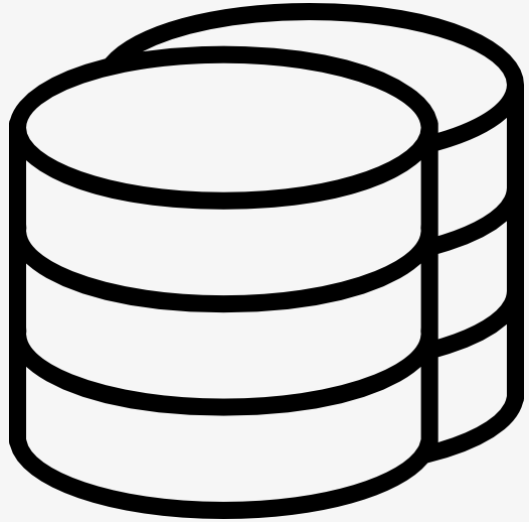
Dataset

- 30분 지하철 30분 단위 이용 통계
- 서울시 지하철 시간대별 승객 수
- 우수중소기업 공간데이터
- 수도권지하철 공간데이터
- 서울시 주요시설과 집객 시설 공간데이터



주제

서울시 시민의 지하철, 대중교통 데이터, 30분 단위 이용 통계, 여러 서울시 공간데이터를 활용해 시간별 혼잡 상황을 파악하고, 혼잡도를 미리 예측한다.
그 후, 새로운 호선을 추가 배치하여 각 동네에 새로운 역을 개통했을 때, 해당 역의 지하철 승하차 인원 수는 과연 얼만큼 될지 예측한다.

STATN19

수도권 지하철 공간데이터(19년)

STATN19_572

	STN_NM	BB_RT	TM_X	TM_Y
0	방화	5	183460.629400	553121.189900
1	개화산	5	182879.188000	552562.267200
2	김포공항	5	182425.217800	551457.329300
3	송정	5	183389.185800	551316.623600
4	마곡	5	184579.398500	551203.240100
...	...	...	...	...
148	도림천	2	189636.922600	546102.029000
149	양천구청	2	188138.364800	545894.398100
150	신정네거리	2	186998.713800	546747.951900
151	용두	2	203364.482800	552726.071500
152	까치산	2	186448.617563	548041.014706

```
# 5호선만 뽑아보리기

STATN19_5 = STATN19[STATN19["BB_RT"].str.contains('5')].reset_index(drop = True)
STATN19_5.head()
```

```
# 7호선만 뽑기

STATN19_7 = STATN19[STATN19["BB_RT"].str.contains('7')].reset_index(drop = True)
STATN19_7.head()
```

```
# 2호선만 뽑기

STATN19_2 = STATN19[STATN19["BB_RT"].str.contains('2')].reset_index(drop = True)[:51]
STATN19_2.head()
```

5, 7, 2호선만 뽑아주었다.

지도 시각화

572선의 시각화

```
# TM좌표계를 위도/경도 좌표로 변환하기 위해 따로 뽑기

STATN19_xy = STATN19_572[["TM_X","TM_Y"]]
```

STATN19_xy

	TM_X	TM_Y
0	183460.629400	553121.189900
1	182879.188000	552562.267200
2	182425.217800	551457.329300
3	183389.185800	551316.623600
4	184579.398500	551203.240100
...	...	...
148	189636.922600	546102.029000
149	188138.364800	545894.398100
150	186998.713800	546747.951900
151	203364.482800	552726.071500
152	186448.617563	548041.014706

```
# 좌표계 정보 설정
p1_type = "epsg:5186"
p2_type = "epsg:4326"

# project_array() 함수 실행
result_sub = project_array(coord, p1_type, p2_type)
result_sub[:5]
```

```
array([[126.81276331,  37.5774913 ],
       [126.80619409,  37.57244484],
       [126.80108169,  37.56248091],
       [126.81199539,  37.56123104],
       [126.82546879,  37.56023014]])
```

위도, 경도 좌표계로 변경해주었다.

STATN19_572

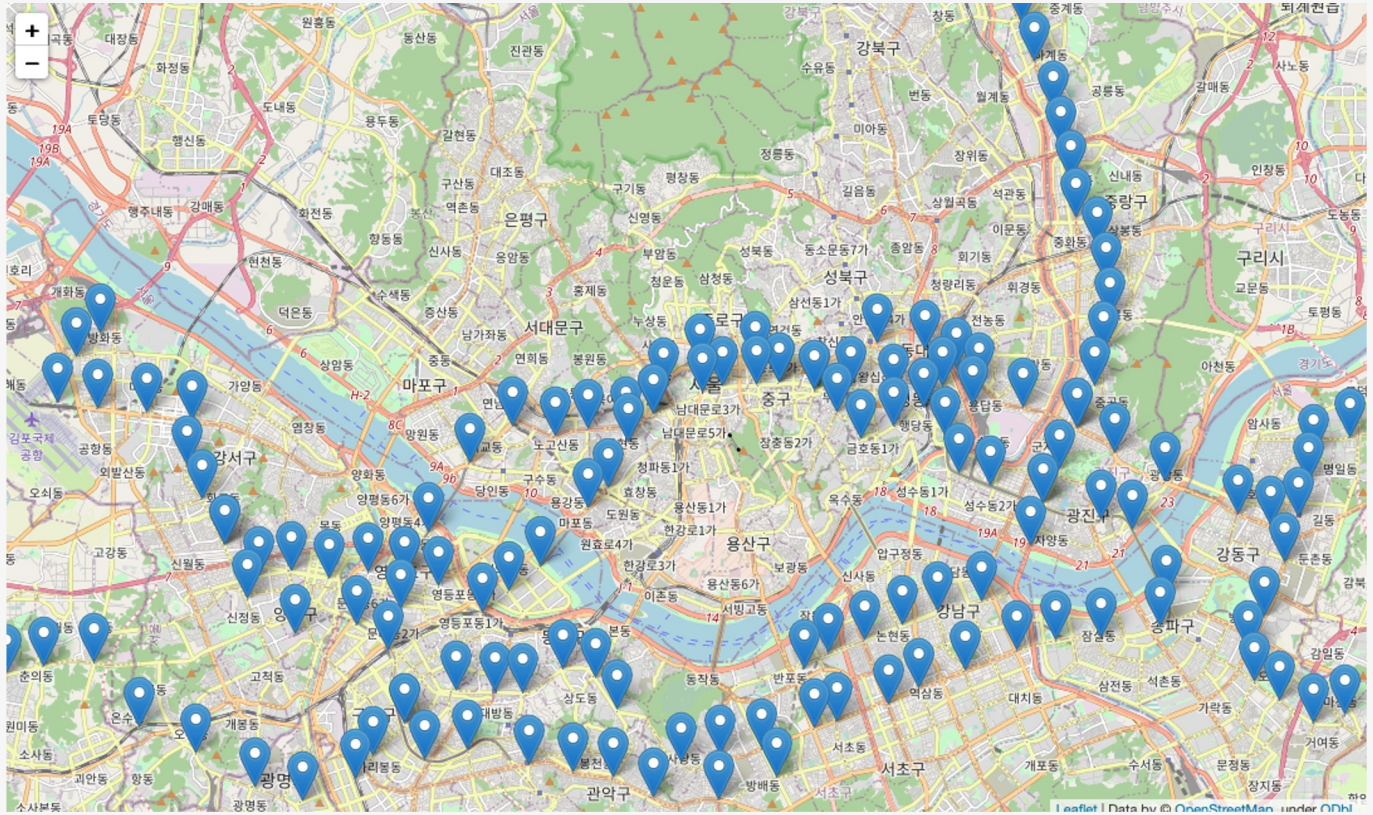
	STN_NM	BB_RT	위도	경도
0	방화	5	37.577491	126.812763
1	개화산	5	37.572445	126.806194
2	김포공항	5	37.562481	126.801082
3	송정	5	37.561231	126.811995
4	마곡	5	37.560230	126.825469
...	...	...	...	...
148	도림천	2	37.514339	126.882782
149	양천구청	2	37.512450	126.865835
150	신정네거리	2	37.520125	126.852930
151	용두	2	37.574074	127.038086
152	까치산	2	37.531768	126.846683

지도 시각화

572선의 시각화

```
g_map = g.Map(location =[37.613333,127.030049],zoom_start = 12)
```

```
# 앞서 구한 위도와 경도 정보를 가지고 실제 위치에 찍은 결과

for item in STATN19_572.index:
    lat = STATN19_572.loc[item, '위도']
    long = STATN19_572.loc[item, '경도']
    g.Marker([lat, long],
                        fill = True).add_to(g_map)
g_map
```



TBVIATR데이터셋

서울시 주요시설, 집객시설 주요데이터(20년)

	LCLASDC	MLSFCDC	X_VALUE	Y_VALUE
0	교육/보건	교육시설	214388.672913	551205.841919
1	산업	서비스산업	214555.910095	550962.356812
2	교육/보건	교육시설	214224.118713	550979.324829
3	교육/보건	교육시설	214171.512024	550841.469116
4	산업	서비스산업	214481.776611	550819.438110
...	...	...	...	...
19456	교육/보건	교육시설	191061.555317	544289.967983
19457	교육/보건	교육시설	191005.451446	542730.985067
19458	산업	서비스산업	191354.443045	541322.297120
19459	교육/보건	보건시설	188687.329239	547251.976724
19460	교육/보건	보건시설	202925.205495	547526.724798

```
# 좌표계 정보 설정
p1_type = "epsg:5186"
p2_type = "epsg:4326"

# project_array() 함수 실행
result_imt = project_array(coord, p1_type, p2_type)
result_imt[:5]
```

```
array([[127.16285188,  37.56027027],
       [127.16473985,  37.55807386],
       [127.16098506,  37.55823192],
       [127.16038702,  37.55699066],
       [127.16389801,  37.55678734]])
```

위도, 경도 좌표계로
변경해주었다.

	LCLASDC	MLSFCDC	위도	경도
0	교육/보건	교육시설	37.560270	127.162852
1	산업	서비스산업	37.558074	127.164740
2	교육/보건	교육시설	37.558232	127.160985
3	교육/보건	교육시설	37.556991	127.160387
4	산업	서비스산업	37.556787	127.163898
...	...	...	...	...
19456	교육/보건	교육시설	37.498027	126.898918
19457	교육/보건	교육시설	37.483980	126.898303
19458	산업	서비스산업	37.471291	126.902265
19459	교육/보건	보건시설	37.524689	126.872023
19460	교육/보건	보건시설	37.527229	127.033093

교육/보건

역별 교육/ 보건 시설의 개수

	LCLASDC	MLSFCDC	위도	경도
0	교육/보건	교육시설	37.560270	127.162852
1	교육/보건	교육시설	37.558232	127.160985
2	교육/보건	교육시설	37.556991	127.160387
3	교육/보건	교육시설	37.498516	127.102067
4	교육/보건	교육시설	37.497201	127.102680
...	...	...	...	...
13044	교육/보건	보건시설	37.450467	126.909474
13045	교육/보건	교육시설	37.498027	126.898918
13046	교육/보건	교육시설	37.483980	126.898303
13047	교육/보건	보건시설	37.524689	126.872023
13048	교육/보건	보건시설	37.527229	127.033093

```python
# 역별로 교육/보건 시설의 개수

print("교육/보건\n")
sb3 = []
i = 0
for a, b in result_sub:
    lyon = (a, b)        # 실제 지하철 역의 위도 경도값 설정
    count = 0
    for c, d in TBVIATR_edu_result_1:
        paris=(c, d)                    # 실제 시설의 위도 경도값 설정
        A = haversine(lyon, paris) < 1      # 지하철역의 위치값을 기준으로 반경 1km내에 포함된 시설 찾기
        if A is True:
            count += 1
    print(name[i], '->', count, '개')
    print("="*30)
    i += 1
    sb3.append(count)
```

```
교육/보건

['방화' '5'] -> 59 개
==============================
['개화산' '5'] -> 68 개
==============================
['김포공항' '5'] -> 34 개
==============================
['송정' '5'] -> 75 개
==============================
['마곡' '5'] -> 38 개
==============================
```

산업

역별 산업 시설의 개수

```
TBVIATR_edu_result_3 = np.array(TBVIATR_edu_3[['위도', '경도']])
TBVIATR_edu_result_3
```

```
array([[ 37.53804211, 127.12821378],
       [ 37.53833182, 127.12895978],
       [ 37.53820195, 127.12814859],
       ...,
       [ 37.50817955, 126.96237648],
       [ 37.50772692, 126.96191758],
       [ 37.47826728, 126.89486763]])
```

```
# 역별로 산업 시설의 개수

print("산업\n")
sb4 = []
i=0
for a, b in result_sub:
    lyon = (a, b)
    count = 0
    for c, d in TBVIATR_edu_result_2:
        paris=(c, d)
        A = haversine(lyon, paris) < 1
        if A is True:
            count += 1
    print(name[i], '->', count, '개')
    print("="*30)
    i += 1
    sb4.append(count)
```

```
산업

['방화' '5'] -> 7 개
==============================
['개화산' '5'] -> 9 개
==============================
['김포공항' '5'] -> 9 개
==============================
['송정' '5'] -> 15 개
==============================
['마곡' '5'] -> 12 개
==============================
```

숙박/음식

역별 숙박/음식
시설의 개수

```
TBVIATR_edu_result_4 = np.array(TBVIATR_edu_4[['위도', '경도']])
TBVIATR_edu_result_4
```

```
array([[ 37.49654608, 127.10993326],
       [ 37.59293371, 127.06575311],
       [ 37.58353155, 126.89368024],
       ...,
       [ 37.58693399, 126.98607813],
       [ 37.64067367, 127.01406449],
       [ 37.60264741, 126.92985768]])
```

```
# 역별로 숙박/음식 시설의 개수

print("숙박/음식\n")
sb5 = []
i=0
for a, b in result_sub:
    lyon = (a, b)
    count = 0
    for c, d in TBVIATR_edu_result_3:
        paris=(c, d)
        A = haversine(lyon, paris) < 1
        if A is True:
            count += 1
    print(name[i], '->', count, '개')
    print("="*30)
    i += 1
    sb5.append(count)
```

```
숙박/음식

['방화' '5'] -> 3 개
==============================
['개화산' '5'] -> 3 개
==============================
['김포공항' '5'] -> 7 개
==============================
['송정' '5'] -> 8 개
==============================
['마곡' '5'] -> 2 개
==============================
```

행정

역별 행정 시설의 개수

```
TBVIATR_edu_result_4 = np.array(TBVIATR_edu_4[['위도', '경도']])
TBVIATR_edu_result_4
```

```
array([[ 37.49654608, 127.10993326],
       [ 37.59293371, 127.06575311],
       [ 37.58353155, 126.89368024],
       ...,
       [ 37.58693399, 126.98607813],
       [ 37.64067367, 127.01406449],
       [ 37.60264741, 126.92985768]])
```

```
# 역별로 행정 시설의 개수

print("행정\n")
sb6 = []
i=0
for a, b in result_sub:
    lyon = (a, b)
    count = 0
    for c, d in TBVIATR_edu_result_4:
        paris=(c, d)
        A = haversine(lyon, paris) < 1
        if A is True:
            count += 1
    print(name[i], '->', count, '개')
    print("="*30)
    i += 1
    sb6.append(count)
```

```
행정

['방화' '5'] -> 2 개
==============================
['개화산' '5'] -> 3 개
==============================
['김포공항' '5'] -> 2 개
==============================
['송정' '5'] -> 2 개
==============================
['마곡' '5'] -> 0 개
==============================
```

레저/관광/예술

역별 레저/관광/예술 시설의 개수

```
TBVIATR_edu_result_5 = np.array(TBVIATR_edu_5[['위도', '경도']])
TBVIATR_edu_result_5[:5]
```

```
array([[ 37.49889341, 127.10445715],
       [ 37.57419305, 126.95891943],
       [ 37.46072539, 126.90504234],
       [ 37.47694521, 126.93793596],
       [ 37.4704393 , 126.91836431]])
```

```
#역별로 레저/관광/예술 시설의 개수

print("레저/관광/예술\n")
sb7 = []
i=0
for a, b in result_sub:
    lyon = (a, b)
    count = 0
    for c, d in TBVIATR_edu_result_5:
        paris=(c, d)
        A = haversine(lyon, paris) < 1
        if A is True:
            count += 1
    print(name[i], '->', count, '개')
    print("="*30)
    i += 1
    sb7.append(count)
```

```
레저/관광/예술

['방화' '5'] -> 1 개
==============================
['개화산' '5'] -> 1 개
==============================
['김포공항' '5'] -> 1 개
==============================
['송정' '5'] -> 1 개
==============================
['마곡' '5'] -> 2 개
==============================
```

공공/환경

역별 공공/환경 시설의 개수

```
TBVIATR_edu_result_6 = np.array(TBVIATR_edu_6[['위도', '경도']])
TBVIATR_edu_result_6[:5]
```

```
array([[ 37.46767352, 126.94480922],
       [ 37.51613554, 126.91214083],
       [ 37.50077829, 126.90856148],
       [ 37.48205824, 126.92792746],
       [ 37.51859504, 126.93167693]])
```

```
# 역별로 공공/환경 시설의 개수

print("공공/환경\n")
sb8 = []
i=0
for a, b in result_sub:
    lyon = (a, b)
    count = 0
    for c, d in TBVIATR_edu_result_6:
        paris=(c, d)
        A = haversine(lyon, paris) < 1
        if A is True:
            count += 1
    print(name[i], '->', count, '개')
    print("="*30)
    i += 1
    sb8.append(count)
```

```
공공/환경

['방화' '5'] -> 0 개
==============================
['개화산' '5'] -> 0 개
==============================
['김포공항' '5'] -> 0 개
==============================
['송정' '5'] -> 0 개
==============================
['마곡' '5'] -> 0 개
==============================
```

시설물

역별 시설물 시설의 개수

```
TBVIATR_edu_result_7 = np.array(TBVIATR_edu_7[['위도', '경도']])
TBVIATR_edu_result_7[:5]
```

```
array([[ 37.57708864, 127.04994455],
       [ 37.4852527 , 126.90160758],
       [ 37.48755509, 126.9132976 ],
       [ 37.51334765, 126.92648565],
       [ 37.50241793, 126.88151226]])
```

```
# 역별로 시설물 시설의 개수

print("시설물\n")
sb9 = []
i=0
for a, b in result_sub:
    lyon = (a, b)
    count = 0
    for c, d in TBVIATR_edu_result_7:
        paris=(c, d)
        A = haversine(lyon, paris) < 1
        if A is True:
            count += 1
    print(name[i], '->', count, '개')
    print("="*30)
    i += 1
    sb9.append(count)
```

```
시설물

['방화' '5'] -> 2 개
==============================
['개화산' '5'] -> 6 개
==============================
['김포공항' '5'] -> 6 개
==============================
['송정' '5'] -> 4 개
==============================
['마곡' '5'] -> 0 개
==============================
```

우수중소기업 시각화

EXC ; 위경도 변환

```
# 필요 컬럼 선택

EXC = EXC[['DEC_SE', 'ADDRES', 'TM_X', 'TM_Y']]
EXC.head()
```

	DEC_SE	ADDRES	TM_X	TM_Y
0	하이브랜드서울기업	서초구 효령로68길 92 (서초동)	201762.893208	542899.588742
1	하이브랜드서울기업	구로구 경인로63길 21-6	189818.338312	545438.678543
2	하이브랜드서울기업	성동구 성수이로 7길 27, 701 (서울숲 코오롱디지털타워2차)	204724.207199	549119.988156
3	하이브랜드서울기업	금천구 가산 디지털 2로 98 롯데아이티 캐슬 2동 6층	189481.128053	541984.760645
4	하이브랜드서울기업	마포구 매봉산로 37, 9층 905호 (DMC 산학협력연구센터)	190337.914065	552950.945367

TM -> 위경도

좌표 변환을 해주었다.

```
# TM -> 위경도 좌표 변환을 위해 선택

EXC_xy = EXC[['TM_X', 'TM_Y']]
```

```
# TM -> 위경도 변환 함수

def project_array(coord, p1_type, p2_type):
    """
    좌표계 변환 함수
    - coord: x, y 좌표 정보가 담긴 NumPy Array
    - p1_type: 입력 좌표계 정보 ex) epsg:5186
    - p2_type: 출력 좌표계 정보 ex) epsg:4326
    """
    p1 = pyproj.Proj(init=p1_type)
    p2 = pyproj.Proj(init=p2_type)
    fx, fy = pyproj.transform(p1, p2, coord[:, 0], coord[:, 1])
    return np.dstack([fx, fy])[0]
```

```
# 좌표계 정보 설정
p1_type = "epsg:5186"
p2_type = "epsg:4326"

# project_array() 함수 실행
result_imt = project_array(coord, p1_type, p2_type)
result_imt[:5]
```

```
array([[127.01993265,  37.48554134],
       [126.88484315,  37.50836426],
       [127.05345553,  37.54157698],
       [126.88107858,  37.4772405 ],
       [126.89062082,  37.57605522]])
```

우수중소기업 시각화

역별 중소기업 시설 개수 확인

```
# 데이터프레임화 시키기

facilities_count = pd.DataFrame(data = list(zip(sb, sb2, sb3, sb4, sb5, sb6, sb7, sb8, sb9)),
                                columns = ['지하철', '우수중소기업', '교육/보건', '산업', '숙박/음식', '행정', '레저/관광/예술', '공공/환경', '시설물'])
```

```
# 역별로 중소기업 시설의 개수

print("중소기업\n")
sb2 = []
i = 0
for a, b in result_sub:
    lyon = (a, b)     # 실제 지하철 역의 위도 경도값 설정
    count = 0
    for c, d in EXC_result:
        paris=(c, d)     # 실제 시설의 위도 경도값 설정
        A = haversine(lyon, paris) < 1    # haversine이용
        if A is True:
            count += 1
    print(name[i], '->', count, '개')
    print("="*30)
    i += 1
    sb2.append(count)
```

```
중소기업

['방화' '5'] -> 1 개
==============================
['개화산' '5'] -> 1 개
==============================
['김포공항' '5'] -> 0 개
==============================
['송정' '5'] -> 1 개
==============================
['마곡' '5'] -> 3 개
==============================
```

```
facilities_count.set_index('지하철')
```

지하철	우수중소기업	교육/보건	산업	숙박/음식	행정	레저/관광/예술	공공/환경	시설물
[방화, 5]	1	59	7	3	2	1	0	2
[개화산, 5]	1	68	9	3	3	1	0	6
[김포공항, 5]	0	34	9	7	2	1	0	6
[송정, 5]	1	75	15	8	2	1	0	4
[마곡, 5]	3	38	12	2	0	2	0	0
...	...	...	...	...	...	...	...	...
[도림천, 2]	3	48	16	4	2	0	0	2
[양천구청, 2]	0	62	8	0	7	1	0	0
[신정네거리, 2]	3	124	14	9	3	1	0	0
[용두, 2]	1	155	31	44	5	4	0	1
[까치산, 2]	0	152	12	49	6	2	0	1

우수중소기업 시각화

각 시설의 수 합치기

```
# 각 시설의 수 합치기

sb10 = facilities_count['우수중소기업'] + facilities_count['교육/보건'] + facilities_count['산업']\
+ facilities_count['숙박/음식'] + facilities_count['행정'] + facilities_count['레저/관광/예술'] + facilities_count['공공/환경']\
+ facilities_count['시설물']
facilities_count['총 합'] = sb10
```

```
# 앞서 구한 총 평균거리 넣기

facilities_count['평균거리'] = final_dist_mean
```

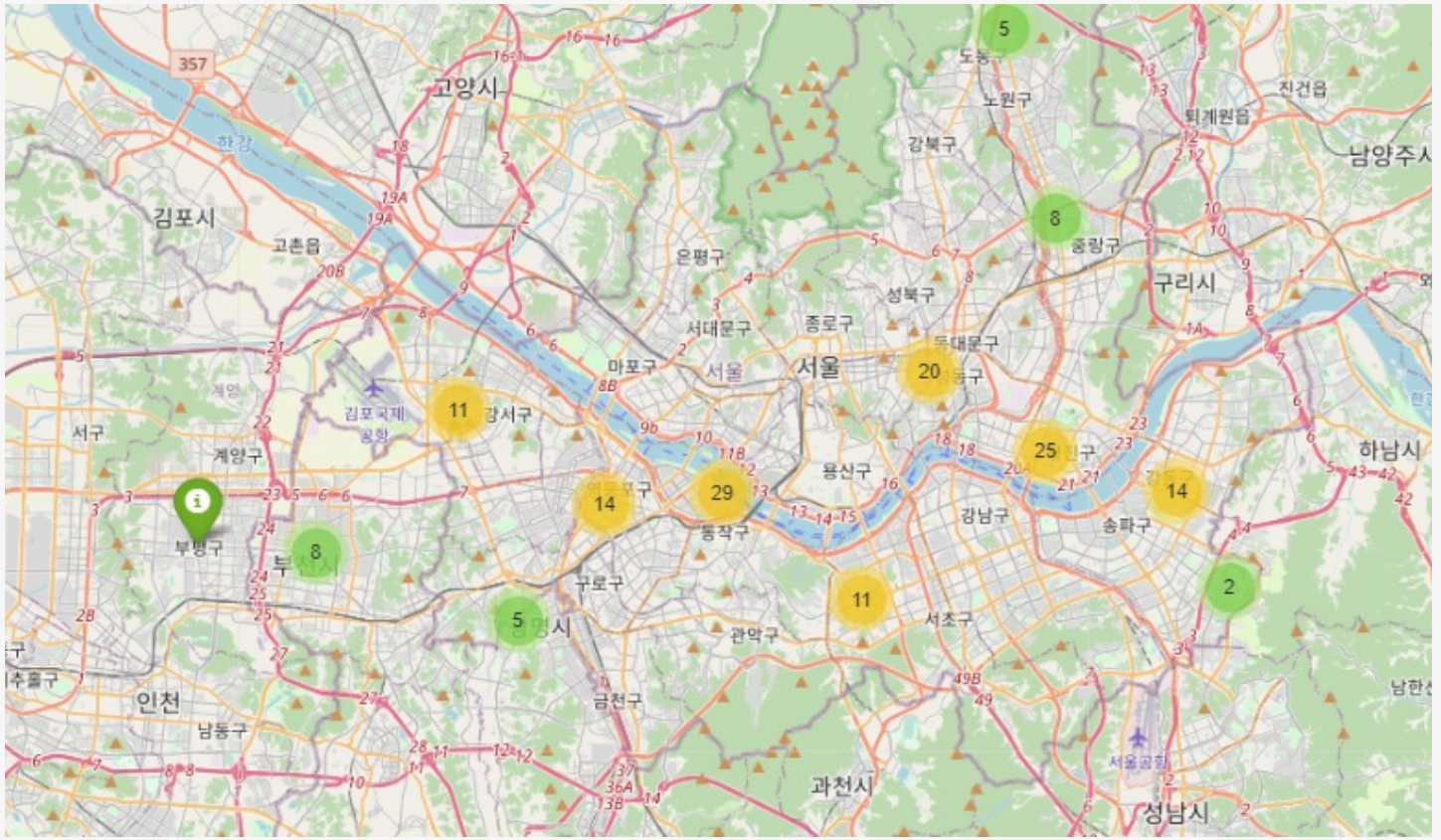
```
facilities_count.tail()
```

	지하철	우수중소기업	교육/보건	산업	숙박/음식	행정	레저/관광/예술	공공/환경	시설물	총 합	평균거리
148	[도림천, 2]	3	48	16	4	2	0	0	2	75	0.51
149	[양천구청, 2]	0	62	8	0	7	1	0	0	78	0.31
150	[신정네거리, 2]	3	124	14	9	3	1	0	0	154	0.44
151	[용두, 2]	1	155	31	44	5	4	0	1	241	0.61
152	[까치산, 2]	0	152	12	49	6	2	0	1	222	0.45

지하철 역 별 시설 총합 시각화

facilities_count를 이용

```
from folium.plugins import MarkerCluster

# 위도
latitude = 37.394946
# 경도
longitude = 127.111104

m = g.Map(
    location=[latitude, longitude],
    zoom_start=15
)

coords = STATN19_572[['위도', '경도']]


marker_cluster = MarkerCluster().add_to(m)

for lat, long in zip(coords['위도'], coords['경도']):
    g.Marker([lat, long], icon = g.Icon(color="green")).add_to(marker_cluster)
m
```



지하철 역 별 시설

facilities_count를 이용

```
sub_list = [0] * 153
a = ''
n = 0
for i, v in facilities_count.지하철:
    a = i + '-' + v
    sub_list[n] = a
    n += 1
sub_list
```

```
['방화-5',
 '개화산-5',
 '김포공항-5',
 '송정-5',
 '마곡-5',
 '발산-5',
 '우장산-5',
 '화곡-5',
 '까치산-5',
 '신정-5',
 '목동-5',
```

```
facilities_count['지하철-호선'] = sub_list
```

```
facilities_count = facilities_count.drop(['지하철'], axis = 1)
facilities_count
```

	우수중소기업	교육/보건	산업	숙박/음식	행정	레저/관광/예술	공공/환경	시설물	총 합	지하철-호선
0	1	59	7	3	2	1	0	2	75	방화-5
1	1	68	9	3	3	1	0	6	91	개화산-5
2	0	34	9	7	2	1	0	6	59	김포공항-5
3	1	75	15	8	2	1	0	4	106	송정-5
4	3	38	12	2	0	2	0	0	57	마곡-5
...	...	...	...	...	...	...	...	...	...	...
148	3	48	16	4	2	0	0	2	75	도림천-2
149	0	62	8	0	7	1	0	0	78	양천구청-2
150	3	124	14	9	3	1	0	0	154	신정네거리-2
151	1	155	31	44	5	4	0	1	241	용두-2
152	0	152	12	49	6	2	0	1	222	까치산-2

153 rows × 10 columns

```
df = facilities_count.set_index('지하철-호선')
#.groupby('지하철-호선')[['총 합']].sum().sort_values('총 합', ascending = False)
```

머신러닝 전 추가 전처리

역별 승하차 인원 확인

```
subway21 = pd.read_csv("data/subway21_df.csv", index_col = ['STATION_NM', 'LINE_NM'])
```

```
# 역별 승하차 인원 확인
subway21['GET_TOTAL'] = subway21['GETON_CNT'] + subway21['GETOFF_CNT']
subway21
subway21_sort = subway21.sort_values(ascending = False, by = 'GET_TOTAL')
subway21_sort = subway21.groupby(['STATION_NM', 'LINE_NM']).sum('GET_TOTAL')[['GET_TOTAL']].sort_values('GET_TOTAL', ascending = False)
```

```
subway21_sort = subway21_sort.reset_index(drop=False)
subway21_sort
```

	STATION_NM	LINE_NM	GET_TOTAL
0	강남	2호선	39321791
1	잠실(송파구청)	2호선	34859312
2	신림	2호선	31166581
3	구로디지털단지	2호선	27919882
4	홍대입구	2호선	24607291
...	...	...	...
330	남태령	4호선	512681
331	남위례	8호선	65574
332	연신내	6호선	227
333	충무로	3호선	184
334	신내	6호선	72

머신러닝 전 추가 전처리

STATION_NM의 괄호 + 괄호안 내용 없애기

```
# STATION_NM의 괄호 + 괄호안 내용 없애기

rmve_bracket = "\(.*\)|\s-\s.*"
name_list = []
for i in subway21_sort.STATION_NM:
    i = re.sub(rmve_bracket, '',i)
    name_list.append(i)
name_list
```

	STATION_NM	LINE_NM	GET_TOTAL
0	강남	2호선	39321791
1	잠실	2호선	34859312
2	신림	2호선	31166581
3	구로디지털단지	2호선	27919882
4	홍대입구	2호선	24607291
...	...	...	...
330	남태령	4호선	512681
331	남위례	8호선	65574
332	연신내	6호선	227
333	충무로	3호선	184
334	신내	6호선	72

5, 7, 2호선
정보만 합침

	STATION_NM	LINE_NM	GET_TOTAL
0	강남	2호선	39321791
1	잠실	2호선	34859312
2	신림	2호선	31166581
3	구로디지털단지	2호선	27919882
4	홍대입구	2호선	24607291
...	...	...	...
152	용마산	7호선	3076887
153	반포	7호선	2873882
154	삼산체육관	7호선	2050929
155	부천종합운동장	7호선	2050168
156	장암	7호선	870210

머신러닝 전 추가 전처리

역이름 및 호선번호만 따로 뽑아 새로운 컬럼에 각각 넣기

```
# 역이름 및 호선번호만 따로 뽑아 새로운 컬럼에 각각 넣기

new2 = []
new3 = []

for i in range(153):
    a = facilities_count.loc[i,'지하철-호선']
    new2.append(a[:-2])

for i in range(153):
    b = facilities_count.loc[i, '지하철-호선']
    new3.append(b[-1])
facilities_count['지하철'] = new2
facilities_count['호선'] = new3
```

```
subway21_572 = subway21_572.rename(columns = {'STATION_NM':'지하철'})
```

```
merge_df = pd.merge(facilities_count,subway21_572, on="지하철")
merge_df = merge_df.drop(['지하철-호선','호선'], axis = 1)
```

```
merge_df = merge_df.drop_duplicates(['지하철'])
merge_df = merge_df.rename(columns = {'LINE_NM':'호선', 'GET_TOTAL':'승하차인원'})
merge_df
```

	우수중소기업	교육/보건	산업	숙박/음식	행정	레저/관광/예술	공공/환경	시설물	총 합	평균거리	지하철	호선	승하차인원
0	1	59	7	3	2	1	0	2	75	0.53	방화	5호선	3148408
1	1	68	9	3	3	1	0	6	91	0.52	개화산	5호선	2506452
2	0	34	9	7	2	1	0	6	59	0.61	김포공항	5호선	3718314
3	1	75	15	8	2	1	0	4	106	0.37	송정	5호선	3297282
4	3	38	12	2	0	2	0	0	57	0.50	마곡	5호선	4570357
...	...	...	...	...	...	...	...	...	...	...	...	...	...
163	2	81	27	57	8	2	0	1	178	0.57	신설동	2호선	1568224
164	3	48	16	4	2	0	0	2	75	0.51	도림천	2호선	635911
165	0	62	8	0	7	1	0	0	78	0.31	양천구청	2호선	3162527
166	3	124	14	9	3	1	0	0	154	0.44	신정네거리	2호선	4709553
167	1	155	31	44	5	4	0	1	241	0.61	용두	2호선	1278391

머신러닝을 통한 예측

선형회귀

```
merge_df.호선 = merge_df.호선.astype('category').cat.codes
merge_df.지하철 = merge_df.지하철.astype('category').cat.codes
merge_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 143 entries, 0 to 167
Data columns (total 13 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   우수중소기업     143 non-null    int64
 1   교육/보건      143 non-null    int64
 2   산업         143 non-null    int64
 3   숙박/음식      143 non-null    int64
 4   행정         143 non-null    int64
 5   레저/관광/예술   143 non-null    int64
 6   공공/환경      143 non-null    int64
 7   시설물        143 non-null    int64
 8   총 합        143 non-null    int64
 9   평균거리       143 non-null    float64
 10  지하철        143 non-null    int16
 11  호선         143 non-null    int8
 12  승하차인원      143 non-null    int64
dtypes: float64(1), int16(1), int64(10), int8(1)
memory usage: 13.8 KB
```

```
data = merge_df[['우수중소기업', '교육/보건', '산업', '숙박/음식', '행정', '레저/관광/예술', '공공/환경', '시설물', '총 합' ,'지하철', '호선', '평균거리']]
target = merge_df['승하차인원']

from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(
    data, target, test_size = 0.2
)
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()

lr.fit(train_input, train_target)

print(lr.coef_, lr.intercept_)
```

```
[  173690.64701195    78561.83696769    81609.69723349    63146.15240513
  -297776.14967563   224073.90962631    29941.45978666  -417105.040714
   -63857.48735833   -15987.56864703 -3741179.48016677 -5188497.27533707] 14391203.465684148
```

```
print(lr.score(train_input, train_target))
print(lr.score(test_input, test_target))
```

```
0.310978400710734
0.3454798046747295
```

머신러닝을 통한 예측

다중회귀

다중회귀

```
# 변환기

from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(include_bias = False)
poly.fit(train_input)
train_poly = poly.transform(train_input)
test_poly = poly.transform(test_input)
```

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train_poly, train_target)
print(lr.score(train_poly, train_target))
print(lr.score(test_poly, test_target))
```

```
0.8580210640729422
-4.5596720162991
```

머신러닝을 통한 예측

릿지

릿지

```
# 표준화

from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
ss.fit(train_poly)
train_scaled = ss.transform(train_poly)
test_scaled = ss.transform(test_poly)
```

```
#그리드 서치 최적의 alpha 값 찾기
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
params = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
gs = GridSearchCV(Ridge(), params, n_jobs=-1)
gs.fit(train_scaled, train_target)
print(gs.best_params_)
```

```
{'alpha': 10}
```

```
from sklearn.linear_model import Ridge
ridge = Ridge(alpha = 10)
ridge.fit(train_scaled, train_target)
print(ridge.score(train_scaled, train_target))
print(ridge.score(test_scaled, test_target))
```

```
0.5065296207152367
0.5575167756444398
```

머신러닝을 통한 예측

라쏘

라쏘

```
#그리드 서치 최적의 alpha 값 찾기
from sklearn.linear_model import Lasso
from sklearn.model_selection import GridSearchCV
params = {'alpha': [0.001, 0.01, 0.1, 1, 10, 100]}
gs = GridSearchCV(Ridge(), params, n_jobs=-1)
gs.fit(train_scaled, train_target)
print(gs.best_params_)
```

```
{'alpha': 10}
```

```
from sklearn.linear_model import Lasso
lasso= Lasso(alpha = 10)
lasso.fit(train_scaled, train_target)
print(lasso.score(train_scaled, train_target))
print(lasso.score(test_scaled, test_target))
```

```
0.8011452334132623
0.08852010221728834
```

머신러닝을 통한 예측

결정트리

결정트리

```
data = merge_df[['우수중소기업', '교육/보건', '산업','행정', '숙박/음식', '레저/관광/예술', '총 합' ,'지하철', '호선', '평균거리']].to_numpy()
target = merge_df['승하차인원'].to_numpy()
train_input, test_input, train_target, test_target = train_test_split(
    data, target, test_size = 0.2)
```

```
from sklearn.model_selection import cross_validate
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state = 42)
dt.fit(train_input, train_target)
print(dt.score(train_input, train_target))
print(dt.score(test_input, test_target))
```

```
1.0
0.0
```

```
from sklearn.model_selection import StratifiedKFold
scores = cross_validate(dt, train_input, train_target, cv = StratifiedKFold(n_splits = 3, shuffle = True, random_state = 42))
print(np.mean(scores['test_score']))

# n_splits값 때문에 계속 오류가 뜨지만 해결 불가..
```