, 2020-06-26 금, 2020-06-29 월, 2020-06-30 화,
 2020-08-17 월, 2020-09-29 화, 2020-09-30 수, 2020-10-01 목, 2020-10-02 금,
 2020-10-09 금, 2020-12-25 금, 2020-12-28 월, 2020-12-29 화, 2020-12-30 수,
 2020-12-31 목, 2021-01-01 금]
```

평일(월화수목금)만 있음.

[휴일] 데이터는 빠져있음.

```
[2016-02-05 금, 2016-02-29 월, 2016-04-12 화, 2016-05-04 수, 2016-06-03 금,
 2016-08-12 금, 2016-09-13 화, 2016-09-30 금, 2017-01-26 목, 2017-01-27 금,
 2017-02-28 화, 2017-04-28 금, 2017-05-02 화, 2017-05-04 목, 2017-05-08 월,
 2017-06-05 월, 2017-08-14 월, 2017-10-06 금, 2017-12-22 금, 2018-12-29 금,
 2018-02-14 수, 2018-02-28 수, 2018-04-30 월, 2018-05-04 금, 2018-05-21 월,
 2018-06-05 화, 2018-06-12 화, 2018-08-14 화, 2018-10-08 월, 2018-12-24 월,
 2019-12-31 월, 2019-02-01 금, 2019-02-28 목, 2019-04-30 화, 2019-05-03 금,
 2019-06-05 수, 2019-08-14 수, 2019-09-11 수, 2019-09-30 월, 2019-10-02 수,
 2019-10-08 화, 2019-12-24 화, 2019-12-31 화, 2020-01-23 목, 2020-01-24 금,
 2020-04-24 화, 2020-04-29 수, 2020-05-04 월, 2020-06-12 금, 2020-08-14 금,
 2020-09-28 월, 2020-10-08 목, 2020-12-24 목]
```

[휴일] 하루 전 근무일은 석식 이용을 피하고 일찍 퇴근하고자 하는 심리를 반영하여 [휴일전날]을 생성

LH 공식 홈페이지 참조

프로젝트 수행 절차 및 방법


```
train['휴일전날'] = False
```

```
before_holiday = [
    '2016-02-05', '2016-02-29', '2016-04-12', '2016-05-04',
    '2016-06-03', '2016-08-12', '2016-09-13', '2016-09-30',
    '2017-01-26', '2017-01-27', '2017-02-28', '2017-04-28',
    '2017-05-02', '2017-05-04', '2017-05-08', '2017-06-05',
    '2017-08-14', '2017-10-06', '2017-12-22', '2018-12-29',
    '2018-02-14', '2018-02-28', '2018-04-30', '2018-05-04',
    '2018-05-21', '2018-06-05', '2018-06-12', '2018-08-14',
    '2018-10-08', '2018-12-24', '2019-12-31', '2019-02-01',
    '2019-02-28', '2019-04-30', '2019-05-03', '2019-06-05',
    '2019-08-14', '2019-09-11', '2019-09-30', '2019-10-02',
    '2019-10-08', '2019-12-24', '2019-12-31', '2020-01-23',
    '2020-01-24', '2020-04-24', '2020-04-29', '2020-05-04',
    '2020-06-12', '2020-08-14', '2020-09-28', '2020-10-08',
    '2020-12-24'
]
for i in before_holiday:
    train.loc[train['일자'] == i, ('휴일전날')] = True
```

1) Test ['휴일']

[2021-02-11 목, 2021-02-12 금, 2021-03-01 월]

2) Test ['휴일전날']

[2021-02-10 수, 2021-02-26 금]

```
[ ]  1 test['휴일전날'] = False
```

```
[ ]  1 before_holiday = [
     2     '2021-02-10', '2021-02-26'
     3 ]
     4 for i in before_holiday:
     5     test.loc[test['일자'] == i, ('휴일전날')] = True
```

LH 공식 홈페이지 참조

프로젝트 수행 절차 및 방법

데이터 전처리 (10) 전체1

✓ 일자데이터 분리

연월일 각각의 상관관계를 찾아보기 위해

문자열로 된 ['일자'] 컬럼을

Datatime을 이용해 ['년'] ['월'] ['일']로 분리

```
def ymd(df):
    df['일자'] = pd.to_datetime(df['일자'], format = '%Y-%m-%d')
    df['년'] = df['일자'].dt.year
    df['월'] = df['일자'].dt.month
    df['일'] = df['일자'].dt.day
    # df = df.drop('일자', axis = 1)
    return df
```

```
train = ymd(train)
test = ymd(test)
submission = ymd(submission)
```

✓ 요일데이터 분리

'월', '화', '수', '목', '금' 문자로 구성된 ['요일']의 더미변수화

```
train['요일'] = train['요일'].map({'월' : 0, '화' : 1, '수' : 2, '목' : 3, '금' : 4})
test['요일'] = test['요일'].map({'월' : 0, '화' : 1, '수' : 2, '목' : 3, '금' : 4})
```

데이콘 제공 한국토지주택공사 구내식당 데이터 참조

프로젝트 수행 절차 및 방법

✓ 데이터 간 상관관계를 살펴보기 위하여['식사 가능자 수']와 ['식사 참여율']을 구함

[식사 가능 인원] = [총 인원] - [휴가 인원] - [출장 인원] - [재택 근무 인원]

```
train['식사가능자수'] = train['본사정원수'] - train['본사휴가자수'] - train['본사출장자수'] - train['현본사소속재택근무자수']
test['식사가능자수'] = test['본사정원수'] - test['본사휴가자수'] - test['본사출장자수'] - test['현본사소속재택근무자수']
```

[참여율] = [이용 인원] / [식사 가능 인원]

```
train['중식참여율'] = train['중식계'] / train['식사가능자수']
train['석식참여율'] = train['석식계'] / train['식사가능자수']
```

데이콘 제공 한국토지주택공사 구내식당 데이터 참조

프로젝트 수행 절차 및 방법

✓ 실수형 데이터 정수형으로 변환

```python
columns = [
    '본사정원수', '본사휴가자수', '본사출장자수',
    '본사시간외근무명령서승인건수', '현본사소속재택근무자수',
    '중식계', '석식계',
    '식사가능자수', '코로나신규확진자', '휴일전날'
]
for i in columns:
    train[i] = train[i].astype(int)

columns = [
    '본사정원수', '본사휴가자수', '본사출장자수',
    '본사시간외근무명령서승인건수', '현본사소속재택근무자수',
    '식사가능자수', '코로나신규확진자', '휴일전날'
]
for i in columns:
    test[i] = test[i].astype(int)
```

실수형으로 표현된

'정원 수', '휴가자 수', '출장자 수', '초과근무자 수', '재택근무자 수'

등 인원 수와 관련된 데이터를 정수형으로 변환,

True, False Boolean으로 표현된 '휴일전날' 데이터도 정수형으로 변환.

데이콘 제공 한국토지주택공사 구내식당 데이터 참조

프로젝트 수행 절차 및 방법

데이터 전처리 (12) 전체3

✓ 데이터셋의 결측치를 확인함

결측치 확인

```
1 train.isna().sum()
```

```
1 test.isna().sum()
```

```
일자                    0
요일                    0
본사정원수                 0
본사휴가자수                 0
본사출장자수                 0
본사시간외근무명령서승인건수          0
현본사소속재택근무자수           0
중식계                   0
석식계                   0
체감온도(중식)               0
불쾌지수(중식)               0
우산(중식)                 0
체감온도(석식)               0
불쾌지수(석식)               0
우산(석식)                 0
코로나신규확진자              0
거리두기단계                 0
휴일전날                   0
년                      0
월                      0
일                      0
중식메뉴_                  0
석식메뉴_                  0
식사가능자수                 0
중식참여율                  0
석식참여율                  0
```

```
일자                    0
요일                    0
본사정원수                 0
본사휴가자수                 0
본사출장자수                 0
본사시간외근무명령서승인건수          0
현본사소속재택근무자수           0
체감온도(중식)               0
불쾌지수(중식)               0
우산(중식)                 0
체감온도(석식)               0
불쾌지수(석식)               0
우산(석식)                 0
코로나신규확진자              0
거리두기단계                 0
휴일전날                   0
년                      0
월                      0
일                      0
중식메뉴_                  0
석식메뉴_                  0
식사가능자수                 0
```

데이콘 제공 한국토지주택공사 구내식당 데이터 참조

프로젝트 수행 절차 및 방법

데이터 전처리 (13) 전체4

✓ 데이터셋의 컬럼을 순서대로 정리하여 csv 파일로 저장

인원 수 예측용 Train 데이터

```
train_pre = train[[
    '일자', '년', '월', '일', '요일', '휴일전날',
    '본사정원수', '본사휴가자수', '본사출장자수',
    '본사시간외근무명령서승인건수', '현본사소속재택근무자수',
    '식사가능자수',
    '중식계', '석식계',
    '중식참여율', '석식참여율',
    '체감온도(중식)', '불쾌지수(중식)', '우산(중식)',
    '체감온도(석식)', '불쾌지수(석식)', '우산(석식)',
    '코로나신규확진자', '거리두기단계'
]]
```

```
train_pre.to_csv('data/train_pre.csv', index = False)
```

인원 수 예측용 Test 데이터

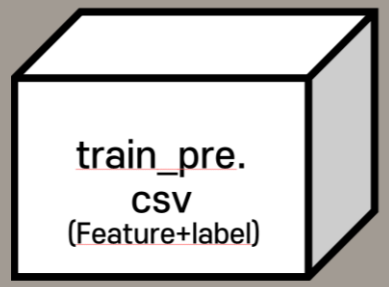
```
test_pre = test[[
    '일자', '년', '월', '일', '요일', '휴일전날',
    '본사정원수', '본사휴가자수', '본사출장자수',
    '본사시간외근무명령서승인건수', '현본사소속재택근무자수',
    '식사가능자수',
    '체감온도(중식)', '불쾌지수(중식)', '우산(중식)',
    '체감온도(석식)', '불쾌지수(석식)', '우산(석식)',
    '코로나신규확진자', '거리두기단계'
]]
```

```
test_pre.to_csv('data/test_pre.csv', index = False)
```

데이콘 제공 한국토지주택공사 구내식당 데이터 참조

프로젝트 수행 절차 및 방법

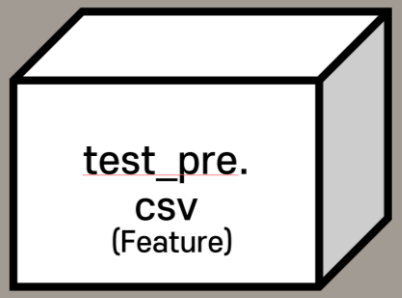
데이터 전처리 (13) 전체4



모델 선택,튜닝,학습용 데이터

✓ 문제(feature)와 정답(label)이 모두 있는 데이터, 모델 선택과 모델 튜닝, 인원 수 예측에 사용됨

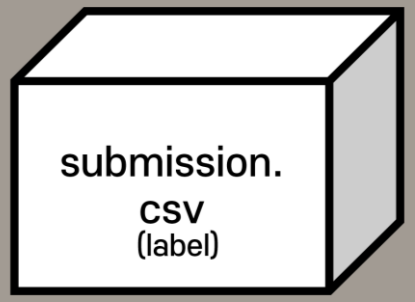
(2016-02-01 ~ 2021-01-26) : 1204개



인원 수 예측용 데이터

✓ 문제(feature)만 있는 데이터, 본격적인 인원 수 예측에 사용됨

(2021-01-27 ~ 2021-04-09) : 49개



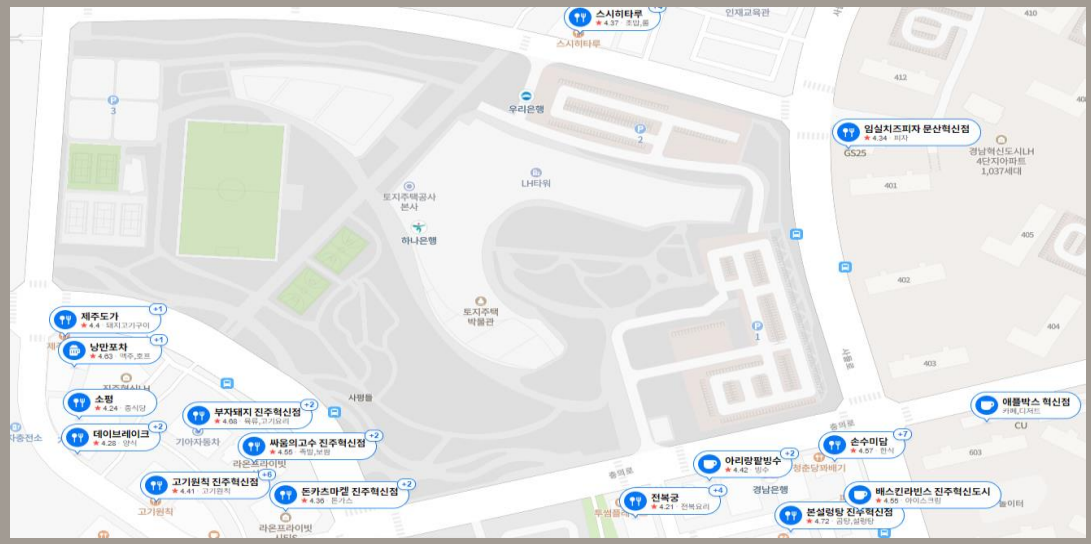
인원 수 예측 값 저장

✓ 예측한 정답(label)을 입력해야하는 데이터프레임

(2021-01-27 ~ 2021-04-09) : 49개

프로젝트 수행 절차 및 방법

데이터 분석 및 시각화 (1)



- ✓ 경상남도 진주시 한국토지주택공사 사옥 주변 식당들을 조사함
- ✓ - 첫 번째 추정 : 상권 활성화 정도에 따라 구내식당 이용률에 영향을 미칠 수 있는가
- ✓ - 두 번째 추정 : 식당 별점에 따라서 구내식당 이용률에 영향을 미칠 수 있는가

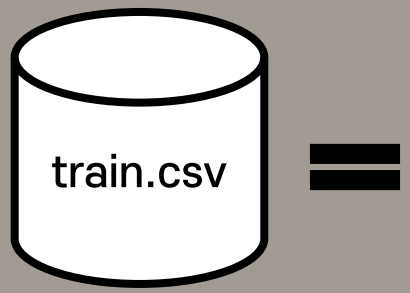
결론

2016년 ~ 2021년 주변 식당가의 중식(11~13시) 석식(17~19시) 카드 결제 건 데이터,

주변 식당의 별점 누적 데이터 등을 수집하고 전처리 하여야 예측에 활용할 수 있으나 일정상 제외함

프로젝트 수행 절차 및 방법

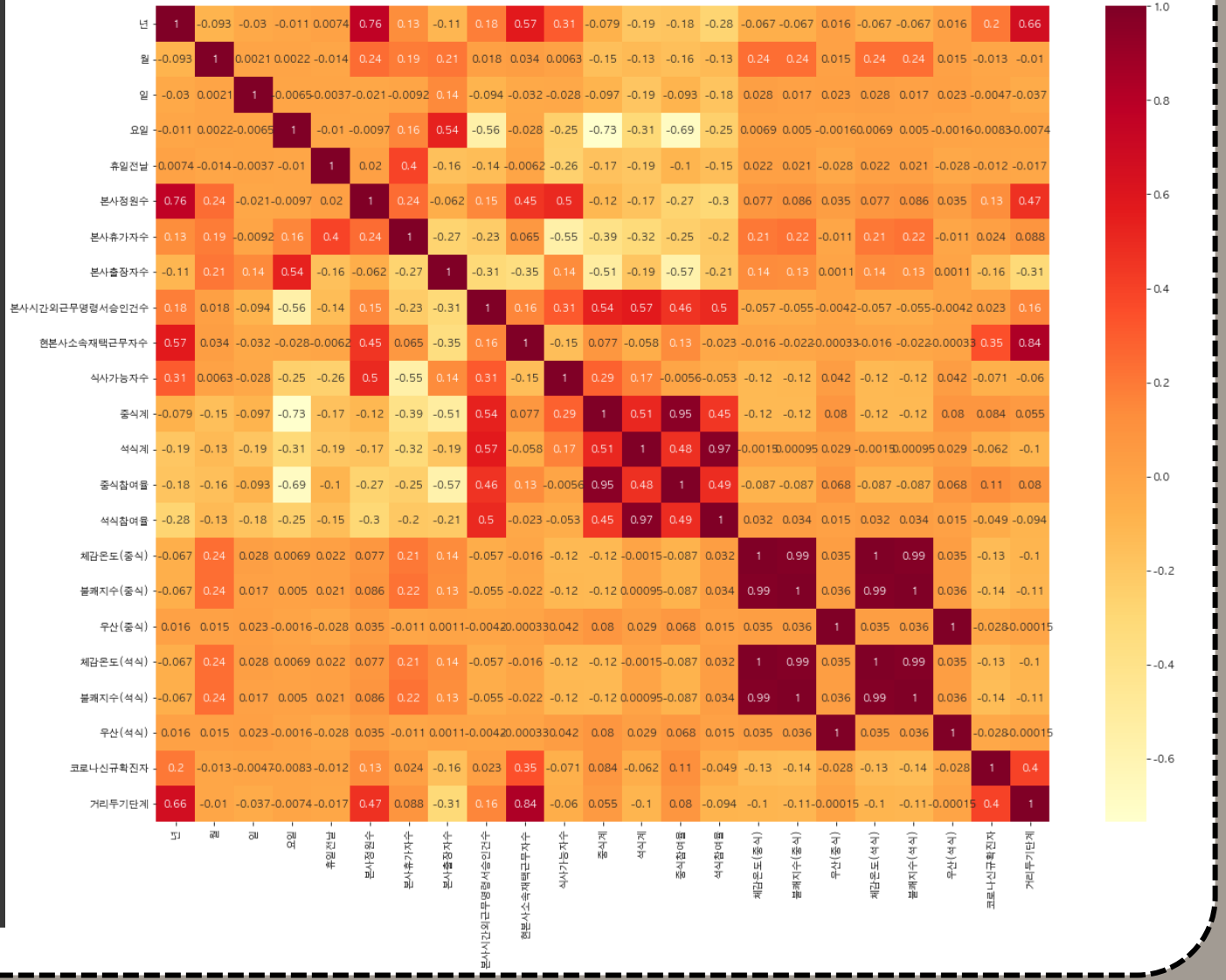
데이터 분석 및 시각화 (2)



Feature +label

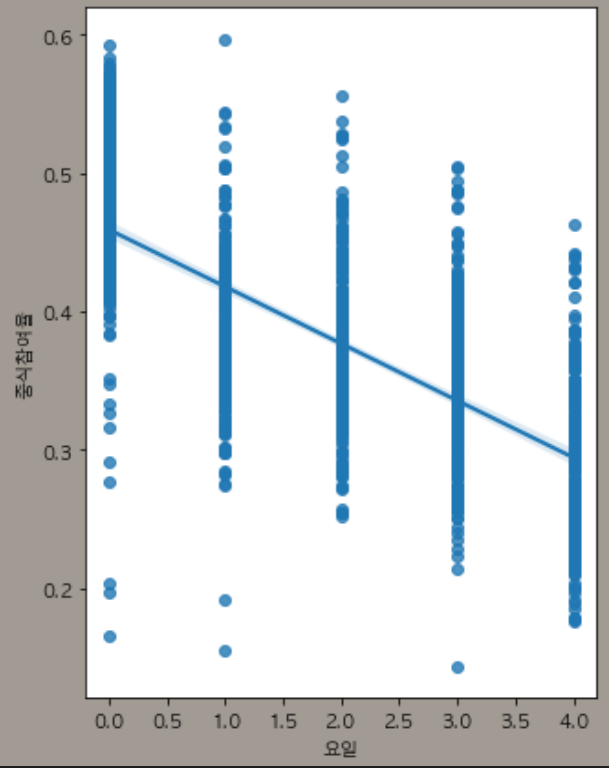
- -1.0 < r < -0.7 : 매우 강한 음의 상관관계
- -0.7 < r < -0.3 : 강한 음의 상관관계
- -0.3 < r < -0.1 : 약한 음의 상관관계
- -0.1 < r < 0.1 : 상관관계 없음
- 0.1 < r < 0.3 : 약한 양의 상관관계
- 0.3 < r < 0.7 : 강한 양의 상관관계
- 0.7 < r < 1.0 : 매우 강한 양의 상관관계

	중식참여율	석식참여율
년	-0.178877	-0.277690
월	-0.158003	-0.130436
일	-0.093265	-0.180888
요일	-0.685481	-0.251989
휴일전날	-0.102500	-0.150425
본사정원수	-0.271385	-0.295678
본사휴가자수	-0.245167	-0.204150
본사출장자수	-0.573241	-0.214768
본사시간외근무명령서승인건수	0.461910	0.498712
현본사소속재택근무자수	0.134819	-0.023177
식사가능자수	-0.005636	-0.052702
중식계	0.954018	0.448772
석식계	0.476374	0.971782
중식참여율	1.000000	0.485553
석식참여율	0.485553	1.000000
체감온도(중식)	-0.087435	0.032322
불쾌지수(중식)	-0.086888	0.033806
우산(중식)	0.068296	0.015070
체감온도(석식)	-0.087435	0.032322
불쾌지수(석식)	-0.086888	0.033806
우산(석식)	0.068296	0.015070
코로나신규확진자	0.113686	-0.048958
거리두기단계	0.079661	-0.093756



프로젝트 수행 절차 및 방법

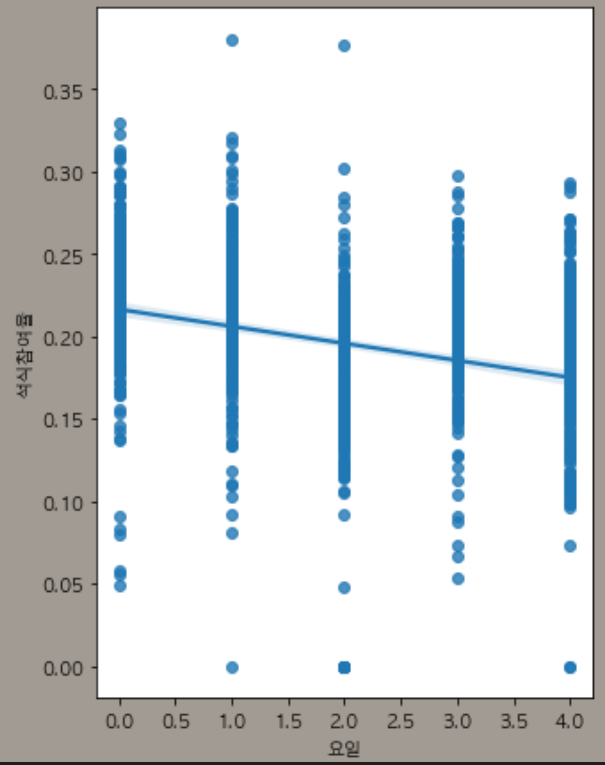
데이터 분석 및 시각화 (3)



['요일'], ['중식참여율']

- -0.685 : 강한 음의 상관관계

```
plt.figure(figsize = (5, 7))
sns.regplot(data = train, x = '요일', y = '중식참여율')
```



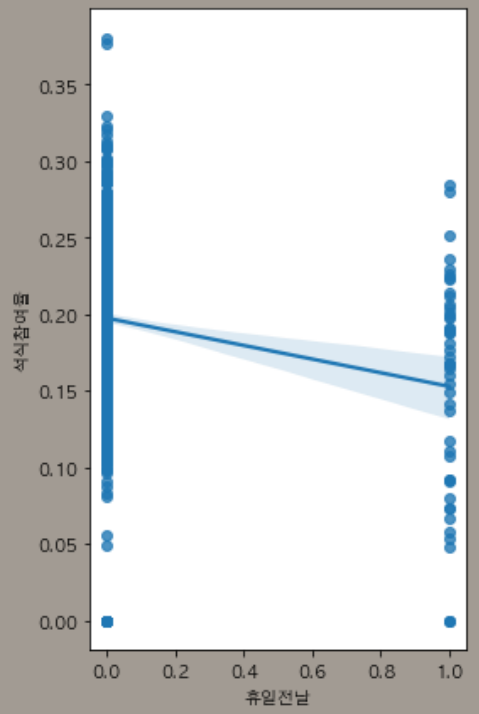
['요일'], ['석식참여율']

- -0.252 : 약한 음의 상관관계

```
plt.figure(figsize = (5, 7))
sns.regplot(data = train, x = '요일', y = '석식참여율')
```

프로젝트 수행 절차 및 방법

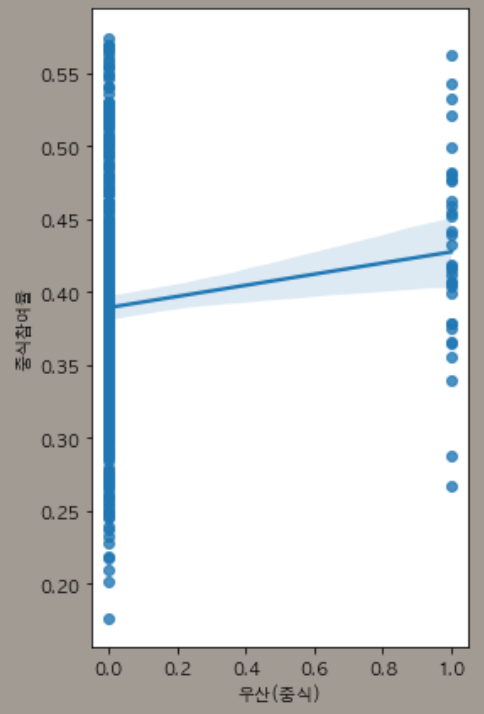
데이터 분석 및 시각화 (4)



['휴일전날'], ['석식참여율']

- -0.150 : 약한 음의 상관관계

```
plt.figure(figsize = (4, 7))
sns.regplot(data = train, x = '휴일전날', y = '석식참여율')
```



갈수기

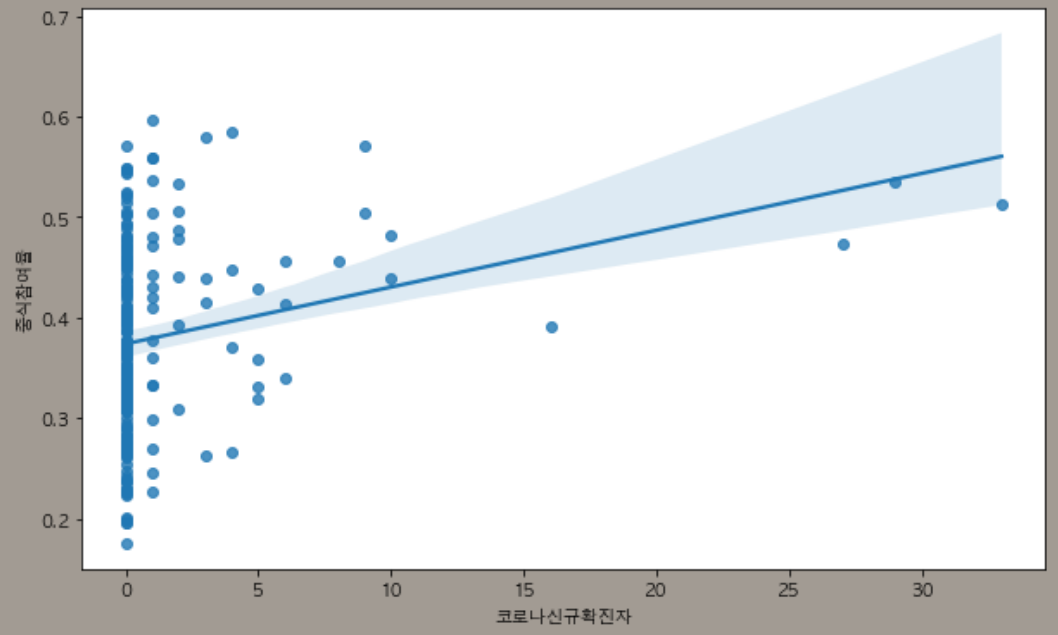
- 갈수기 (2, 3, 4, 5월) 에 비가 온다면?
- ['우산(중식)'] 0.127 : 약한 양의 상관관계
- ['우산(석식)'] 0.019 : 상관관계 없음

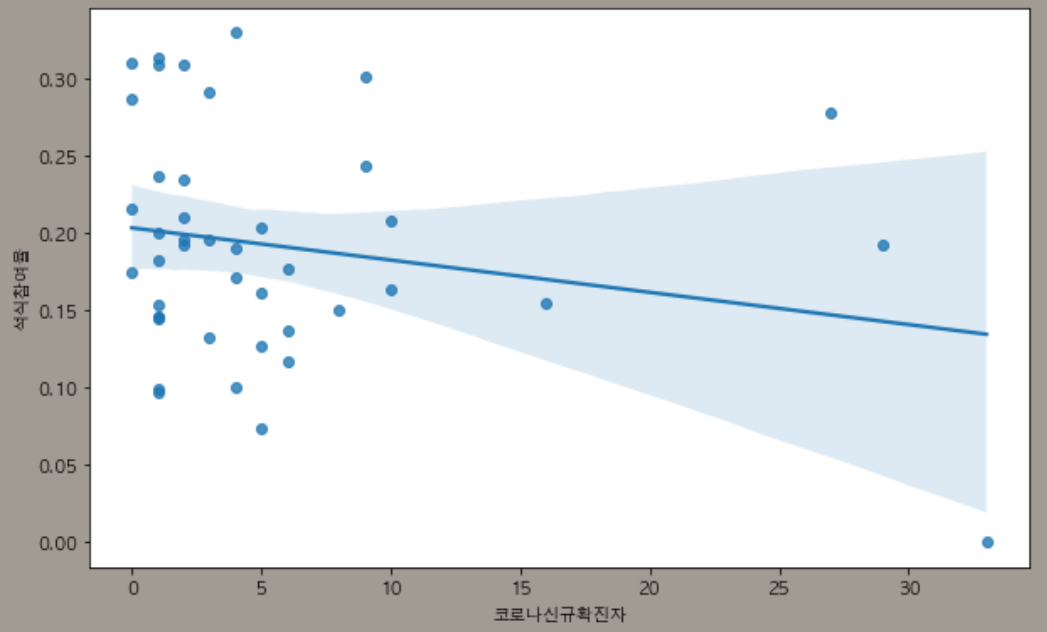
```
thirst_month = [2, 3, 4, 5]
train_thirst = train.query('월 in @thirst_month')
```

```
plt.figure(figsize = (4, 7))
sns.regplot(data = train_thirst, x = '우산(중식)', y = '중식참여율')
```

프로젝트 수행 절차 및 방법

데이터 분석 및 시각화 (5)





2020-02-28 ~ 2021-01-26

- 경상남도 진주시 첫 신규 확진자 발생 일자 2020-02-28
- ['중식참여율'] 0.245 : 약한 양의 상관관계
- ['석식참여율'] -0.056 : 상관관계 없음

```
train_corona1 = train.loc[995:]
```

```
plt.figure(figsize = (10, 6))
sns.regplot(data = train_corona1, x = '코로나신규확진자', y = '중식참여율')
```

2020-11-19 ~ 2021-01-26

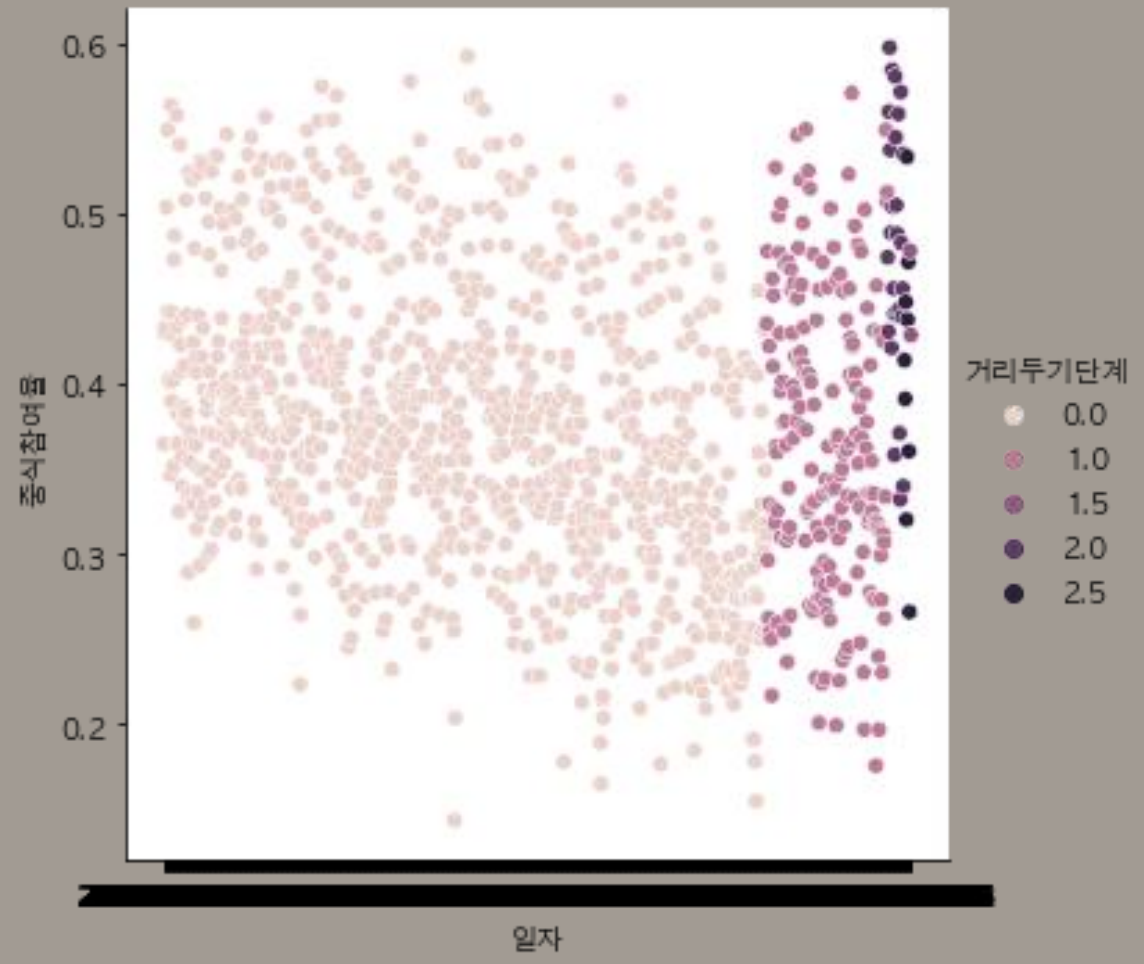
- 경상남도 진주시 코로나 신규 확진자가 급격히 증가하는 일자 2020-11-19
- ['중식참여율'] 0.063 : 상관관계 없음
- ['석식참여율'] -0.213 : 약한 음의 상관관계

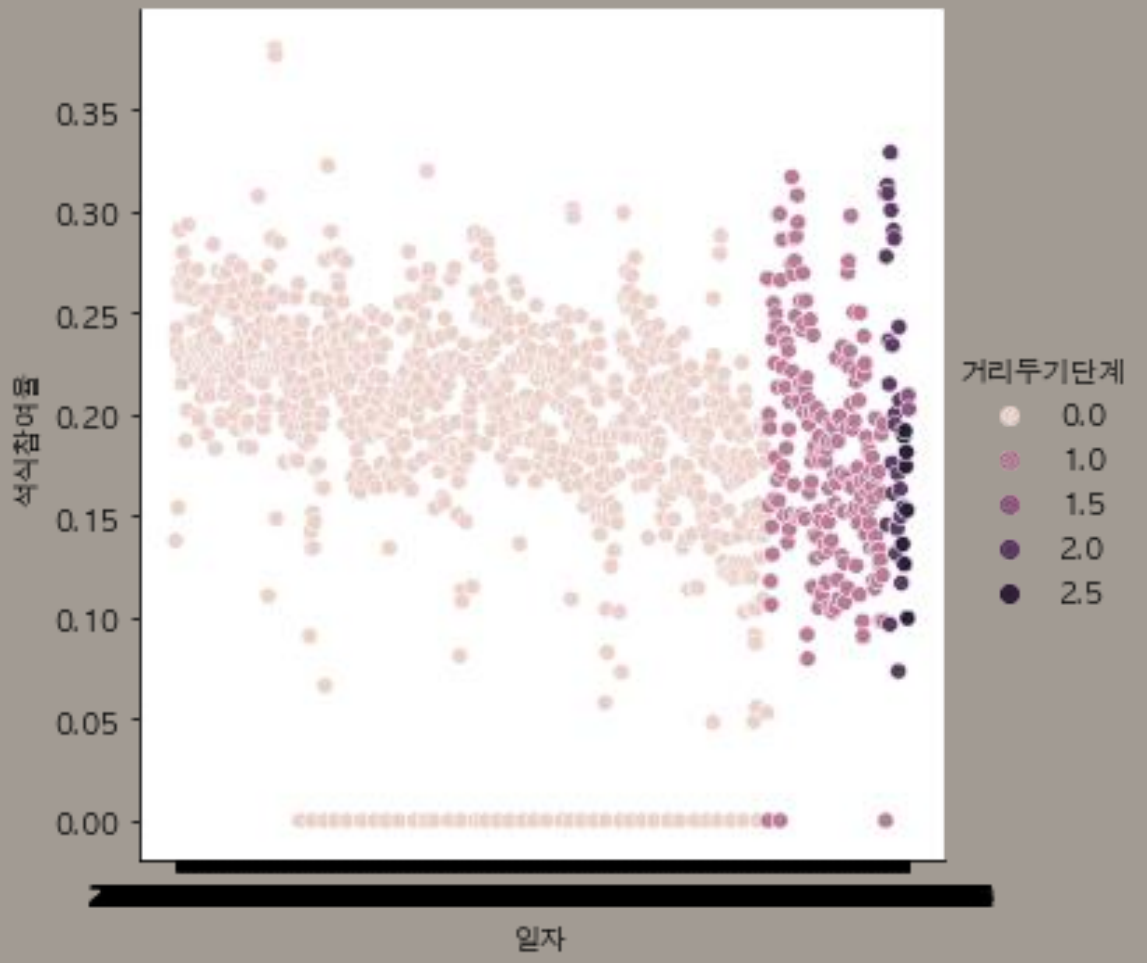
```
train_corona2 = train.loc[1162:]
```

```
plt.figure(figsize = (10, 6))
sns.regplot(data = train_corona2, x = '코로나신규확진자', y = '석식참여율')
```

프로젝트 수행 절차 및 방법

데이터 분석 및 시각화 (6)






```
sns.relplot(data = train, x = '일자', y = '중식참여율', hue = '거리두기단계')
```

```
sns.relplot(data = train, x = '일자', y = '석식참여율', hue = '거리두기단계')
```

프로젝트 수행 절차 및 방법

예측 모델 선택 과정

Feature +label

✓ 4가지 모델을 기본 설정으로 테스트하여

✓ 가장 수치가 좋은 모델을 사용

- R2 : 분산 기반, 1에 가까울수록 예측 정확도 높음
- MSE : (실제값-예측값)² 평균
- RMSE : √(MSE)
- MAE : |실제값-예측값|

모델	Linear Reg	Lasso	Ridge	Random Forest Reg
R2 score	0.58	0.59	0.58	0.69
MSE	11236.88	11185.03	11230.47	8550.35
RMSE	106.00	105.76	105.97	92.49
MAE	78.71	78.47	78.68	66.78

Random Forest Regression 모델 사용

프로젝트 수행 절차 및 방법

예측 모델 튜닝 준비 과정

랜덤포레스트

다수의 결정 트리들을 학습하는
앙상블 기법

장점: 쉽고 좋은 성능

단점: 많은 튜닝이 필요

=

- n_estimators : int, default = 100, 트리 수
- criterion : {'squared_error', 'absolute_error', 'poisson'}, Default = 'squared_error', 분할 품질 측정
- (1) 'squared_error' : 평균 제곱 오차
- (2) 'absolute_error' : 평균 절대값 오차
- (3) 'poisson' : 포아송 이탈도 감소
- max_depth : int, default = None, 트리 최대 깊이
- min_samples_split : int/float, default = 2, 내부 노드 분할 위해 필요 최소 샘플 수
- min_samples_leaf : int/float, default = 1, 리프 노드에 있어야 할 최소 샘플 수
- min_weight_fraction_leaf : float, default = 0.0, 가중치 합계의 최소 가중비
- max_features : {'auto', 'sqrt', 'log2'}, int/float, default = auto, Feature 수
- (1) 'auto', max_features = n_features
- (2) 'sqrt', max_features = sqrt(n_features)
- (3) 'log2', max_features = log2(n_features)
- (4) None, max_features = n_features
- max_leaf_nodes : int, default = None, 리프 노드의 최대 개수

프로젝트 수행 절차 및 방법

예측 모델 튜닝 과정 (1)

Feature +label

Randomized Search CV

```
'n_estimators' : [700, 800, 900],
'max_depth' : [10, 20, 30],
'bootstrap' : [True, False]
```

Grid Search CV

```
'n_estimators' : [700, 800, 900],
'max_depth' : [10, 20, 30],
'bootstrap' : [True, False]
```

- ✓ Bootstrap 적용 여부를 판별하기 위해
- ✓ 임의 탐색, 그리드 탐색 시행

```
{'bootstrap': True, 'max_depth': 20, 'n_estimators': 800}
```

```
R2 Score - Default : 0.69, Set : 0.69
MSE - Default : 8552.27, Set : 8429.55
MAE - Default : 66.85, Set : 66.24
```

프로젝트 수행 절차 및 방법

예측 모델 튜닝 과정 (2)

Feature +label

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import cross_val_score
rfr = RandomForestRegressor(random_state = 2021)
scores = cross_val_score(
    rfr, X_train, y_train, cv = 5, n_jobs = -1,
    scoring = 'neg_mean_squared_error'
)
print('RMSE : {}'.format(np.sqrt(-scores)))
print('RMSE평균 : {}'.format(np.sqrt(-scores.mean())))
```

교차검증

```
RMSE : [94.51223771 85.02233721 94.85365883 94.30588492 91.10598015]
RMSE평균 : 92.03529436609418
```

그리드탐색

```
최적 하이퍼파라미터 :  {'max_features': 11, 'min_samples_leaf': 2, 'n_estimators': 200}
최적 하이퍼파라미터 RMSE : 0.8376
```

임의탐색

```
최적 하이퍼파라미터 :  {'max_features': 7, 'min_samples_leaf': 1, 'n_estimators': 209}
최적 하이퍼파라미터 RMSE : 0.8373
```

프로젝트 수행 절차 및 방법

예측 모델 튜닝 과정 (3)

Feature +label

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
rfr = RandomForestRegressor(
    max_features = 11, min_samples_leaf = 2,
    n_estimators = 200, random_state = 2021
)
rfr.fit(X_train, y_train)
pred_2 = rfr.predict(X_test)
np.sqrt(mean_squared_error(y_test, pred_2))
```

첫번째 모델

- Default

두번째 모델

- 'max_features': 11, 'min_samples_leaf': 2, 'n_estimators': 200

세번째 모델

- 'max_features': 7, 'min_samples_leaf': 1, 'n_estimators': 209

네번째 모델

- 'n_estimators' = 800, 'max_depth' = 20, 'bootstrap' = True

프로젝트 수행 절차 및 방법

예측 모델 튜닝 과정 (4)

Feature +label

```
from sklearn.metrics import r2_score
print('R²값(1) : %.3f' % r2_score(y_test, pred_1)) # 첫 번째 모델
print('R²값(2) : %.3f' % r2_score(y_test, pred_2)) # 두 번째 모델
print('R²값(3) : %.3f' % r2_score(y_test, pred_3)) # 세 번째 모델
print('R²값(4) : %.3f' % r2_score(y_test, pred_4)) # 네 번째 모델
```

ANOVA

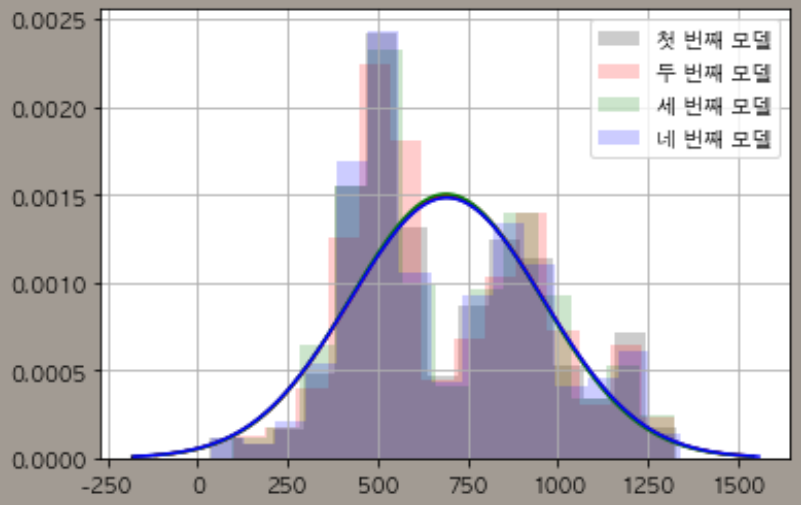
- R^2(결정계수) : RSS / TSS = 1 - SSE / TSS
- TSS : $(y - y_{평균})^2$
- RSS : $(y_{예측} - y_{평균})^2$
- SSE : $(y_{예측} - y)^2$

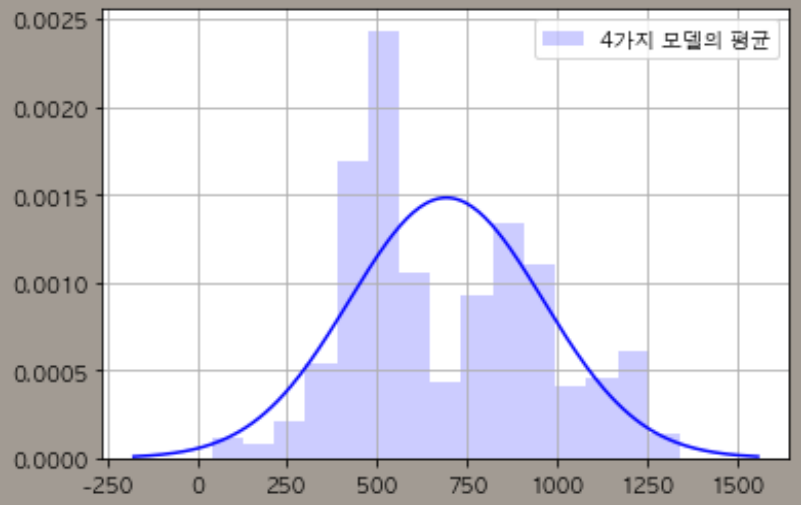
```
R²값(1) : 0.692
R²값(2) : 0.698
R²값(3) : 0.705
R²값(4) : 0.692
```

프로젝트 수행 절차 및 방법

예측 모델 튜닝 과정 (5)

Feature +label



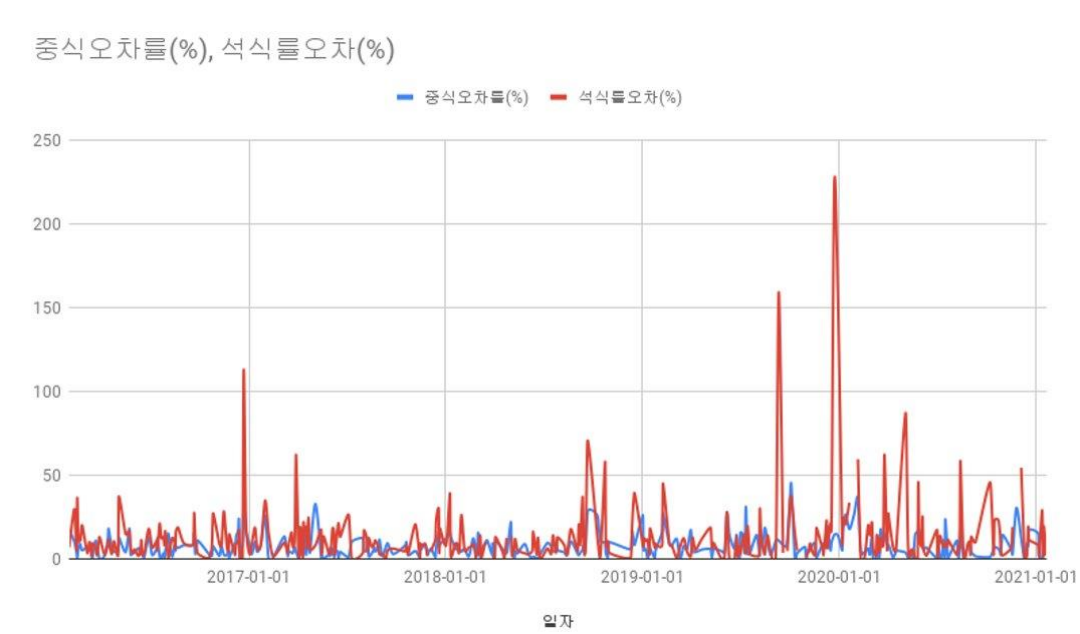


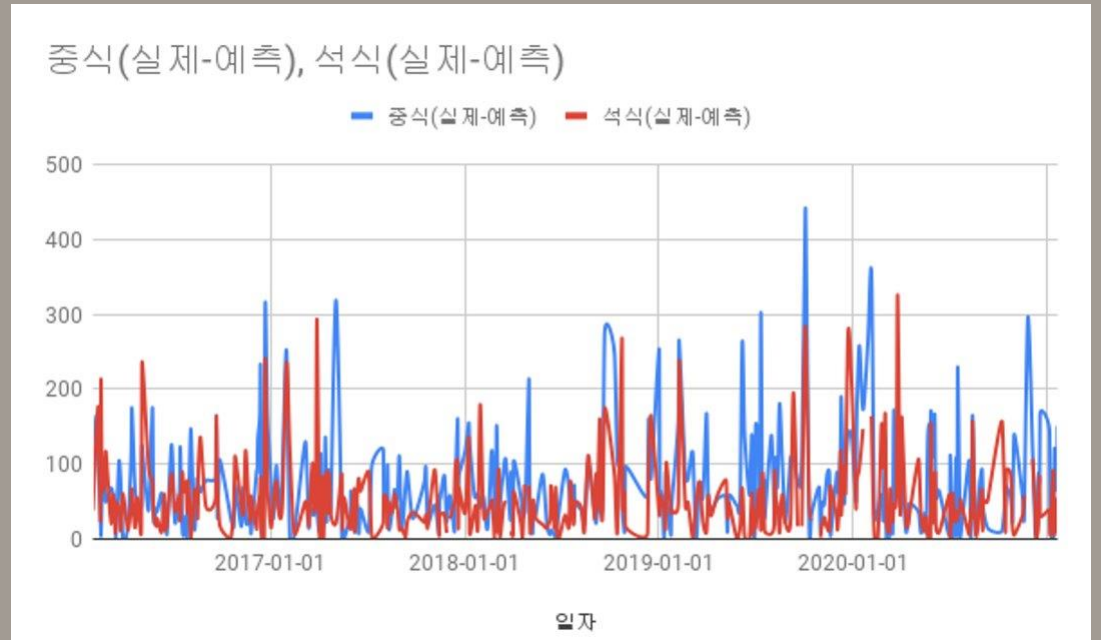

```
sns.distplot(pred_1, kde = False, fit =
sns.distplot(pred_2, kde = False, fit =
sns.distplot(pred_3, kde = False, fit =
sns.distplot(pred_4, kde = False, fit =
plt.legend()
plt.grid()
plt.show()
```

```
mean = (pred_1 + pred_2 + pred_3 + pred_4) / 4
sns.distplot(pred_4, kde = False, fit = stats.norm,
plt.legend()
plt.grid()
plt.show()
```

프로젝트 수행 절차 및 방법

예측 모델 튜닝 과정(5)





'중식 이용 인원' 예측에 비해서 '석식 이용 인원' 예측의 오차가 크게 발생하였음

첫 번째 이유 : '가정의 달', '자기 계발의 날' 등의 이유로 석식 운영하는 날이 없었던 데이터들이 있음

두 번째 이유 : '석식 이용 인원'은 '중식 이용 인원'의 절반 수치로 변동폭이 심한 데이터

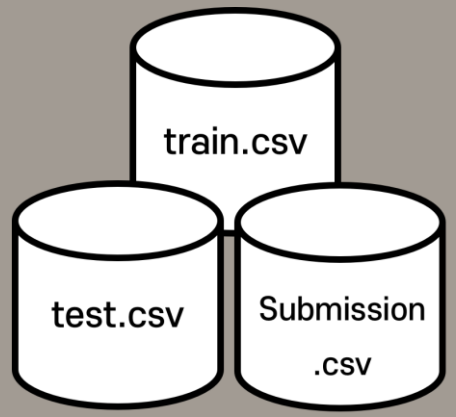
세 번째 이유 : 전체 데이터는 2016년부터 이나 코로나 데이터는 2020년초부터 시작되어 예측에 어려움 발생

네 번째 이유 : Light GBM 등 성능이 좋은 모델도 함께 수행해야했으나

1,000개 안팎의 적은 데이터셋이라 과적합 문제 우려 때문에 사용할 수 없었음

프로젝트 수행 절차 및 방법

예측 결과




```
1 X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
((1205, 19), (50, 19), (1205, 2), (50, 2))
```

첫번째 모델

두번째 모델

- 'max_features': 11, 'min_samples_leaf': 2, 'n_estimators': 200

세번째 모델

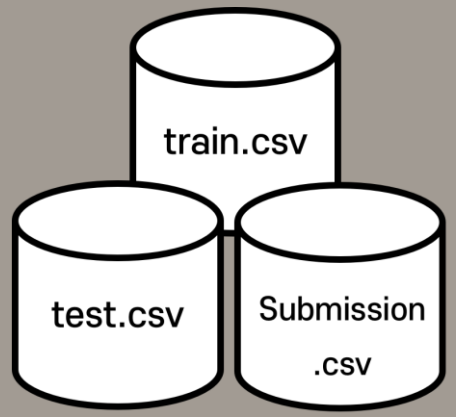
- 'max_features': 7, 'min_samples_leaf': 1, 'n_estimators': 209

네번째 모델

- 'n_estimators' = 800, 'max_depth' = 20, 'bootstrap' = True

프로젝트 수행 절차 및 방법

예측 결과




```
1 pred_lunch.head(1)
```

	중식계1	중식계2	중식계3	중식계4
0	978.06	976.130581	993.444976	1000.4675

```
1 pred_dinner.head(1)
```

	석식계1	석식계2	석식계3	석식계4
0	204.24	208.829968	212.672249	176.37125

```
1 submission.tail(1)
```

	일자	중식계	석식계
49	2021-04-09	588	284

04

프로젝트
수행 결과 및 기대 효과

- ✓ 수행 결과
- ✓ 기대 효과

프로젝트 수행 결과 및 기대 효과

수행 결과

데이터 주어진 기간

2016-02-01 ~ 2021-01-26

- index : 1204개
- feature : 19개

데이터 예측 기간

2021-01-27 ~ 2021-04-09

- index : 49개
- label : 2개

수 행 결 과

총 49일 동안의 중식 이용 인원과 석식 이용 인원을 예측하기 위해서
R2, RMSE 점수가 비슷한 Random Forest Regression의 하이퍼 파라미터가
각기 다른 4가지 모델로 예측하여 4개의 결과값의 평균을 도출함

프로젝트 수행 결과 및 기대 효과

기 대 효 과

- 국가적 차원

1. 음식물 쓰레기를 처리할 때 발생하는 각종 환경 요소들을 감소시킬 수 있다.
2. 음식물 쓰레기 처리비용을 줄여 국고를 아낄 수 있다.

- 기업적 차원

1. 사내 식수인원을 예측해 음식 원재료의 과매입과 배출량을 줄일 수 있다.
2. 잉여 재료와 음식을 위해 소비되는 인력과 시간, 재원을 관리하여 회사 재정 리스크를 줄이는 효과를 기대할 수 있음

- 거시적

1. 비슷한 요인으로 식수 인원이 결정되는 타사 구내식당 데이터에 적용 가능
2. 조금 더 깨끗한 환경을 미래 아이들에게 물려줄 수 있음

Thank you