Last update : 2022. 8. 21

ROTHY 걸음데이터 분석

사람들은 언제, 얼마나 걸을까

Global No.1 Health Data Tech Company

배 경

- 사람은 자유의지에 따라 행동하는 것으로 알려져 있다.
- 하지만 다양한 외부요인에 의해 사람의 행동이 영향을 받는다는 주장도 있다.
- 개인의 행동 기록 중 하나인 걸음 데이터를 분석하여 패턴이 존재하는지 확인한다.
- 걸음에 영향을 주는 인자를 도출할 수 있다면, 시간대 별 Rothy 사용자의 걸음 수를 예측할 수 있다.
- 이 예측 결과를 바탕으로 타겟 마케팅, 서비스 고도화에 직/ 간접적으로 활용할 수 있을 것이다.

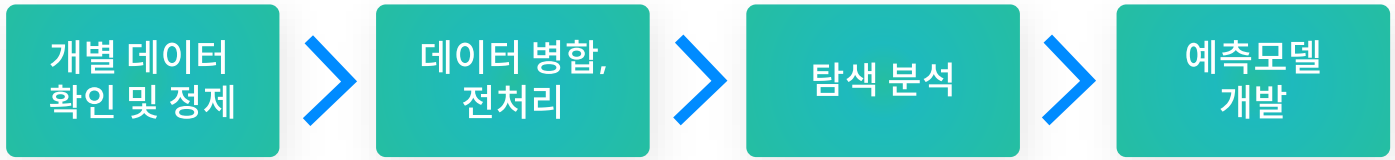
가 설

- 사람의 걸음 수는 시간대와 관련이 깊다.
- 사람의 걸음 수는 계절, 기후에 영향을 받는다.
- 시간대별 걸음 수(패턴)는 연령 대, 경제활동 유무에 영향을 받는다.
- 시간대별 걸음 패턴은 영업일, 비영업일(주말, 공휴일)에 따라 다르다.

데이터 개요

- 분석에 사용한 데이터는 다음과 같다.
  - ROTHY 고객정보 : GB_SVC_USER
  - 일별 걸음 데이터 : GB_BYDT_STEP
  - 상세 걸음 데이터 : GB_STEP_HIST
  - 기후 데이터(기상청) : weather.csv

작업 순서



개별 데이터 확인 및 정제

- 1) ROTHY 고객정보(GB_SVC_USER)

- 8월 15일 기준 가입 상태가 `정상`인 고객만 추출 : 10920명

```
user = user[user['STATUS']=='정상']
```

	USER_ID	NICK_NM	USER_EMAIL	USER_NM	USER_MOBILE	BIRTHDAY	GENDER	TALL	WEIGHT	OS	APP_VER_NO	STATUS
9168	51	H	s@hanmail.net	한	010	19600821.0	M	167.5	60.0	Android	1.6.31	정상
9169	52	돈	21@naver.com	백	010	19850521.0	M	172.0	60.0	Android	1.6.31	정상
9170	53	s	on@gmail.com	문	010	19880817.0	F	164.6	60.0	Android	1.6.29	정상
9171	54	2	3@naver.com	신	010	19721220.0	M	176.0	60.0	Android	1.6.31	정상
9172	55	진짜	4@naver.com	이	010	19850102.0	M	174.0	60.0	Android	1.6.31	정상

- 생년월일(BIRTHDAY) 정보를 바탕으로 `나이(AGE)` 속성 추가

```
user['AGE'] = datetime.datetime.today().year - user['BIRTHDAY'].dt.year
```

BIRTHDAY	GENDER	TALL	WEIGHT	OS	APP_VER_NO	STATUS	AGE
1960-08-21	M	167.5	60.0	Android	1.6.31	정상	62
1985-05-21	M	172.0	60.0	Android	1.6.31	정상	37
1988-08-17	F	164.6	60.0	Android	1.6.29	정상	34
1972-12-	M	176.0	60.0	Android	1.6.31	정상	50

개별 데이터 확인 및 정제

- 1) ROTHY 고객정보(GB_SVC_USER)

- `GROUP` 속성을 추가하여 연령을 카테고리 변수화 한다. (26세 미만: `youth` / 27~57세 : `adult` / 58세 이상 : `elderly`

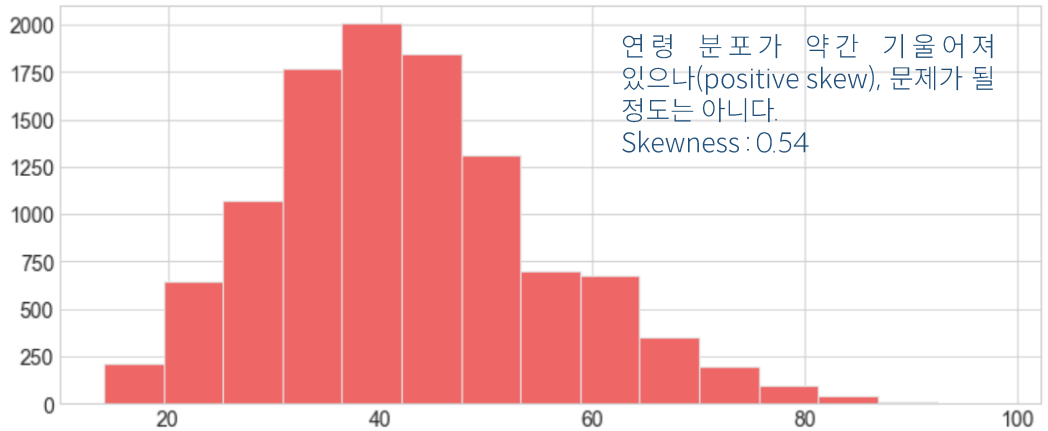
```
user['GROUP'] = user['AGE'].apply(lambda x: 'youth' if x < 26 else ('adult' if x < 58 else 'elderly'))
```

BIRTHDAY	GENDER	TALL	WEIGHT	OS	APP_VER_NO	STATUS	AGE	GROUP
19	M	167.5	60.0	Android	1.6.31	정상	62	elderly
19	M	172.0	60.0	Android	1.6.31	정상	37	adult
19	F	164.6	60.0	Android	1.6.29	정상	34	adult
19	M	176.0	60.0	Android	1.6.31	정상	50	adult
1985-01-	M	174.0	60.0	Android	1.6.31	정상	37	adult

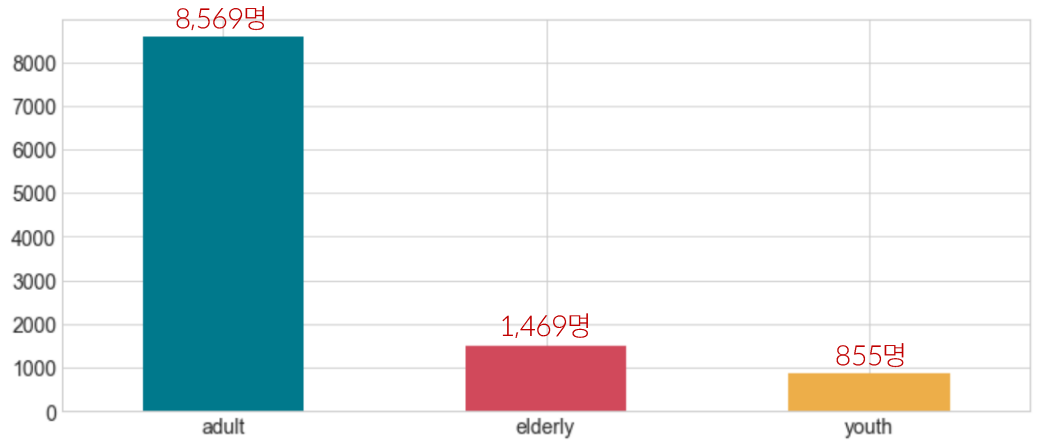
경제활동 유무에 따라 걸음 패턴의 차이가 있는지 확인하기 위함

편의를 위해 청년(youth), 성인(adult), 장년(elderly)로 부르자.

- 중간정리



고객 연령 분포(평균 42.46세)



연령 그룹 별 고객 수

개별 데이터 확인 및 정제

- 2) 일별 걸음 데이터(GB_BYDT_STEP)

  - 별도의 정제 과정 없이 주요 속성을 살펴보았다. (수집기간 : 22-10-26 ~ 22-08-02)

	MSRE_DTM	USER_ID	STEP_CNT	MOVE_DIST	CNPT_CALR	MOVE_SPEED
0	2021-10-26	4	5417.0	4104.298828	204.639999	1.429030
1	2021-10-27	3	110.0	83.983040	4.258073	1.354610
2	2021-10-27	4	5455.0	4152.960938	200.949982	1.453837
3	2021-10-27	5	4918.0	3757.562744	197.159180	1.430885
4	2021-10-27	7	5980.0	4739.403320	302.210968	1.557562

~

	날짜 MSRE_DTM	사용자 ID USER_ID	걸음 수 STEP_CNT	이동거리(m) MOVE_DIST	소모 칼로리 CNPT_CALR	이동속도 MOVE_SPEED
188135	2022-08-02	26	150.0	97.539999	4.440000	0.977520
188136	2022-08-02	31	6704.0	5123.590001	267.949984	1.411110
188137	2022-08-02	32	27.0	19.950000	1.390000	1.107110
188138	2022-08-02	33	59.0	41.600003	2.260000	1.104627
188139	2022-08-02	36	490.0	397.819993	15.490000	2.046290

  - 일별 걸음 수의 총합 VS 활성 고객 수

```
daily_step[['MSRE_DTM', 'STEP_CNT']].set_index('MSRE_DTM').resample('D').sum()['STEP_CNT']\
                                .plot(ax =ax1, color=colors[0], label = 'daily steps', alpha = 0.8)
daily_step[['MSRE_DTM', 'USER_ID']].set_index('MSRE_DTM').resample('D').count()['USER_ID']\
                                .plot(ax =ax2, color=colors[1], label = 'daily users', alpha = 0.8, style = '--')
```

22년 7월까지 걸음수가 완만하게 증가하다가
7월 중순(Rothy 오픈)에 급격하게 증가

걸음 수가 활성 고객 수와 동반 상승
(고객 유입이 걸음 수 증가에 기여)

일단 일별 걸음은 여기까지 확인함
분석, 모델링에는 `상세 걸음 데이터`를 사용할 예정

개별 데이터 확인 및 정제

- 3) 상세 걸음 데이터(GB_STEP_HIST)

  - 상세 걸음은 1분 단위로 집계하고 있다. (수집기간 : 22-01-31 ~ 22-07-20)

  - 랩탑에서 분석하기 위해 각 속성별 값의 범위에 최적화된 변수 타입 지정 -> 메모리 점유 최소화

```
step_hist = reduce_mem_usage(step)
```

  - 데이터를 살펴본다.

	MSRE_BEGIN_DTM	USER_ID	DVIC_TP	STEP_CNT	MOVE_DIST	CNPT_CALR	MOVE_SPEED
0	2022-01-04 19:44:00	7	WATCH	6	4.00	6.00	1.111111
1	2022-01-05 20:44:00	7	WATCH	6	4.00	6.00	1.111111
2	2022-01-31 00:01:00	0	360003	11	8.47	0.39	1.500000
3	2022-01-31 00:01:00	3	360003	1	0.83	0.04	2.138889
4	2022-01-31 00:01:00	4	360001	18	13.59	0.74	1.284435

~

1분 단위로 측정

	측정 날짜, 시간 MSRE_BEGIN_DTM	사용자 ID USER_ID	측정장비 (폰, 워치) DVIC_TP	걸음 수 STEP_CNT	이동거리(m) MOVE_DIST	소모 칼로리 CNPT_CALR	이동속도 MOVE_SPEED
25813406	2022-07-20 09:02:00	13	360003	13	10.027487	0.499278	1.461404
25813407	2022-07-20 09:02:00	74	360001	83	64.490005	4.159998	1.388299
25813408	2022-07-20 09:02:00	19	360001	114	83.220055	3.419997	1.387001
25813409	2022-07-20 09:03:00	86	360003	99	79.705078	3.434418	1.328418
25813410	2022-07-20 09:03:00	74	360001	112	77.320000	5.500002	1.453039

총 25813411개 샘플,
그러니까 25813411분의
걸음 기록이 저장되어 있음

개별 데이터 확인 및 정제

- 3) 상세 걸음 데이터(GB_STEP_HIST)

- 정제 #1 걸음 중복 집계 처리 : 14850479개 샘플이 중복 집계됨(스마트폰, 워치)

```
step_hist[step_hist[['MSRE_BEGIN_DTM', 'USER_ID']].duplicated(keep=False)]
```

	측정 날짜, 시간 MSRE_BEGIN_DTM	사용자 ID USER_ID	측정장비 (폰,워치) DVIC_TP	걸음 수 STEP_CNT	MOVE_DIST	CNPT_CALR	MOVE_SPEED
9	2022-01-31 00:01:00	464	360001	97	70.699997	5.060000	1.178333
10	2022-01-31 00:01:00	464	360003	106	78.483742	5.369317	1.308062
16	2022-01-31 00:01:00	581	360001	89	68.379997	3.560000	1.361111
17	2022-01-31 00:01:00	581	360003	83	65.669998	3.290000	1.388889
21	2022-01-31 00:01:00	691	360001	110	74.589989	5.500001	1.418392
...	...	...	...	...	...	...	...

같은 시간대에, 같은 사용자로 부터 서로 다른 걸음 수가 집계됨

하나는 워치로부터, 하나는 스마트폰으 부터 수집된 것임

- 정제 #1 걸음 중복 집계 처리 : 큰 걸음 수로 측정된 것만 남기고 나머지는 버림

```
step_hist = step_hist.sort_values(by='STEP_CNT', ascending=False).\
                              drop_duplicates(subset=['MSRE_BEGIN_DTM', 'USER_ID'], keep='first').sort_index()
```

	MSRE_BEGIN_DTM	USER_ID	DVIC_TP	STEP_CNT	MOVE_DIST	CNPT_CALR	MOVE_SPEED
9	2022-01-31 00:01:00	464	360001	97	70.699997	5.060000	1.178333
17	2022-01-31 00:01:00	581	360003	83	65.669998	3.290000	1.388889
21	2022-01-31 00:01:00	691	360001	110	74.589989	5.500001	1.418392
27	2022-01-31 00:02:00	360	360001	9	6.390000	0.380000	0.989849
34	2022-01-31 00:02:00	437	360001	7	4.970000	0.270000	0.916667
...	...	...	...	...	...	...	...

샘플 수 : 25,813,411 → 18,387,580 개로 감소

개별 데이터 확인 및 정제

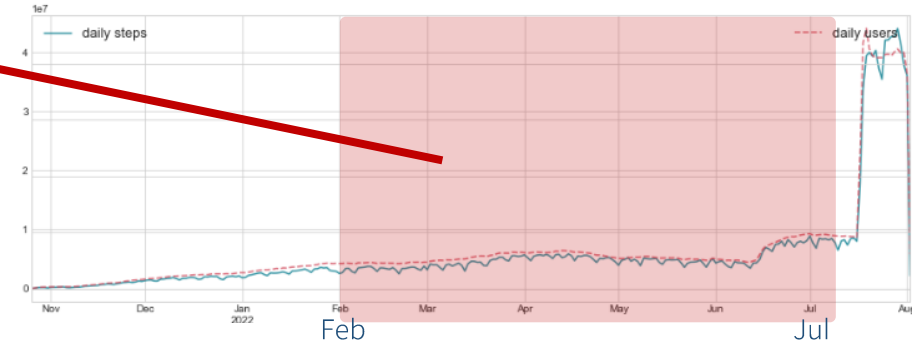
- 3) 상세 걸음 데이터(GB_STEP_HIST)

- 정제 #2 데이터 분석 기간 조정 : 2022-02-01 ~ 07-10 기간의 데이터만 추출

```
step_hist = step_hist[(step_hist['MSRE_BEGIN_DTM']>= pd.to_datetime('2022-02-01'))]
step_hist = step_hist[(step_hist['MSRE_BEGIN_DTM']<= pd.to_datetime('2022-07-10 23:55:00'))]
```

	MSRE_BEGIN_DTM	USER_ID	DVIC_TP	STEP_CNT	MOVE_DIST	CNPT_CALR	MOVE_SPEED
0	2022-02-01 00:00:00	262	360003	11	8.060000	0.530000	1.111111
1	2022-02-01 00:00:00	294	360003	25	17.833750	0.909395	1.176433
2	2022-02-01 0	16078176	2022-07-10 23:54:00	2559	360003	24	20.450001	1.190000	2.583333
3	2022-02-01 0	16078177	2022-07-10 23:54:00	2560	360001	102	69.430000	3.060000	1.157167
4	2022-02-01 0	16078178	2022-07-10 23:54:00	2650	360003	11	8.470000	0.410000	1.444444
		16078179	2022-07-10 23:54:00	2653	360001	14	10.460000	0.580000	1.220215
		16078180	2022-07-10 23:54:00	2658	360003	41	32.230000	1.420000	1.750000

~



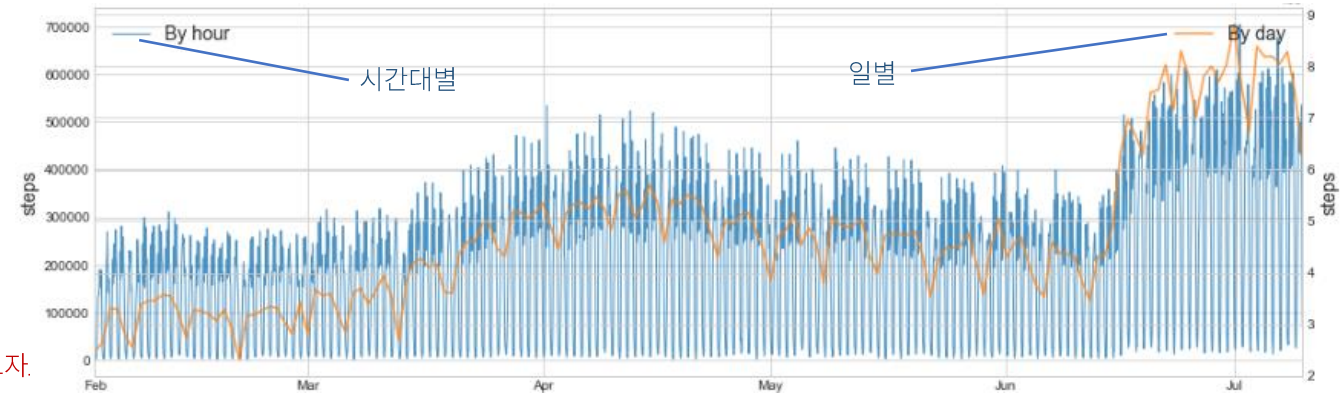
- 정제 #3 데이터 분석 대상자 조정 : 걸음 분석에 의미가 있는 상위 1,500명(active users)의 걸음만 추출

```
active_users = step['USER_ID'].value_counts()[:1500].index.tolist()
step_hist = step_hist[step_hist['USER_ID'].isin(active_users)]
```

개별 데이터 확인 및 정제

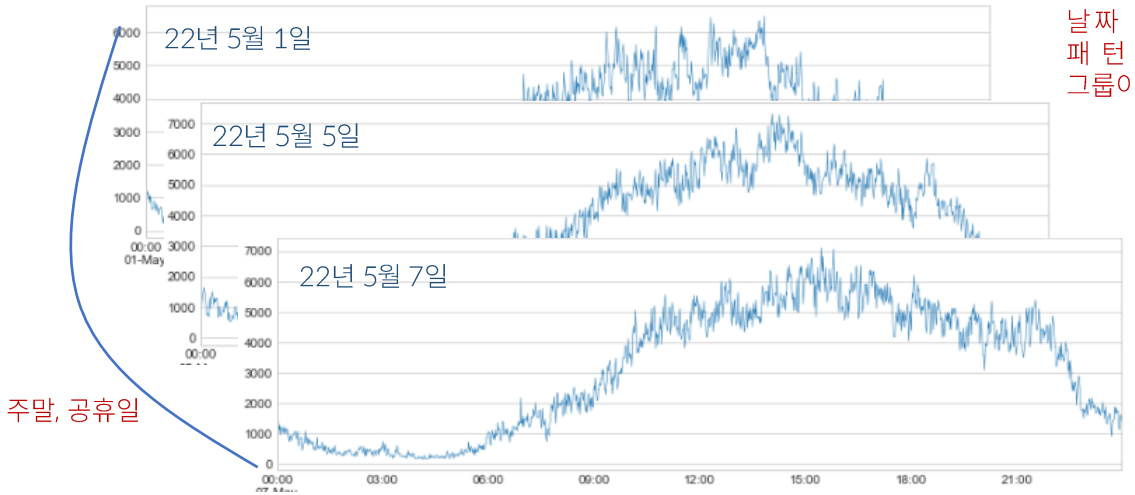
- 3) 상세 걸음 데이터(GB_STEP_HIST)

- 중간 정리 : 시간대별/ 일별 걸음 합계 트렌드 확인



패턴을 확인하기 어렵다.
날짜 별로 차트를 그려서 확인해보자.

- 중간 정리 : 특정 날짜의 트렌드



주말, 공휴일

날짜 별로 유사한 패턴을 보이는 그룹이 있다.

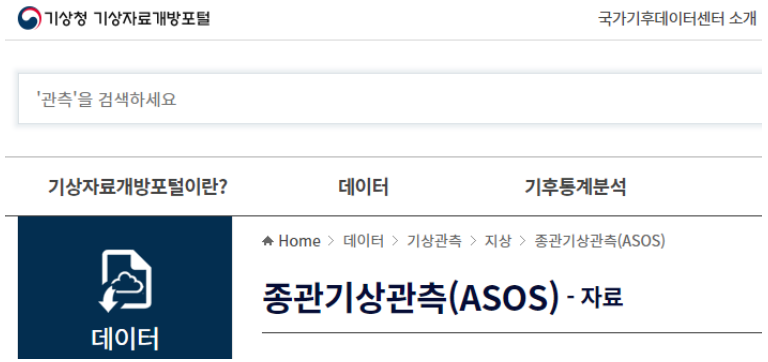
평일
(working day)

일단 여기까지만 확인하고,
상세한 분석은 뒤에서
진행한다.

개별 데이터 확인 및 정제

- 4) 기후 데이터(weather.csv)

- 기상자료개방포털(https://data.kma.go.kr/cmmn/main.do)에서 기후 데이터 추출



Rothy 고객의 거주, 활동 지역을 확인할 수 없어 `서울` 지역의 데이터를 추출

- 데이터를 살펴본다. (수집 기간 : 2022-02-01 ~ 07-10)

	측정 날짜, 시간 datetime	기온 temp	강수량 rain	풍속 wind	풍향 wind_d	습도 humidity	적설량 snow	운량 cloud
0	2022-02-01 0:00	0.4	1.2	1.8	270	92	1.8	9
1	2022-02-01 1:00	-0.2	NaN	2.9	250	85	1.8	2
2	2022-02-01 2:00	-0.9	NaN	2.8	250	74	1.7	7
3	2022-02-01 3:00	-0.9	0.0	3.6	250	72	1.7	9
4	2022-02-01 4:00	-1.7	NaN	3.1	270	87	2.0	10

1시간 단위로 집계

~

결측치

3836	2022-07-10 20:00	30.1	NaN	2.6	270	64	NaN	10
3837	2022-07-10 21:00	29.2	NaN	2.6	250	64	NaN	10
3838	2022-07-10 22:00	28.5	0.0	2.2	270	59	NaN	10
3839	2022-07-10 23:00	28.2	NaN	2.6	250	55	NaN	10
3840	2022-07-11 0:00	27.9	NaN	3.5	270	54	NaN	8

수집 기간
22-02-01
~
22-07-11

개별 데이터 확인 및 정제

- 4) 기후 데이터(weather.csv)

- 정제 #1 결측치 처리 : 집계가 되지 않은 부분을 0으로 대체

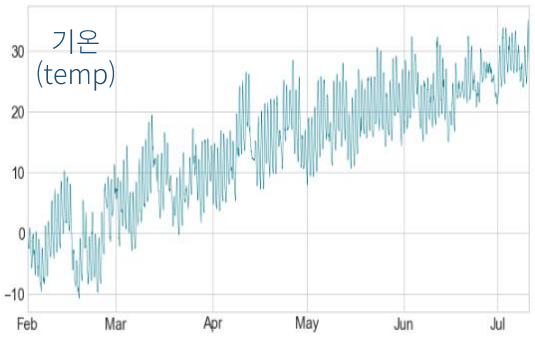
```
weather = weather.fillna(0)
```

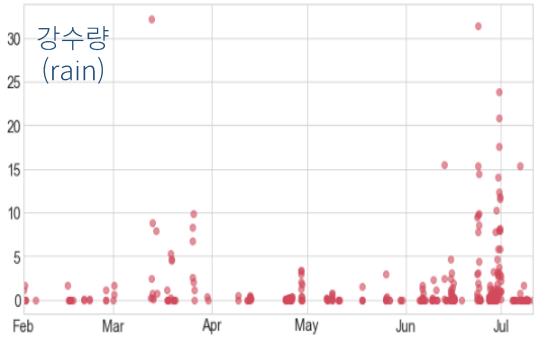
	datetime	temp	rain	wind	wind_d	humidity	snow	cloud
0	2022-02-01 00:00:00	0.4	1.2	1.8	270	92	1.8	9
1	2022-02-01 01:00:00	-0.2	NaN	2.9	250	85	1.8	2
2	2022-02-01 02:00:00	-0.9	NaN	2.8	250	74	1.7	7
3	2022-02-01 03:00:00	-0.9	0.0	3.6	250	72	1.7	9
4	2022-02-01 04:00:00	-1.7	NaN	3.1	270	87	2.0	10

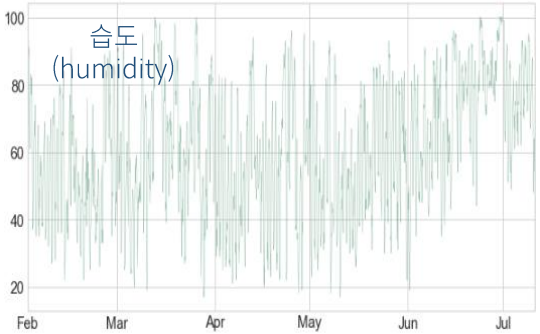
→

	datetime	temp	rain	wind	wind_d	humidity	snow	cloud
0	2022-02-01 00:00:00	0.4	1.2	1.8	270	92	1.8	9
1	2022-02-01 01:00:00	-0.2	0.0	2.9	250	85	1.8	2
2	2022-02-01 02:00:00	-0.9	0.0	2.8	250	74	1.7	7
3	2022-02-01 03:00:00	-0.9	0.0	3.6	250	72	1.7	9
4	2022-02-01 04:00:00	-1.7	0.0	3.1	270	87	2.0	10

- 중간 정리 : 트렌드 확인







일단 여기까지만
확인한다.

데이터 병합 및 전처리

- 데이터 병합

- 병합 #1 상세 걸음 데이터(GB_STEP_HIST)와 ROTHY 고객정보(GB_SVC_USER) 병합

```
step_hist = step_hist.merge(user, on='USER_ID', how='left')
```

상세 걸음 데이터

고객정보

MSRE_BEGIN_DTM	USER_ID	DVIC_TP	STEP_CNT	MOVE_DIST	CNPT_CALR	MOVE_SPEED	GENDER	AGE	GROUP
2022-02-01 00:04:00	559	360003	43	33.349998	1.49	1.611111	F	28.0	adult
2022-02-01 00:04:00	658	360003	12	7.660000	0.47	0.833333	F	35.0	adult
2022-02-01 00:04:00	674	360001	18	13.980000	0.76	1.388889	M	46.0	adult
2022-02-01 00:04:00	712	360003	12	8.290000	0.37	0.972222	F	48.0	adult

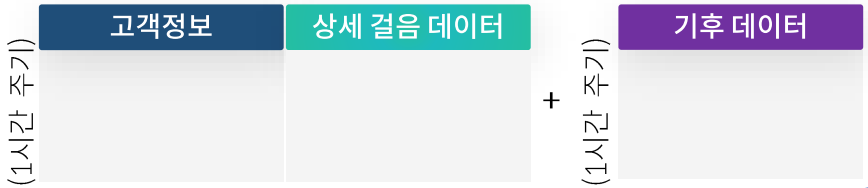
- 병합 #2 데이터를 1시간 주기로 리샘플링(Resampling) 후 기후 데이터와 병합한다.

1시간 동안 수집된 걸음의 총합(sum)

```
step_total[['timestamp', 'STEP_CNT']] = step_hist.set_index('MSRE_BEGIN_DTM').resample('H').sum()['STEP_CNT'].reset_index()
step_total['users'] = step_hist.set_index('MSRE_BEGIN_DTM').resample('H').count()['USER_ID'].values
step_total = step_total.merge(weather, left_on='timestamp', right_on='datetime', how = 'left').drop('datetime', axis = 1)
```

`users` 속성을 추가하여 각 시간대에 걸음 수집에 관여한 사용자 (active_users)수를 입력

Resampling



해당 시간대에 수집된 걸음 수

해당 시간대에 걸음 수집에 관여한 사용자 (active_users)수

1시간 주기

timestamp	users	STEP_CNT	temp	rain	wind	wind_d	humidity	snow	cloud
2022-02-01 00:00:00	528	13107	0.4	1.2	1.8	270	92	1.8	9
2022-02-01 01:00:00	322	9079	-0.2	0.0	2.9	250	85	1.8	2
2022-02-01 02:00:00	170	3294	-0.9	0.0	2.8	250	74	1.7	7
2022-02-01 03:00:00	128	2705	-0.9	0.0	3.6	250	72	1.7	9
2022-02-01 04:00:00	78	1146	1.7	0.0	3.1	270	87	2.0	10

데이터 병합 및 전처리

- 속성(feature) 추가하기 : 데이터셋으로부터 피처를 추가 생성한다.

- 추가 #1 날짜(Calendar) 속성 : 연도, 월, 일, 시간, 요일, 공휴일

```
step_total['year'] = step_total['timestamp'].dt.year
step_total['month'] = step_total['timestamp'].dt.month
step_total['day'] = step_total['timestamp'].dt.day
step_total['hour'] = step_total['timestamp'].dt.hour
step_total['dayofweek'] = step_total['timestamp'].dt.dayofweek #월0, 화1 ...일6

holiday = [datetime.date(2022, 2, 1), datetime.date(2022, 2, 2),  datetime.date(2022, 3, 1), datetime.date(2022, 3, 9),
                        datetime.date(2022, 5, 5), datetime.date(2022, 6, 1), datetime.date(2022, 6, 9)]
step_total[‘holiday’] = 0
Step_total.loc[step_youth['timestamp'].dt.date.isin(holiday), 'holiday'] = 1
```

- 추가 #2 계절(Season) 속성 : 봄, 여름, 가을, 겨울

```
step_total['season'] = step_total['month'].apply(season)
```

```
def season(month):
    if month in [3, 4, 5]: # 봄 0
        return 0
    elif month in [6, 7, 8]: # 여름 1
        return 1
    elif month in [9, 10, 11]: # 가을 2
        return 2
    elif month in [12, 1, 2]: # 겨울 3
        return 3
```

추가 생성된 피처

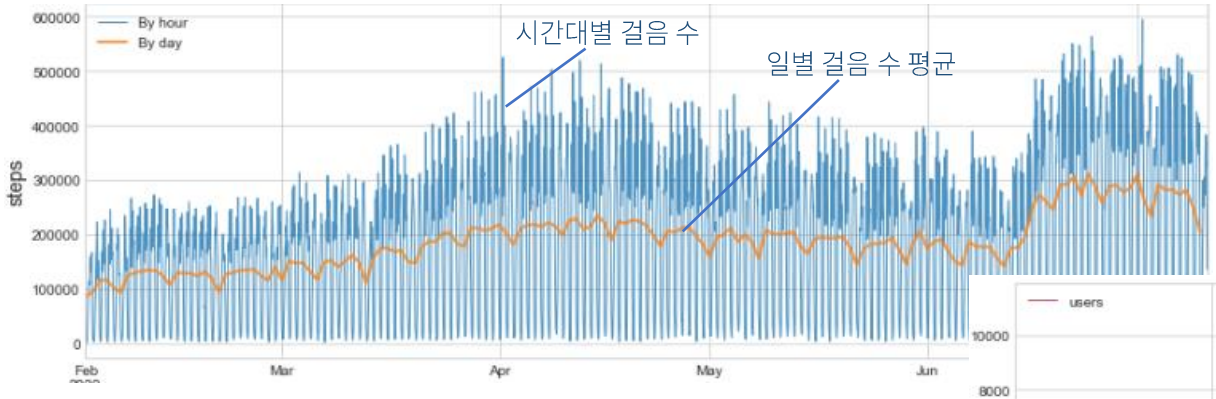
timestamp	STEP_CNT	users	year	month	day	hour	dayofweek	season	holiday	temp	rain	wind	wind_d	humidity	snow	cloud
2022-02-01 00:00:00	13107	528	2022	2	1	0	1	4	1	0.4	1.2	1.8	270	92	1.8	9
2022-02-01 01:00:00	9079	322	2022	2	1	1	1	4	1	-0.2	0.0	2.9	250	85	1.8	2
2022-02-01 02:00:00	3294	170	2022	2	1	2	1	4	1	-0.9	0.0	2.8	250	74	1.7	7
2022-02-01 03:00:00	2705	128	2022	2	1	3	1	4	1	-0.9	0.0	3.6	250	72	1.7	9
2022-02-01 04:00:00	1146	78	2022	2	1	4	1	4	1	-1.7	0.0	3.1	270	87	2.0	10

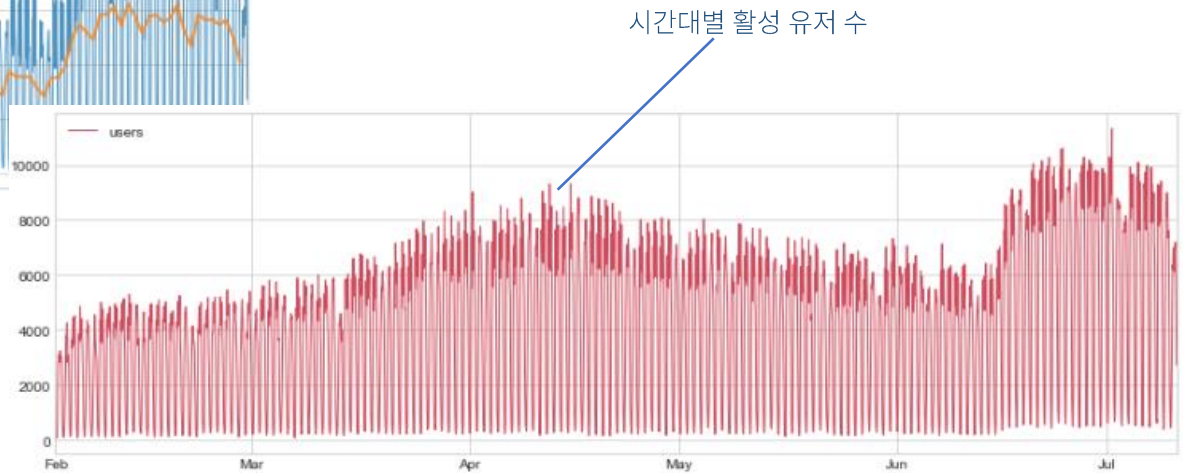
탐색 분석

- 타겟 변수(STEP_CNT, 걸음수)

- #1 트렌드 그려보기 : 시간대별 걸음 수, 일별 걸음 수. 그리고 활성 유저 수(users)

```
fig, axes = plt.subplots(2, 1, figsize = (14, 10))
step_total.set_index('timestamp')['STEP_CNT'].plot(ax=axes[0], label = 'By hour', color='tab:blue',
                                                   alpha = 0.8, linewidth=1).set_ylabel('steps', fontsize= 14)
step_total.set_index('timestamp').resample('D').mean()['STEP_CNT'].plot(ax=axes[0], label = 'By day', color='tab:orange',
                                                 alpha = 0.8, linewidth=2).set_ylabel('steps', fontsize= 14)
step_total.set_index('timestamp')['users'].plot(ax = axes[1], color=colors[1], linewidth=1)
axes[0].legend()
axes[1].legend()
```





- 시간대의 걸음 수의 변동성이 크다.
- 걸음 수의 평균은 날짜에 영향을 받는 것 같으나 현재까지 알 수 없다.
- 시간대별 활성 유저 수와 걸음 수는 상관성이 커 보인다.(차트 형태 유사)

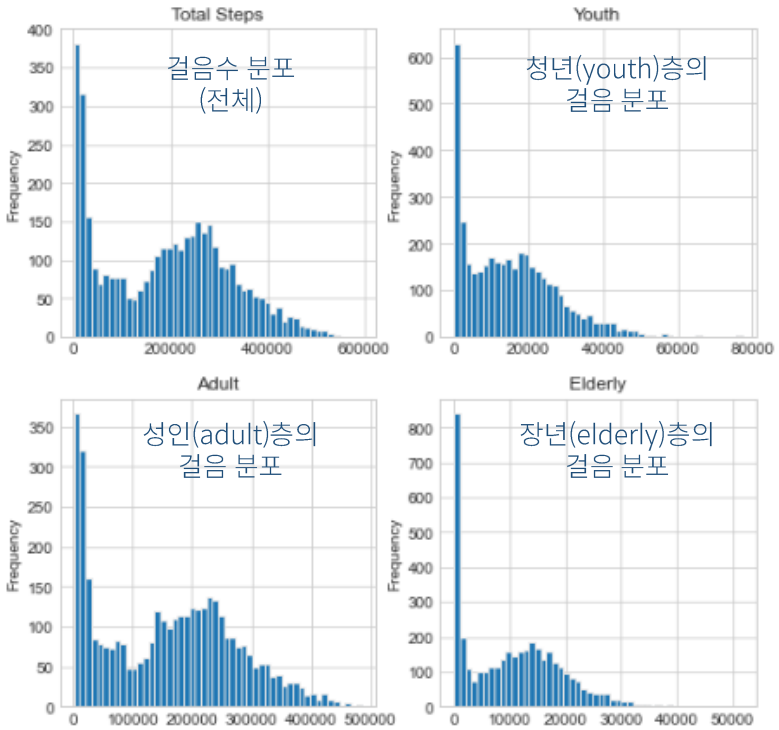
본 차트만으로 패턴을 명확하게 확인하기 어렵다.

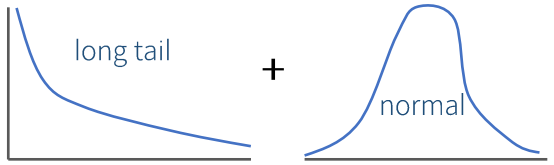
탐색 분석

- 타겟 변수(STEP_CNT, 걸음수)

- #2 분포 확인하기

```
fig, axes = plt.subplots(2, 2, figsize = (8, 8))
step_total['STEP_CNT'].plot(kind ='hist', bins = 50, edgecolor='#E6E6E6', ax = axes[0, 0])
step_youth['STEP_CNT'].plot(kind ='hist', bins = 50, edgecolor='#E6E6E6', ax = axes[0, 1])
step_adult['STEP_CNT'].plot(kind ='hist', bins = 50, edgecolor='#E6E6E6', ax = axes[1, 0])
step_eldery['STEP_CNT'].plot(kind ='hist', bins = 50, edgecolor='#E6E6E6', ax = axes[1, 1])
```

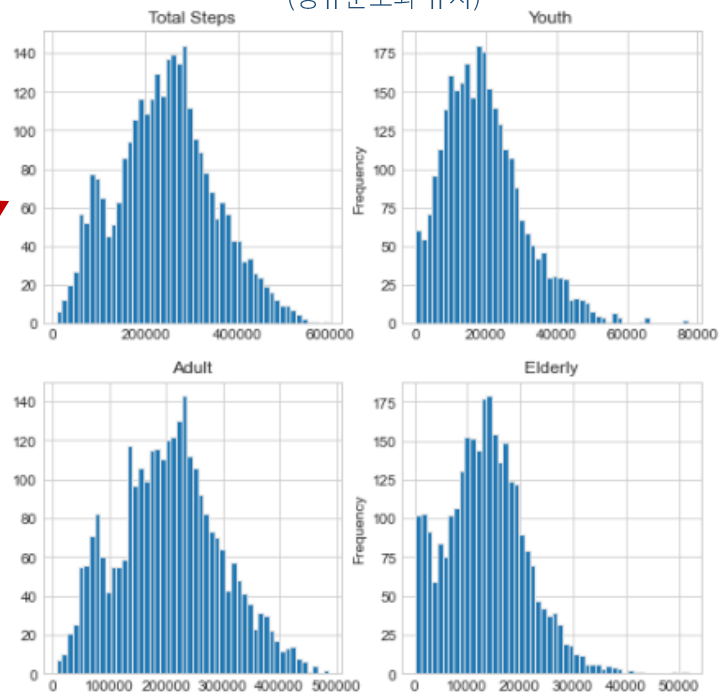




- 긴 꼬리(long tail) 분포와 정규분포(normal)의 혼합된 형태이다.
- 사용자의 걸음 수가 심야(long tail), 주간(normal)에 다른 패턴을 띄기 때문이다.
- 긴꼬리는 로그(log) 변환을 통해 정규화 가능하나 정규 분포가 혼합되어 분리 작업이 필요하다.

일단 여기까지만 알고 넘어간다.

주간 시간대의 걸음 분포
(정규분포와 유사)

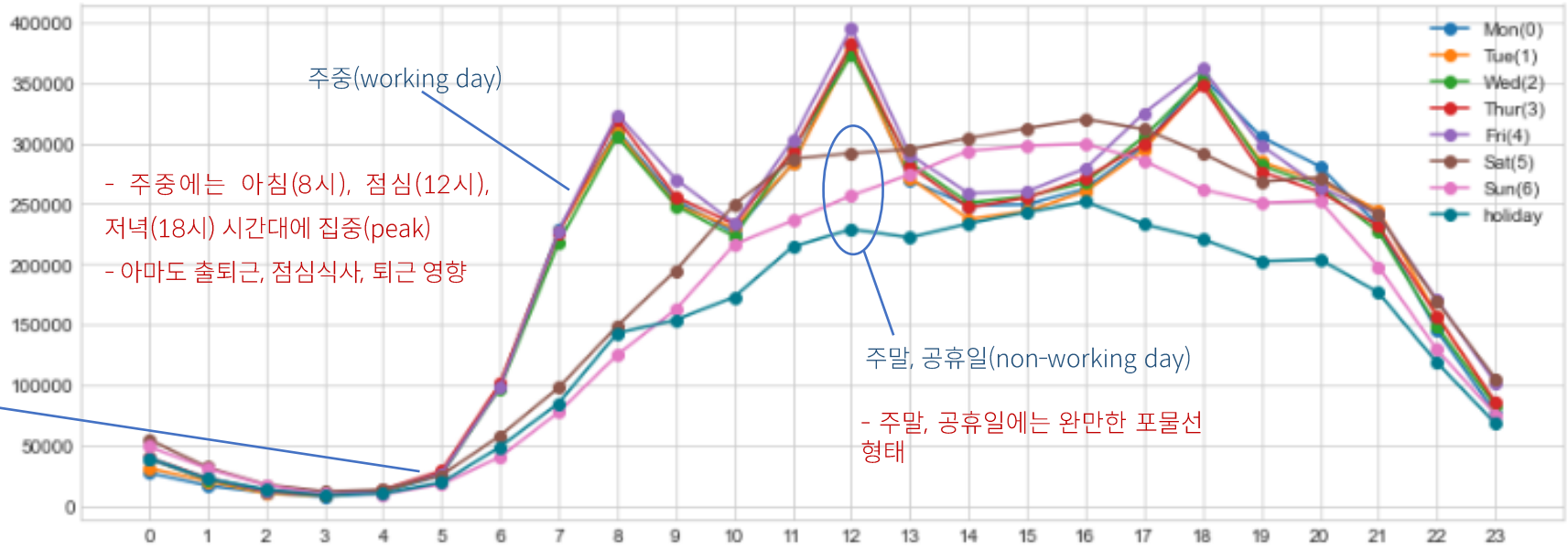


탐색 분석

- 상관도 분석 : 카테고리 변수

- 각 요일별 걸음 수가 어떤 패턴을 보이는지 확인한다.

```
fig, ax = plt.subplots(figsize = (14, 5))
by_weekday_hr = step_total[['hour', 'dayofweek', 'STEP_CNT']].groupby(['hour', 'dayofweek'])
by_weekday_hr.mean()['STEP_CNT'].unstack().plot(ax = ax, marker='o')
df_total[df_total['holiday']==1].groupby('hour').mean()['STEP_CNT'].plot(ax = ax, marker='o', color = colors)
ax.set_xticks([i for i in range(0, 24)])
ax.legend(['Mon(0)', 'Tue(1)', 'Wed(2)', 'Thur(3)', 'Fri(4)', 'Sat(5)', 'Sun(6)', 'holiday'])
```

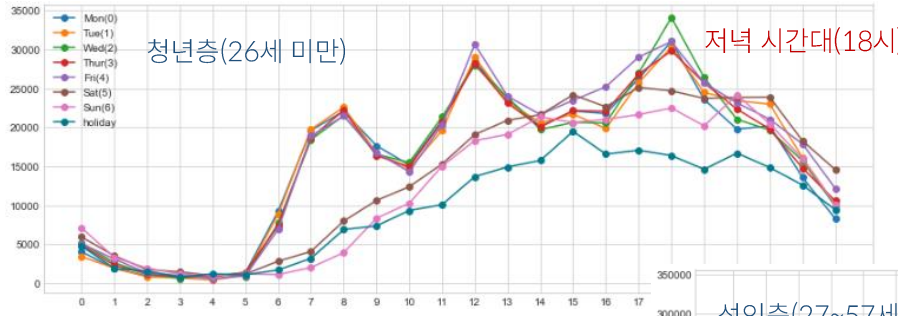


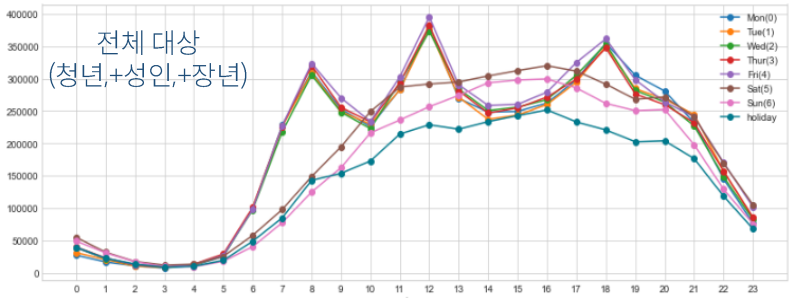
* insight
- 고객의 걷기 행위는 특정 시간대에 집중되는 경향이 있다.
- 이 시간 대의 걸음을 효율적으로 활용하도록 가이드 하면 어떨까.
- 예를 들자면 AI/ 걸음 가이드 서비스는 걷기가 집중되는 시간 전에 refresh 되어야 많은 사용자가 참고할 것이다.
(ex 오전 5시전)

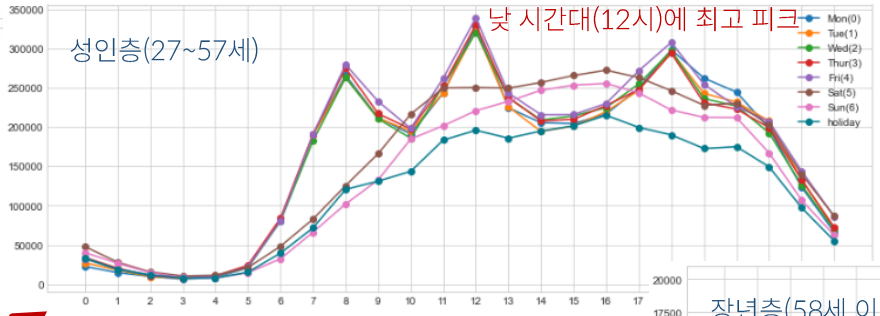
탐색 분석

- 상관도 분석 : 카테고리 변수

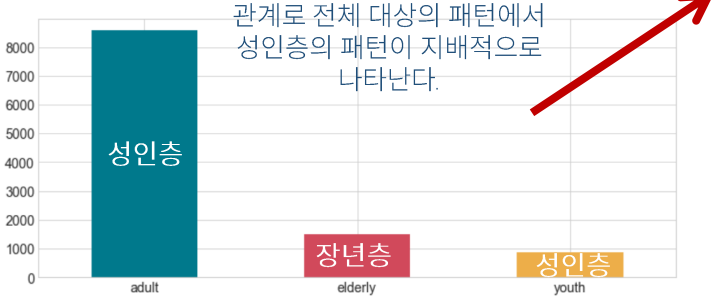
  - 연령대별 걸음 수 패턴의 차이가 있는지 확인한다.

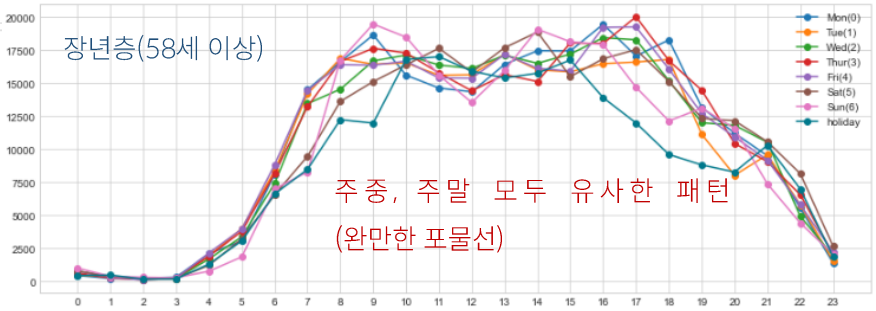






* insight
- 고객의 걸음 패턴은 연령대, 경제활동 유무에 따라 상이할 것으로 추정된다.
- 이를 활용하여 마케팅, 서비스에 반영하면 효과적..



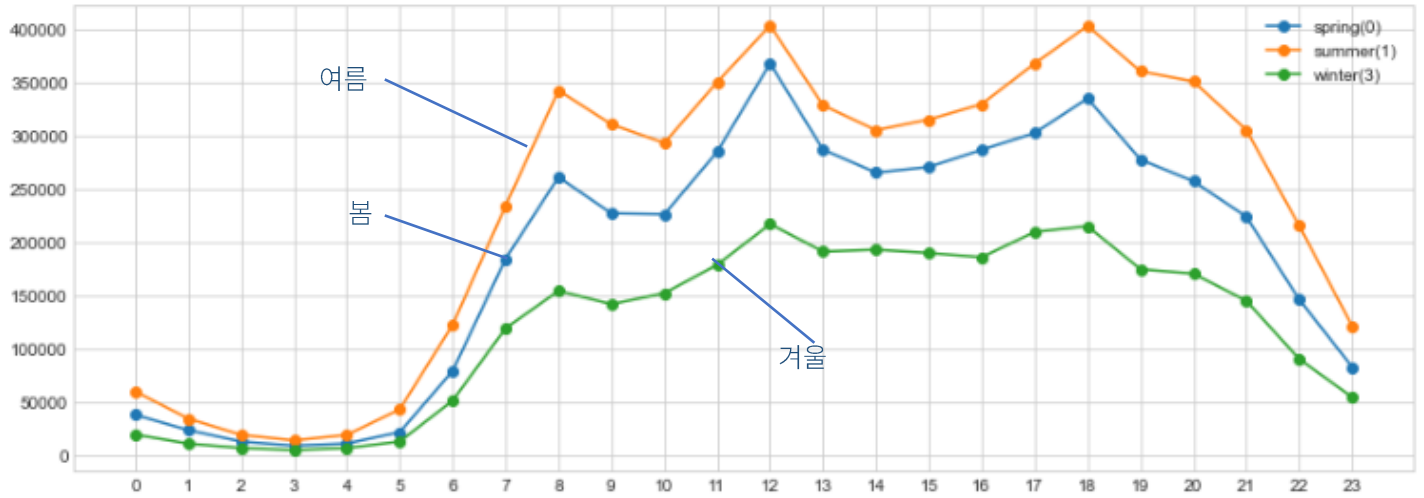


탐색 분석

- 상관도 분석 : 카테고리 변수

- 계절별 걸음 수 패턴의 차이가 있는지 확인한다.

```
fig, ax = plt.subplots(figsize = (14, 5))
by_weekday_hr = step_total[['hour', 'season','STEP_CNT']].groupby(['hour', 'season'])
by_weekday_hr.mean()['STEP_CNT'].unstack().plot(ax = ax, marker='o')
ax.set_xticks([i for i in range(0, 24)])
ax.legend(['spring(1)', 'summer(2)', 'winter(4)'])
```



- 여름, 봄, 겨울 순서로 걸음 수가 많다.
- 그러나 계절적 특성을 확인하기에는 샘플 수가 부족하다.(최소 2년치 이상 필요)

탐색 분석

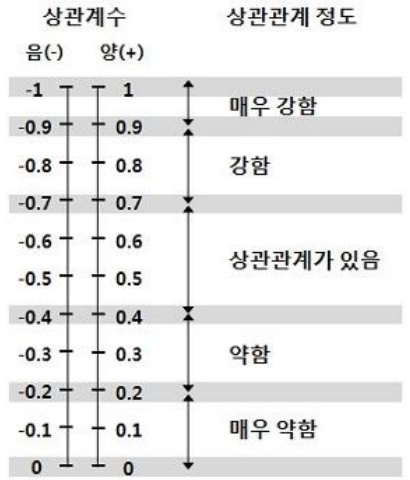
- 상관도 분석 : 실수형 변수

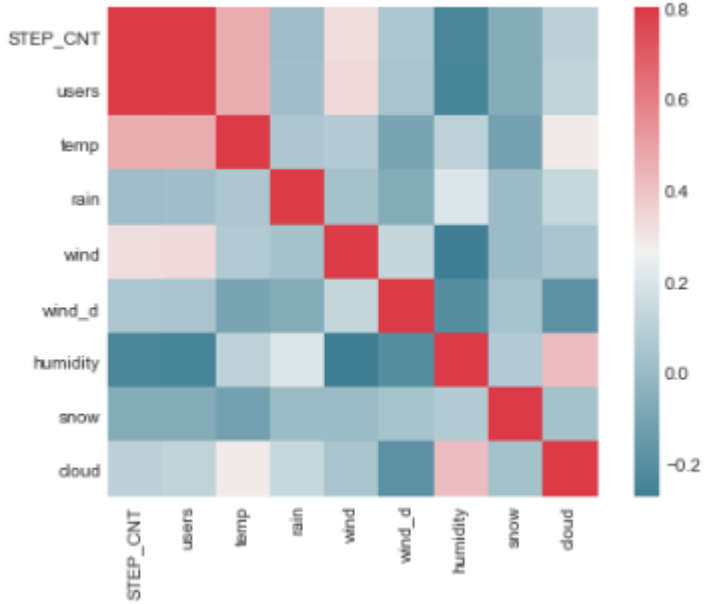
- 타겟(걸음 수)과 실수형 변수간의 선형 상관도를 확인한다.

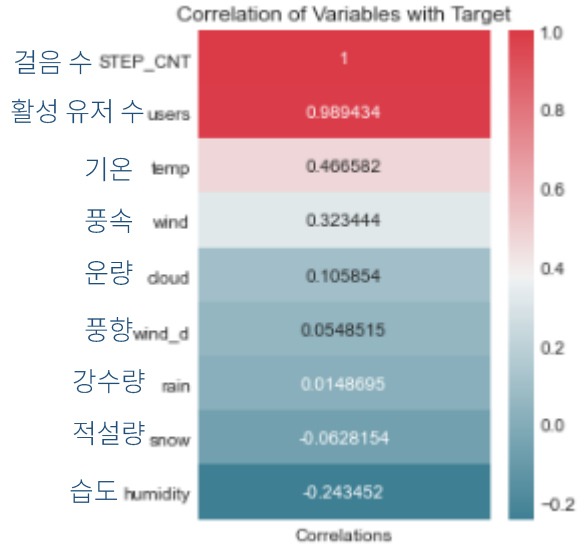
```
fig, axes = plt.subplots(1, 2, figsize = (12, 6), gridspec_kw={'width_ratios': [2, 1]})
corr = step_total.corr()

sns.heatmap(corr, vmax = 0.8, square = True, cmap =cmap, ax = axes[0])

corr = data.corrwith(data['STEP_CNT']).reset_index()
corr.columns = ['Index','Correlations']
corr = corr.set_index('Index') .sort_values(by=['Correlations'], ascending = False)
sns.heatmap(corr, annot=True, fmt="g", cmap=cmap, ax = axes[1])
axes[1].set_title("Correlation of Variables with Class", fontsize = 12)
```







- 특정 시간대의 걸음 수는 활성 고객 수(users)와 강한 상관성을 가진다.
- 기온, 풍속 등이 그 다음으로 상관성이 높다.
- 습도는 음의 상관성을 가진다.
- 기후 정보가 예상외로 상관성이 낮다.
- 사용자의 거주지역을 알지 못해 서울지역의 기후 데이터를 사용했기 때문으로 추정된다.

- 왼쪽 표는 타겟과 실수형 피처 간 선형 상관 관계(Pearson Correlation)를 나타낸 것으로 비선형 상관성은 확인할 수 없다.
- 비선형 상관성을 확인하기 위해서는 시각화나 다른 계산법(ex: DTW) 등을 사용해야 한다.

예측모델 개발 1

- 앞에서 유의미한 상관성을 보인 속성들을 사용하여 시간대별 걸음 수를 예측(forecast)하는 머신러닝 모델을 만들어보자.

  - 학습 데이터 : 2022-02-01 ~ 2022-06-30
    검증 데이터 : 2022-07-01 ~ 2022-07-10

```
train = df[df['timestamp']<datetime.datetime(2022, 7, 1)]
test = df[df['timestamp']>datetime.datetime(2022, 7, 1)]
```

  - 피처 선정 : 활성 사용자수(users), 시간(hour), 요일(dayofweek),계절(season), 공휴일 여부(holiday), 기온(temp), 풍속(wind_d), 습도(humidity), 적설량(snow), 운량(cloud)

```
X_train = train.drop(['timestamp','year', 'month', 'day', 'STEP_CNT'], axis = 1)
y_train = train['STEP_CNT']

X_test = test.drop(['timestamp','year', 'month', 'day', 'STEP_CNT'], axis = 1)
y_test = test['STEP_CNT']
```

users	hour	dayofweek	season	holiday	temp	rain	wind	wind_d	humidity	snow	cloud
528	0	1	4	1	0.4	1.2	1.8	270	92	1.8	9
322	1	1	4	1	-0.2	0.0	2.9	250	85	1.8	2
170	2	1	4	1	-0.9	0.0	2.8	250	74	1.7	7
128	3	1	4	1	-0.9	0.0	3.6	250	72	1.7	9
78	4	1	4	1	-1.7	0.0	3.1	270	87	2.0	10
...	...	...	...	...	...	...	...	...	...	...	...

학습 데이터 피처

예측모델 개발 1

- 모델 학습 : 트리기반의 모델인 LightGBM을 사용하며, 기본 파라미터를 사용함
  평가지표는 MAPE(Mean Absolute Percentage Error)를 사용

```
model = lgb.LGBMRegressor()
model.fit(X_train, y_train)
pred = model.predict(X_test)
test['STEP_CNT_pred'] = pred

MAPE(y_test, pred)
```

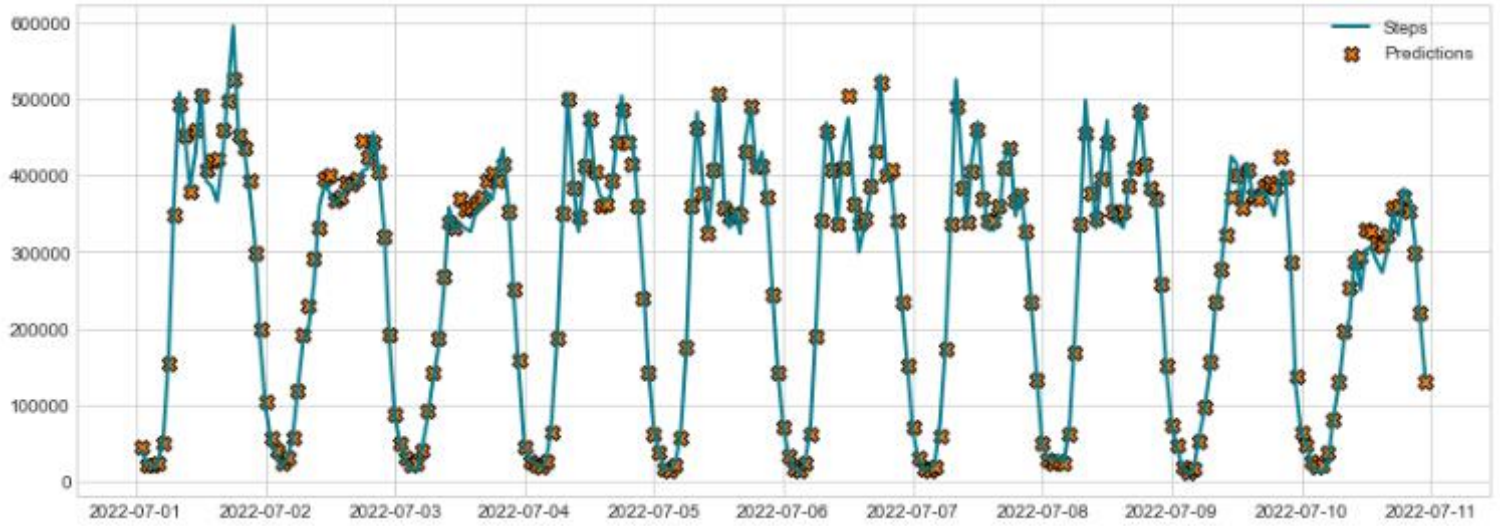
```
def MAPE(y_test, y_pred):
            return np.mean(np.abs((y_test - y_pred) / y_test)) * 100
```

MAPE	Interpretation
<10	Highly accurate forecasting
10-20	Good forecasting
20-50	Reasonable forecasting
>50	Inaccurate forecasting

Source: Lewis (1982, p. 40)

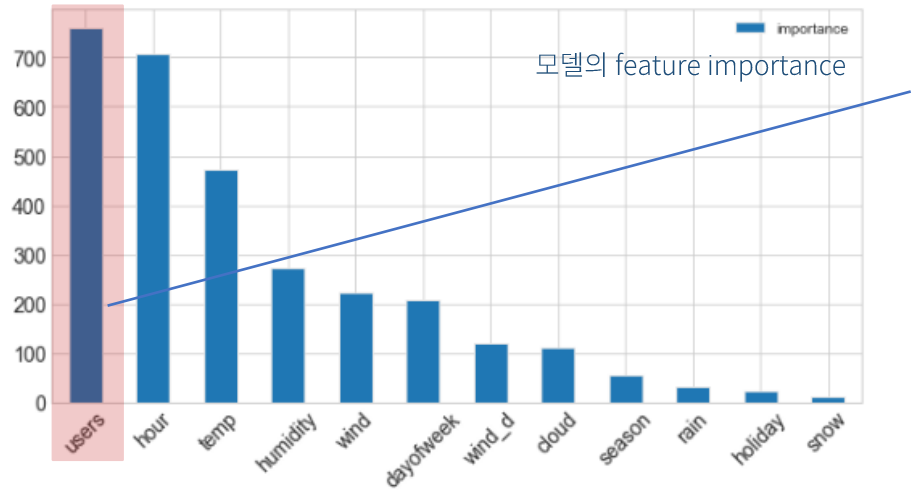
예측모델 개발 1

- 모델 평가 : MAPE 6.54%



MAPE	Interpretation
<10	Highly accurate forecasting
10-20	Good forecasting
20-50	Reasonable forecasting
>50	Inaccurate forecasting

Source: Lewis (1982, p. 40)



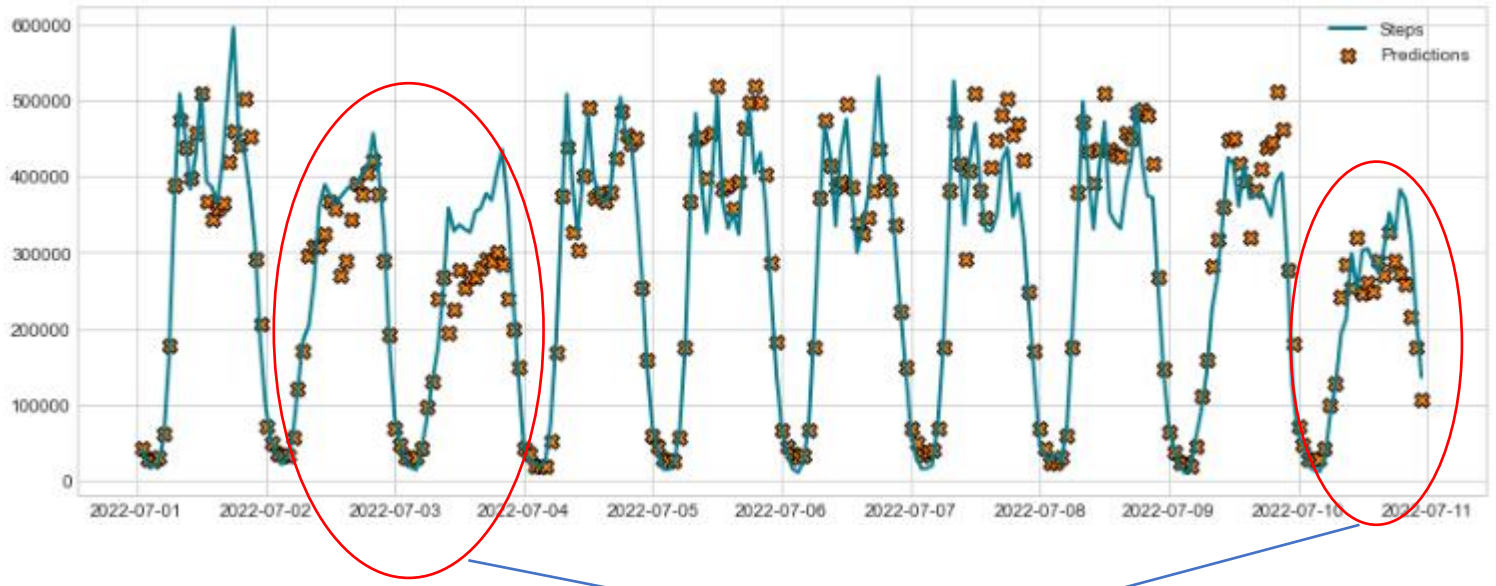
- 모델의 예측 성능이 매우 우수하다.(오차율↓)
- 모델이 예측할 때 활성 유저 수(users)가 가장 중요도가 높게 작동한다.
- 그런데 이 속성은 실제 모델이 예측할 시점에 사용할 수 없다.
- 예측할 미래 시점에는 활성 유저 수 정보가 없기 때문이다.(Data leakage)
- 따라서 이 모델은 실제에서 사용할 수 없다.

- users 속성을 제외한 데이터 셋으로 모델을 다시 구축해보자.

예측모델 개발 2

활성 사용자 수(users) 제외

- 피처 선정 : 시간(hour), 요일(dayofweek),계절(season), 공휴일여부(holiday), 기온(temp), 풍속(wind_d), 습도(humidity), 적설량(snow), 운량(cloud)
- 모델 평가 : MAPE 19.55%



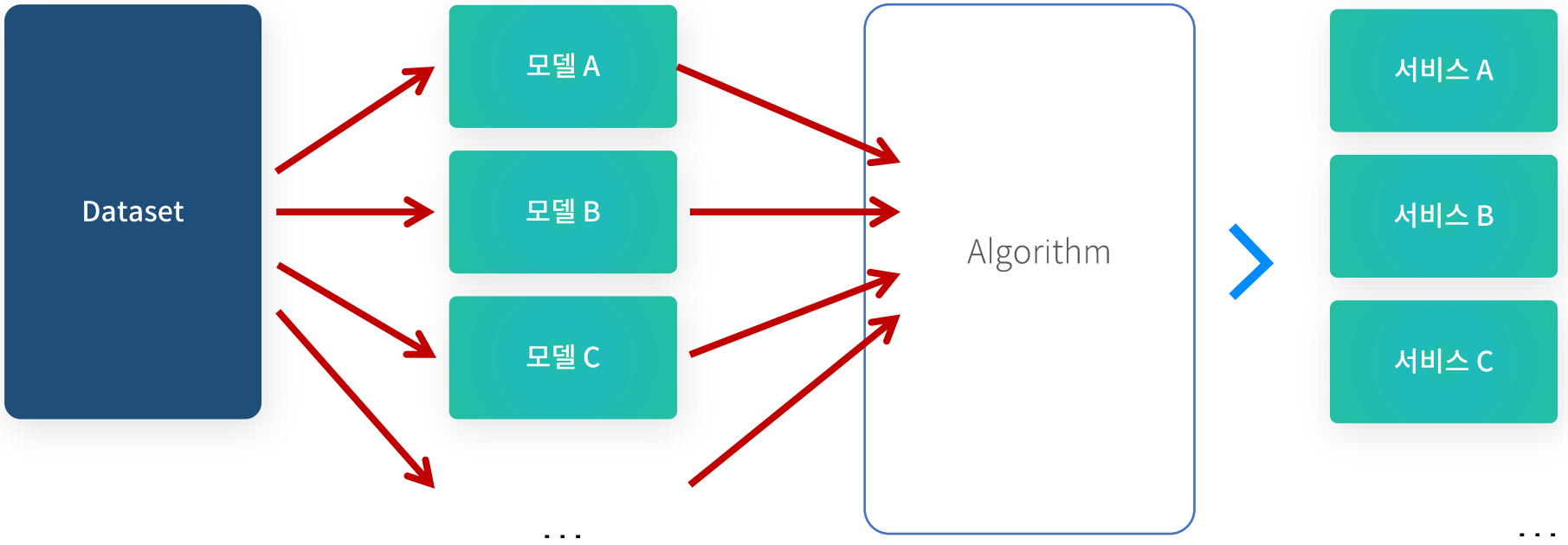
주말 시간대에 예측 성능이 낮게 나타난다.

- 예측 성능은 낮아졌지만 실제에서 사용할 수 있는 모델이다.
- 하이퍼파라미터 튜닝, 타겟 변환, 추가 속성을 발굴하면 예측 성능을 더 향상시킬 수 있다.

결론

- 간단한 통계 분석을 통해 Rothy 사용자의 걸음 수에 영향을 주는 요소들과 패턴을 확인할 수 있었다.
- 이 요소들을 사용해서 시간대별 Rothy 고객의 걸음 수를 예측하는 모델을 개발하였다.
- 모델 성능 향상을 위해서 속성 발굴, 학습 데이터 추가 수집이 필요하다.
- 예측 모델 활용 예
  - Rothy 서비스, AI 걸음 가이드 refresh 시점 결정
  - 고객 그룹 별 패턴 분석을 통한 타겟 마케팅, 걸음 가이드
  - 고객 이탈 조기 감지/ 예측 등

- 대형 시스템이나 플랫폼에서 머신러닝을 활용할 때 단일 모델이 적용되는 경우는 드물다.
- 다수의 모델이 복합적으로 연계되기도 하고, 엔드 프로덱트(서비스)에 직/간접적으로 기여하기도 한다.
- 머신러닝이나 데이터분석으로 문제 해결하는 것이 유리한 부분을 찾고, 문제를 정의하는 능력을 키우는 것이 중요하다.



GI VITA

info@gi-vita.io
5, 8, 9F, 507, Gangnam-daero, Seocho-gu, Seoul | www.gi-vita.io

Global No.1 Health Data Tech Company