0. 사용 모듈

```
# Data Handling
import pandas as pd
import datetime as dt
import numpy as np

# Data Visualizing
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
from tqdm import tqdm

#model
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error, mean_squared_log_er
import joblib

# etc.
import warnings
warnings.filterwarnings('ignore')
```

1. 데이터 불러오기

```
# colab 환경설정
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
path = "/content/drive/MyDrive/BusProject/NumofPassenger/CSV/"
df_list = []

for i in tqdm( range(1,60) ):
    df_list.append( pd.read_csv(path+str(i)+".csv", encoding='cp949') )

df = pd.concat( df_list ).reset_index()
```

```
100%|██████████| 59/59 [00:21<00:00,  2.72it/s]
```

```
df.head()
```

	index	일자	노선명	정류장명	ARS_ID	시간	승하차	거래건수
0	0	20220301	1187	4수원지	4263.0	10	승차	1
1	1	20220301	1187	4수원지	4264.0	12	승차	1
2	2	20220301	1187	4수원지	4264.0	13	하차	1
3	3	20220301	1187	4수원지위	4268.0	11	하차	1

```
df = df.drop(['index'], axis=1)
df['일자'] = pd.to_datetime(df.일자, format='%Y%m%d')
```

```
df.head()
```

	일자	노선명	정류장명	ARS_ID	시간	승하차	거래건수
0	2022-03-01	1187	4수원지	4263.0	10	승차	1
1	2022-03-01	1187	4수원지	4264.0	12	승차	1
2	2022-03-01	1187	4수원지	4264.0	13	하차	1
3	2022-03-01	1187	4수원지위	4268.0	11	하차	1
4	2022-03-01	1187	4수원지위	4268.0	13	하차	2

```
print( "총 관측치의 수: {:,}".format(len(df)) )
```

```
총 관측치의 수: 3,670,384
```

2. EDA

1) Target 데이터 분포 확인

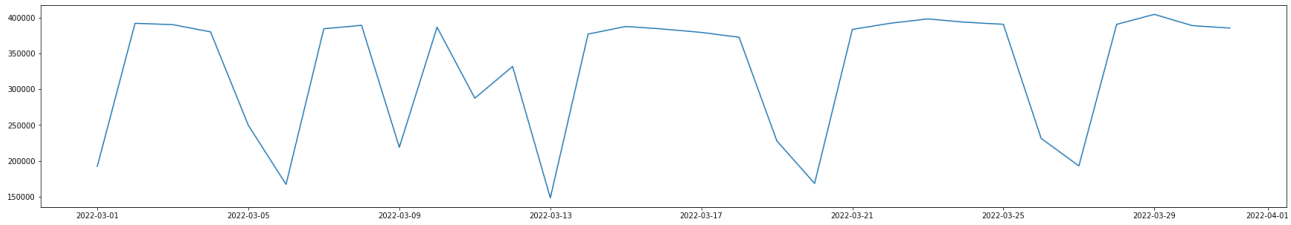
```
print(df.거래건수.value_counts())
plt.plot(df.거래건수.value_counts())

df.거래건수.value_counts().unique()
```

```
1       1769752
2        743009
3        385039
4        225179
5        144212
         ...
123           1
118           1
144           1
125           1
147           1
Name: 거래건수, Length: 149, dtype: int64
array([1769752,  743009,  385039,  225179,  144212,   97109,   68111,
         49193,   36570,   27739,   21615,   16942,   13289,   10916,
          9023,    7320,    5945,    5056,    4336,    3651,    3129,
          2683,    2308,    1997,    1730,    1567,    1315,    1156,
          1035,     874,     853,     773,     661,     597,     485,
           447,     404,     365,     341,     300,     257,     248,
           244,     194,     188,     180,     170,     138,     123,
           107,     105,      92,      87,      73,      70,      66,
            64,      59,      56,      40,      38,      35,      34,
            33,      32,      29,      27,      25,      23,      22,
            19,      15,      14,      12,      11,      10,       9,
             8,       7,       6,       5,       4,       3,       2,
             1])
```


```
fig = plt.figure(figsize=(30,5))
fig.add_subplot()
plt.plot(df.groupby('일자')['거래건수'].sum())
```

```
[<matplotlib.lines.Line2D at 0x7fd787a28a90>]
```



2) 정류장명과 ARS_ID의 1:1 대응여부 확인

```
print(len(df.정류장명.unique()), len(df.ARS_ID.unique()), sep='₩n')
len(df.정류장명.unique()) == len(df.ARS_ID.unique())
```

```
1488
2770
False
```

```
nunique_df = df.groupby('정류장명')['ARS_ID'].nunique()

print( nunique_df.unique(), end='₩n₩n₩n' )
print( nunique_df, end='₩n₩n₩n' )
print( nunique_df.value_counts(), end='₩n₩n₩n' )
```

```
[2 1 4 3]


정류장명
31사단          2
4.19기념관       2
4수원지           2
4수원지위         2
5.18기념공원      2
              ..
흑석사거리(동)      2
흑석사거리(서)      2
희망가아파트         2
힐스테이트 각화      2
힐스테이트연제아파트    2
Name: ARS_ID, Length: 1487, dtype: int64


2    1268
1     213
3       4
4       2
Name: ARS_ID, dtype: int64
```

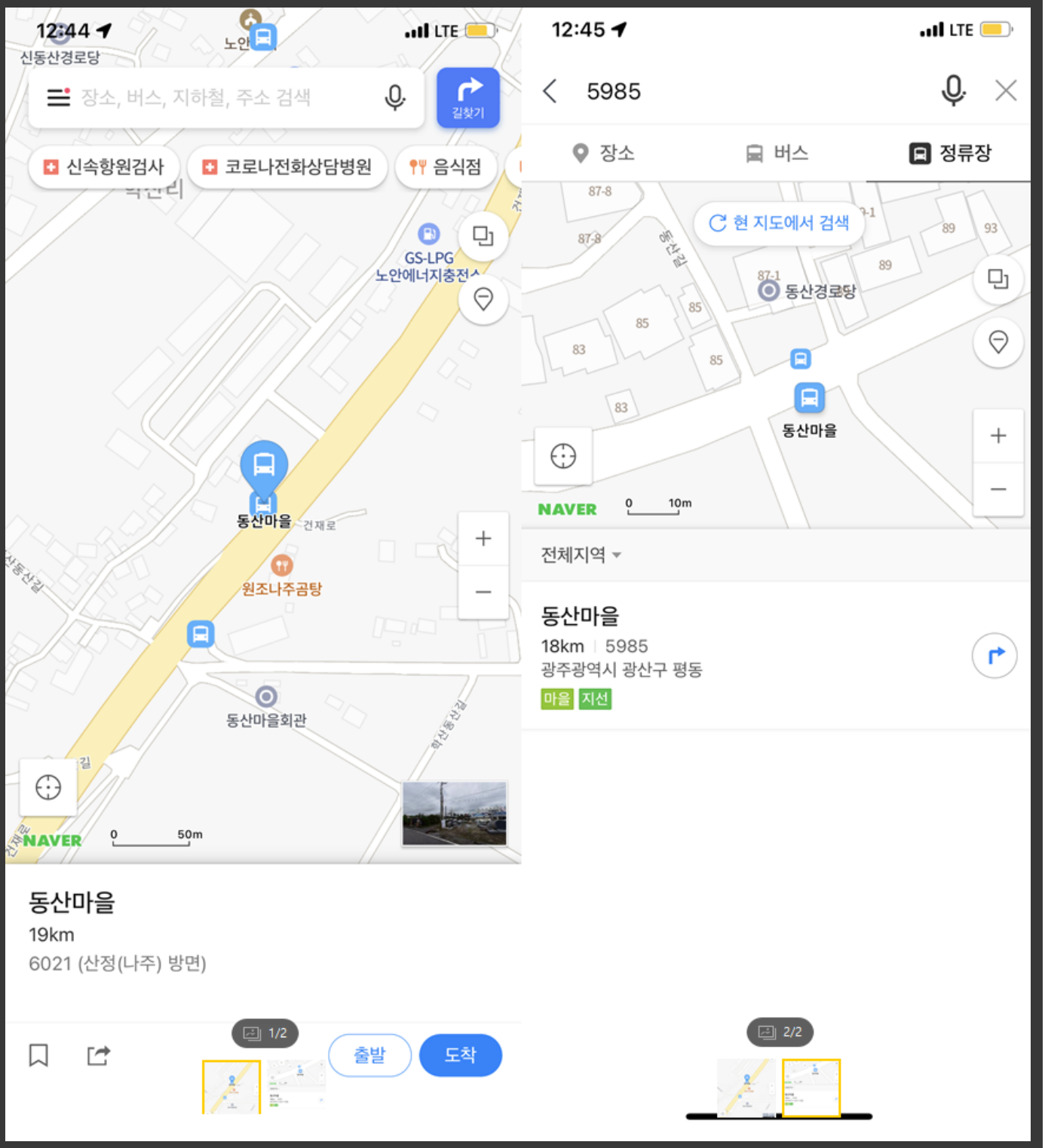
대부분의 정류장은 정류장명마다 2개의 ARS_ID를 가지고 있다
-> 하나의 정류장마다 가는 방향, 오는 방향이 있기 때문에 총 2개의 ARS_ID를 가지고 있는 것
3개 이상인 정류장도 있길래 확인해보았다.

```
nunique_df[nunique_df>=3]
```

```
정류장명
동산마을        4
명도삼거리       3
문화전당역       3
살레시오고입구     4
평촌          3
효령노인복지타운    3
Name: ARS_ID, dtype: int64
```

```
df[df.정류장명=='동산마을'].ARS_ID.unique()
```

```
array([5985.0, 6021.0, 6022.0, 5965.0], dtype=object)
```

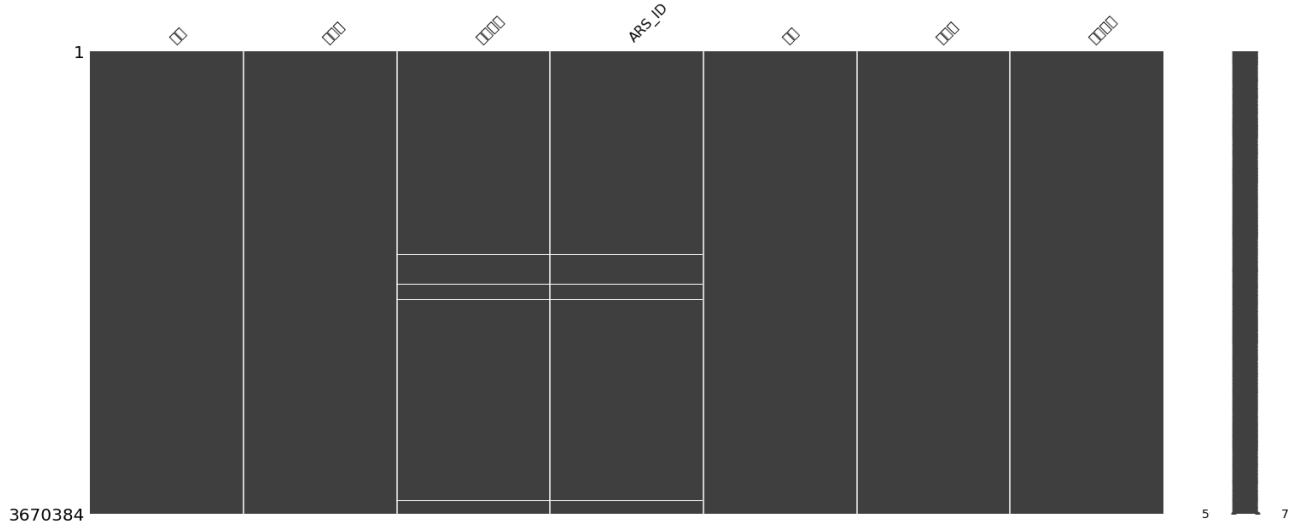


* 동일 이름의 정류장이 있었던 것으로 확인
* 따라서 정류장의 고유값으로는 정류장명이 아닌 ARS_ID를 사용하는 것이 바람직 함.

3) 결측치 확인

```
msno.matrix(df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f85ad0c2ed0>
```




```
df.isnull().sum()
```

```
일자            0
노선명           0
정류장명      24032
ARS_ID    24197
시간            0
승하차           0
거래건수          0
dtype: int64
```

```
df[df.ARS_ID.isnull()]
```

	일자	노선명	정류장명	ARS_ID	시간	승하차	거래건수
552	20220301	1187	NaN	NaN	9	승차	6
553	20220301	1187	NaN	NaN	10	승차	1
554	20220301	1187	NaN	NaN	11	승차	7
555	20220301	1187	NaN	NaN	12	승차	20
556	20220301	1187	NaN	NaN	13	승차	5
...	...	...	...	...	...	...	...
3670121	20220331	첨단95	NaN	NaN	21	승차	1
3670122	20220331	첨단95	NaN	NaN	22	승차	2

```
df[df.정류장명.isnull()]
```

	일자	노선명	정류장명	ARS_ID	시간	승하차	거래건수
552	20220301	1187	NaN	NaN	9	승차	6
553	20220301	1187	NaN	NaN	10	승차	1
554	20220301	1187	NaN	NaN	11	승차	7
555	20220301	1187	NaN	NaN	12	승차	20
556	20220301	1187	NaN	NaN	13	승차	5
...	...	...	...	...	...	...	...
3670121	20220331	첨단95	NaN	NaN	21	승차	1
3670122	20220331	첨단95	NaN	NaN	22	승차	2
3670375	20220331	충효187	NaN	NaN	8	승차	1
3670376	20220331	충효187	NaN	NaN	14	승차	2
3670377	20220331	충효187	NaN	NaN	17	승차	1

24032 rows × 7 columns

정류장명과 ARS_ID간의 null값 개수가 다르고, 정류장명과 정류장 ID가 Null값인 관측치가 상당수 존재.
전체 데이터 셋에 비해서 매우 적은 양(약 0.5%), 또한 정류장명과 ARS_ID는 회귀식에 데이터가 아님
따라서 정류장명과 ARS_ID간의 대응관계만 맞춰주고, 나머지 Null값은 모두 Drop

```
df[df.정류장명.notnull() & df.ARS_ID.isnull()]
```

	일자	노선명	정류장명	ARS_ID	시간	승하차	거래건수
14662	20220301	마을788	시화문화마을문화관	NaN	8	승차	1
14663	20220301	마을788	시화문화마을문화관	NaN	12	승차	1
112519	20220302	마을788	시화문화마을문화관	NaN	7	승차	4
112520	20220302	마을788	시화문화마을문화관	NaN	8	승차	1
112521	20220302	마을788	시화문화마을문화관	NaN	10	승차	1
...	...	...	...	...	...	...	...
3563174	20220331	마을788	시화문화마을문화관	NaN	12	승차	1
3563175	20220331	마을788	시화문화마을문화관	NaN	13	승차	1
3563176	20220331	마을788	시화문화마을문화관	NaN	14	승차	1

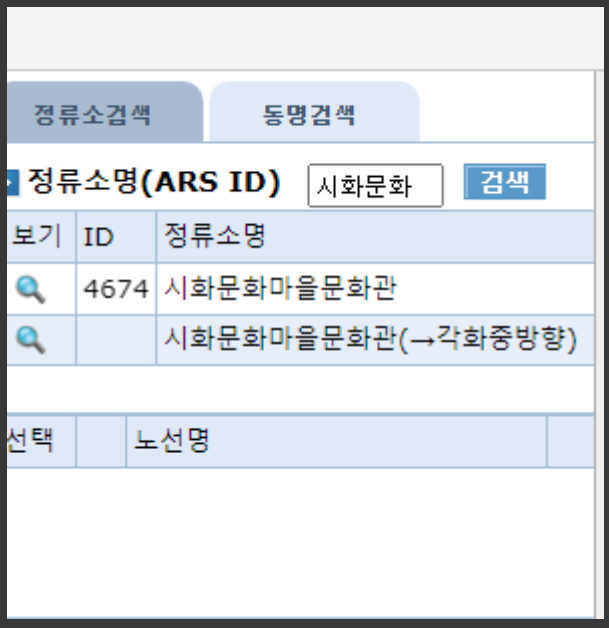
```
df[df.정류장명.isnull() & df.ARS_ID.notnull()]
```

일자	노선명	정류장명	ARS_ID	시간	승하차	거래건수

```
df[df.정류장명.notnull() & df.ARS_ID.isnull()].정류장명.unique()
```

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

정류장명은 존재하지만 ARS_ID가 존재하지 않는 경우 존재
확인 결과, 각화중 방향 사회문화마을문화관 정류장은 따로 ARS_ID가 존재하지 않았음



3. 데이터 전처리

0) 광주광역시 노선 정보 데이터 불러오기

```
# url = 'D:/python_projects/BusStopCongestionProject/Data/Original_data/GwangjuBusInfo/Excel/광주광
# BusInfo = pd.read_excel(url, sheet_name=None)
```

```
# bus_dict = {}
# for i in enumerate(BusInfo):
#     key = i[1]
#     item = BusInfo[ key ].filter(regex='출발')
#     if (len(item.columns)>1):
#         item = list( set(item.iloc[:,0]) | set(item.iloc[:,1]) )

#     bus_dict[key] = item
```

```
# for bus_names in df[df.ARS_ID.isnull()].노선명.unique():
#     if(bus_names in bus_dict.keys()):
#         print("[%s]₩n총 정류장 수: %d₩n데이터셋의 정류장 수: %d₩n₩n"
#               %(bus_names,len(bus_dict[bus_names]), df[df.노선명==bus_names].정류장명.nunique()))
```

숨겨진 출력 표시

```
# for bus_names in df[df.ARS_ID.isnull()].노선명.unique():
#     if(bus_names in bus_dict.keys()):
#         whole_bus = set( bus_dict[bus_names] )
#         dataset_bus = set( df[df.노선명==bus_names].정류장명.unique() )
#         print("%s"%bus_names)
#         print( list(whole_bus - dataset_bus ), end='₩n₩n' )
```

숨겨진 출력 표시

1) 결측치 제거

- 임의의 ARS_ID 값 부여

```
null_ARS_ID = df[(df.정류장명=='시화문화마을문화관') & df.ARS_ID.isnull()].ARS_ID
df.loc[null_ARS_ID.index, 'ARS_ID'] = "46740"
```

각화중 방향 ARS_ID에는 임의의 코드 값 부여

```
df[df.정류장명.notnull() & df.ARS_ID.isnull()]
```

	일자	노선명	정류장명	ARS_ID	시간	승하차	거래건수

```
df.isnull().sum()
```

```
일자            0
노선명           0
정류장명       24032
ARS_ID     24032
시간            0
승하차           0
거래건수          0
dtype: int64
```

정류장명과 ARS_ID 간 결측치 개수 일치

- drop NA

```
df.shape
```

```
(3670384, 7)
```

```
df = df.dropna()
df.shape
```

```
(3646352, 7)
```

```
df.isnull().sum()
```

```
일자         0
노선명        0
정류장명       0
ARS_ID     0
시간         0
승하차        0
거래건수       0
dtype: int64
```

2) 데이터 정류장 별 그룹화

```
df['ARS_ID'] = df.ARS_ID.astype('str')
df['일자'] = df.일자.astype('str')
grp_df = df.groupby(['일자','ARS_ID','시간','승하차']).sum()
grp_df = grp_df.reset_index()
```

```
grp_df
```

	일자	ARS_ID	시간	승하차	거래건수
0	2022-03-01	1002.0	5	하차	2
1	2022-03-01	1002.0	6	승차	1
2	2022-03-01	1002.0	6	하차	1
3	2022-03-01	1002.0	7	승차	6
4	2022-03-01	1002.0	8	승차	6
...	...	...	...	...	...
1399369	2022-03-31	6622.0	18	승차	18
1399370	2022-03-31	6622.0	19	승차	5
1399371	2022-03-31	6622.0	20	승차	4
1399372	2022-03-31	6622.0	21	승차	3
1399373	2022-03-31	6622.0	22	승차	3

1399374 rows × 5 columns

3) 요일 데이터 One-hot Encoding

```
grp_df['weekday'] = pd.to_datetime(grp_df['일자'], format="%Y-%m-%d").dt.weekday
grp_df = pd.get_dummies(grp_df, columns=['weekday'])
grp_df.columns
```

```
Index(['일자', 'ARS_ID', '시간', '승하차', '거래건수', 'weekday_0', 'weekday_1',
       'weekday_2', 'weekday_3', 'weekday_4', 'weekday_5', 'weekday_6'],
      dtype='object')
```

moday is 0, sunday is 6

4) 추가 피처 생성

- 기상청 데이터 활용

```
path = '/content/drive/MyDrive/BusProject/기상청 데이터/OBS_ASOS_TIM_20220504132652.csv'

whether_data = pd.read_csv(path, encoding='cp949')
whether_data = whether_data.iloc[:,2:]
whether_data['일시'] = pd.to_datetime(whether_data['일시'])
whether_data.columns = ['일시', '기온', '강수량', '풍속', '습도']
whether_data
```

	일시	기온	강수량	풍속	습도
0	2022-03-01 00:00:00	9.1	NaN	0.7	71
1	2022-03-01 01:00:00	8.6	NaN	0.0	67
2	2022-03-01 02:00:00	8.3	NaN	0.0	65
3	2022-03-01 03:00:00	8.8	NaN	1.2	60
4	2022-03-01 04:00:00	6.4	NaN	0.2	97
...	...	...	...	...	...
739	2022-03-31 19:00:00	9.6	NaN	0.8	83
740	2022-03-31 20:00:00	9.0	NaN	0.5	87
741	2022-03-31 21:00:00	8.6	NaN	1.1	88
742	2022-03-31 22:00:00	8.3	NaN	0.5	87
743	2022-03-31 23:00:00	8.2	NaN	0.1	86

```
grp_df['시간'] = grp_df.apply(lambda x: '0'+str(x.시간) if len(str(x.시간))<2 else str(x.시간), ax
grp_df['일시'] = grp_df.apply(lambda x: pd.to_datetime(str(x.일자)[:-8] + str(x.시간), format='%Y-%
```

```
new_df = pd.merge(grp_df,whether_data, how='left', on='일시')
new_df = new_df[ ['일시','ARS_ID', 'weekday_0', 'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4',
        'weekday_5', 'weekday_6', '기온', '강수량', '풍속', '습도', '승하차', '거래건수'
        ] ]
new_df.fillna(0, inplace=True)
new_df
```

	일시	ARS_ID	weekday_0	weekday_1	weekday_2	weekday_3	weekday_4	w
0	2022-03-01 05:00:00	1002.0	0	1	0	0	0	
1	2022-03-01 06:00:00	1002.0	0	1	0	0	0	
2	2022-03-01 06:00:00	1002.0	0	1	0	0	0	
3	2022-03-01 07:00:00	1002.0	0	1	0	0	0	
4	2022-03-01 08:00:00	1002.0	0	1	0	0	0	

- 공휴일 유무 피처

```
path = '/content/drive/MyDrive/BusProject/holiday.csv'
month = 3

holiday_df = pd.read_csv(path, encoding='cp949')
holiday_df = holiday_df[(holiday_df.년>=2022) & (holiday_df.월==month)]
holiday_df
```

	순서	년	월	일	설명
75	76	2022	3	1	삼일절
76	77	2022	3	9	대통령선거일

```
new_df['hollyday'] = new_df.apply(lambda x: 1 if x.일시.day in holiday_df.일.tolist() else 0, axis=
new_df['weekend'] = new_df.apply(lambda x: 1 if x.일시.weekday()>4 else 0, axis=1)

new_df
```

	일시	ARS_ID	weekday_0	weekday_1	weekday_2	weekday_3	weekday_4	w
0	2022-03-01 05:00:00	1002.0	0	1	0	0	0	
1	2022-03-01 06:00:00	1002.0	0	1	0	0	0	
2	2022-03-01 06:00:00	1002.0	0	1	0	0	0	
3	2022-03-01 07:00:00	1002.0	0	1	0	0	0	
4	2022-03-01 08:00:00	1002.0	0	1	0	0	0	

5) 시간/승하차별 거래건수 Feature화

```
time = new_df['일시'].apply( lambda x: str(x.hour) )
ride_takeoff = new_df['승하차'].apply( lambda x: '_r' if (x=='승차') else '_t' )
corr_df = pd.get_dummies( time + ride_takeoff )
corr_df = pd.concat([corr_df, new_df.거래건수], axis=1)
```

```
corr_df = corr_df.apply(lambda x: x.replace(1, x.거래건수), axis=1).drop('거래건수', axis=1)
corr_df = corr_df.astype('int')
```

	0_r	0_t	10_r	10_t	11_r	11_t	12_r	12_t	13_r	13_t	...	5_r	5_t	6_
0	0	0	0	0	0	0	0	0	0	0	...	0	2	
1	0	0	0	0	0	0	0	0	0	0	...	0	0	
2	0	0	0	0	0	0	0	0	0	0	...	0	0	
3	0	0	0	0	0	0	0	0	0	0	...	0	0	
4	0	0	0	0	0	0	0	0	0	0	...	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1399369	0	0	0	0	0	0	0	0	0	0	...	0	0	
1399370	0	0	0	0	0	0	0	0	0	0	...	0	0	
1399371	0	0	0	0	0	0	0	0	0	0	...	0	0	
1399372	0	0	0	0	0	0	0	0	0	0	...	0	0	
1399373	0	0	0	0	0	0	0	0	0	0	...	0	0	

1399374 rows × 41 columns

```
for i in range(5,24,2):
  if (i==23):
    corr_df[ str(23)+'~'+str(0)+'_r' ] = corr_df[ str(23)+'_r' ] + corr_df[ str(0)+'_r' ]
    corr_df[ str(23)+'~'+str(0)+'_t' ] = corr_df[ str(23)+'_t' ] + corr_df[ str(0)+'_t' ]
    break

  corr_df[ str(i)+'~'+str(i+1)+'_r' ] = corr_df[ str(i)+'_r' ] + corr_df[ str(i+1)+'_r' ]
  corr_df[ str(i)+'~'+str(i+1)+'_t' ] = corr_df[ str(i)+'_t' ] + corr_df[ str(i+1)+'_t' ]
```

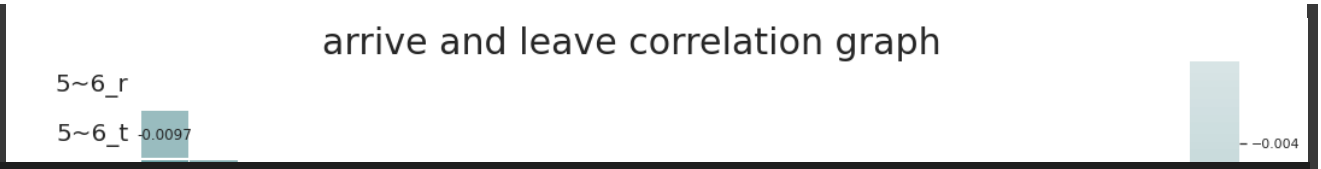
```
corr_df.columns.unique
```

```
<bound method Index.unique of Index(['0_r', '0_t', '10_r', '10_t', '11_r', '11_t', '12_r', '
       '13_t', '14_r', '14_t', '15_r', '15_t', '16_r', '16_t', '17_r', '17_t',
       '18_r', '18_t', '19_r', '19_t', '1_t', '20_r', '20_t', '21_r', '21_t',
       '22_r', '22_t', '23_r', '23_t', '5_r', '5_t', '6_r', '6_t', '7_r',
       '7_t', '8_r', '8_t', '9_r', '9_t', '5~6_r', '5~6_t', '7~8_r', '7~8_t',
       '9~10_r', '9~10_t', '11~12_r', '11~12_t', '13~14_r', '13~14_t',
       '15~16_r', '15~16_t', '17~18_r', '17~18_t', '19~20_r', '19~20_t',
       '21~22_r', '21~22_t', '23~0_r', '23~0_t'],
      dtype='object')>
```

```
cor = corr_df[['5~6_r', '5~6_t', '7~8_r', '7~8_t',
       '9~10_r', '9~10_t', '11~12_r', '11~12_t', '13~14_r', '13~14_t',
       '15~16_r', '15~16_t', '17~18_r', '17~18_t', '19~20_r', '19~20_t',
       '21~22_r', '21~22_t', '23~0_r', '23~0_t']].corr()

sns.set(style="white")
mask=np.zeros_like(cor,dtype=np.bool)
mask[np.triu_indices_from(mask)]=True
```

```
f,ax=plt.subplots(figsize=(20,15))
cmap=sns.diverging_palette(200,10,as_cmap=True)
sns.heatmap(cor,mask=mask,cmap=cmap,center=0,square=True,linewidths=0.5,cbar_kws={"shrink":1},anno
plt.xticks(size=20)
plt.yticks(size=20,rotation=0)
plt.title("arrive and leave correlation graph",size=30);
```




```
for i in grp_df.loc[:,'시간'].unique():
    grp_df[str(i)+'_ride'] = 0
    grp_df[str(i)+'_takeoff'] = 0
```

9~10 t -0.015 -0.012 -0.018 -0.016 -0.019

```
def makeFeatures(data):
        time = str(data.시간)

        if (data.승하차=='승차'): tmp = '_ride'
        else: tmp = '_takeoff'

        num = data.거래건수

        return [time+tmp, num]
```

```
for i in tqdm(range(len(grp_df))):
    data_list = makeFeatures(grp_df.loc[i,:])
    grp_df.loc[i,data_list[0]] = data_list[1]
```

21~22 t -0.014 -0.011 -0.017 -0.015 -0.018 -0.017 -0.017 -0.017 -0.017 -0.018 -0.016 -0.017 -0.016 -0.017 -0.013 -0.017 -0.012 -0.018

4. 모델 학습 및 저장

1) 모델 학습

```
final_df = new_df[new_df.승하차=='승차']
final_df
```

```
                일시  ARS_ID  weekday_0  weekday_1  weekday_2  weekday_3  weekday_4  w
```

```
input_var = ['weekday_0', 'weekday_1', 'weekday_2', 'weekday_3', 'weekday_4',
             'weekday_5', 'weekday_6', '기온', '강수량', '풍속', '습도']
target = new_df['거래건수']
```

```
          06:00:00
```

```
x_train, x_test, y_train, y_test = train_test_split(new_df.loc[:,input_var].astype('int'), target.a
```

```
3       03-01  1002.0          0          1          0          0          0
```

```
rf = RandomForestRegressor(random_state=1217)
rf.fit(x_train, y_train)
```

```
RandomForestRegressor(random_state=1217)
```

```
          2022-
```

```
y_pred = rf.predict(x_test)

MAE =  mean_absolute_error(y_test, y_pred)
MSE = mean_squared_error(y_test, y_pred)
RMSE = np.sqrt(MSE)
MSLE = mean_squared_log_error(y_test, y_pred)
RMSLE = np.sqrt(mean_squared_log_error(y_test, y_pred))
R2 = r2_score(y_test, y_pred)

print(MAE, MSE, RMSE, MSLE, RMSLE, R2, sep='₩n')
```

```
6.851206532072684
161.78070298982476
12.719304343784874
1.0090492777777955
1.0045144487650715
0.0286476060087687
```

```
score = np.mean(np.abs(y_test-y_pred)/y_test)
print('점수 :',score)
```

```
점수 : 2.489868306590451
```

2) 모델 저장

```
joblib.dump(rf, '/content/drive/MyDrive/BusProject/Model/bus_model_2.pkl')
```

```
['/content/drive/MyDrive/BusProject/Model/bus_model_2.pkl']
```

- 1차 모델

```
path = "/content/drive/MyDrive/BusProject/Model/bus_model_prototype.pkl"
loaded_model = joblib.load(path)
loaded_model
```

```
RandomForestRegressor(criterion='mse', random_state=1217)
```

```
y_pred = loaded_model.predict(x_test.iloc[:,:-4])
```

- 2차 모델

```
path2 = "/content/drive/MyDrive/BusProject/Model/bus_model_2.pkl"
loaded_model2 = joblib.load(path2)
loaded_model2
```