

소프트웨어응용 프로젝트 최종 발표

동대문구 내 공공자전거 따릉이 대여소 방역 스케줄 추천 서비스

I. 주제 선정 배경 II. 데이터 분석 III. 머신 러닝 IV. 스케줄 추천 알고리즘 설계 V. 웹 서비스 구축 VI. 고찰 VII. 참고 자료

목차

- I. 주제 선정 배경
- II. 데이터 분석
- III. 머신 러닝
- IV. 스케줄 추천 알고리즘 설계
- V. 웹 서비스 구축
- VI. 고찰
- VII. 참고 자료



주제 선정 배경



코로나19 확산으로 인한 사회적 거리두기



서울시 공공자전거 따릉이 이용 건수 증가

<서울시 공공자전거 따릉이 방역 강화>

- ▶ 매일 1,540여개 대여소 방문 후 전수 소독
- ▶ 하지만 특정 시간대에 따릉이 쏠림 현상 존재

- 소독대상 : 따릉이 이용시 손이 자주 닿는 곳 중점 소독

자전거 ... 손잡이, 단말기, 안장 / 거치대 ... 잠금장치



* [공공자전거 따릉이 방역단 운영 계획]에서 발췌

“보다 효율적인 서울시 공공자전거 따릉이 방역 스케줄을 예측 및 추천”

I. 주제 선정 배경 II. 데이터 분석 III. 머신 러닝 IV. 스케줄 추천 알고리즘 설계 V. 웹 서비스 구축 VI. 고찰 VII. 참고 자료

타겟 데이터셋

서울특별시 공공자전거 대여이력 정보 (2019-01-01 ~ 2020-06-30)

자전거번호	대여일시	대여 대여소명	대여대여소명	반납대여소명	반납대여소명	대여시간	이용거리
SPB-04392	2019-11-28 23:07	2377 수서역 5번출구 앞	9	2019-11-28 23:18	1203 밀리어나2빌딩	5	11980
SPB-23502	2019-11-28 22:43	2613 잠실나들목	3	2019-11-29 0:07	1203 밀리어나2빌딩	1	82 11130
SPB-11498	2019-11-29 0:04	1256 문정현대아파트	6	2019-11-29 0:11	1203 밀리어나2빌딩	14	6 800
SPB-16665	2019-11-29 0:02	1278 송파구청 교차로	9	2019-11-29 0:21	1203 밀리어나2빌딩	2	18 3730
SPB-14688	2019-11-29 0:17	1230 송파중학교 정문	5	2019-11-29 0:26	1203 밀리어나2빌딩	3	9 1370
SPB-21479	2019-11-29 0:06	1050 둔촌역 3번 출입구	15	2019-11-29 0:38	1203 밀리어나2빌딩	20	30 4490
SPB-18660	2019-11-29 0:43	1207 마천CU우방점 앞	2	2019-11-29 1:09	1203 밀리어나2빌딩	8	23 3370
SPB-17128	2019-11-29 0:58	1271 송파도서관	1	2019-11-29 1:09	1203 밀리어나2빌딩	9	10 2110
SPB-24249	2019-11-29 1:27	1286 워래중앙푸르지오	3	2019-11-29 2:20	1203 밀리어나2빌딩	1	46 4830
SPB-06096	2019-11-29 2:07	1259 방이역 1번출구	12	2019-11-29 2:21	1203 밀리어나2빌딩	10	13 2340
SPB-04230	2019-11-29 3:41	1280 송파파크데일 2단	16	2019-11-29 4:03	1203 밀리어나2빌딩	5	22 4310
SPB-22980	2019-11-29 4:12	1221 삼전사거리 포스	6	2019-11-29 4:36	1203 밀리어나2빌딩	20	22 3660
SPB-10543	2019-11-29 7:18	1218 방이역 4번출구	4	2019-11-29 7:27	1203 밀리어나2빌딩	19	8 1970
SPB-11298	2019-11-29 7:28	1243 문정 법조단지7	19	2019-11-29 7:39	1203 밀리어나2빌딩	4	10 0
SPB-24038	2019-11-29 8:20	2614 가락고등학교 앞	10	2019-11-29 8:28	1203 밀리어나2빌딩	7	7 1280
SPB-23749	2019-11-29 8:28	1288 문정중교 사거리	6	2019-11-29 8:42	1203 밀리어나2빌딩	2	12 1210
SPB-19067	2019-11-29 7:59	1222 잠실새내역 5번 출	12	2019-11-29 8:52	1203 밀리어나2빌딩	10	52 5550
SPB-10826	2019-11-29 8:40	1240 문정 법조단지4	7	2019-11-29 8:54	1203 밀리어나2빌딩	8	14 2120
SPB-14613	2019-11-29 8:39	1247 문정 법조단지11	3	2019-11-29 8:56	1203 밀리어나2빌딩	12	16 2210

서울특별시 공공자전거 대여소 정보

대여소구	대여소구	대여소명	대여소주소	위도	경도	거치대수
마포구	101	101. (가)광명동 주민센터	서울특별시 마포구 홍교로8길 58	37.549561	126.905754	2015-09-06 23:40 5
마포구	102	102. 양원역 1번출구 앞	서울특별시 마포구 월드컵로 72	37.555640	126.910629	2015-09-06 23:42 20
마포구	103	103. 양원역 2번출구 앞	서울특별시 마포구 월드컵로 79	37.554951	126.910835	2015-09-06 23:43 14
마포구	104	104. 양원역 3번출구 앞	서울특별시 마포구 월드컵로 59	37.550629	126.914986	2015-09-06 23:44 13
마포구	105	105. 양원역 4번출구 앞	서울특별시 마포구 월드컵로 48	37.550007	126.914625	2015-09-06 23:45 5
마포구	106	106. 양원역 5번출구 앞	서울특별시 마포구 월드컵로 4	37.548645	126.912827	2015-09-06 23:46 10
마포구	107	107. 신원로명 서교동공공연립전 앞	서울특별시 마포구 월드컵북로 33	37.557151	126.918503	2015-09-06 23:47 5
마포구	108	108. 서교동 사거리	서울특별시 마포구 월드컵로 93	37.552740	126.918617	2015-09-06 23:51 10
마포구	109	109. 서교동 사거리	서울특별시 마포구 월드컵로 93	37.552740	126.918617	2015-09-07 1:27 10
서대문구	110	110. 홍대입구 1번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:28 20
마포구	111	111. 홍대입구 2번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:30 10
마포구	112	112. 홍대입구 3번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:31 10
마포구	113	113. 홍대입구 4번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:32 25
마포구	114	114. 홍대입구 5번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:33 15
마포구	115	115. 홍대입구 6번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:34 10
서대문구	116	116. 홍대입구 7번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:35 5
서대문구	117	117. 홍대입구 8번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:36 25
마포구	118	118. 홍대입구 9번출구	서울특별시 서대문구 홍익로 109	37.551199	126.915841	2015-09-07 1:37 10
마포구	119	119. 서교동 2번출구	서울특별시 마포구 월드컵로 93	37.552740	126.918617	2015-09-07 1:38 10
마포구	120	120. 신원로명 서교동공공연립전 앞	서울특별시 마포구 월드컵북로 33	37.557151	126.918503	2015-09-07 1:39 10
마포구	121	121. 마포소방서 앞	서울특별시 마포구 양천로 76	37.549767	126.931714	2015-09-07 10:30 5
마포구	122	122. 신원로명 서교동공공연립전 앞	서울특별시 마포구 월드컵북로 33	37.557151	126.918503	2015-09-07 10:31 10
마포구	123	123. 양원역 1번출구	서울특별시 마포구 월드컵로 72	37.555640	126.910629	2015-09-07 10:32 20
마포구	124	124. 서교동 명문 건너편	서울특별시 마포구 서교로 16길 72	37.551114	126.930889	2015-09-07 10:33 20
마포구	125	125. 서교동 명문 앞	서울특별시 마포구 서교로 16길 72	37.551114	126.930889	2015-09-07 10:34 15
마포구	126	126. 서교동 명문 앞	서울특별시 마포구 서교로 16길 72	37.551114	126.930889	2015-09-07 10:34 20

서울특별시 일별 기상관측 정보 (2019-01-01 ~ 2020-06-30)

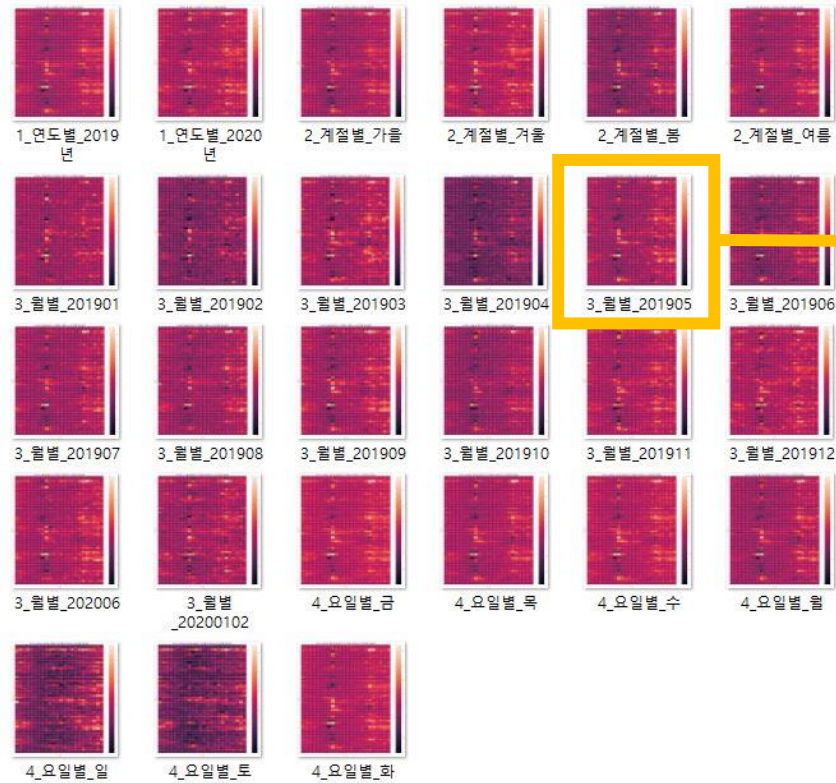
지점	지점명	일시	평균기온(°C)	최저기온(°C)	최고기온(°C)	일강수량(mm)	최대 풍속(m/s)	평균 상대습도(%)
1	108 서울	2019-01-01	-5	-8.2	-0.6		4.3	49.5
2	108 서울	2019-01-02	-4.9	-8.8	0.2		3.6	42.8
3	108 서울	2019-01-03	-3.5	-8.4	3.2		2.9	38.8
5	108 서울	2019-01-04	-1.1	-6.2	4.1		3	55.5
6	108 서울	2019-01-05	-2.8	-5.5	1.1		4.3	40.3
7	108 서울	2019-01-06	-2.8	-6.3	2.7		3.2	35
8	108 서울	2019-01-07	-1.9	-6.2	3.1		4.7	46.4
9	108 서울	2019-01-08	-3.5	-7.2	0.5		6.5	31
10	108 서울	2019-01-09	-4.1	-9.4	-0.3		2.8	28.8
11	108 서울	2019-01-10	-0.1	-5.5	5.3		4.8	48.8
12	108 서울	2019-01-11	2.4	-4.4	9.4		5.8	58.8
13	108 서울	2019-01-12	1.4	-5.5	8.5		4.7	47.6
14	108 서울	2019-01-13	1.4	-5.5	8.5		3.2	54.3
15	108 서울	2019-01-14	1.4	-5.5	8.5		2.5	68.6
16	108 서울	2019-01-15	-1.7	-7.2	2.8	0	6.1	58.8
17	108 서울	2019-01-16	-5.2	-10.1	-1.1	0	4.3	45.9
18	108 서울	2019-01-17	-0.3	-3.2	4	0	5	45.8
19	108 서울	2019-01-18	0.6	-4.6	6.7		4.5	44
20	108 서울	2019-01-19	3.6	0	8.5		3	59.8
21	108 서울	2019-01-20	-1.4	-5.5	4		5.5	38.5
22	108 서울	2019-01-21	-1.2	-6.7	4.2	0	4.3	57.8
23	108 서울	2019-01-22	1.7	-3.3	7.1		4.4	62
24	108 서울	2019-01-23	2	-0.8	7		5.1	48.9

* 2020년 데이터의 경우, 01월, 02월 절반, 06월의 데이터만 제공됨

* 664번 대여소의 경우, 2019년 11월 말에 설치되어 데이터 부족으로 제외함

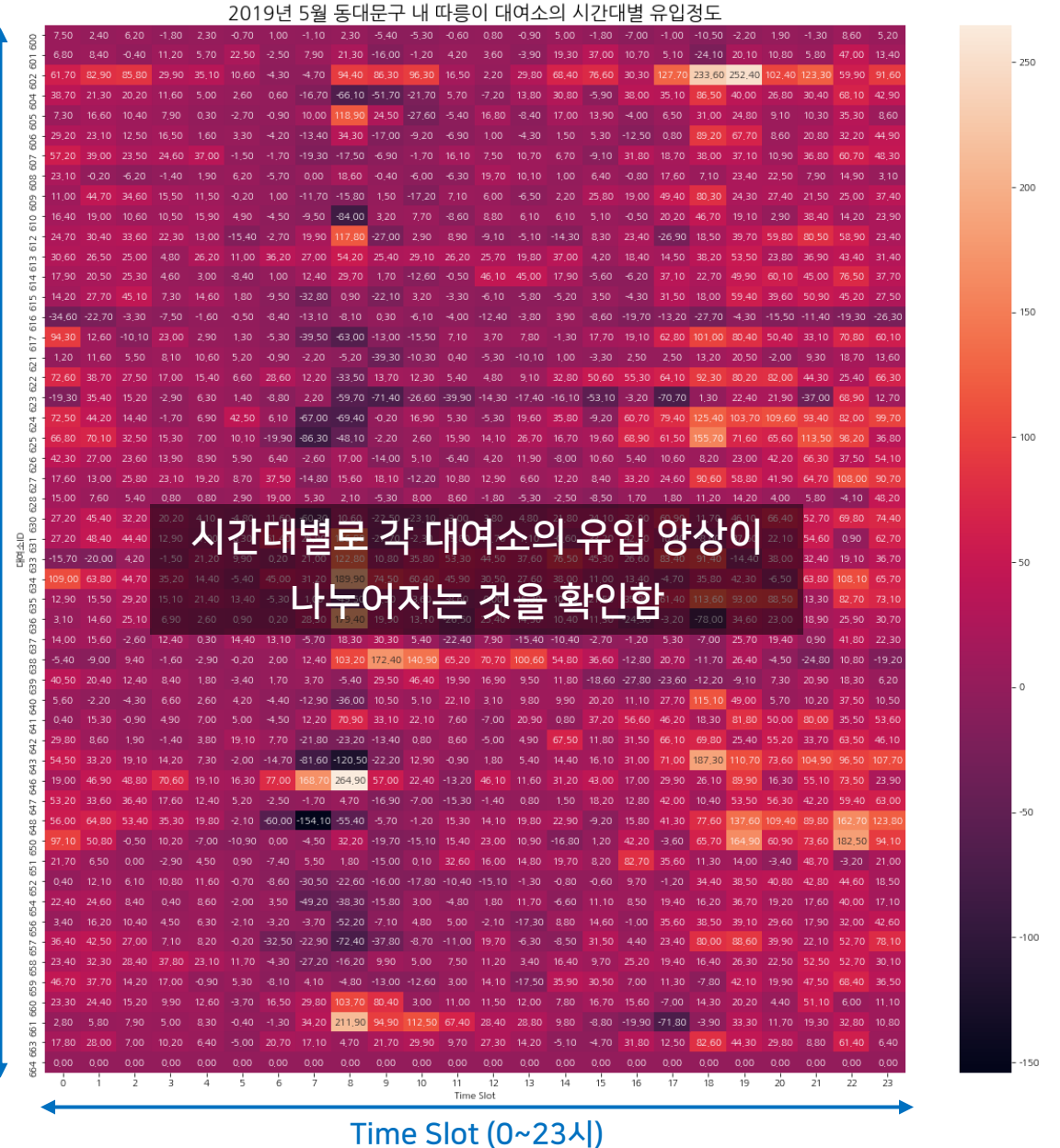
* 서울 열린데이터 광장(<http://data.seoul.go.kr/>), 기상자료개방포털(<https://data.kma.go.kr/>) 제공

분석 결과 - Heatmap



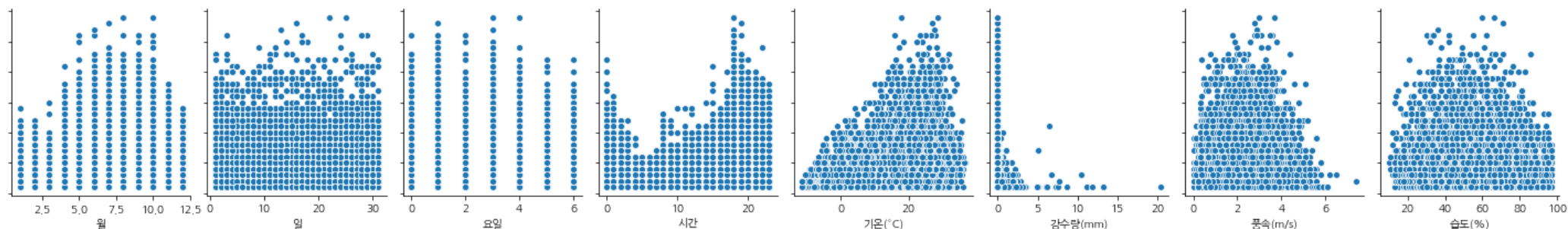
- Time Slot이 밝은 색일수록 유입 ↑
- Time Slot이 어두운 색일수록 유입 ↓

대여소 ID

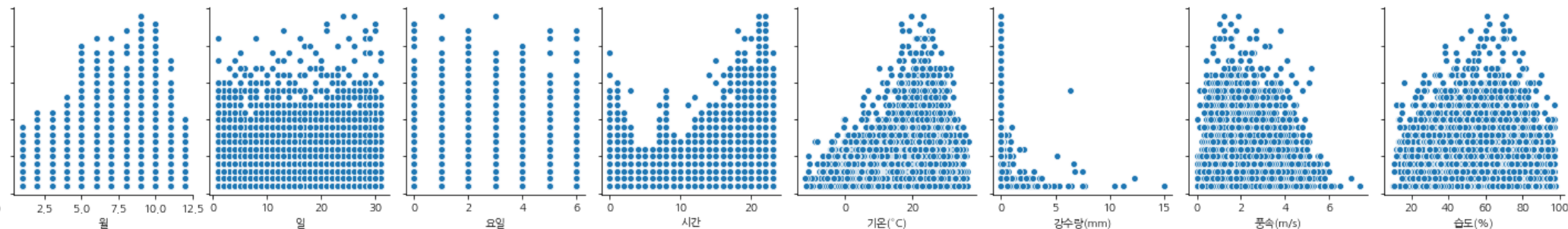


분석 결과 - Pair Plot

반납 건수
(Inflow)

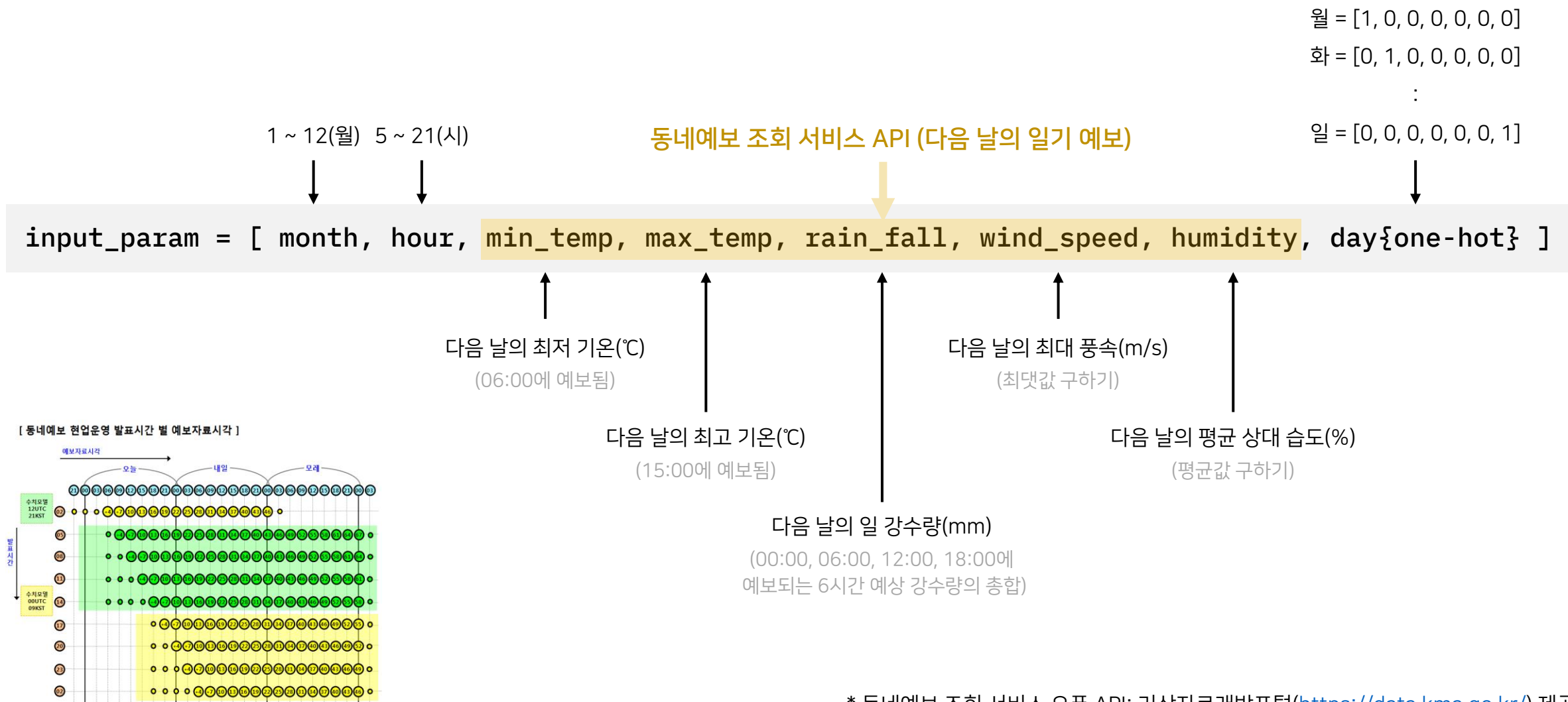


대여 건수
(Outflow)



Inflow & Outflow가 여러 파라미터들과 상관 관계가 있음을 대략적으로 파악함

모델의 Input Parameter



여러 Regression 모델의 성능 비교

{ Scikit-Learn의 linear_model 라이브러리에 포함된
Linear Regression 모델과 XGBoost 모델의 학습 }

```
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import GridSearchCV
from sklearn import linear_model as lm
import xgboost as xgb
from scipy.stats import uniform, randint

models = [
    ('lr', lm.LinearRegression(n_jobs=-1)),
    ('ridge', lm.Ridge()),
    ('lasso', lm.Lasso()),
    ('elastic', lm.ElasticNet()),
    ('LassoLars', lm.LassoLars()),
    ('LogisticRegression', lm.LogisticRegression()),
    ('SGDRegressor', lm.SGDRegressor()),
    ('Perceptron', lm.Perceptron(n_jobs=-1)),
    ('xgboost', xgb.XGBRegressor())
]
```

```
best_model, best_mae = None, float('inf')
for model_name, model in models:
    param_grid = params[model_name]
    grid = GridSearchCV(model, cv=5, n_jobs=-1, param_grid=param_grid)
    grid = grid.fit(x_train, y_train)

    model = grid.best_estimator_
    predictions = model.predict(x_test)
    mae = mean_absolute_error(y_test, predictions)

    print(model_name, mae)

    if mae < best_mae:
        best_model = model
```

{ 각 Regression 모델의 MAE(Mean Absolute Error) 비교 }

```
lr 2.5597978259333316
ridge 2.5597915740096853
lasso 2.565557232750446
elastic 2.562006247837826
LassoLars 2.5655572268419196
LogisticRegression 2.6759776536312847
SGDRegressor 2.989231202931475
Perceptron 3.6575418994413407
xgboost 1.7322861785828734
```

XGBoost 모델의 MAE 값이 가장 작았음

In [56]: best_model

Out[56]: XGBRegressor(gamma=0.24714164807668376, max_depth=5, n_estimators=119)

XGBoost Regression 학습

```
cols = ['월', '요일', '시간', '최저기온(°C)', '최고기온(°C)', '일강수량(mm)', '최대 풍속(m/s)', '평균 상대습도(%)']

Result_Inflow = {}
total = Bike_Inflow['반납대여소ID'].nunique()

for idx, (name, df) in enumerate(Bike_Inflow.groupby('반납대여소ID'), 1):
    print("[%d/%d] %s" % (idx, total, name), end=' ')

    try:
        df = df.groupby(cols)['반납건수'].sum().reset_index()
        df = df.join(pd.get_dummies(df['요일'], prefix="요일"))

        print(df.shape)

        x_train, x_test, y_train, y_test = train_test_split(df[features], df['반납건수'], test_size=0.2, random_state=42)

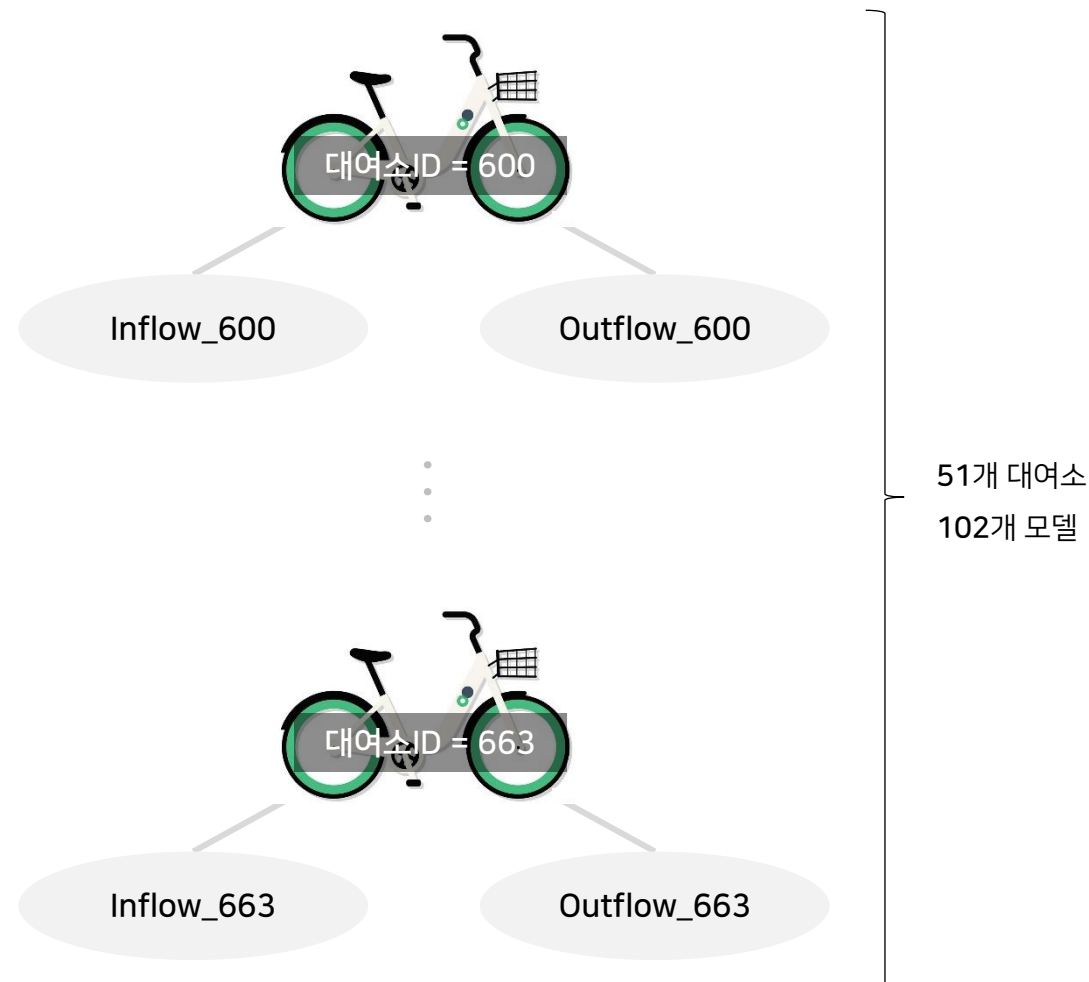
        param_grid = {
            "gamma": uniform(0, 0.5).rvs(n),
            "max_depth": range(2, 7), # default 3
            "n_estimators": randint(100, 150).rvs(n), # default 100
        }

        grid = GridSearchCV(xgb.XGBRegressor(objective='reg:squarederror'), cv=5, n_jobs=-1, param_grid=param_grid)
        grid = grid.fit(x_train, y_train)

        model = grid.best_estimator_
        predictions = model.predict(x_test)
        mae = mean_absolute_error(y_test, predictions)

        Result_Inflow[str(name)] = {}
        Result_Inflow[str(name)]['model'] = model
        Result_Inflow[str(name)]['mae'] = mae
        Result_Inflow[str(name)]['errors'] = predictions - y_test

    except:
        continue
```

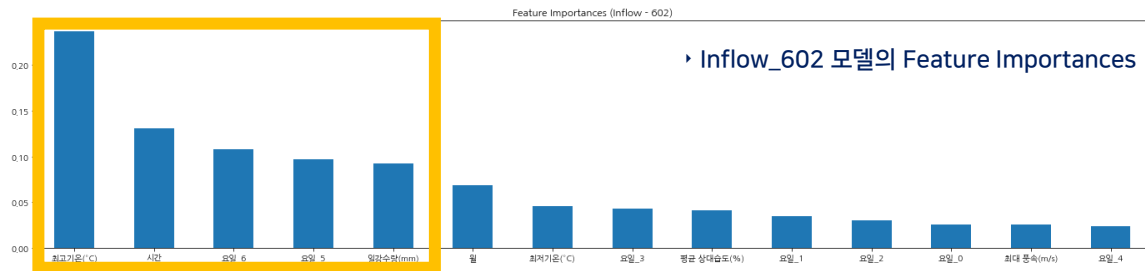
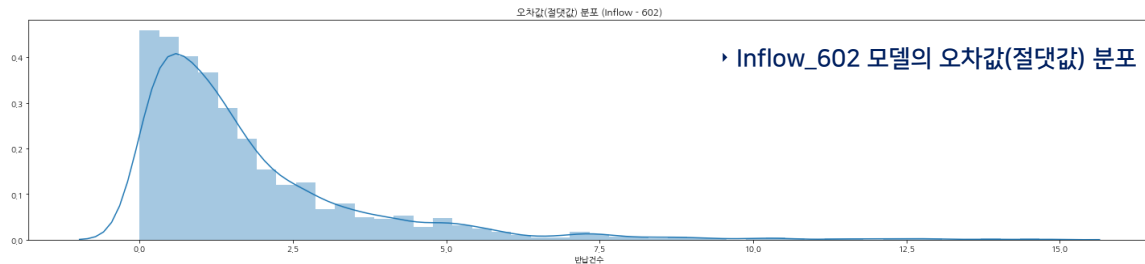


XGBoost Regression 학습 결과

{ Inflow 모델 }

```
In [53]: errors = np.array([result['mae'] for result in Result_Inflow.values()])
errors.mean()
```

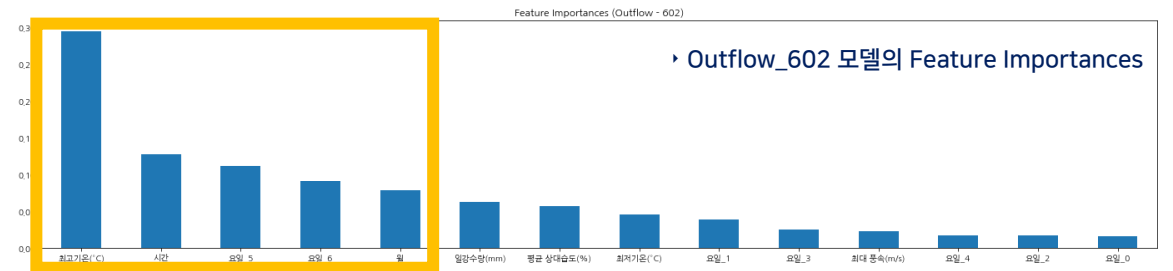
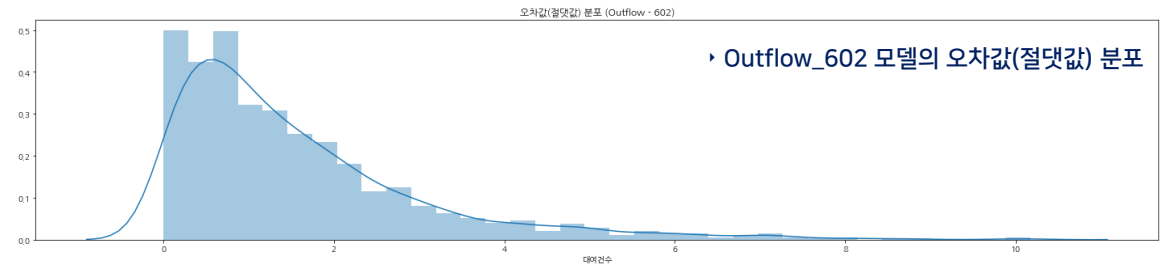
Out[53]: 1.0110311263420817 ▶ Inflow 모델 51개의 평균 MAE(Mean Absolute Error)



{ Outflow 모델 }

```
In [58]: errors = np.array([result['mae'] for result in Result_Outflow.values()])
errors.mean()
```

Out[58]: 1.0802892975824931 ▶ Outflow 모델 51개의 평균 MAE(Mean Absolute Error)



* 대여소의 단위 시간당 Inflow와 Outflow에 영향을 미치는 요소 TOP 5

: 최고 기온, 시간대, 요일, 일 강수량, 월

XGBoost Regression 예측 결과

예측된 Inflow & Outflow 값을 통해 계산한 다음 날의
유입 정도(Delta)를 Data Frame으로 저장

예측된 다음 날의 유입 정도에 따른 Heatmap 출력

```
In [47]: Result2 = pd.DataFrame(Result_All2['600'])

for id_stop in Bike_Stop_ID:
    if id_stop == 600:
        continue

tmppp = pd.DataFrame(Result_All2[str(id_stop)])

Result2 = pd.concat([Result2, tmppp])

Result2 = Result2.reset_index(drop = True)
Result2
```

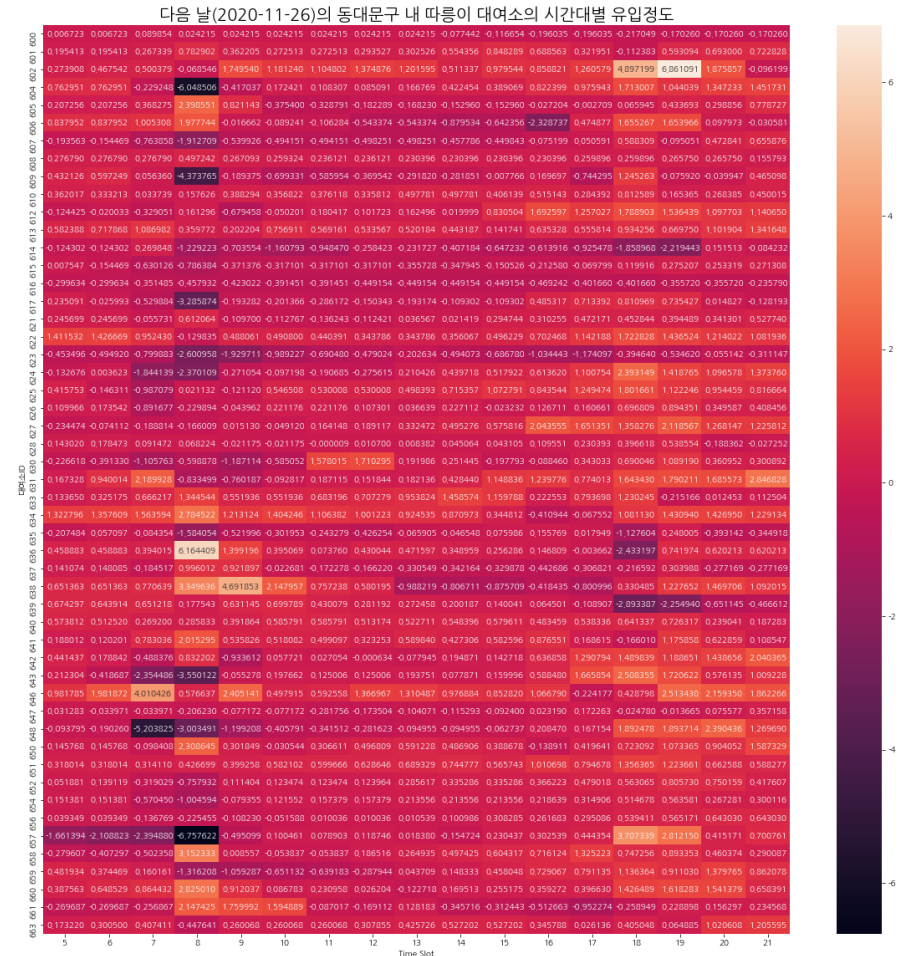
Out[47]:

	대여소ID	Time Slot	Inflow	Outflow	Delta
0	600	5	1.143319	1.337173	0.006723
1	600	6	1.143319	1.337173	0.006723
2	600	7	1.226451	1.337173	0.089854
3	600	8	1.226451	1.414396	0.024215

* 유입 정도(Delta) = $\alpha \times (\text{Inflow} - \text{Outflow}) + (1 - \alpha) \times \text{Inflow}$

이때, $\alpha = 0.85$

867 rows × 5 columns



스케줄 추천 알고리즘 설계 시 필요한 정보

- ▶ **Bike_Stop_Time** : 각 대여소 간 자동차로 걸리는 시간
 - ▶ 자동차 경로안내 API를 이용하여 51×51개의 경로에 대한 소요 시간을 구함
 - ▶ 소요 시간의 평균 값은 836.969초 ⇒ 13분 56.969초 ⇒ 약 15분
 - ▶ 각 대여소 간 소요 시간을 모두 동일하게 15분으로 가정함

- ▶ **top_13_sorted** : 유입 정도의 합이 큰 TOP 13 대여소의 ID
- ▶ **desired_order** : 각 대여소마다 유입 정도가 큰 순서로 (Time Slot, 유입 정도) 정렬
- ▶ **largest_stop** : 각 시간대마다 그 시간대가 제일 유입 정도가 큰 대여소의 ID
- ▶ **visit_count** : 각 대여소를 방문한 횟수

```
In [16]: temp3 = []  
         for k in temp2.values():  
             temp3 += [k]  
  
         avg_time = np.mean(np.array(temp3))  
         avg_time
```

Out[16]: 836.9686274509804

```
In [15]: print(datetime.timedelta(0, avg_time))
```

0:13:56.968627

스케줄 추천 알고리즘

▶ 가정

- ▶ 방역에 참여하는 팀은 2개 (A팀, B팀)
- ▶ 대여소간 소요 시간 = 15분, 대여소별 소독 시간 = 15분
 - ▶ 한 팀은 한 시간마다 2개의 대여소 소독 가능 \Rightarrow 두 팀은 총 4개의 대여소 소독 가능
- ▶ 소독은 이전 Time Slot의 유입 정도를 토대로 수행함
 - ▶ ex) Time Slot 8(08~09시)에 유입 정도가 큰 대여소는 Time Slot 9(09~10시)에 소독함
 - ▶ 유입 정도 예측은 5~21시에 대해서, 방역 스케줄 추천은 6~22시에 대해서 이루어짐
- ▶ TOP 13 대여소는 한 번 더 소독함

▶ 각 Time Slot에 소독할 대여소 배정 순서 (**visit_order**)

1. 직전 Time Slot에서 제일 유입 정도가 컸던 대여소들을 배정함 (최대 4개)
2. 제일 유입 정도가 컸던 대여소의 개수가 4개가 넘는 Time Slot의 경우, 그 다음 Time Slot으로 가득 채워 배정함
3. Time Slot 21의 배정이 끝난 경우, 방문 횟수(**visit_count**)가 0인 대여소와 TOP 13 대여소를 무작위로 남은 Time Slot에 배정함
4. A팀에는 각 Time Slot의 홀수 번째 대여소, B팀에는 각 Time Slot의 짝수 번째 대여소를 배정함

```
In [272]: visit_order
Out[272]: {'6': ['616', '612', '622', '604'],
          '7': ['642', '651', '609', '610'],
          '8': ['646', '631', '614', '600'],
          '9': ['636', '658', '660', '634'],
          '10': ['638', '605', '650', '661'],
          '11': ['639', '641', '606', '637'],
          '12': ['621', '608', '617', '640'],
          '13': ['630', '656', '607', '654'],
          '14': ['628', '615', '635', '623'],
          '15': ['633', '602', '646', '634'],
          '16': ['601', '622', '638', '631'],
          '17': ['660', '651', '613', '627'],
          '18': ['647', '633', '636', '625'],
          '19': ['657', '643', '624', '625'],
          '20': ['602', '627', '626', '652'],
          '21': ['648', '659', '613', '663']}
```

```
In [274]: # team_A와 team_B에게 배분
team_A = {}
team_B = {}

for t in range(6, 22):
    team_A[str(t)] = []
    team_B[str(t)] = []
    team_A[str(t)].append(visit_order[str(t)][0])
    team_A[str(t)].append(visit_order[str(t)][2])
    team_B[str(t)].append(visit_order[str(t)][1])
    team_B[str(t)].append(visit_order[str(t)][3])

print(team_A)
print(team_B)

{'6': ['616', '622'], '7': ['642', '609'], '8': ['646', '614'],
'9': ['636', '660'], '10': ['638', '650'], '11': ['639', '606'],
'12': ['621', '617'], '13': ['630', '607'], '14': ['628', '635'],
'15': ['633', '646'], '16': ['601', '638'], '17': ['660', '613'],
'18': ['647', '636'], '19': ['657', '624'], '20': ['602', '626'],
'21': ['648', '613']}

{'6': ['612', '604'], '7': ['651', '610'], '8': ['631', '600'],
'9': ['658', '634'], '10': ['605', '661'], '11': ['641', '637'],
'12': ['608', '640'], '13': ['656', '654'], '14': ['615', '623'],
'15': ['602', '634'], '16': ['622', '631'], '17': ['651', '627'],
'18': ['633', '625'], '19': ['643', '625'], '20': ['627', '652'],
'21': ['659', '663']}
```


효용성 검증 - 추천 스케줄과 랜덤 스케줄의 비교

▶ 예상 유효 소독 대수

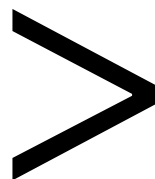
- ▶ $\sum \{(\text{각 대여소의 기존 거치대수}) + (\text{각 대여소가 소독되기 직전 Time Slot까지의 (Inflow - Outflow) 값의 합})\} = \sum (\text{disinfected}[\text{stop_id}])$
- ▶ 이때, $\text{disinfected}[\text{stop_id}] < 0$ 인 경우, 0으로 처리함
- ▶ 추천된 방역 스케줄과 랜덤으로 짜여진 방역 스케줄의 효용성을 비교하는 척도

▶ Case ① :: 2020-11-26-목

```
input_param = [ 11, {hour}, 3.0, 10.0, 0.0, 1.7, 67.5, 3{one-hot} ]
```

<추천 방역 스케줄>

추천 방역 스케줄	예상 유효 소독 대수
※ {'시간대': ['대여소ID-1', '대여소ID-2'], ...} 형식입니다.	
> A 팀의 추천 방역 스케줄은 다음과 같습니다.	
{6: [616, 622], 7: [642, 609], 8: [646, 614], 9: [636, 660], 10: [638, 650], 11: [639, 606], 12: [621, 617], 13: [630, 607], 14: [628, 635], 15: [633, 646], 16: [601, 638], 17: [660, 613], 18: [647, 636], 19: [657, 624], 20: [602, 626], 21: [648, 613]}	약 629.29 대
> B 팀의 추천 방역 스케줄은 다음과 같습니다.	
{6: [612, 604], 7: [651, 610], 8: [631, 600], 9: [658, 634], 10: [605, 661], 11: [641, 637], 12: [608, 640], 13: [656, 654], 14: [615, 623], 15: [602, 634], 16: [622, 631], 17: [651, 627], 18: [633, 625], 19: [643, 625], 20: [627, 652], 21: [659, 663]}	629.29대



<랜덤 방역 스케줄>

랜덤 방역 스케줄	예상 유효 소독 대수
In [80]: disinfected_day_ran = 0 for stop_id in Bike_Stop_ID: #print(disinfected_ran[str(stop_id)]) if disinfected_ran[str(stop_id)] > 0: disinfected_day_ran += disinfected_ran[str(stop_id)] print("랜덤 경로의 하루 예상 유효 소독 대수: ", disinfected_day_ran)	588.29대
In [88]: disinfected_day_ran = 0 for stop_id in Bike_Stop_ID: #print(disinfected_ran[str(stop_id)]) if disinfected_ran[str(stop_id)] > 0: disinfected_day_ran += disinfected_ran[str(stop_id)] print("랜덤 경로의 하루 예상 유효 소독 대수: ", disinfected_day_ran)	581.83대

효용성 검증 - 추천 스케줄과 랜덤 스케줄의 비교

▶ Case ② :: 2020-08-14-금

```
input_param = [8, {hour}, 26.5, 28.0, 0.4, 3.5, 86.1, 4{one-hot} ]
```

<추천 방역 스케줄>

추천 방역 스케줄

※ {시간대}: ['대여소ID-1', '대여소ID-2', ...] 형식입니다.

> A 팀의 추천 방역 스케줄은 다음과 같습니다.

```
{'6': ['643', '604'], '7': ['650', '651'], '8': ['646', '600'], '9': ['636', '606'], '10': ['638', '633'], '11': ['602', '628'], '12': ['663', '656'], '13': ['627', '654'], '14': ['608', '641'], '15': ['631', '609'], '16': ['625', '626'], '17': ['659', '658'], '18': ['614', '657'], '19': ['625', '657'], '20': ['631', '630'], '21': ['648', '626']}
```

> B 팀의 추천 방역 스케줄은 다음과 같습니다.

```
{'6': ['609', '642'], '7': ['652', '617'], '8': ['613', '616'], '9': ['661', '660'], '10': ['605', '637'], '11': ['639', '640'], '12': ['615', '610'], '13': ['607', '623'], '14': ['621', '634'], '15': ['624', '646'], '16': ['660', '643'], '17': ['601', '622'], '18': ['647', '635'], '19': ['624', '634'], '20': ['635', '622'], '21': ['641', '612']}
```

예상 유효 소독 대수

약 619.4대

619.4대

<랜덤 방역 스케줄>

```
In [119]: disinfect_day_ran = 0
for stop_id in Bike_Stop_ID:
    #print(disinfect_ran[str(stop_id)])
    if disinfect_ran[str(stop_id)] > 0:
        disinfect_day_ran += disinfect_ran[str(stop_id)]
print("랜덤 경로의 하루 예상 유효 소독 대수: ", disinfect_day_ran)
```

랜덤 경로의 하루 예상 유효 소독 대수: 581.2920391559601

581.29대

```
In [127]: disinfect_day_ran = 0
for stop_id in Bike_Stop_ID:
    #print(disinfect_ran[str(stop_id)])
    if disinfect_ran[str(stop_id)] > 0:
        disinfect_day_ran += disinfect_ran[str(stop_id)]
print("랜덤 경로의 하루 예상 유효 소독 대수: ", disinfect_day_ran)
```

랜덤 경로의 하루 예상 유효 소독 대수: 569.8832812309265

569.88대

효용성 검증 - 추천 스케줄과 랜덤 스케줄의 비교

▶ Case ③ :: 2020-10-11-일

```
input_param = [10, {hour}, 12.7, 20.8, 0.0, 1.8, 68.5, 6{one-hot} ]
```

<추천 방역 스케줄>

추천 방역 스케줄

※ {'시간대': ['대여소ID-1', '대여소ID-2', ...]} 형식입니다.

> A 팀의 추천 방역 스케줄은 다음과 같습니다.

```
{'6': [624, 654], '7': [622, 651], '8': [646, 600], '9': [636, 606], '10': [638, 658], '11': [639, 608], '12': [631, 657], '13': [630, 614], '14': [628, 615], '15': [663, 634], '16': [602, 641], '17': [660, 651], '18': [643, 640], '19': [612, 609], '20': [634, 627], '21': [648, 660]}
```

> B 팀의 추천 방역 스케줄은 다음과 같습니다.

```
{'6': [623, 616], '7': [642, 656], '8': [613, 652], '9': [661, 641], '10': [637, 625], '11': [605, 626], '12': [601, 607], '13': [610, 647], '14': [621, 602], '15': [622, 648], '16': [646, 630], '17': [661, 627], '18': [650, 613], '19': [604, 633], '20': [617, 635], '21': [663, 659]}
```

예상 유효 소득 대수

약 600.23 대

600.23대

<랜덤 방역 스케줄>

```
In [135]: disinfect_day_ran = 0
for stop_id in Bike_Stop_ID:
    #print(disinfect_ran[str(stop_id)])
    if disinfect_ran[str(stop_id)] > 0:
        disinfect_day_ran += disinfect_ran[str(stop_id)]
print("랜덤 경로의 하루 예상 유효 소득 대수: ", disinfect_day_ran)
```

랜덤 경로의 하루 예상 유효 소득 대수: 565.515043258667

565.52대

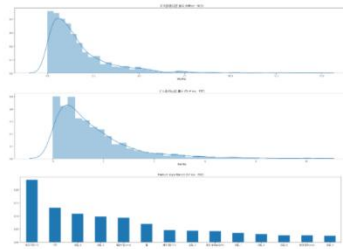
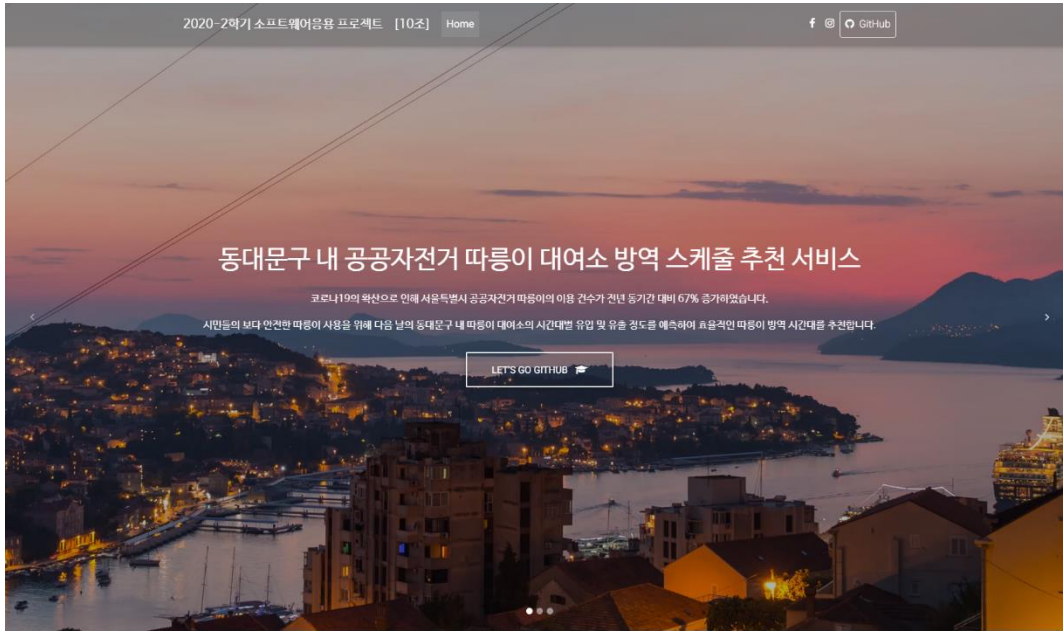
```
In [143]: disinfect_day_ran = 0
for stop_id in Bike_Stop_ID:
    #print(disinfect_ran[str(stop_id)])
    if disinfect_ran[str(stop_id)] > 0:
        disinfect_day_ran += disinfect_ran[str(stop_id)]
print("랜덤 경로의 하루 예상 유효 소득 대수: ", disinfect_day_ran)
```

랜덤 경로의 하루 예상 유효 소득 대수: 577.5639467835426

577.56대

I. 주제 선정 배경 II. 데이터 분석 III. 머신 러닝 IV. 스케줄 추천 알고리즘 설계 V. 웹 서비스 구축 VI. 고찰 VII. 참고 자료

Flask를 활용한 웹 서비스 구축



따릉이 대여소의 시간대별 유입 및 유출 정도 예측 모델

XGBoost Regression 모델을 활용하여 동대문구 내 각 따릉이 대여소의 시간대별 유입 및 유출 정도를 예측하고 학습 결과를 확인했습니다.

2019-01-01 ~ 2020-06-30 기간 동안의 서울특별시 공공자전거 대여정보 이력과 서울특별시 일별 기상관측 자료를 활용하여 평균 MAE 1.045 정도의 오차를 보이는 예측 모델을 개발했습니다.

학습된 모델은 다음 날의 월, 요일, 기온, 강수량, 풍속, 습도를 입력받아 각 대여소의 시간대별 유입 및 유출 정도를 예측합니다.

모델에 의해 예측된 다음 날 05 ~ 21시의 유입 및 유출 정도를 토대로 다음 날 06 ~ 22시의 방역 스케줄을 추천합니다. 두 개의 팀이 방역에 참여한다고 가정합니다.

프로젝트 제출물 다운로드

API로 불러온 다음 날의 기상 정보

파라미터 입력
(사용자 마음대로 입력 가능)

* 무료 템플릿: MDBootstrap (<https://mdbootstrap.com/freebies/>) 제공

I. 주제 선정 배경 II. 데이터 분석 III. 머신 러닝 IV. 스케줄 추천 알고리즘 설계 V. 웹 서비스 구축 VI. 고찰 VII. 참고 자료

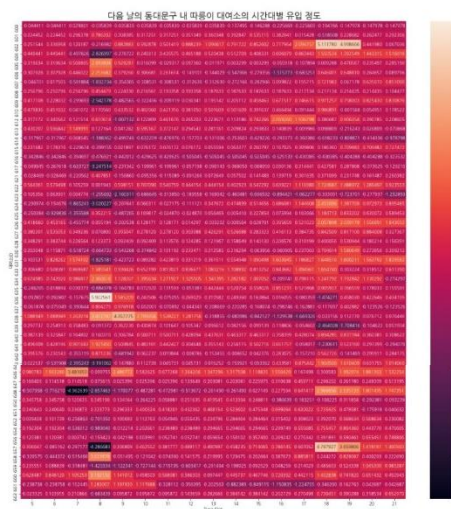
Flask를 활용한 웹 서비스 구축

2020-2학기 소프트웨어융합 프로젝트 [10조] | Home

f @ GitHub

예상 유입 정도 Heatmap

※ 범은 해당수독 유입 정도가 큰 시간대입니다.



예상 유입 정도 Heatmap

추천 방역 스케줄

예상 유효 소독 대수

※ [시간대]: [대여소ID-1', '대여소ID-Z'], ...] 형식입니다.

> A 팀의 추천 방역 스케줄은 다음과 같습니다.

[6: [623, 643], 7: [633, 625], 8: [646, 614], 9: [636, 658], 10: [638, 652], 11: [639, 661], 12: [621, 621], 13: [635, 651], 14: [650, 652], 15: [617, 602], 16: [635, 651], 17: [612, 657], 18: [647, 655], 19: [602, 648], 20: [631, 654], 21: [659, 656]]

> B 팀의 추천 방역 스케줄은 다음과 같습니다.

[6: [616, 622], 7: [604, 610], 8: [613, 600], 9: [660, 605], 10: [637, 606], 11: [641, 642], 12: [608, 609], 13: [640, 663], 14: [607, 628], 15: [646, 654], 16: [660, 658], 17: [622, 626], 18: [612, 627], 19: [657, 624], 20: [630, 615], 21: [627, 626]]

약 632.29 대

추천 방역 스케줄



예상 유효 소독 대수

LET'S GO GITHUB

© 2020 Copyright 유승민

고찰 및 확장 가능성

- ▶ 파라미터를 추가하여 더 많은 횟수의 실험을 통해 최적의 유입 정도를 계산하는 식을 도출해낸다면 추천 방역 스케줄의 정확도를 향상시킬 수 있을 것이다.
- ▶ 51개의 대여소 간 소요 시간의 평균값이 아닌 51×51개의 소요 시간, 각 대여소별 유입 정도가 큰 시간대, 유입 정도의 크기를 모두 고려하여 방역 스케줄을 도출해내도록 알고리즘을 개선한다면 대략적인 시간(hour) 단위가 아닌 minute 단위의 스케줄 추천이 가능할 것이다.
- ▶ 코로나19 이후 따릉이 이용 건수는 전년 동기간 대비 67% 증가했으나, 공공데이터포털에는 2020년 1월, 2월 절반, 6월의 따릉이 대여이력 데이터만 업로드 되어 있다. 따라서 증가된 이용 건수를 학습 모델에 반영하기에는 데이터가 부족했을 것으로 예상된다.
- ▶ 본 서비스에서 머신 러닝 모델을 만들 때, 여러 종류의 Regression 모델 중에서 실험 결과 가장 성능이 좋았던 XGBoost를 선택했으나 여러 이상치까지 예측을 할 수 있도록 LSTM 등의 딥러닝 모델로도 학습을 진행하여 성능 비교를 해볼 수 있을 것이다.
- ▶ 보다 정확한 다음 날의 일기 예보를 토대로 각 대여소의 시간대별 유입 정도와 유출 정도를 예측할 수 있으므로 따릉이 재배치 솔루션 혹은 따릉이 관련 데이터 연구에 적용될 수 있을 것으로 보인다. 또한, 동대문구와 따릉이 이용량이 많은 마포구, 영등포구, 송파구 뿐만 아니라 서울시 내 모든 구에 대해서도 확장이 가능할 것이다.

참고 자료

- ▶ 서울시설공단 공공자전거 운영처, 『코로나19 극복 희망일자리사업』 공공자전거 따릉이 방역단 운영 계획
- ▶ 기상청, 동네예보 조회서비스 Open API 활용가이드
- ▶ 서울 열린데이터 광장 - <http://data.seoul.go.kr/>
- ▶ 기상청 - <https://www.weather.go.kr/>
- ▶ 기상자료개방포털 - <https://data.kma.go.kr/>
- ▶ tmap API - <http://tmapapi.sktelecom.com/>
- ▶ MDBootstrap - <https://mdbootstrap.com/freebies/>
- ▶ All about 따릉이 EDA - <https://dailyheumsi.tistory.com/>
- ▶ 다변인 선형회귀를 활용한 배추 가격 예측 AI 개발하기 - <https://ndb796.tistory.com/> , <https://github.com/ndb796/Vegita/>

감사합니다. 