



빅데이터 프로젝트

서울교통공사 지하철 혼잡도 분석

201944047.정경훈

목차

1 주제 설명

2 프로젝트 과정

3 결론

1

주제 설명

프로젝트 목표 : 서울교통공사 지하철 혼잡도를 파악하여 나타내는 것

분석 할 내용 : 승하차 별 이용객 수 분석, 호선, 역 별 이용객 수 분석

데이터 수집 방법 : 공공데이터 포털

프로젝트 범위 : 데이터 정제 및 가공 – 데이터 분석 – 데이터 시각화



2

프로젝트 과정


데이터 수집 및 다운로드

파일데이터 정보

데이터 다운로드

데이터 개선요청

오류신고 및 문의

파일데이터명	서울특별시_지하철 호선별 역별 승하차 인원 정보_10/28/2021		
분류체계	교통및물류 - 철도	제공기관	서울특별시
관리부서명	교통정책과	관리부서 전화번호	02-2133-2237
보유근거		수집방법	
업데이트 주기	수시 (1회성 데이터)	차기 등록 예정일	
매체유형	텍스트	전체 행	1
확장자	CSV	키워드	지하철 승하차 인원, 교통
데이터 한계		다운로드(바로그가)	2914
등록일	2020-10-27	수정일	2021-10-28
제공형태	기관 자체에서 다운로드(제공데이터URL기재)		
URL	https://data.seoul.go.kr/dataList/OA-12914/5/1/datasetView.do		
설명	교통카드(선후불교통카드 및 1회용 교통카드)를 이용한 지하철 호선별 역별(서울교통공사, 한국철도공사, 공항철도, 9호선) 승하차인원을 나타내는 정보입니다. (일단위) ※ Sheet 서비스는 마지막 한달치 데이터만 서비스 합니다. (* 데이터 적재는 매일 3일전 데이터를 갱신합니다.)		
기타 유의사항			
비용부과유무	무료	비용부과기준 및 단위	건
이용허락범위	  공공저작물_출처표시		

데이터 로드, 정제 및 가공 (1)

- 지하철역 공백 제거
- 승 하차 구분
- 승 하차 칼럼 명 재정의 후 info와 합침

```
import pandas as pd
import numpy as np

df = pd.read_csv('지하철.csv')

df.drop(['작업일자'], axis=1, inplace=True)
df['지하철역'] = df['지하철역'].str.strip()
#df['지하철역']

info = df.iloc[:, :3]
info.columns = ['년월', '호선', '역명']
#info.head()

# 승차
riding = df.iloc[:, 3:]
riding = riding.iloc[:, [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46]]
riding.rename(columns={'04시-05시 승차인원':'4시', '05시-06시 승차인원':'5시', '06시-07시 승차인원':'6시', '07시-08시 승차인원':'7시',
'08시-09시 승차인원':'8시', '09시-10시 승차인원':'9시', '10시-11시 승차인원':'10시', '11시-12시 승차인원':'11시',
'12시-13시 승차인원':'12시', '13시-14시 승차인원':'13시', '14시-15시 승차인원':'14시', '15시-16시 승차인원':'15시',
'16시-17시 승차인원':'16시', '17시-18시 승차인원':'17시', '18시-19시 승차인원':'18시', '19시-20시 승차인원':'19시',
'20시-21시 승차인원':'20시', '21시-22시 승차인원':'21시', '22시-23시 승차인원':'22시', '23시-24시 승차인원':'23시',
'00시-01시 승차인원':'24시', '01시-02시 승차인원':'1시', '02시-03시 승차인원':'2시', '03시-04시 승차인원':'3시'}, inplace=True)

riding = pd.concat([info, riding], axis=1)

# 하차
stopover = df.iloc[:, 3:]
stopover = stopover.iloc[:, [1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47]]
stopover.rename(columns={'04시-05시 하차인원':'4시', '05시-06시 하차인원':'5시', '06시-07시 하차인원':'6시', '07시-08시 하차인원':'7시',
'08시-09시 하차인원':'8시', '09시-10시 하차인원':'9시', '10시-11시 하차인원':'10시', '11시-12시 하차인원':'11시',
'12시-13시 하차인원':'12시', '13시-14시 하차인원':'13시', '14시-15시 하차인원':'14시', '15시-16시 하차인원':'15시',
'16시-17시 하차인원':'16시', '17시-18시 하차인원':'17시', '18시-19시 하차인원':'18시', '19시-20시 하차인원':'19시',
'20시-21시 하차인원':'20시', '21시-22시 하차인원':'21시', '22시-23시 하차인원':'22시', '23시-24시 하차인원':'23시',
'00시-01시 하차인원':'24시', '01시-02시 하차인원':'1시', '02시-03시 하차인원':'2시', '03시-04시 하차인원':'3시'}, inplace=True)

stopover = pd.concat([info, stopover], axis=1)
#stopover.head()
```

2

프로젝트 과정

데이터 정제 및 가공 (2)

- 승 하차 데이터에 합계 칼럼 추가
- 필요한 정보로만 데이터 프레임 구성
- 월 평균으로 일일 이용객 합계 만들기

```
import pandas as pd
import numpy as np

# 승하차 인원 합계
riding['합계'] = riding.sum(axis=1)
stopover['합계'] = stopover.sum(axis=1)
#riding.head()

# 승차
df_r = riding[['년월', '호선', '역명', '합계']]
#df_r.head()

# 하차
df_s = stopover[['년월', '호선', '역명', '합계']]
#df_s.head()

# 평균으로 일일 이용객 합계
# 승차
dfg_r = df_r.groupby(['년월', '호선', '역명'])['합계'].mean()
#dfg_r.head()
# 하차
dfg_s = df_s.groupby(['년월', '호선', '역명'])['합계'].mean()
#dfg_s.head()
```

데이터 정제 및 가공 (2)

- 데이터프레임 보기 좋게 하기 위해 index 재정렬
- 합계 칼럼을 이용객수로 바꿈 (보기 좋게 하기 위해)
- 승 하차 평균값으로 하루 평균 이용객의 수를 알아봄
- 데이터프레임 이용객수로 정렬
- 보기 좋게 하기 위해서 index 재정렬

```
# 데이터프레임 수정
# 승차
dfg2_r = pd.DataFrame(dfg_r)
dfg2_r = dfg2_r.reset_index()
#dfg2_r.head()

# 하차
dfg2_s = pd.DataFrame(dfg_s)
dfg2_s = dfg2_s.reset_index()
#dfg2_s.head()

# 합계 -> 이용객수
dfg2_r.columns = ['년월', '호선', '역명', '이용객수']
dfg2_s.columns = ['년월', '호선', '역명', '이용객수']

dfg3_r = pd.DataFrame(round(dfg2_r.groupby(['호선', '역명'])['이용객수'].mean()))
dfg3_s = pd.DataFrame(round(dfg2_s.groupby(['호선', '역명'])['이용객수'].mean()))

# 이용객수로 정렬
dfg3_r = dfg3_r.sort_values(by='이용객수', ascending=False)
dfg3_s = dfg3_s.sort_values(by='이용객수', ascending=False)
#dfg3_s.head(20)

# 인덱스 리빌드
dfg4_r = pd.DataFrame(dfg3_r)
dfg4_r = dfg4_r.reset_index()
#dfg4_r.head()

dfg4_s = pd.DataFrame(dfg3_s)
dfg4_s = dfg4_s.reset_index()
#dfg4_s.head()
```

2

프로젝트 과정

데이터 정제 및 가공 (3)

- 위도, 경도 데이터 로드
- 알아보기 쉽게 칼럼명 변경
- 역명의 공백 제거
- 1~9호선까지만 분석하기 위해 승 하차별로 분리

```
import pandas as pd
import numpy as np

dinfo = pd.read_csv('map_utf.csv')

dinfo.columns = ['역명', '호선', '위도', '경도']
dinfo.head()

dinfo['역명'] = dinfo['역명'].str.strip()
dinfo['역명'] = dinfo['역명'].str.replace(" ", "")

dfg4_r['호선'].unique()

# 1~9 호선 까지만 분리.
dfg4_r = dfg4_r[dfg4_r['호선'].isin(['1호선', '2호선', '3호선', '4호선', '5호선', '6호선', '7호선', '8호선', '9호선'])]
dfg4_s = dfg4_s[dfg4_s['호선'].isin(['1호선', '2호선', '3호선', '4호선', '5호선', '6호선', '7호선', '8호선', '9호선'])]

dfg4_r.loc[dfg4_r['호선']=='1호선', '호선'] = 1
dfg4_r.loc[dfg4_r['호선']=='2호선', '호선'] = 2
dfg4_r.loc[dfg4_r['호선']=='3호선', '호선'] = 3
dfg4_r.loc[dfg4_r['호선']=='4호선', '호선'] = 4
dfg4_r.loc[dfg4_r['호선']=='5호선', '호선'] = 5
dfg4_r.loc[dfg4_r['호선']=='6호선', '호선'] = 6
dfg4_r.loc[dfg4_r['호선']=='7호선', '호선'] = 7
dfg4_r.loc[dfg4_r['호선']=='8호선', '호선'] = 8
dfg4_r.loc[dfg4_r['호선']=='9호선', '호선'] = 9

dfg4_s.loc[dfg4_s['호선']=='1호선', '호선'] = 1
dfg4_s.loc[dfg4_s['호선']=='2호선', '호선'] = 2
dfg4_s.loc[dfg4_s['호선']=='3호선', '호선'] = 3
dfg4_s.loc[dfg4_s['호선']=='4호선', '호선'] = 4
dfg4_s.loc[dfg4_s['호선']=='5호선', '호선'] = 5
dfg4_s.loc[dfg4_s['호선']=='6호선', '호선'] = 6
dfg4_s.loc[dfg4_s['호선']=='7호선', '호선'] = 7
dfg4_s.loc[dfg4_s['호선']=='8호선', '호선'] = 8
dfg4_s.loc[dfg4_s['호선']=='9호선', '호선'] = 9
```

데이터 정제 및 가공 (3)

- 위도, 경도 데이터를 합침
- 상위, 하위 30개 데이터 추출

```
# 역정보와 이용객수 병합
# 승차
df_m_r = pd.merge(dfg4_r, dinfo, how='inner')
df_m_r.head()
# 하차
df_m_s = pd.merge(dfg4_s, dinfo, how='inner')
df_m_s

# 중복제거
df_m_r_2 = df_m_r.drop_duplicates('이용객수', keep='first')
df_m_s_2 = df_m_s.drop_duplicates('이용객수', keep='first')

# 상위, 하위 30개 역 표시
# 승차
dfm_top_r = df_m_r_2.head(30)
dfm_bottom_r = df_m_r_2.tail(30)

# 하차
dfm_top_s = df_m_s_2.head(30)
dfm_bottom_s = df_m_s_2.tail(30)
```

2

프로젝트 과정

시각화 (1)

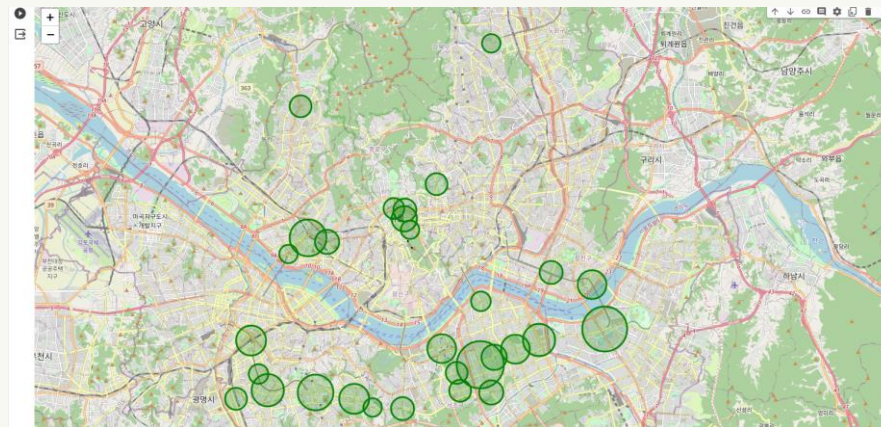
- Folium을 사용하여 새로운 지도 객체(map1)를 생성

```
# 지도시각화
import folium

# 승차 상위 이용객 30개역 표시
map1 = folium.Map(location=[37.5502, 126.982], zoom_start=11)

for item in dfm_top_r.index:
    lat = dfm_top_r.loc[item, '위도']
    long = dfm_top_r.loc[item, '경도']
    folium.CircleMarker([lat, long],
                        radius=dfm_top_r.loc[item, '이용객수']/70000,
                        color='green',
                        fill = True).add_to(map1)

map1
```



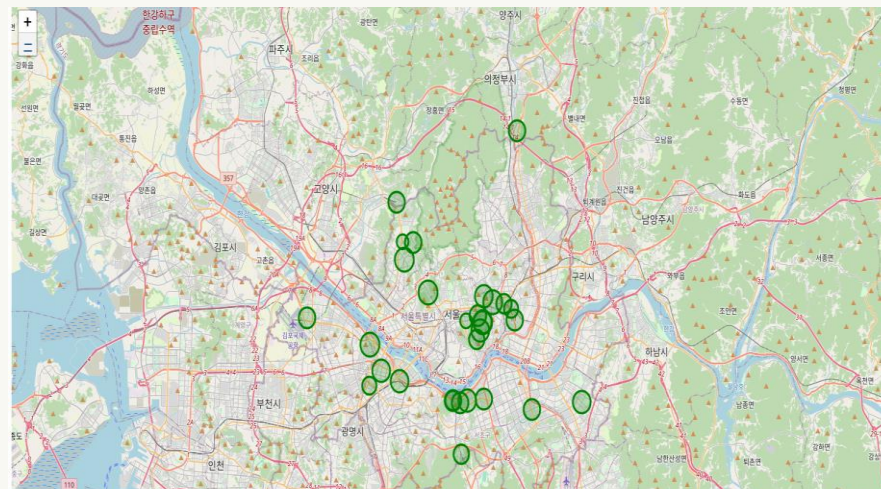
시각화 (1)

- Folium을 사용하여 새로운 지도 객체(map2)를 생성

```
# 승차 하위 이용객 30개역 표시
map2 = folium.Map(location=[37.5502, 126.982], zoom_start=11)

for item in dfm_bottom_r.index:
    lat = dfm_bottom_r.loc[item, '위도']
    long = dfm_bottom_r.loc[item, '경도']
    folium.CircleMarker([lat, long],
                        radius=dfm_bottom_r.loc[item, '이용객수']/20000,
                        color='green',
                        fill = True).add_to(map2)

map2
```



2

프로젝트 과정

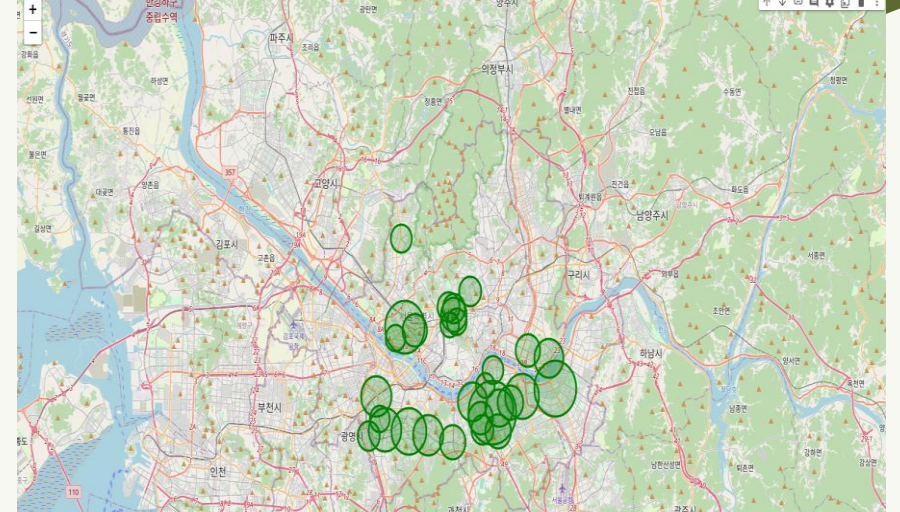
시각화 (2)

- Folium을 사용하여 새로운 지도 객체(map3)를 생성

```
# 하차 상위 이용객 30개역 표시
map3 = folium.Map(location=[37.5502, 126.982], zoom_start=11)

for item in dfm_top_s.index:
    lat = dfm_top_s.loc[item, '위도']
    long = dfm_top_s.loc[item, '경도']
    folium.CircleMarker([lat, long],
                        radius=dfm_top_s.loc[item, '이용객수']/70000,
                        color='green',
                        fill = True).add_to(map3)

map3
```



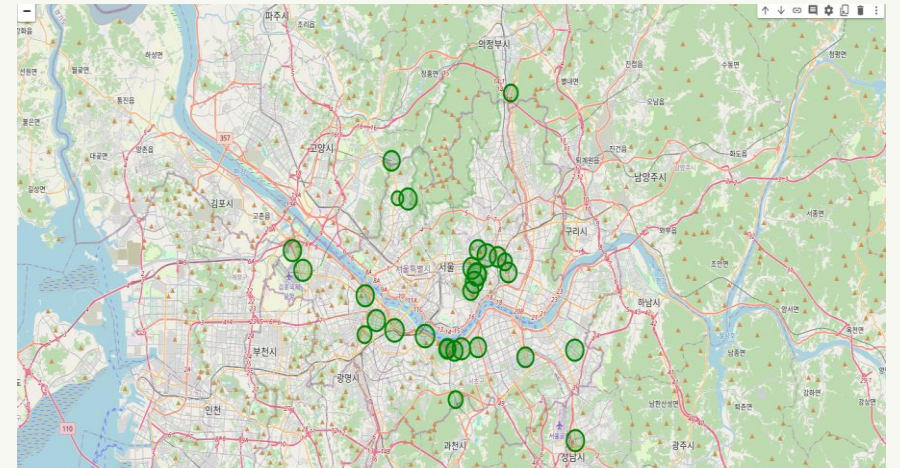
시각화 (2)

- Folium을 사용하여 새로운 지도 객체(map4)를 생성

```
# 하차 하위 이용객 30개역 표시
map4 = folium.Map(location=[37.5502, 126.982], zoom_start=11)

for item in dfm_bottom_s.index:
    lat = dfm_bottom_s.loc[item, '위도']
    long = dfm_bottom_s.loc[item, '경도']
    folium.CircleMarker([lat, long],
                        radius=dfm_bottom_s.loc[item, '이용객수']/20000,
                        color='green',
                        fill = True).add_to(map4)

map4
```



2

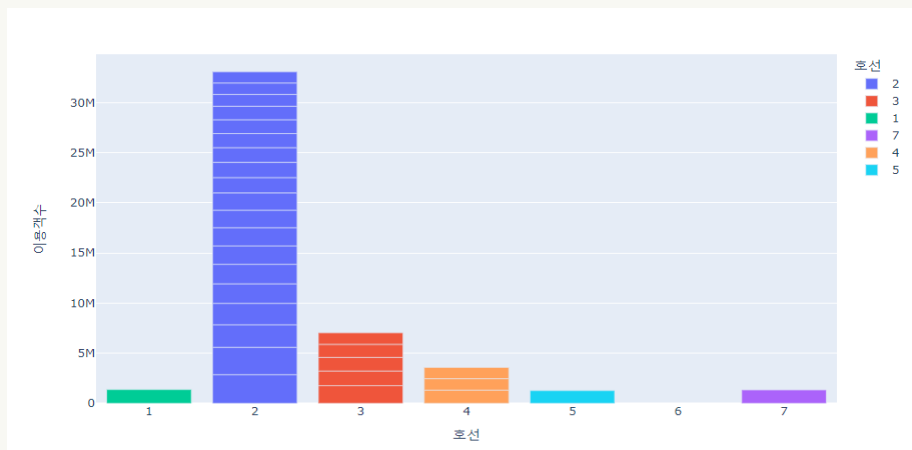
프로젝트 과정

시각화 (3)

- 호선 별 승차 이용객 수를 막대그래프로 생성 (상위30)

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

```
# 호선별 승차 이용객수 (상위30)
fig=px.bar(dfm_top_r, x='호선', y='이용객수',
            color='호선',
            color_continuous_scale=px.colors.diverging.Spectral)
fig.update_layout(width=900)
```

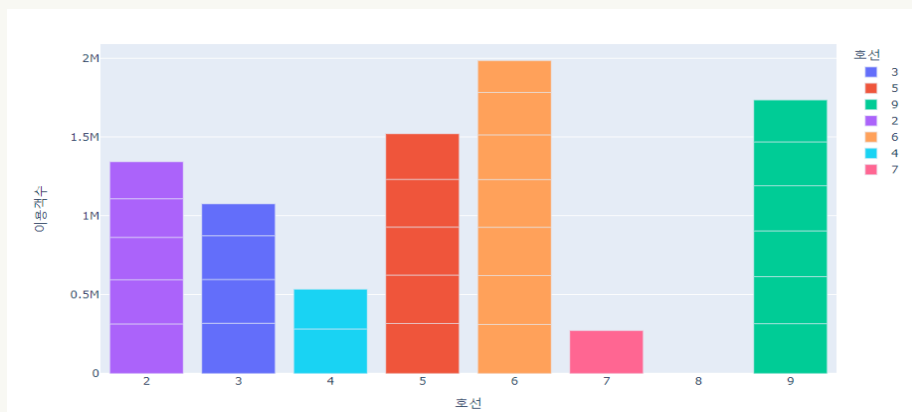


시각화 (3)

- 호선 별 승차 이용객 수를 막대그래프로 생성 (하위30)

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

```
# 호선별 승차 이용객수 (하위30)
fig=px.bar(dfm_bottom_r, x='호선', y='이용객수',
            color='호선',
            color_continuous_scale=px.colors.diverging.Spectral)
fig.update_layout(width=900)
```



2

프로젝트 과정

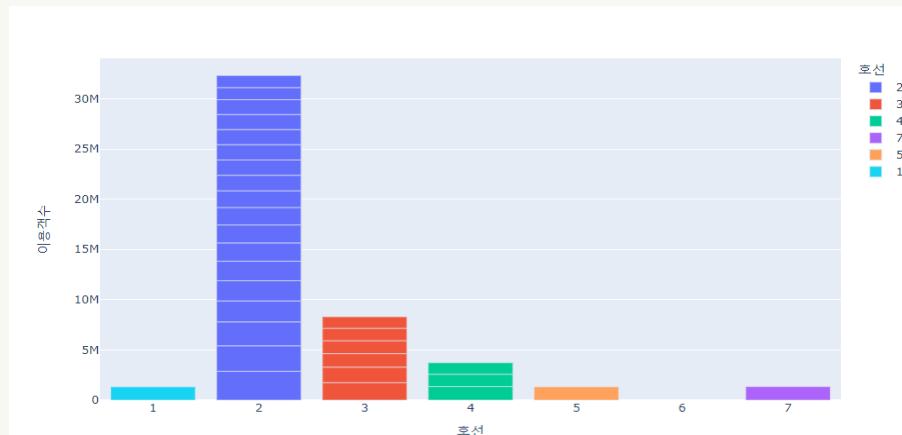
시각화 (4)

- 호선 별 하차 이용객 수를 막대그래프로 생성 (상위30)

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

호선별 하차 이용객수 (상위30)

```
fig=px.bar(dfm_top_s, x='호선', y='이용객수',
color='호선',
color_continuous_scale=px.colors.diverging.Spectral)
fig.update_layout(width=900)
```



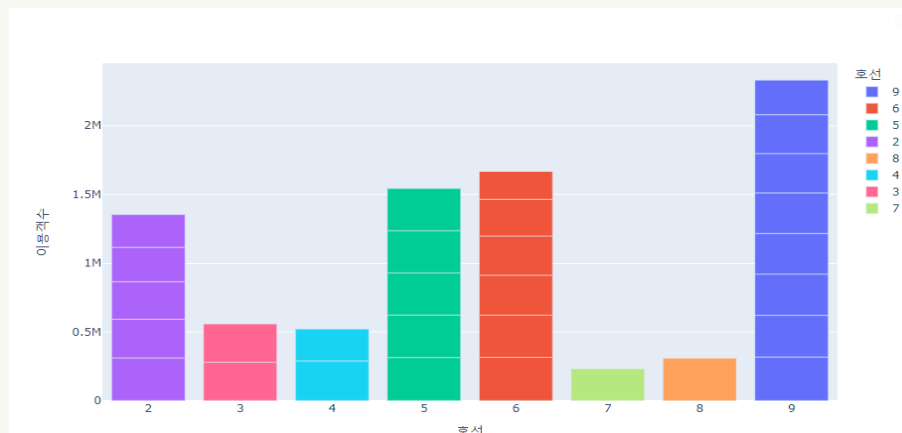
시각화 (4)

- 호선 별 하차 이용객 수를 막대그래프로 생성 (하위30)

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

호선별 하차 이용객수 (하위30)

```
fig=px.bar(dfm_bottom_s, x='호선', y='이용객수',
color='호선',
color_continuous_scale=px.colors.diverging.Spectral)
fig.update_layout(width=900)
```



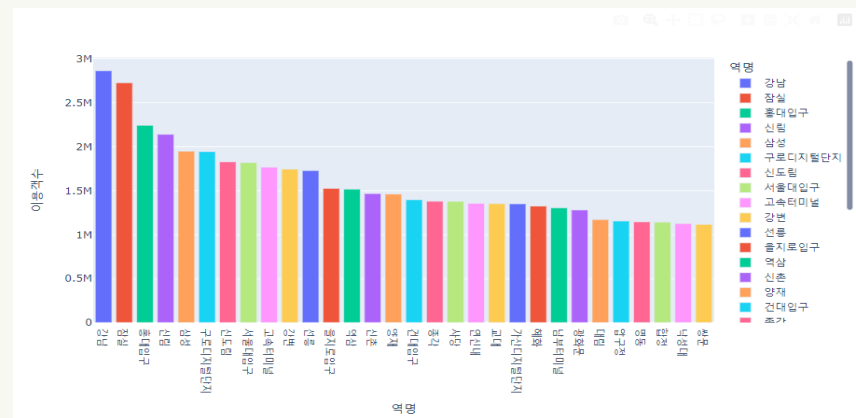
프로젝트 과정

시각화 (5)

- 역 별 승차 이용객 수를 막대그래프로 생성 (상위30)

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

```
# 역별 층차 이용객수 (상위30)
fig=px.bar(dfm_top_r, x='역명', y='이용객수',
color='역명',
color_continuous_scale=px.colors.diverging.Spectral)
fig.update_layout(width=900)
```



시각화 (5)

- 역 별 승차 이용객 수를 막대그래프로 생성 (하위30)

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

```
# 역별 승차 이용객수 (하위30)
fig=px.bar(dfm_bottom_r, x='역명', y='이용객수',
color='역명',
color_continuous_scale=px.colors.diverging.Spectral)
fig.update_layout(width=900)
```



프로젝트 과정

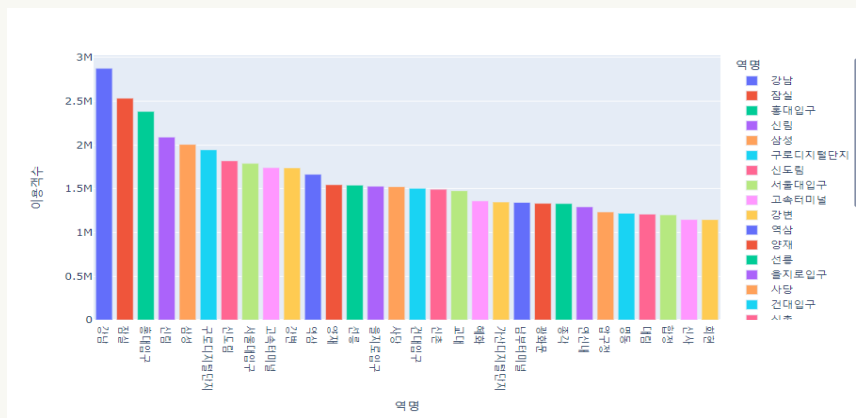
시각화 (6)

- 역 별 하차 이용객 수를 막대그래프로 생성 (상위30)

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

역할 하차 이용객수 (상위30)

```
fig=px.bar(dfm_top_s, x='역명', y='이용객수',
color='역명',
color_continuous_scale=px.colors.diverging.Spectral)
fig.update_layout(width=900)
```



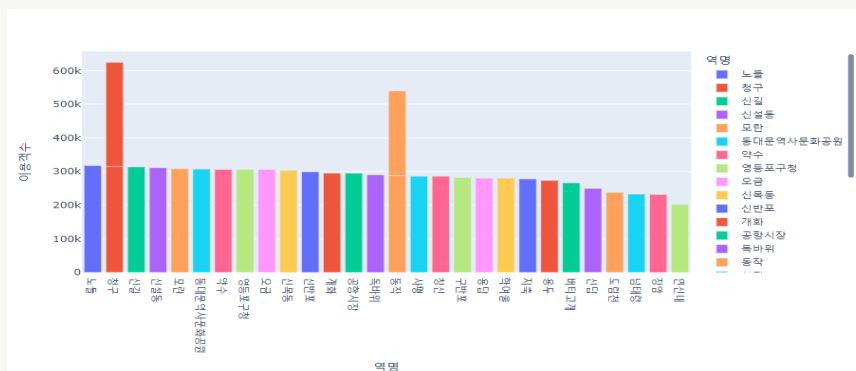
시각화 (6)

- 역 별 하차 이용객 수를 막대그래프로 생성 (하위30)

```
import pandas as pd
import plotly.express as px
import matplotlib.pyplot as plt
```

역별 하차 이용객수 (하위30)

```
fig=px.bar(dfm_bottom_s, x='역명', y='이용객수',
color='역명',
color_continuous_scale=px.colors.diverging.Spectral)
fig.update_layout(width=900)
```



3

결론

주요 결과: 분석 결과, 2호선이 가장 높은 평균 혼잡도를 보였다. 특히 강남역, 잠실역, 홍대입구에서 승 하차 둘 다 높은 혼잡도를 보였다.

정책 제언: 지하철 이용객 분산을 위해 추가적인 차량 배차 및 환승 시스템의 강화가 필요하다. 또한, 지역별 도시 계획을 고려한 새로운 교통 인프라의 도입이 혼잡을 완화할 수 있는 방안으로 제시된다.



