데이터 과학

- [과제 1] 타슈 데이터 분석하기 -

제 출 일	2019.05.21.
분 반	00반
학 과	컴퓨터공학과
학 번	201402323
이 름	김 낙 현

1. 과제 목표

(1) 2013년도, 2014년도, 2015년도 타슈 데이터와 기상데이터 가공하기

(2) 탐색적 분석

- 계절, 요일, 월, 시간 별 대여량 막대그래프 작성
- 인기 경로 Top10 Google Map에 표시
- 이용 경로 Chord 다이어그램 그리기

(3) 2015년 대여량 실제 데이터와 예측

- 예측 모델 각 요소 영향도 그래프 그리기
- 2015/1/1 3번 정류장으로 예측 결과 평가

2. 진행 과정

2-1. 타슈 데이터와 기상데이터 가공하기

(1) 타슈 데이터, 기상 데이터 준비

- 공공데이터 포털과 기상자료개방포털에서 데이터 다운로드

(2) 타슈 데이터 날짜 형식 맞추기

- Libre office를 이용하여 2013, 2014, 2015년도 날짜 데이터 형식을 하나로 맞춰준다. (아래 예시처럼 날짜 데이터 형식을 바꿔준다.)

대여일시
20130101055603

(3) 2013~2015년도 데이터 합치기

- Terminal의 명령어인 cat을 이용하여 2013~2015년도 타슈 데이터를 tashu.csv 로 합쳐준다.
- 같은 방법으로 2013~2015년도 기상 데이터도 합쳐준다.

(4) 2013~2015 시간별 정류장 날씨 및 대여수 데이터 만들기

- 먼저 필요한 library 들을 import 시켜준다.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import gmplot
import statistics

from pandas import DataFrame
```

- 정류장 번호 데이터와 기상데이터를 join 시켜준다. (train_rent)

	STATION	YEAR	MONTH	DAY	HOUR	WEEKDAY	SEASON	TEMPERATURE	RAINFALL	WINDSPEED	HUMLDITY	SNOWFALL	RENTCOUNT
0	1	2013	1	1	0	2	3	-8.8	NaN	0.1	90.0	8.8	0
1	1	2013	1	1	1	2	3	-8.5	NaN	0.9	90.0	8.8	0
2	1	2013	1	1	2	2	3	-8.5	NaN	1.0	89.0	8.8	0
3	1	2013	1	1	3	2	3	-9.0	NaN	0.7	91.0	8.8	0
4	1	2013	1	1	4	2	3	-9.1	NaN	0.6	92.0	8.8	0
5	1	2013	1	1	5	2	3	-9.4	NaN	0.5	92.0	8.8	0
6	1	2013	1	1	6	2	3	-9.0	NaN	1.4	93.0	8.8	0

- tashu.csv 파일도 다음과 같이 준비한다.

	RENT_STATION	RENT_YEAR	RENT_MONTH	RENT_DAY	RENT_HOUR	RETURN_STATION	RETURN_YEAR	RETURN_MONTH	RETURN_DAY	RETURN_HOU
0	43.0	2013	1	1	5	34.0	2013.0	1.0	1.0	
1	97.0	2013	1	1	6	NaN	2013.0	1.0	1.0	1(
2	2.0	2013	1	1	6	10.0	2013.0	1.0	1.0	
3	106.0	2013	1	1	10	105.0	2013.0	1.0	1.0	1(
4	4.0	2013	1	1	11	4.0	2013.0	1.0	1.0	1:
5	21.0	2013	1	1	11	105.0	2013.0	1.0	1.0	1
6	90.0	2013	1	1	12	91.0	2013.0	1.0	1.0	1:

- 데이터파일에서 필요없는 column들을 제거해준다.

```
rent_clean = train_rent.drop(columns=['WEEKDAY', 'SEASON', 'TEMPERATURE','RAINFALL',
                                      'WINDSPEED','HUMLDITY','SNOWFALL'])
# join 하기 위해 column명 변경
rent = tashu.rename(columns = {'RENT_STATION': 'STATION', 'RENT_YEAR': 'YEAR',
                               'RENT_MONTH': 'MONTH', 'RENT_DAY': 'DAY', 'RENT_HOUR': 'HOUR'})
```

- 제거 된 타슈 대여 데이터와 기상데이터를 join 해준다.

```
rent_join = pd.merge(rent_clean, rent, how='outer')
```

```
rent_join
```

	STATION	YEAR	MONTH	DAY	HOUR	RENTCOUNT	RETURN_STATION	RETURN_YEAR	RETUR
0	1.0	2013	1	1	0	0.0	NaN	NaN	
1	1.0	2013	1	1	1	0.0	NaN	NaN	
2	1.0	2013	1	1	2	0.0	NaN	NaN	
3	1.0	2013	1	1	3	0.0	NaN	NaN	
4	1.0	2013	1	1	4	0.0	NaN	NaN	
5	1.0	2013	1	1	5	0.0	NaN	NaN	
6	1.0	2013	1	1	6	0.0	NaN	NaN	
7	1.0	2013	1	1	7	0.0	NaN	NaN	
8	1.0	2013	1	1	8	0.0	NaN	NaN	
9	1.0	2013	1	1	9	0.0	NaN	NaN	
10	1.0	2013	1	1	10	0.0	NaN	NaN	
11	1.0	2013	1	1	11	0.0	NaN	NaN	
12	1.0	2013	1	1	12	0.0	NaN	NaN	
13	1.0	2013	1	1	13	0.0	1.0	2013.0	

- NaN 데이터가 있는 row를 dropna() 사용하여 제거한다.

- grouby를 사용하여 RENTCOUNT를 제외한 모든 칼럼들로 그룹화한다.
  그러면 RENTCOUNT가 계산되어서 temp에 저장이 된다.

```
temp = rent_join.groupby(['STATION','YEAR','MONTH','DAY','HOUR']).size().to_frame('RENTCOUNT').reset_index()
```

- temp와 train_rent를 outer join을 이용하여 합쳐주고 RENTCOUNT에 있는 NaN 데이터 값을 모두 0.0 으로 바꿔준다.

	STATION	YEAR	MONTH	DAY	HOUR	WEEKDAY	SEASON	TEMPERATURE	RAINFALL	WINDSPEED	HUMLDITY	SNOWFALL	RENTCOUNT
0	1	2013	1	1	0	2	3	-8.8	NaN	0.1	90.0	8.8	0.0
1	1	2013	1	1	1	2	3	-8.5	NaN	0.9	90.0	8.8	0.0
2	1	2013	1	1	2	2	3	-8.5	NaN	1.0	89.0	8.8	0.0
3	1	2013	1	1	3	2	3	-9.0	NaN	0.7	91.0	8.8	0.0
4	1	2013	1	1	4	2	3	-9.1	NaN	0.6	92.0	8.8	0.0
5	1	2013	1	1	5	2	3	-9.4	NaN	0.5	92.0	8.8	0.0

- 위의 데이터파일을 total_rent.csv 로 저장한다.

(5) 2013~2015 시간별 정류장 날씨 및 반납수 데이터 만들기

- (4)의 진행과정과 동일
- total_return.csv로 저장된다.

2-2. 탐색적 분석

(1) 계절, 요일, 월, 시간 별 대여량 막대그래프 작성

- 필요한 Library를 import 시켜준다.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import gmplot
import statistics

from pandas import DataFrame
```

- total_rent.csv 파일을 불어와 train_rent에 저장한다.

- 계절 별로 그룹화 시켜 RENTCOUNT를 계산하고 x 축을 계절, y 축을 대여량으로 하여 막대그래프를 작성한다.

```
#1-a season graph
group_season = train_rent.groupby(['SEASON'])['RENTCOUNT'].sum().reset_index()
ax = sns.barplot(x=group_season['SEASON'], y=group_season['RENTCOUNT'])
ax.set(xlabel='Season', ylabel='Rent count')
season = ['Spring', 'Summer', 'Fall', 'Winter']
plt.xticks(np.arange(4), season)
plt.title('The number of rented bike by season')
plt.show()
```

- 요일, 월, 시간 별 대여량 막대그래프는 그룹화 방법만 다르게 하여 막대그래프를 작성한다.

```
#1-a weekday graph
group_weekday = train_rent.groupby(['WEEKDAY'])['RENTCOUNT'].sum().reset_index()
```

```
#1-a month graph
group_mn = train_rent.groupby(['MONTH'])['RENTCOUNT'].sum().reset_index()
```

```
#1-a hour graph
group_hr = train_rent.groupby(['HOUR'])['RENTCOUNT'].sum().reset_index()
```

(2) 인기 경로 Top10 Google Map에 표시

- tashu.csv 파일을 불러와 rent에 저장한다.
- rent에서 정류장 번호가 145이상인 데이터를 모두 지워준다.

```
rent = rent[rent.RENT_STATION < 145]
rent = rent[rent.RETURN_STATION < 145]
```

- rent 데이터를 RENT_STATION, RETURN_STATION으로 그룹화 시켜준다.

	RENT_STATION	RETURN_STATION	0
0	3.0	3.0	84496
1	31.0	31.0	21749
2	56.0	56.0	18343
3	21.0	105.0	17220

- 위에 나온 것처럼 대여장소에서 반납한 장소까지의 경로를 알 수 있다.
  이 데이터들 중 가장 값이 큰 10개를 뽑아서 top10에 저장시킨다.
  이 데이터가 인기경로 top10 데이터가 된다.

- Google Map에 표시를 하기 위해서는 각 정류장마다 위도와 경도를 알아야 하므로 반복문을 이용하여 대여장소의 위도와 경도, 반납장소의 위도와 경도를 station.csv에서 찾아 각각의 list에 저장한다.

```
for j in range(0,10):
    rent_station = int(top10.iloc[j]['RENT_STATION'])
    rt_station = int(top10.iloc[j]['RETURN_STATION'])
    for i in station_df.번호:
        if i == rent_station:
            temp = station_df.iloc[i-1][7].split(',')
            rent_top10_lat.append(float(temp[0]))
            rent_top10_lon.append(float(temp[1]))
        if i == rt_station:
            temp = station_df.iloc[i-1][7].split(',')
            rt_top10_lat.append(float(temp[0]))
            rt_top10_lon.append(float(temp[1]))
```

- 인기 대여, 반납 장소를 지도에 표시하고 경로 또한 지도에 표시 해준다.

```
gmapl = gmplot.GoogleMapPlotter(statistics.median(rent_top10_lat),
                                statistics.median(rent_top10_lon), 14)
gmapl.scatter(rent_top10_lat, rent_top10_lon, '#FF0000',size=30, marker=False)
gmapl.scatter(rt_top10_lat, rt_top10_lon, '#FF0000',size=30, marker=False)

for i in range(0,10):
    top10_lat = [rent_top10_lat[i], rt_top10_lat[i]]
    top10_lon = [rent_top10_lon[i], rt_top10_lon[i]]
    gmapl.plot(top10_lat, top10_lon, 'cornflowerblue', edge_width = 3.0)

gmapl.draw('top10.html')
```

- (추가)대여 지점으로 돌아오는 경로를 제외한 top10을 그리기 위해서 위 있던 데이터들 중 대여장소와 반납장소가 같은 데이터들을 모두 지워주고 top10_bonus에 Rent_station과 Return_station에 대한 값들을 저장시킨 후 위의 방법처럼 지도에 표시한다.

(3) 이용 경로 Chord 다이어그램 그리기 (https://github.com/fengwangPhysics/matplotlib-chord-diagram/blob/master/matplotlib-chord.py) 사이트를 참고하여 코드 작성

- 필요한 Library를 import 한다.

```
####################
# chord diagram
import matplotlib.pyplot as plt
from matplotlib.path import Path
import matplotlib.patches as patches
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.font_manager as fm
####################################
```

- 출처 사이트에는 colors의 개수가 적으므로 정류장 수만큼 color 개수를 추가해준다.

```
if colors is None:
# use d3.js category10 https://github.com/d3/d3-3.x-api-reference/bl
    colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd',
              '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf',
              '#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd',
              '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf',
              '#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd',
              '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf',
              '#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd',
              '#8c564b', '#e377c2', '#7f7f7f', '#bcbd22', '#17becf',
```

- 이동 경로 데이터를 2차원 배열에 저장시킨다.

```
for i in range(18295):
    x = rent.iloc[i]['RENT_STATION']
    y = rent.iloc[i]['RETURN_STATION']
    arr[x-1][y-1] = rent.iloc[i][0]
```

- plt에서 한글이 깨지므로 font도 새로 설정해준다.

```
path = '/usr/share/fonts/truetype/nanum/NanumBarunGothic.ttf'
font_name = fm.FontProperties(fname=path, size=20).get_name()
plt.rc('font', family=font_name)
plt.style.use('ggplot')
```

- node를 정류장 명칭으로 바꿔주고, link를 이동 경로 데이터 2차원 배열로 바꿔준다. arr에는 이동 경로 데이터가 저장되어 있다.

```
fig = plt.figure(figsize=(10,10))
flux = arr
ax = plt.axes([0,0,1,1])
#nodePos = chordDiagram(flux, ax, colors=[hex2rgb(x) fo
nodePos = chordDiagram(flux, ax)
ax.axis('off')
prop = dict(fontsize=16*0.8, ha='center', va='center')
nodes = station_df['명칭']
```

- Chord 다이어그램을 이미지 파일로 저장한다.

```
plt.savefig("chord.png", dpi=600,
        transparent=True,
        bbox_inches='tight', pad_inches=0.02)
```

2-3. 2015년 대여량 실제 데이터와 예측

(1) 예측 영향도

- 필요한 Library를 import 시켜준다.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

- total_rent.csv 파일을 불러와 train_data에 저장하여 트레이닝 셋을 만든다. 필요없는 column들을 모두 지워주고 데이터 타입을 object로 바꿔준다. NaN 데이터 값은 0.0 으로 채워준다.

```
train_data = pd.read_csv('./total_rent.csv')
train_data= train_data.fillna(value=0.0)
train_data = train_data.drop(columns = ['STATION', 'YEAR', 'DAY'])
train_data['MONTH'] = train_data['MONTH'].astype('object')
train_data['HOUR'] = train_data['HOUR'].astype('object')
train_data['WEEKDAY'] = train_data['WEEKDAY'].astype('object')
train_data['SEASON'] = train_data['SEASON'].astype('object')
```

- 입력변수 X와 출력변수 y로 트레이닝 셋을 나누어 준다. 입력변수 X는 RENTCOUNT를 제외한 모든 변수이고, 출력변수 y 는 RENTCOUNT 이다.

```
X_train = train_data.iloc[:, :-1].values
y_train = train_data.iloc[:, -1].values
y_train += np.ones(len(y_train)) # to avoid divide_by_0
```

- Randoforest를 통하여 학습을 진행한다.

```
# Randomforest를 통한 학습 진행
rf = RandomForestRegressor(n_estimators = 50)
rf.fit(X_train, y_train)
```

- 모델의 영향도를 list로 저장한다. 영향도를 소수점 둘째 자리까지 반올림한다. 그 후 영향도를 저장한 리스트를 y축으로 하고 입력변수 x를 x축으로 해서 막대그래프를 그린다.

```
ax = sns.barplot(x=x_train.columns, y=importances)
ax.set(xlabel='Variable', ylabel='Importance')
plt.title('Variable Importances')
plt.show()
```

(2) 하루 시간별 대여량 예측 모델

- 2015년 1월 1일 3번 정류장의 데이터만 쓰기 위해서 필요없는 데이터들은 모두 지워준다.

```
test_data = train[train.YEAR == 2015]
test_data = test_data[test_data.MONTH == 1]
test_data = test_data[test_data.DAY == 1]
test_data = test_data[test_data.STATION == 3]
```

- train 데이터와 마찬가지로 필요없는 column들은 모두 지워주고 데이터 타입을 object로 바꿔준다. NaN 데이터 값은 0.0. 으로 채워준다.
- 입력변수와 출력변수를 train 데이터와 똑같이 나누어 준다.
- RSME를 계산하기 위해 predict 함수를 사용하여 아래의 수식에 적용한다.

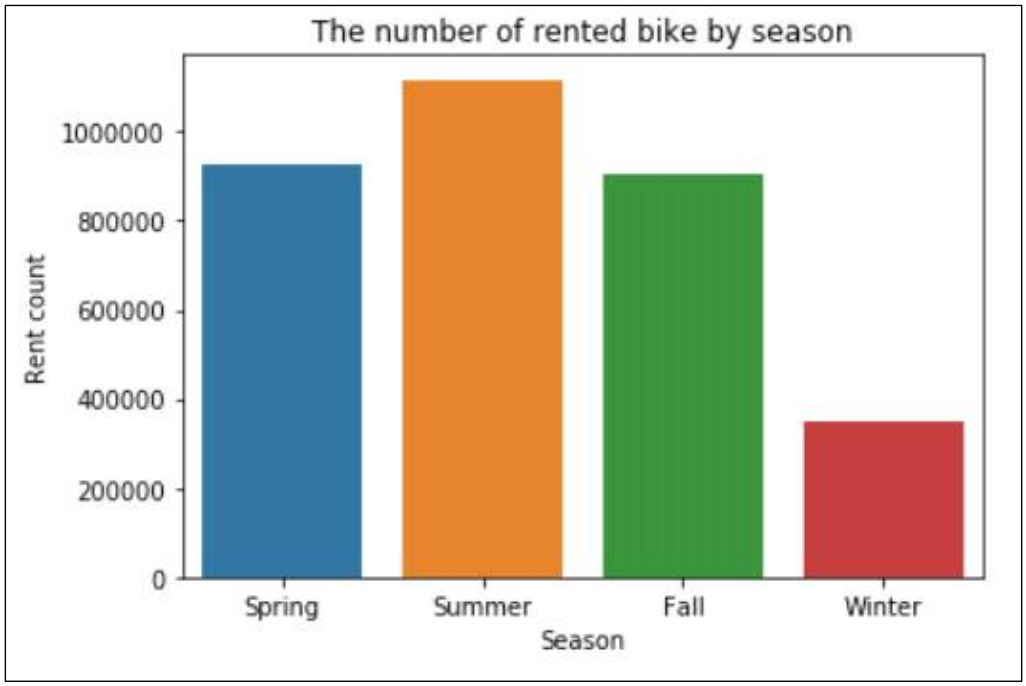
```
# RSME 계산
predictions = rf.predict(X_test)
errors = np.sqrt(np.mean((predictions - y_test)**2))
print('Error: ', round(errors, 2), 'degrees.')
```

- 실제 데이터와 예측된 데이터 값을 시간 별로 비교하여 꺾은선 그래프를 그린다.

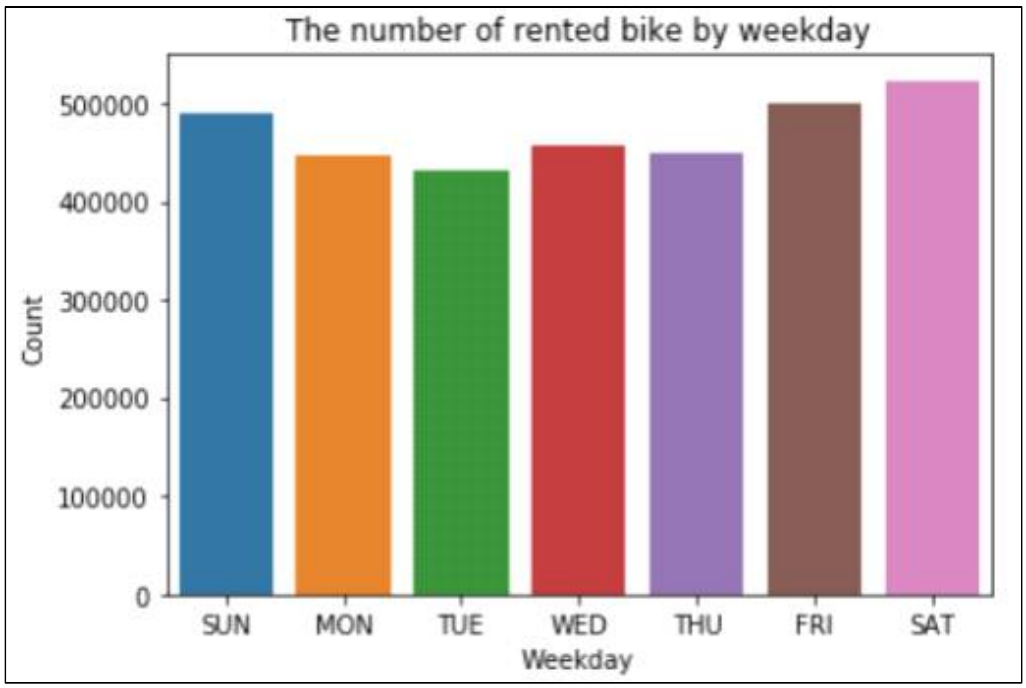
3. 결과 화면

3-1. 과제 1번

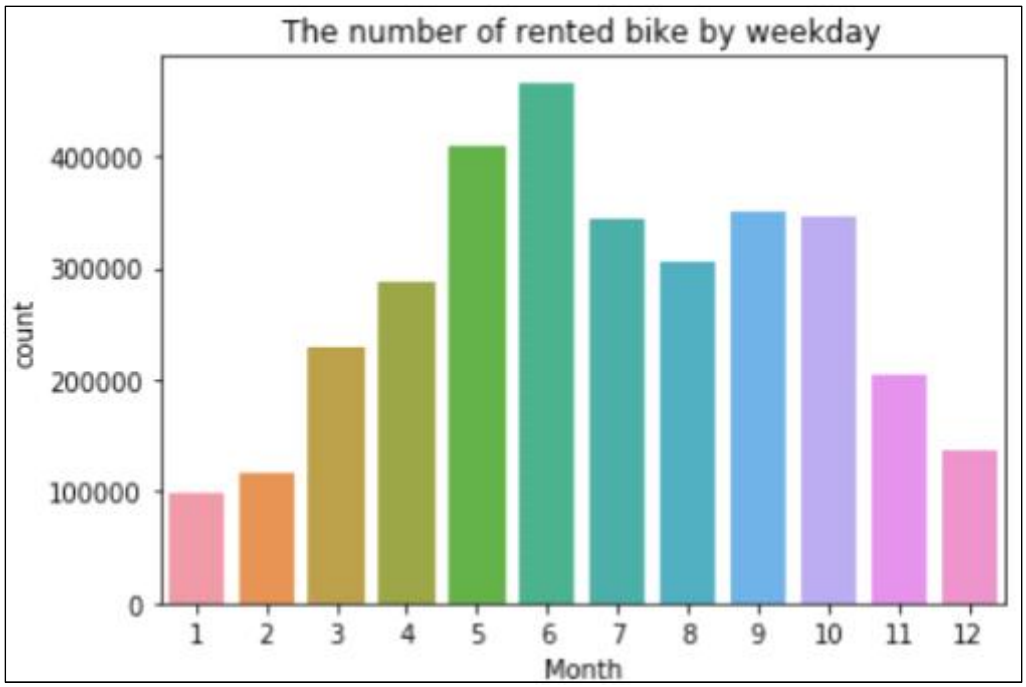
(1) 1-A 계절, 요일, 월, 시간 별 대여량 막대 그래프



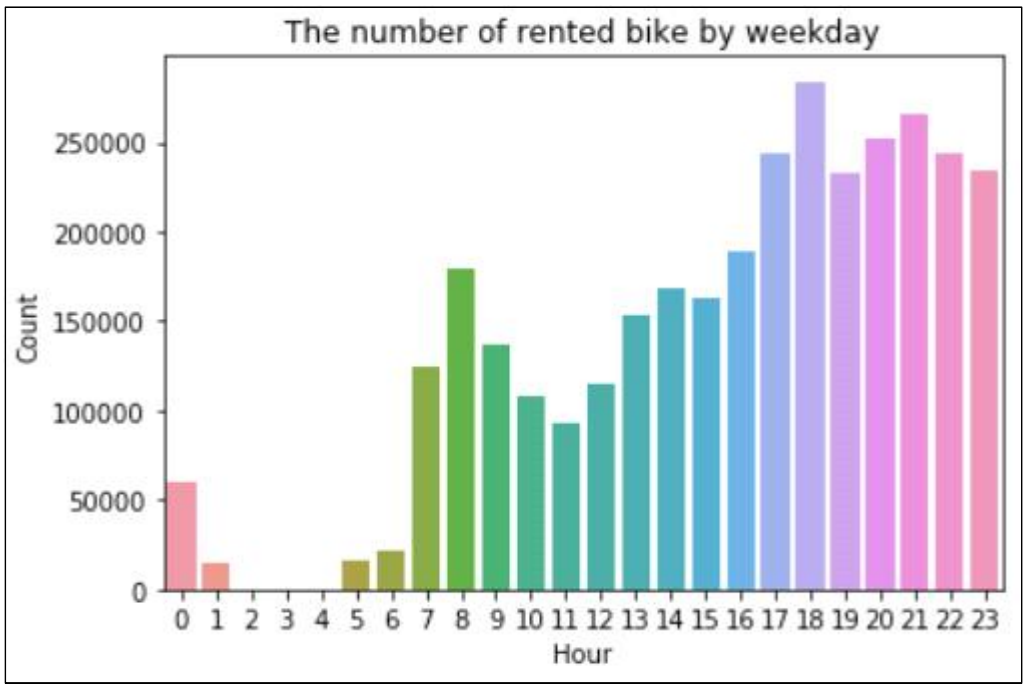
< 계절 별 대여량 막대 그래프 >



< 요일 별 대여량 막대 그래프 >

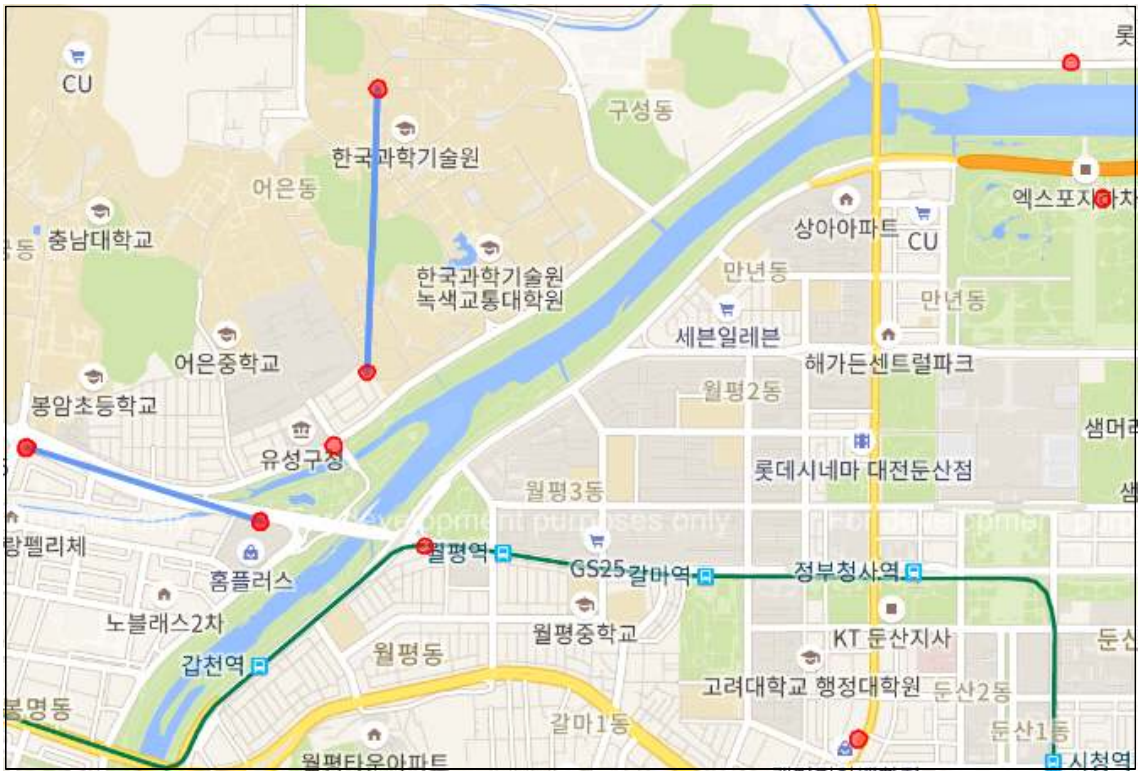


< 월 별 대여량 막대 그래프 >

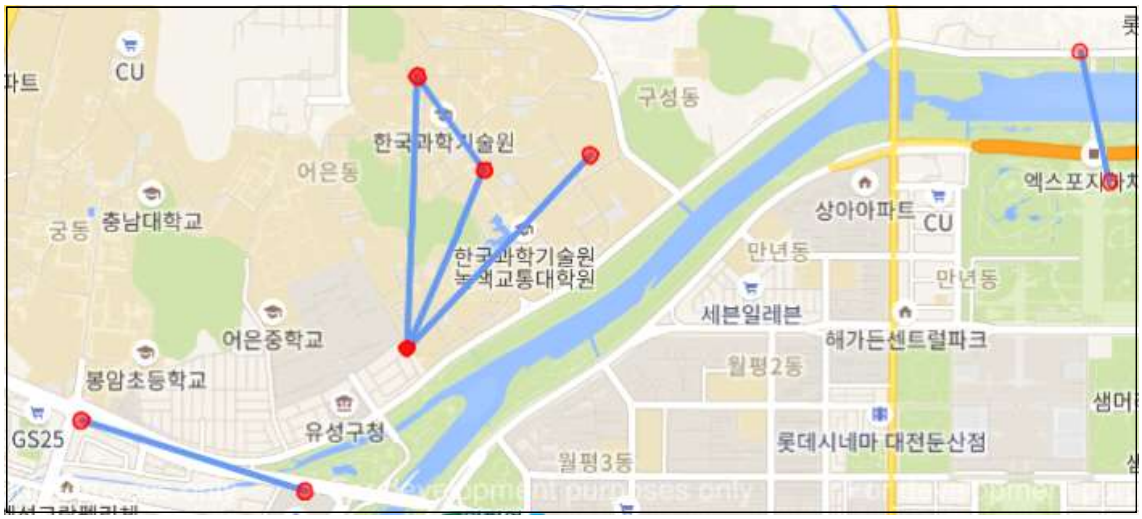


< 시간 별 대여량 막대 그래프 >

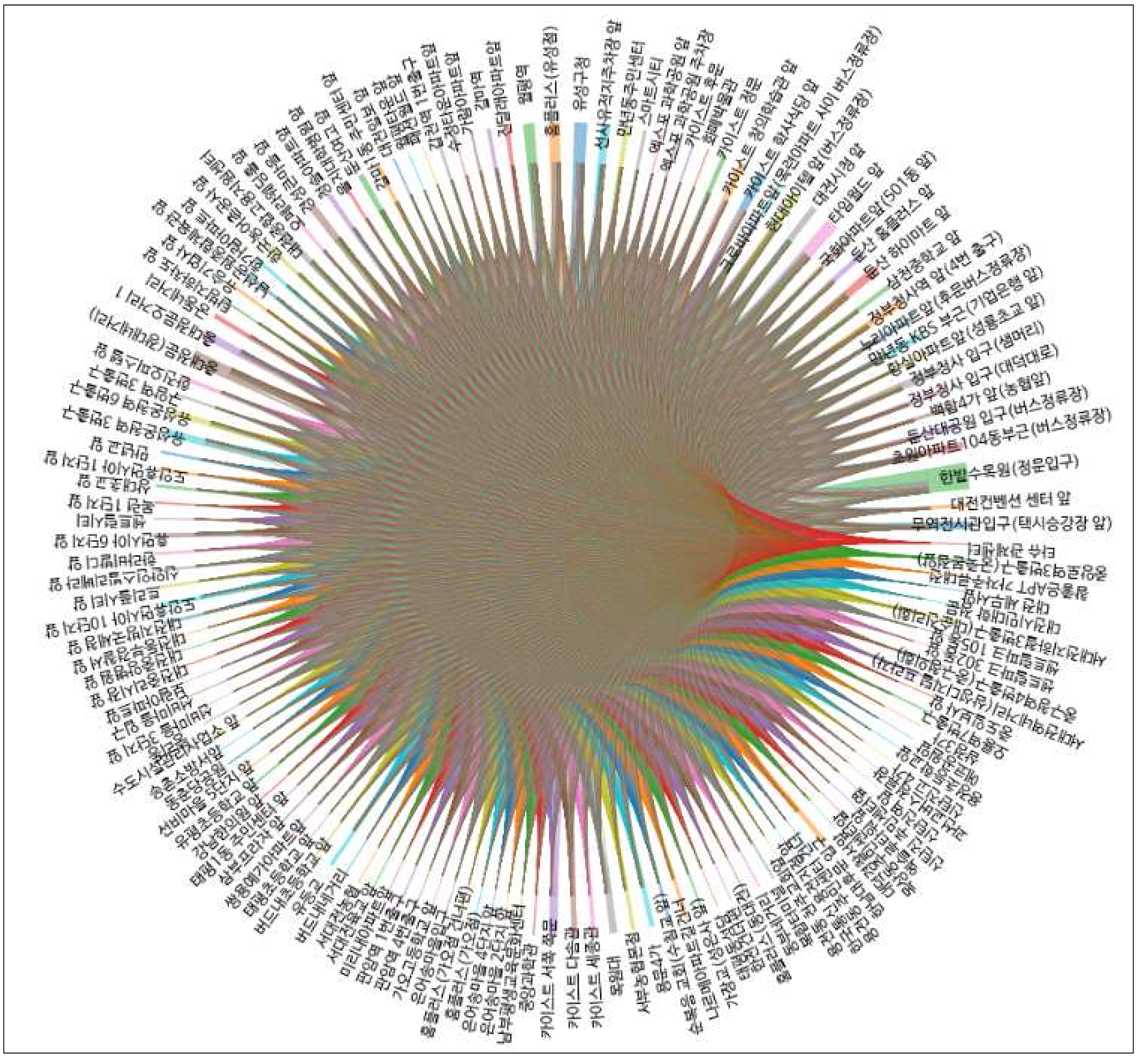
(2) 1-B 인기 경로 Top10



(3) (추가)1-B 대여지점으로 돌아오는 경로를 제외한 Top10

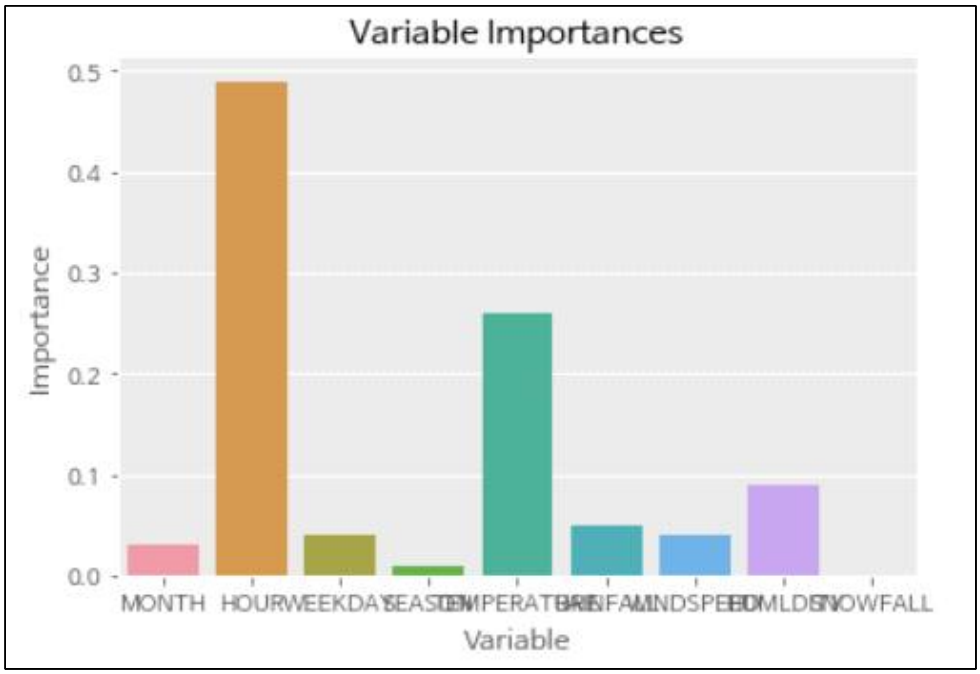


(4) 1-C Chord 다이어그램



3-2. 과제 2번

(1) 2-A 예측 영향도



(2) 2-B 하루 시간별 대여량 예측 모델