지하철 칸 별 혼잡도 분석을 통한 지하철 이용 만족도 향상

경영학과 강민석
문헌정보학과 김민우
문헌정보학과 안지원
문헌정보학과 정만기

칸 별 혼잡도를 분석하여 혼잡도를 줄여보자

지하철 칸 별 혼잡도를 분석하여
비교적 한산한 칸으로 사람들을 유도해보기

주제선정

문제상황정의
프로젝트 진행 개요

발표 개요

서울 지하철 4호선 대상
실시간 데이터 분석

데이터분석 과정

데이터 선별 과정
실시간 칸 별 혼잡도 데이터 분석
시각화

결론

시사점 및 한계점
데이터분석 실패 사례

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

칸 별 혼잡도를 분석하여
혼잡도를 줄여보자

지하철 칸 별 혼잡도를 분석하여
비교적 한산한 칸으로 사람들을 유도해보기

발표 개요

서울 지하철 4호선 대상
실시간 데이터 분석

1. 주제 선정

문제상황정의
프로젝트 진행 개요

주제선정

문제상황정의
프로젝트 진행 개요

데이터분석 과정

데이터 선별 과정
칸 별 혼잡도 차이 분석

결론

시사점
한계점

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

1.1 문제 정의

서울시의 오랜 고민거리, 출퇴근 시간 지하철 인구 밀집도

한겨레 특집 1438호

서울, 밀도를 줄여야 사람이 산다

도쿄보다 좁고 파리보다 인구 4배 많은 서울…
혼잡시간대 지하철도 인구 밀집도 높아 사고 위험

기관별 주요 철도차량 구매단가

	차종	세부차종	운영노선	사업비	량수	단가
김포골드라인운영	전동	경량전철	김포골드라인	462억원	12량	38.5억원
서울교통공사	전동	중전철	1호선	2240억원	160량	14억원
	전동	중전철	5호선	2800억원	200량	14억원
	전동	중전철	6호선 1·2차	4592억원	328량	14억원
	전동	중전철	7호선 1·2차	5152억원	368량	14억원
부산교통공사	전동	경전철	사상하단선	294억원	18량	16.3억원
에스지레일	일반	여객(EMU-180)	GTX-A노선	3339억원	120량	27.8억원
한국철도공사	고속	EMU-320	남부내륙선	5831억원	104량	56억원
에스알	고속	EMU-320	고속선	5375억원	112량	48억원

*자료: 국토교통부 철도차량 중장기(2023~2027) 구매계획 및 지자체 · 기관 취합

현실적인 해결책 마련 필요

하지만 지하철 배차간격 조정과 증차 등의 해결책은 굉장히 까다롭고 비용도 많이 든다.

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

1.1 문제 정의

서울시의 오랜 고민거리, 출퇴근 시간 지하철 인구 밀집도

현실적인 해결책 마련 필요

현재 2호선은 열차 내부에 실시간 칸 별 혼잡도를 표시해주고 있음

열차 1칸당 정원 : 160명 *최대 수용 인원과 다름

여유 ~ 약 127명 ~79%

보통 약 128~207명 80~129%

혼잡 약 208명~ 130%~

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

1.1 문제 정의

서울시의 오랜 고민거리, 출퇴근 시간 지하철 인구 밀집도

현실적인 해결책 마련 필요

현재 2호선은 열차 내부에 실시간 칸 별 혼잡도를 표시해주고 있음

실시간 데이터는 신형 전동차에서만 제공 가능

신형 전동차 바닥의 무게를 측정하는 하중감지 센서로 칸 별 탑승객 수 역산

내부 무게 ÷ 성인 평균 체중 = 탑승객 수

여전히 한계점 존재

이 마저도 열차 내부에만 표시되어 미리 혼잡한 칸을 피해서 타기 힘듦

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

1.2 프로젝트 진행 개요

서울시의 오랜 고민거리, 출퇴근 시간 지하철 인구 밀집도

PUZZLE

현실적인 해결책

'지오비전 퍼즐'에서 여러 데이터를 종합해 칸 별 혼잡도 통계 데이터를 제공해준다.

신형 열차로 바꾸지 않아도 칸 별 혼잡도 예측 가능

실시간 칸 별 혼잡도 정보를 제공하지 않더라도
특정 시간대에 어느 칸에 사람이 몰릴 것인지 예측이 가능해져
승객을 골고루 분산시킬 수 있다.

분석 방향

칸 별 혼잡도에 큰 영향을 주는 시간, 역, 지하철 노선, 요일을
독립변수로 설정

특정 역을 기준으로 가장 많이 몰리는 칸과 비교적 한적한 칸을 분리.
붐비는 시간대에 사람들을 한산한 칸으로 이동할 수 있도록
안내하는 것을 목표로 함

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

1.2 프로젝트 진행 개요

서울시의 오랜 고민거리, 출퇴근 시간 지하철 인구 밀집도

PUZZLE

현실적인 해결책

'지오비전 퍼즐'에서 여러 데이터를 종합해 칸 별 혼잡도 통계 데이터를 제공해준다.

분석 방향

칸 별 혼잡도에 큰 영향을 주는 시간, 역, 지하철 노선을 독립변수로 설정

특정 역을 기준으로 가장 많이 몰리는 칸과 비교적 한적한 칸을 분리.
붐비는 시간대에 사람들을 한산한 칸으로 이동할 수 있도록
안내하는 것을 목표로 함

기대 효과

이를 통해 열차 증편, 배차간격 조정 없이도 좀 더 쾌적하게
지하철을 이용할 수 있을 것이다.

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

1.2 프로젝트 진행 개요

서울시의 오랜 고민거리, 출퇴근 시간 지하철 인구 밀집도

2. 데이터 분석 과정

데이터 선별 과정
칸 별 혼잡도 차이 분석
혼잡도 예측

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.1 데이터 선별 과정

Q) 왜 '4호선'인가?

호선별 평균 혼잡도

```
#호선별 평균 혼잡도
df2 = pd.read_excel('/content/drive/MyDrive/Data Science_CAU/Team Project/지하철+혼잡도_20230516171821.xlsx')
df2_1 = df2[df2['시점'] == '2021'].reset_index(drop = True)
df2_2 = df2_1.loc[:, '서울교통공사':'우이신설도시철도']
df2_2.columns = ['1~8호선평균', '1호선', '2호선', '3호선', '4호선', '5호선', '6호선', '7호선', '8호선', '9호선소계', '9호선1단계', '9호선2단계', '9호선3단계', '우이신설도시철도']
df2_3 = df2_2.replace('-', np.nan)
df2_3
```

[지하철+혼잡도_20230516171821.xlsx 파일]

시점	서울교통공사									서울시메트로9호선	
	평균	1호선	2호선	3호선	4호선	5호선	6호선	7호선	8호선	소계	1단계
2005	-	-	-	-	-	-	-	-	-	-	-
2007	-	-	-	-	-	-	-	-	-	-	-
2009	-	-	-	-	-	-	-	-	-	-	-
2011	-	-	-	-	-	-	-	-	-	-	-
2013	-	-	-	-	-	-	-	-	-	115(196)	-
2015	-	-	-	-	-	-	-	-	-	114(205)	-
2017	135	105	170	140	160	110	111	156	130	91(175)	-
2019	134	106	173	144	162	107	106	146	131	-	87(145)
2021	127	84	149	141	151	132	96	127	134	-	75(135)

2021년 데이터 사용
9호선 혼잡도는 가장 혼잡한 상위 5개역의 목측기준으로 07:00~08:45 평균값이며, 전체 노선의 평균 혼잡도 산출시 제외.( )는 급행열차의 혼잡도임
우이신설선은 혼잡 4개역(솔샘역 - 성신여대입구역) 기준, 07:00~09:00 평균값임

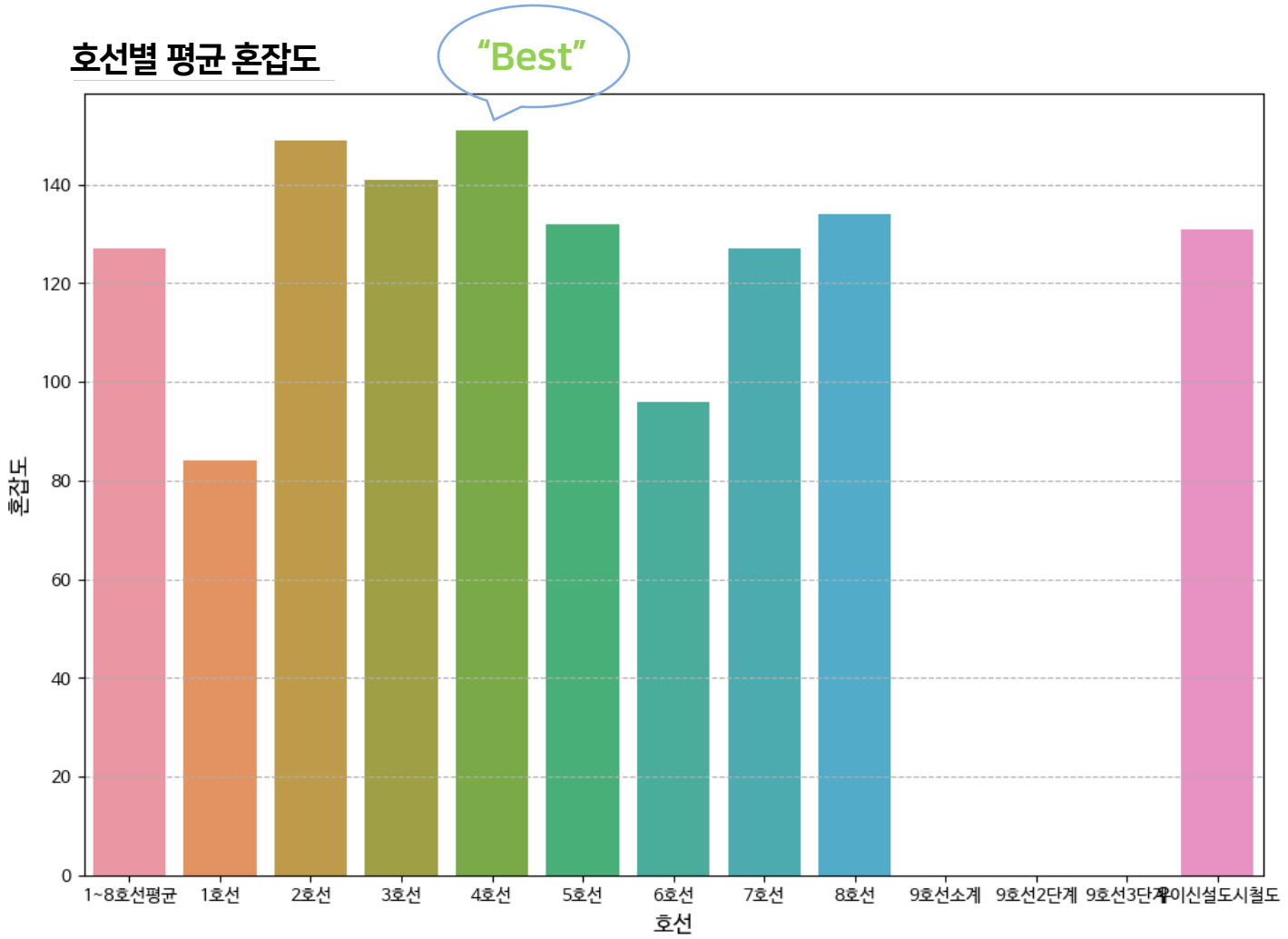
유독 2021년만 2호선보다 4호선이 혼잡도가 근소하게나마 높게 나옴.

```
plt.figure(figsize = (12,8))
sns.barplot(data = df2_3)
plt.grid(True, axis = 'y', linestyle = '--')
plt.xlabel('호선', fontsize = 13)
plt.ylabel('혼잡도', fontsize = 13)
```

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.1 데이터 선별 과정

Q) 왜 '4호선'인가?



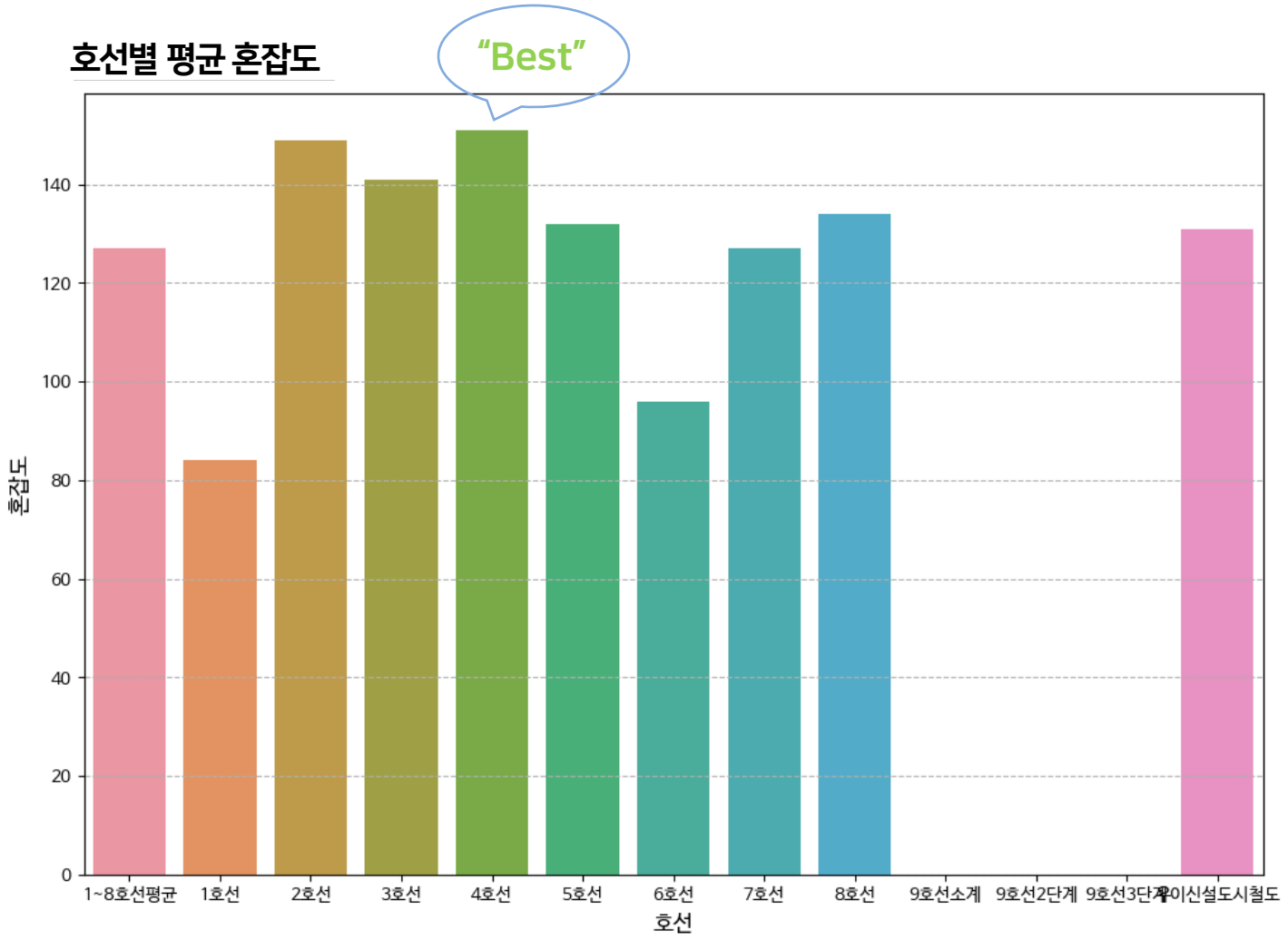
'4호선'을 대상으로 분석하는 것이 Best

이미 2,3호선은 실시간 칸 별 혼잡도 데이터가 제공되고 있음
그러므로 2, 3호선 제외하고 평균 혼잡도가 가장 높은 4호선을 분석.

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.1 데이터 선별 과정

Q) 왜 '4호선'인가?



'4호선'을 대상으로 분석하는 것이 Best

이미 2,3호선은 실시간 칸 별 혼잡도 데이터가 제공되고 있음
그러므로 2, 3호선 제외하고 평균 혼잡도가 가장 높은 4호선을 분석.

4호선 중 가장 붐비는 역과 붐비는 시간을 찾아보자

자료 양이 너무 많으므로 4호선 중에서도 가장 붐비는 역을 먼저 특정
해당 역을 기준으로 데이터 분석 진행

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.1 데이터 선별 과정

4호선 중 가장 붐비는 역을 찾아보자

4호선 역별 혼잡도 - 주말

```
#기본 데이터 전처리
scdata = pandas.read_csv('서울교통공사_지하철혼잡도정보_20221231.csv', encoding = 'cp949')
scdata = scdata[(scdata['호선'] == 4) & (scdata['요일구분'] != '평일')]
scdata = scdata.drop(['연번', '요일구분', '호선', '역번호', '상하구분'], axis = 1)
scdata = scdata.rename(columns = {'출발역' : '역명'})
scdata
```

[서울교통공사_지하철혼잡도정보_20221231.csv 파일]

연번	요일구분	호선	역번호	출발역	상하구분	5시30분	6시00분	6시30분	7시00분	7시30분	8시00분	8시30분	9시00분	9시30분	10시00분	10시30분	11시00분	11시30분	12시00분	12시30분
1	평일	1	150	서울역	상선	7.3	18.1	18.1	30.9	56.6	69.1	82.7	57.5	52.3	36.9	31.8	30.9	30.9	25.8	39.2
2	평일	1	150	서울역	하선	11.5	11	13.2	21.4	38	38.7	24.6	24	19.8	18.8	21	19.4	19.5	23.3	27.2
3	평일	1	151	시청	상선	6.6	15.4	14.7	25	42.4	55	56	44.3	48.2	30.5	29.8	29.2	28.6	26.2	36.6
4	평일	1	151	시청	하선	9	9.1	14.6	20	35.4	35.7	26.9	26	20.6	18.9	19.7	18.6	18.2	22.3	24.8
5	평일	1	152	종각	상선	6.3	14.4	10.7	17.6	26.7	36	31.2	28	36.5	27.4	24.8	26.4	26.4	25.9	31.8
6	평일	1	152	종각	하선	9.8	9.6	15.6	24.4	44.7	47.5	45.6	34.6	26	20.6	21.4	19.2	18.8	22.2	25.6
7	평일	1	153	종로3가	상선	6.2	13.8	10.6	17.2	25.9	40.7	32.5	28.7	37.2	28.4	27.3	25.8	26.1	27.4	35.2
8	평일	1	153	종로3가	하선	11.6	9.1	19.6	28.7	58.3	61.2	72	48.9	36.2	23.6	25.2	20.4	21.3	23.8	24.8
9	평일	1	154	종로5가	상선	5.7	12.8	8.7	13.1	20.4	31.4	23.1	24.1	27.4	24.9	21.8	22.2	22.6	26.7	28
10	평일	1	154	종로5가	하선	13.5	9.8	21.4	34.6	61.7	72	74.4	56.3	39.9	28.7	31.3	25.6	22.8	31.6	30.6
11	평일	1	155	동대문	상선	7.8	12.7	8.5	12.7	20.2	26.8	25.8	21.7	25.6	24.2	19.2	22.9	22.8	25.3	27.9
12	평일	1	155	동대문	하선	13.2	14	19.8	36.1	69.1	75.9	81	57.8	43.2	31.3	32.4	25.7	23	28.1	33
13	평일	1	156	신설동	상선	7.1	12.5	7.5	9.6	13.9	20	19.3	16.6	27.4	18.2	17.7	24.8	18.8	20	24.6
14	평일	1	156	신설동	하선	20.2	20.1	28.4	35.5	63.6	76.3	60.3	47.9	41.1	31.9	29.1	21.5	21.1	26.4	29.2
15	평일	1	157	제기동	상선	8.2	10.1	6.8	8	12.6	18.2	16.8	13.9	23	14.9	14.2	18.4	16.2	16.7	19.8
16	평일	1	157	제기동	하선	19.5	19.1	24.6	36.1	60.2	70.7	60.1	46	35	34.1	28.1	20.4	20	25.8	28
17	평일	1	158	청량리	상선	7.6	6	4.8	6.5	10.6	15.4	13.3	10.8	18.1	10.9	12.7	15.6	13	14.9	16.7
18	평일	1	158	청량리	하선	19.3	18.1	27.1	36.4	63.8	64.9	59	44.6	34.2	34.2	27.1	16.1	21.1	22.8	22.2

u. All Rights Reserved.

2.1 데이터 선별 과정

4호선 중 가장 붐비는 역을 찾아보자

4호선 역별 혼잡도 - 주말

시간대별 혼잡도 평균 측정

```
#역 혼잡도 데이터 전처리
time_label = [col for col in scdata.columns if col != '역명']
scdata_station = scdata.copy()
scdata_station['혼잡도'] = scdata_station[time_label].mean(axis = 1)
scdata_station = scdata_station.drop(time_label, axis = 1)
scdata_station
```

역별로 합친 후 전체 평균 측정

```
#역 혼잡도 데이터 분석
scdata_station_mean = scdata_station.groupby('역명')['혼잡도'].mean()
scdata_station_mean = scdata_station_mean.sort_values(ascending = False).to_frame()
scdata_station_mean.nlargest(5,'혼잡도')
```

[결과]

역명	혼잡도
남태령	45.626351
동대문	42.656081
혜화	39.890541
동대문역사문화공원	38.997973
한성대입구	36.476351

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.1 데이터 선별 과정

4호선 중 가장 붐비는 역을 찾아보자

[4호선 역별 혼잡도] - 평일

역명	혼잡도
동대문	50.064103
남태령	48.396154
혜화	48.297436
한성대입구	45.343590
동대문역사문화공원	44.229487

남태령, 혜화역, 성신여대역 기준으로 데이터 분석 진행

평일은 동대문, 남태령, 혜화, 한성대입구, 동대문역사문화공원 순으로 평균 혼잡도가 가장 높음

이 중 상위 3개 역(동대문, 남태령, 혜화)을 대상으로 분석 진행

[4호선 역별 혼잡도] - 주말

역명	혼잡도
남태령	45.626351
동대문	42.656081
혜화	39.890541
동대문역사문화공원	38.997973
한성대입구	36.476351

남태령, 동대문역, 혜화역 기준으로 데이터 분석 진행

주말 또한 남태령, 동대문, 혜화, 동대문역사문화공원, 한성대입구 순으로 평균 혼잡도가 가장 높음

이 중 상위 3개 역을 대상으로 분석 진행

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.1 데이터 선별 과정

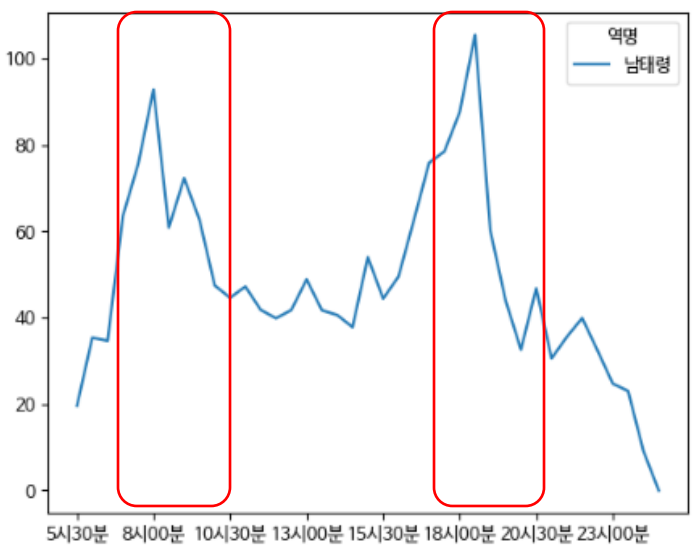
4호선 중 가장 붐비는 시간을 찾아보자

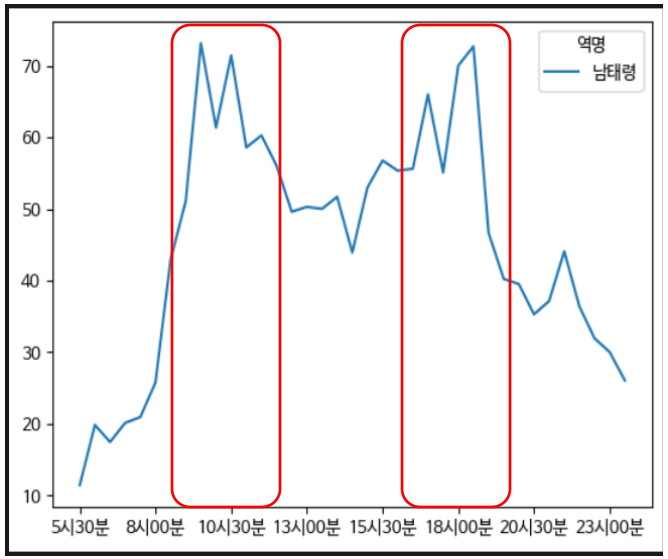
[남태령 역] 혼잡시간 데이터 정리

```
#남태령 역 혼잡시간 데이터 전처리
scdata_namt = scdata[scdata.역명 == '남태령']

#남태령 역 혼잡시간 데이터 분석

scdata_namt_time = scdata_namt.groupby('역명').mean()
scdata_namt_time = scdata_namt_time.T
scdata_namt_time.plot()
```

*** 전반적으로 평일이 주말보다 훨씬 혼잡도가 높음**

[남태령 역] 결과





[평일]

출퇴근 시간이 가장 혼잡

출근: 8~9시
퇴근: 16시~17시로 특정

[주말]

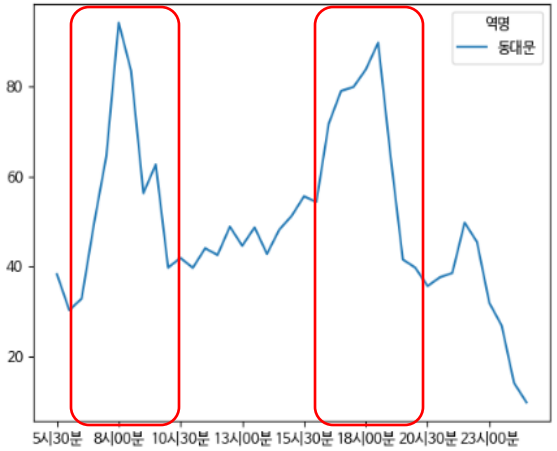
10시 ~ 11시 / 18시 ~ 19시가 가장 혼잡

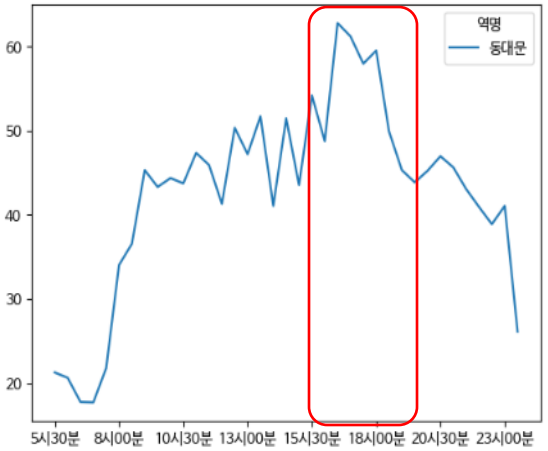
Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.1 데이터 선별 과정

4호선 중 가장 붐비는 시간을 찾아보자

[동대문 역] 결과





[평일]

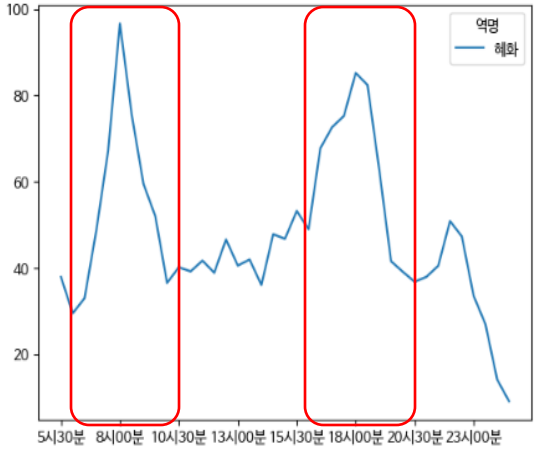
출퇴근시간이 가장 혼잡

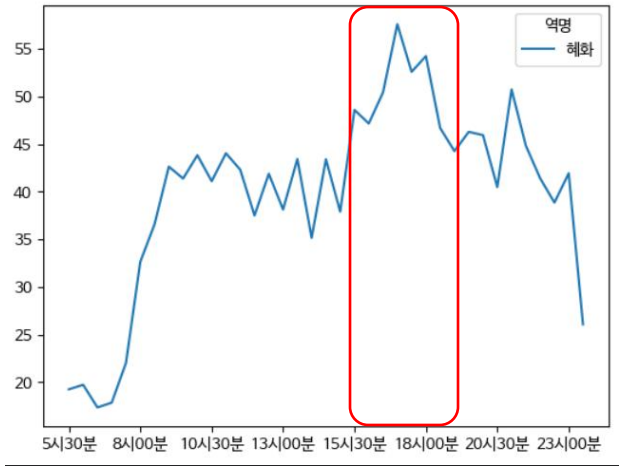
출근: 8~9시
퇴근: 16시~17시로 특정

[주말]

17~18시가 가장 혼잡

[혜화 역] 결과





[평일]

출퇴근시간이 가장 혼잡

출근: 8~9시
퇴근: 16시~17시로 특정

[주말]

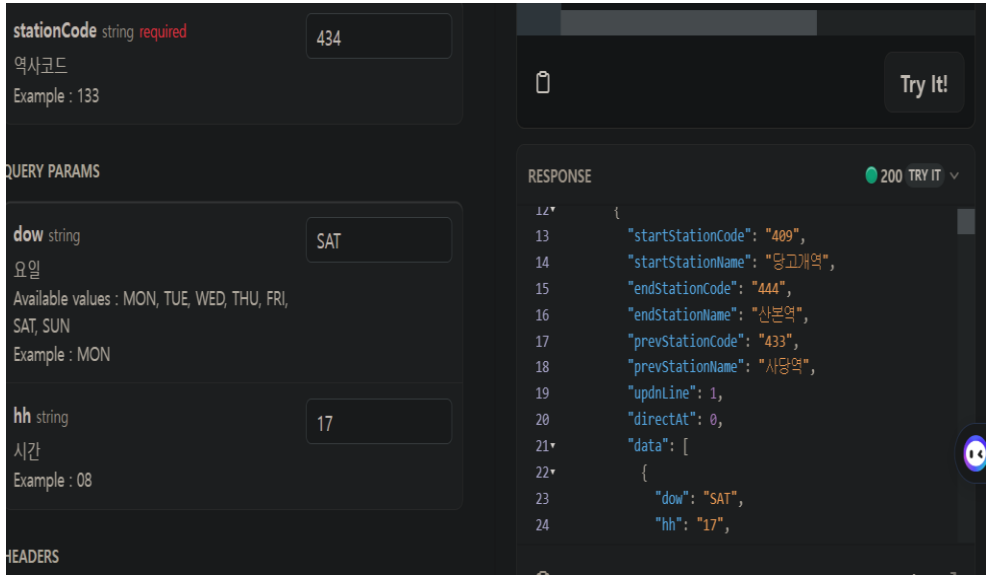
17~18시가 가장 혼잡

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.2 실시간 칸 별 혼잡도 데이터 분석

시간별 데이터 분석

[SK open API]



https://skopenapi.readme.io/reference/%EC%A7%84%EC%9E%85-%EC%97%AD%EC%82%AC-%EA%B8%B0%EC%A4%80-%EC%B9%B8-%ED%98%BC%EC%9E%A1%EB%8F%84

칸 별 혼잡도 구하는 함수 정의

```
[27] # 데이터 프레임 생성을 위한 데이터 가공
     def process_data(data):
       processed_data = []
       for stat in data['contents']['stat']:
         for datum in stat['data']:
           if any(datum['congestionCar']):  # congestionCar 값이 모두 0인 데이터는 제외
                 processed_data.append({
                     '현재역': data['contents']['stationName'],
                     '출발역': stat['startStationName'],
                     '종착역': stat['endStationName'],
                     '이전역': stat['prevStationName'],
                     '요일': datum['dow'],
                     'congestionCar': datum['congestionCar'],
                     'hh': datum['hh'],
                     'mm': datum['mm'],
                 })
       df = pd.DataFrame(processed_data)
       grouped_df = df.groupby(['현재역', '출발역', '종착역', '이전역', '요일'])
       result = []
       for name, group in grouped_df:
         congestion_data = [sum(car) / len(group) for car in zip(*group['congestionCar'])]
         congestion_data = [int(val) for val in congestion_data]  # 소수점 이하 버림
         result.append({
             '현재역': name[0],
             '출발역': name[1],
             '종착역': name[2],
             '이전역': name[3],
             '요일': name[4],
             '평균혼잡도': congestion_data,
         })

       # 최종 결과 데이터 프레임 생성
       result_df = pd.DataFrame(result)

       # 칸별로 데이터를 분리
       for i in range(1, 11):
         result_df[f'{i}호칸'] = result_df['평균혼잡도'].apply(lambda x: x[i-1])

       # 불필요한 '평균혼잡도' 컬럼 제거
       result_df = result_df.drop(columns=['평균혼잡도'])

       # 'prevStationCode'의 값이 낮은 순으로 정렬
       result_df = result_df.sort_values(by=["이전역"])

       return result_df
```

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.2 실시간 칸 별 혼잡도 데이터 분석

시간별 데이터 분석

칸 별 혼잡도 구하는 함수 정의

```
[27] # 데이터 프레임 생성을 위한 데이터 가공
     def process_data(data):
       processed_data = []
       for stat in data['contents']['stat']:
         for datum in stat['data']:
           if any(datum['congestionCar']):  # congestionCar 값이 모두 0인 데이터는 제외
                processed_data.append({
                    '현재역': data['contents']['stationName'],
                    '출발역': stat['startStationName'],
                    '종착역': stat['endStationName'],
                    '이전역': stat['prevStationName'],
                    '요일': datum['dow'],
                    'congestionCar': datum['congestionCar'],
                    'hh': datum['hh'],
                    'mm': datum['mm'],
                })
       df = pd.DataFrame(processed_data)
       grouped_df = df.groupby(['현재역', '출발역', '종착역', '이전역', '요일'])
       result = []
       for name, group in grouped_df:
         congestion_data = [sum(car) / len(group) for car in zip(*group['congestionCar'])]
         congestion_data = [int(val) for val in congestion_data]  # 소수점 이하 버림
         result.append({
             '현재역': name[0],
             '출발역': name[1],
             '종착역': name[2],
             '이전역': name[3],
             '요일': name[4],
             '평균혼잡도': congestion_data,
         })

        # 최종 결과 데이터 프레임 생성
       result_df = pd.DataFrame(result)

       # 칸별로 데이터를 분리
       for i in range(1, 11):
         result_df[f'{i}호칸'] = result_df['평균혼잡도'].apply(lambda x: x[i-1])

       # 불필요한 '평균혼잡도' 컬럼 제거
       result_df = result_df.drop(columns=['평균혼잡도'])

       # 'prevStationCode'의 값이 낮은 순으로 정렬
       result_df = result_df.sort_values(by=["이전역"])

       return result_df
```

함수 적용 예시

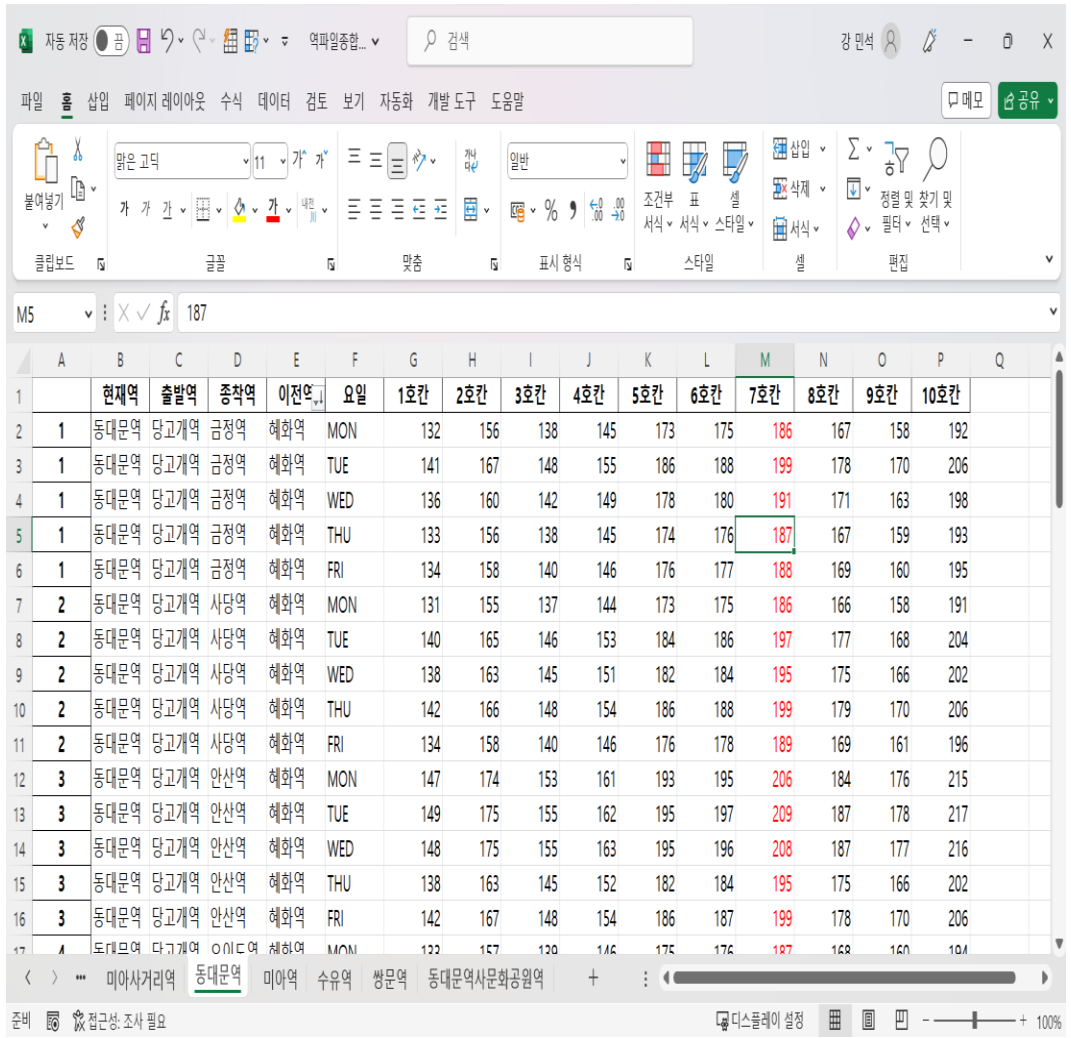
	현재역	출발역	종착역	이전역	요일	1호칸	2호칸	3호칸	4호칸	5호칸	6호칸	7호칸	8호칸	9호칸	10호칸
0	남태령역	당고개역	안산역	사당역	SAT	51	69	68	68	57	57	49	30	34	21
1	남태령역	당고개역	오이도역	사당역	SAT	67	90	95	86	74	71	63	37	41	27
2	남태령역	오이도역	당고개역	선바위역	SAT	61	71	69	76	87	93	74	67	87	72
3	남태령역	당고개역	안산역	사당역	SUN	47	64	61	60	52	51	44	28	30	19
4	남태령역	당고개역	오이도역	사당역	SUN	54	74	77	71	62	60	52	31	34	23
5	남태령역	오이도역	당고개역	선바위역	SUN	41	48	43	52	57	64	49	45	61	44
6	남태령역	당고개역	오이도역	사당역	SAT	57	66	62	62	61	67	58	47	46	32
7	남태령역	안산역	당고개역	선바위역	SAT	49	60	56	52	51	49	47	45	57	52
8	남태령역	오이도역	당고개역	선바위역	SAT	70	89	81	75	72	68	65	63	80	76
9	남태령역	당고개역	오이도역	사당역	SUN	41	47	44	45	44	48	44	34	34	23
10	남태령역	안산역	당고개역	선바위역	SUN	49	60	58	53	52	52	50	47	59	53
11	남태령역	오이도역	당고개역	선바위역	SUN	57	69	65	59	58	56	53	52	64	60

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.2 실시간 칸 별 혼잡도 데이터 분석

시간별 데이터 분석

[평일]



	A	현재역	출발역	종착역	이전역	요일	1호칸	2호칸	3호칸	4호칸	5호칸	6호칸	7호칸	8호칸	9호칸	10호칸
2	1	동대문역	당고개역	금정역	혜화역	MON	132	156	138	145	173	175	186	167	158	192
3	1	동대문역	당고개역	금정역	혜화역	TUE	141	167	148	155	186	188	199	178	170	206
4	1	동대문역	당고개역	금정역	혜화역	WED	136	160	142	149	178	180	191	171	163	198
5	1	동대문역	당고개역	금정역	혜화역	THU	133	156	138	145	174	176	187	167	159	193
6	1	동대문역	당고개역	금정역	혜화역	FRI	134	158	140	146	176	177	188	169	160	195
7	2	동대문역	당고개역	사당역	혜화역	MON	131	155	137	144	173	175	186	166	158	191
8	2	동대문역	당고개역	사당역	혜화역	TUE	140	165	146	153	184	186	197	177	168	204
9	2	동대문역	당고개역	사당역	혜화역	WED	138	163	145	151	182	184	195	175	166	202
10	2	동대문역	당고개역	사당역	혜화역	THU	142	166	148	154	186	188	199	179	170	206
11	2	동대문역	당고개역	사당역	혜화역	FRI	134	158	140	146	176	178	189	169	161	196
12	3	동대문역	당고개역	안산역	혜화역	MON	147	174	153	161	193	195	206	184	176	215
13	3	동대문역	당고개역	안산역	혜화역	TUE	149	175	155	162	195	197	209	187	178	217
14	3	동대문역	당고개역	안산역	혜화역	WED	148	175	155	163	195	196	208	187	177	216
15	3	동대문역	당고개역	안산역	혜화역	THU	138	163	145	152	182	184	195	175	166	202
16	3	동대문역	당고개역	안산역	혜화역	FRI	142	167	148	154	186	187	199	178	170	206

[주말]

```
#남태령 역 18시_토요일 데이터
result_NT_18_SAT = process_data(NTdata18)

#남태령 역 18시_일요일 데이터
result_NT_18_SUN = process_data(SUN_NTdata18)
```

아까 정의했던 함수 사용

```
# 세 역 데이터 전부 종합
result_df_all = pd.concat([result_df_NT,result_df_H, result_df_D]).reset_index(drop = True)
```

	현재역	출발역	종착역	이전역	요일	1호칸	2호칸	3호칸	4호칸	5호칸	6호칸	7호칸	8호칸	9호칸	10호칸
0	남태령역	당고개역	안산역	사당역	SAT	51	69	68	68	57	57	49	30	34	21
1	남태령역	당고개역	오이도역	사당역	SAT	67	90	95	86	74	71	63	37	41	27
2	남태령역	오이도역	당고개역	선바위역	SAT	61	71	69	76	87	93	74	67	87	72
3	남태령역	당고개역	안산역	사당역	SUN	47	64	61	60	52	51	44	28	30	19
4	남태령역	당고개역	오이도역	사당역	SUN	54	74	77	71	62	60	52	31	34	23
5	남태령역	오이도역	당고개역	선바위역	SUN	41	48	43	52	57	64	49	45	61	44
6	남태령역	당고개역	오이도역	사당역	SAT	57	66	62	62	61	67	58	47	46	32
7	남태령역	안산역	당고개역	선바위역	SAT	49	60	56	52	51	49	47	45	57	52
8	남태령역	오이도역	당고개역	선바위역	SAT	70	89	81	75	72	68	65	63	80	76
9	남태령역	당고개역	오이도역	사당역	SUN	41	47	44	45	44	48	44	34	34	23
10	남태령역	안산역	당고개역	선바위역	SUN	49	60	58	53	52	52	50	47	59	53
11	남태령역	오이도역	당고개역	선바위역	SUN	57	69	65	59	58	56	53	52	64	60
12	혜화역	남태령역	진접역	동대문역	SAT	46	72	74	79	72	66	63	55	62	51
13	혜화역	사당역	당고개역	동대문역	SAT	42	67	68	74	67	61	58	51	57	47
14	혜화역	사당역	진접역	동대문역	SAT	46	72	74	79	72	66	63	55	62	51
15	혜화역	안산역	당고개역	동대문역	SAT	46	72	74	79	72	66	62	55	61	51
16	혜화역	오이도역	당고개역	동대문역	SAT	48	75	77	83	75	69	66	58	64	53
17	혜화역	당고개역	사당역	한성대입구역	SAT	21	31	26	33	33	37	44	35	32	37
18	혜화역	당고개역	오이도역	한성대입구역	SAT	26	38	32	40	41	44	53	42	39	44
19	혜화역	진접역	사당역	한성대입구역	SAT	23	34	29	36	36	40	48	38	35	39
20	혜화역	진접역	서울역	한성대입구역	SAT	23	34	29	36	36	40	48	38	35	39

hts Reserved.

2.3 시각화

요일별 데이터 분석 - 평일

[시각화 코드]

```python
# 요일을 순서대로 정렬하기 위한 목록
weekdays = ['MON', 'TUE', 'WED', 'THU', 'FRI']

# 호칸 목록
bins = ['1호칸', '2호칸', '3호칸', '4호칸', '5호칸', '6호칸', '7호칸', '8호칸', '9호칸', '10호칸']

# 데이터 재구성
df_melted = result_df_칸별혼잡도3개역수정.melt(id_vars=['현재역', '요일'], value_vars=bins, var_name='호칸', value_name='혼잡도')

# 요일을 카테고리형 데이터로 변환하고 순서를 지정
df_melted['요일'] = pd.Categorical(df_melted['요일'], categories=weekdays, ordered=True)

# 현재역의 유일한 값 리스트를 가져옴
stations = df_melted['현재역'].unique()

# 현재역 별로 그래프를 그림
for station in stations:
    plt.figure(figsize=(10, 6))
    sns.lineplot(x="요일", y="혼잡도", hue="호칸", data=df_melted[df_melted['현재역']==station], ci=None)
    plt.title(f'{station}의 요일별 호칸별 혼잡도')
    plt.show()
```

<참고> 100%는 좌석은 모두 차 있고 출입문마다 사람이 서있는 정도.

분석 결과

1) 남태령은 4호칸이, 동대문과 혜화는 7호칸이 가장 붐빔

2) 동대문역은 7호칸 바로 앞에 환승통로 마련되어있기 때문으로 보임. 혜화는 동대문의 영향을 받은 것으로 추정

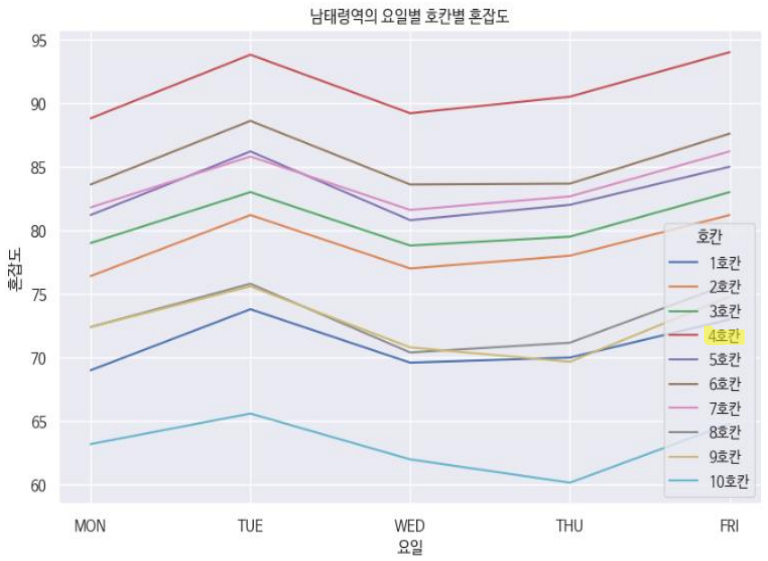
3) 남태령역은 인접한 사당역의 환승통로 중 하나가 4호칸과 가까운 영향으로 보임

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

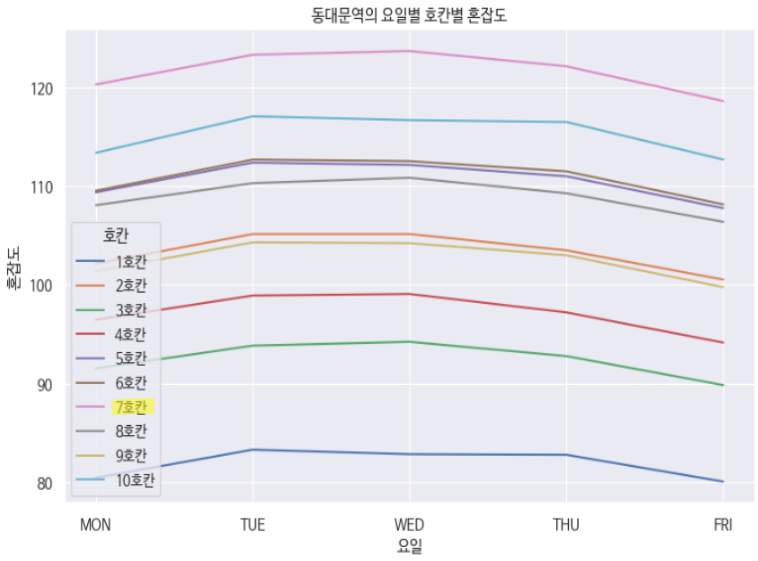
2.3 시각화

요일별 데이터 분석 - 평일

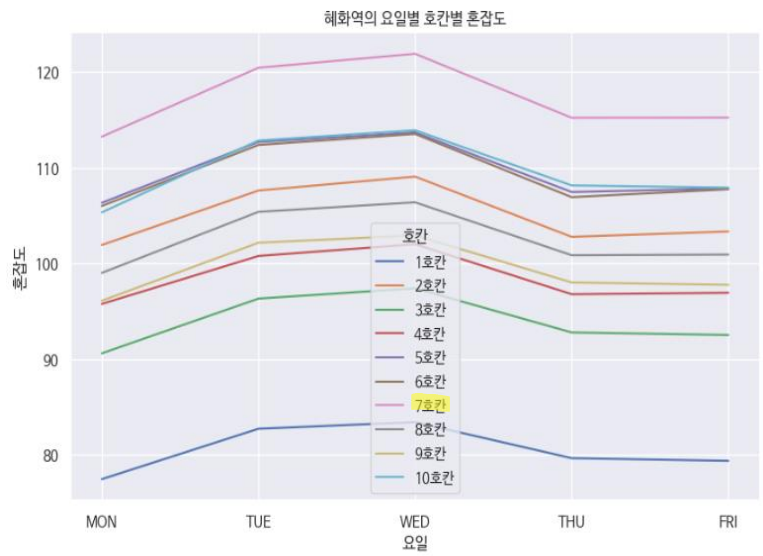
[남태령역] 요일별 칸 별 혼잡도



[동대문역] 요일별 칸 별 혼잡도



[혜화역] 요일별 칸 별 혼잡도



<참고> 100%는 좌석은 모두 차 있고 출입문마다 사람이 서있는 정도.

분석 결과

I. 남태령은 4호칸이, 동대문과 혜화는 7호칸이 가장 붐빔

II. 동대문역은 7호칸 바로 앞에 환승통로 마련되어있기 때문으로 보임. 혜화는 동대문의 영향을 받은 것으로 추정

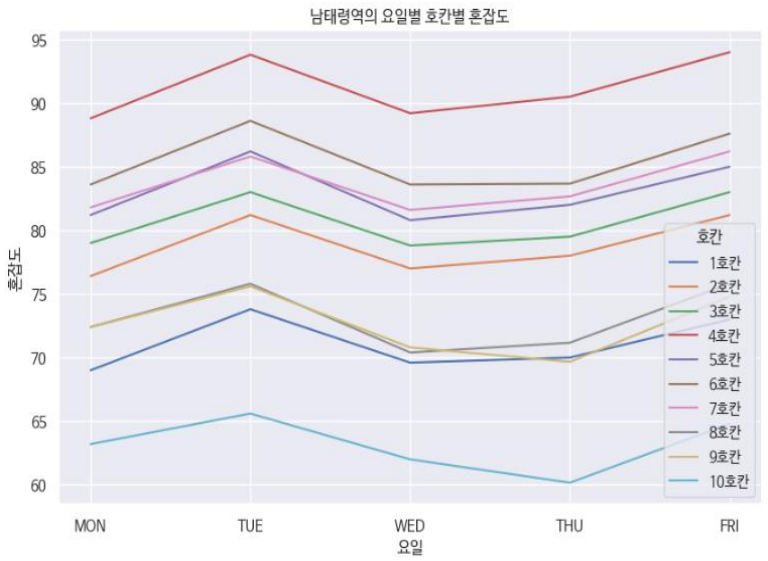
III. 남태령역은 인접한 사당역의 환승통로 중 하나가 4호칸과 가까운 영향으로 보임

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

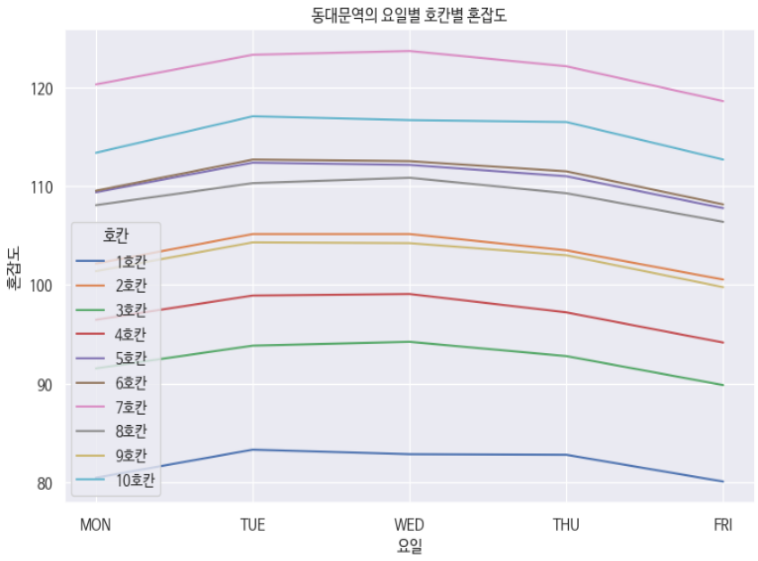
2.3 시각화

요일별 데이터 분석 - 평일

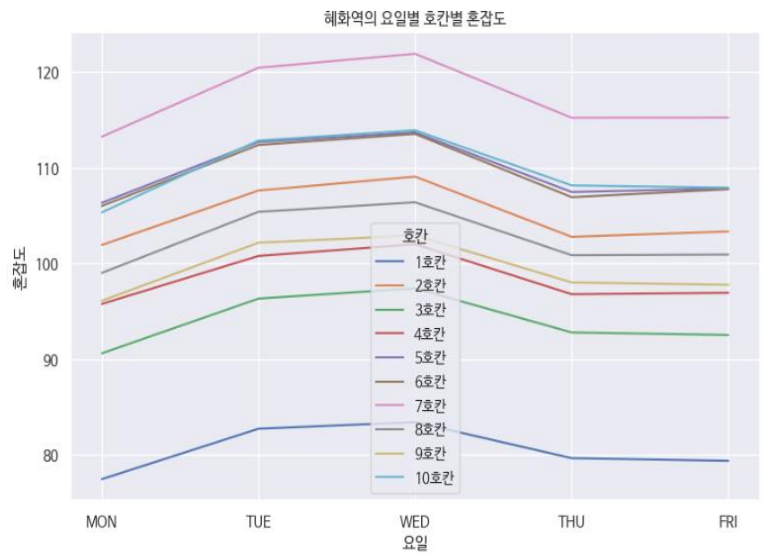
[남태령역] 요일별 칸 별 혼잡도



[동대문역] 요일별 칸 별 혼잡도



[혜화역] 요일별 칸 별 혼잡도



<참고> 100%는 좌석은 모두 차 있고 출입문마다 사람이 서있는 정도.

시사점

요일별로 붐비는 칸은 비슷한 양상을 띈다. 즉, 특정 역에서는 항상 붐비는 칸이 계속 붐비는 걸 알 수 있음

> 그렇기에 역 별로 붐비는 시간대에 승객들에게 미리 한산한 칸을 안내한 후 승객을 골고루 분산시킬 수 있을 것

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.3 시각화

노선별 데이터 분석 - 평일

[시각화 코드]

```
# 데이터 프레임 재구성
df_melted = result_df_칸별혼잡도3개역수정.melt(id_vars=['현재역', '출발역', '종착역', '이전역', '요일'], value_vars=['1호칸', '2호칸', '3호칸', '4호칸', '5호칸', '6호칸', '7호칸', '8호칸', '9호칸', '10호칸'], var_name='호칸', value_name='혼잡도')

# 현재역별로 데이터를 분리하여 그래프 생성
current_stations = df_melted['현재역'].unique()

# 요일 별로 그래프 생성
weekdays = ['MON', 'TUE', 'WED', 'THU', 'FRI']
for station in current_stations:
    for day in weekdays:
        plt.figure(figsize=(10, 6))

        # 해당 역의 데이터만 필터링
        df_station = df_melted[(df_melted['현재역']==station) & (df_melted['요일']==day)]

        # 출발역-종착역의 유일한 값 리스트를 가져옴
        train_lines = df_station[['출발역', '종착역']].drop_duplicates().values.tolist()

        for line in train_lines:
            departure_station, destination_station = line
            # 해당 열차 라인의 데이터만 필터링
            df_line = df_station[(df_station['출발역']==departure_station) &
                                 (df_station['종착역']==destination_station)]

            # 각 열차 라인별로 그래프 그리기
            plt.plot(df_line['호칸'], df_line['혼잡도'], label=f'{departure_station}-{destination_station}')

        plt.xlabel('호칸')
        plt.ylabel('혼잡도')
        plt.title(f'{station} - {day} 혼잡도')
        plt.legend()
        plt.show()
```

역 이름만 따로 분리

각각의 역별로 분리해서 저장

출발역, 종착역 리스트를 뽑음

line에서 나오는 출발역, 종착역을 각각의 변수에 할당한 다음 해당하는 데이터만 필터링

분석 결과

I. 혜화역과 동대문역의 경우 사당방면 지하철노선의 혼잡도가 매우 높은 것을 볼 수 있음

II. 남태령역의 경우 당고개방면 지하철노선의 혼잡도가 높음

즉, 혜화와 남태령 사이의 역을 가기 위해 지하철을 이용하는 승객이 많다고 유추할 수 있음

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.3 시각화

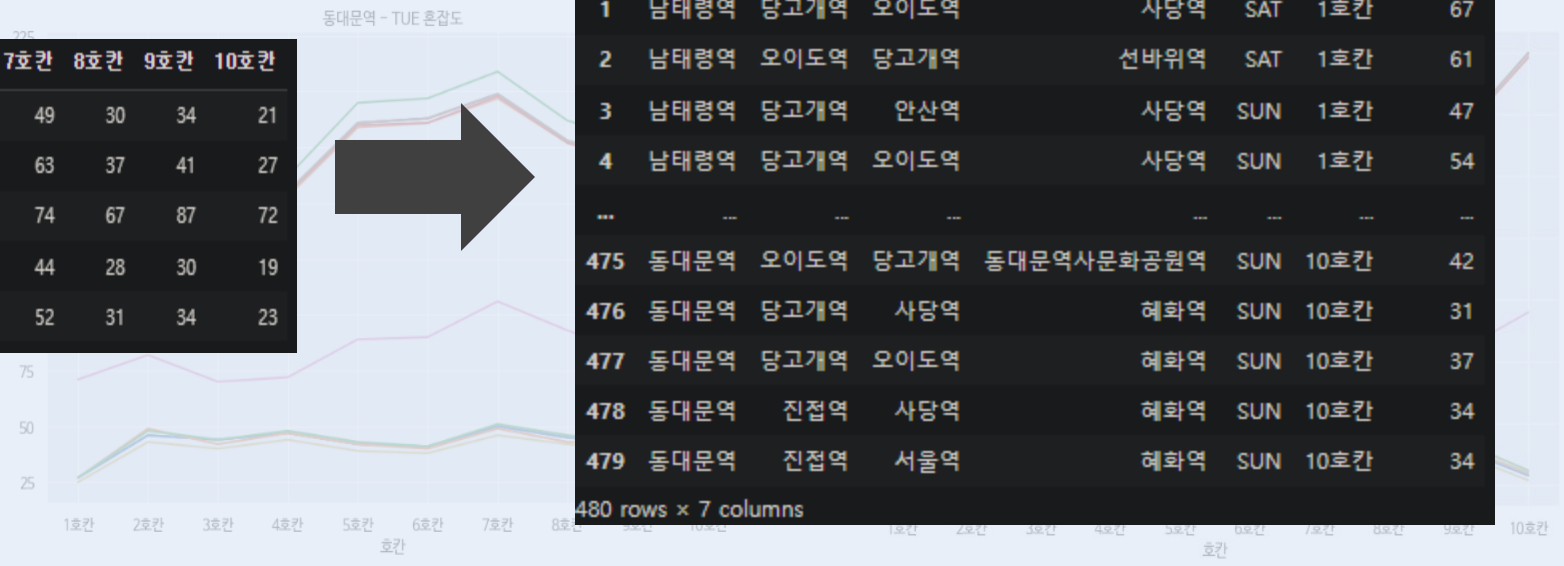
노선별 데이터 분석 - 평일

[시각화 코드] df.melt

[남태령역] 노선별 칸 별 혼잡도

	현재역	출발역	종착역	이전역	요일	1호칸	2호칸	3호칸	4호칸	5호칸	6호칸	7호칸	8호칸	9호칸	10호칸
0	남태령역	당고개역	안산역	사당역	SAT	51	69	68	68	57	57	49	30	34	21
1	남태령역	당고개역	오이도역	사당역	SAT	67	90	95	86	74	71	63	37	41	27
2	남태령역	오이도역	당고개역	선바위역	SAT	61	71	69	76	87	93	74	67	87	72
3	남태령역	당고개역	안산역	사당역	SUN	47	64	61	60	52	51	44	28	30	19
4	남태령역	당고개역	오이도역	사당역	SUN	54	74	77	71	62	60	52	31	34	23

[동대문역] 노선별 칸 별 혼잡



id.vars()

	현재역	출발역	종착역	이전역	요일	호칸	혼잡도
0	남태령역	당고개역	안산역	사당역	SAT	1호칸	51
1	남태령역	당고개역	오이도역	사당역	SAT	1호칸	67
2	남태령역	오이도역	당고개역	선바위역	SAT	1호칸	61
3	남태령역	당고개역	안산역	사당역	SUN	1호칸	47
4	남태령역	당고개역	오이도역	사당역	SUN	1호칸	54
...	...	...	...	...	...	...	...
475	동대문역	오이도역	당고개역	동대문역사문화공원역	SUN	10호칸	42
476	동대문역	당고개역	사당역	혜화역	SUN	10호칸	31
477	동대문역	당고개역	오이도역	혜화역	SUN	10호칸	37
478	동대문역	진접역	사당역	혜화역	SUN	10호칸	34
479	동대문역	진접역	서울역	혜화역	SUN	10호칸	34

480 rows × 7 columns

가장 혼잡도가 높았던 화요일 기준으로 분석 진행. 요일별로 비슷한 양상 띔

분석 결과

I. 혜화역과 동대문역의 경우 사당방면 지하철노선의 혼잡도가 매우 높은 것을 볼 수 있음

II. 남태령역의 경우 당고개방면 지하철노선의 혼잡도가 높음

즉, 혜화와 남태령 사이의 역을 가기 위해 지하철을 이용하는 승객이 많다고 유추할 수 있음

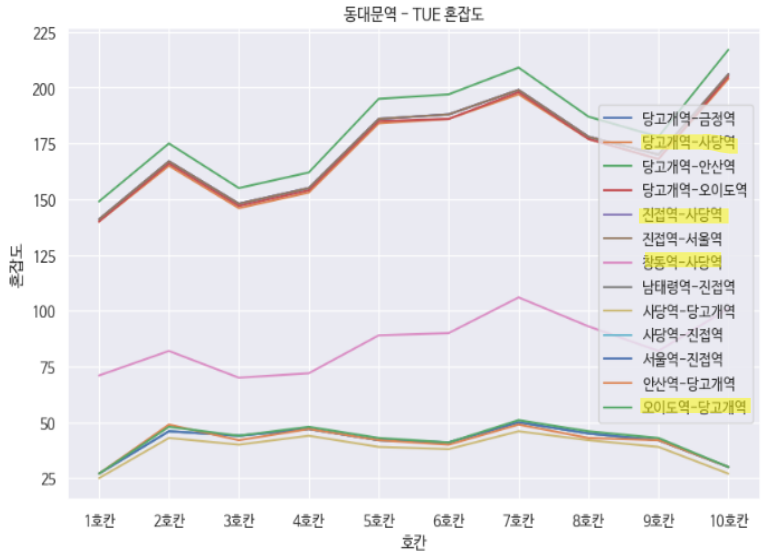
Copyright©. Saebyeol Yu. All Rights Reserved.

2.3 시각화

노선별 데이터 분석 - 평일

[남태령역] 노선별 칸 별 혼잡도

[동대문역] 노선별 칸 별 혼잡도



[혜화역] 노선별 칸 별 혼잡도

가장 혼잡도가 높았던 화요일 기준으로 분석 진행. 요일별로 비슷한 양상 띔

분석 결과

I. 혜화역과 동대문역의 경우 사당방면 지하철노선의 혼잡도가 매우 높은 것을 볼 수 있음/7호, 10호칸 혼잡도 가장 높음

II. 남태령역의 경우 당고개방면 지하철노선의 혼잡도가 높음/6, 9호칸 혼잡도가 가장 높게 측정됨

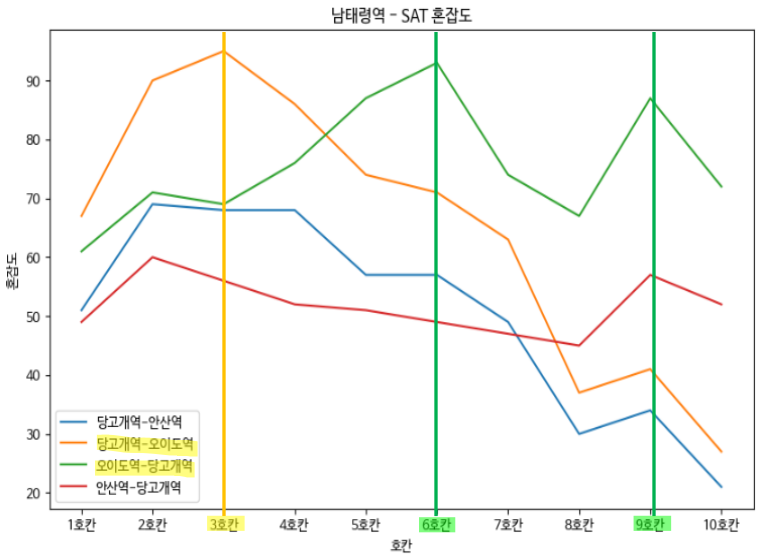
| 즉, 혜화와 남태령 사이의 역을 가기 위해 지하철을 이용하는 승객이 많다고 유추할 수 있음

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

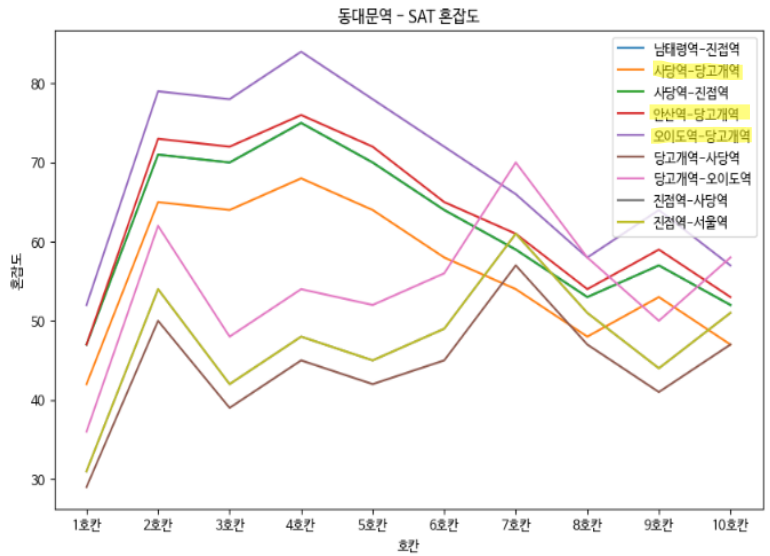
2.3 시각화

노선별 데이터 분석 - 주말

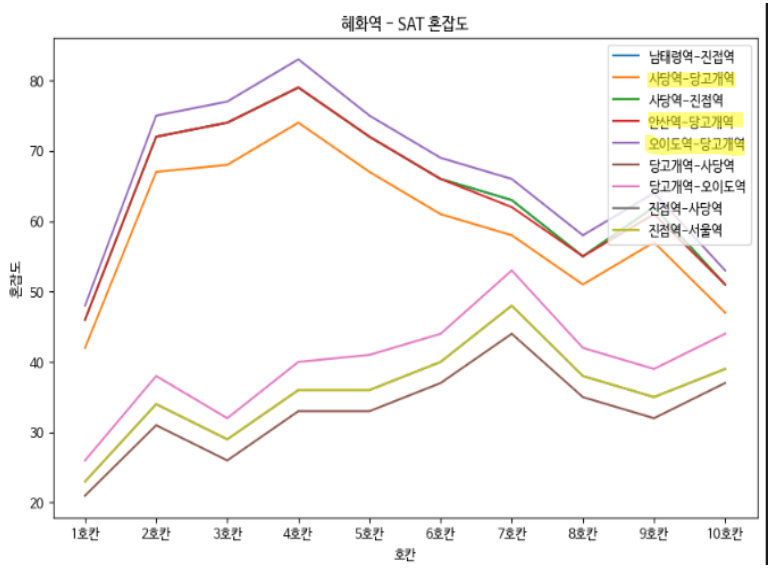
[남태령역] 노선별 칸 별 혼잡도



[동대문역] 노선별 칸 별 혼잡도



[혜화역] 노선별 칸 별 혼잡도



분석 결과

가장 혼잡도가 높았던 화요일 기준으로 분석 진행. 요일별로 비슷한 양상 띔

I. 혜화역과 동대문역의 경우 평일과 반대로 당고개 방면 지하철노선의 혼잡도가 매우 높은 것을 볼 수 있음

II. 남태령역도 마찬가지로 당고개 방면 혼잡도가 평균적으로 높게 나옴.

특히 오이도 방면 열차는 2,3,4호칸이, 오이도-당고개행 열차는 6,9호칸의 혼잡도가 높은 것을 알 수 있음

*남태령 역 개찰구로 올라가는 통로가 3호칸 앞에 있기 때문 / 오이도 역의 통로는 6, 9호칸 앞에 위치

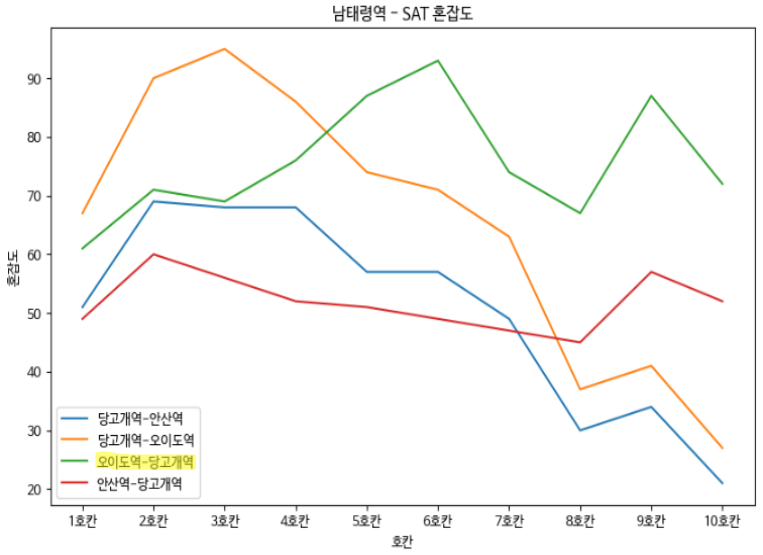
Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.3 시각화

노선별 데이터 분석

[남태령역_평일] 노선별 칸 별 혼잡도

[남태령역_주말] 노선별 칸 별 혼잡도



시사점

I. 하나의 역에서도 열차의 '출발역'-'종착역'에 따라서 혼잡도가 다르게 나타난다.

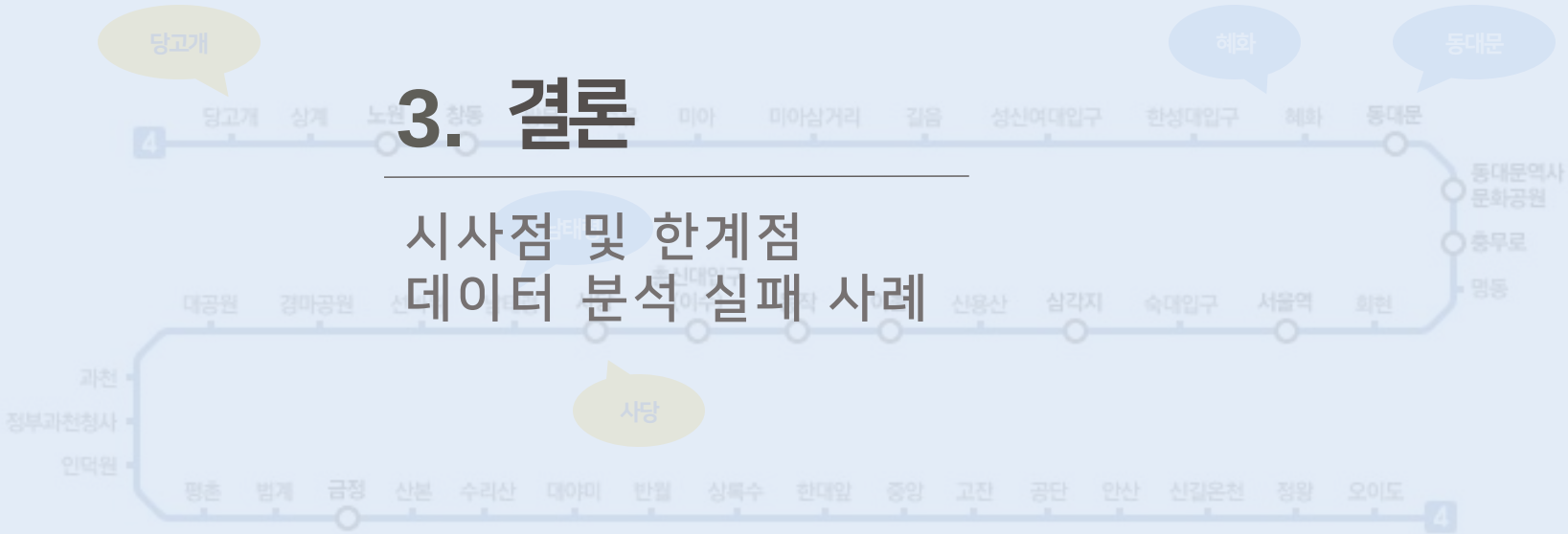
II. 평일과 주말의 특성을 이용하여 가장 밀집도가 높은 시간대에 가장 밀집도가 높은 노선은 특히 주의해서 관리할 필요가 있음
e.g) 오이도발 열차는 6, 9호칸을 피해서 타고, 오이도 방면 열차는 2,3,4호칸을 피해서 타는 것이 효율적

III. 주말, 평일 모두 역에 상관없이 오이도에서 출발한 열차에 대한 혼잡도가 가장 높게 나타남.

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

2.3 시각화

노선별 데이터 분석 - 주말



3. 결론

시사점 및 한계점
데이터 분석 실패 사례

Copyright©. Saebyeol Yu. All Rights Reserved.

3.1 시사점 및 한계점

데이터 분석 결과 정리

데이터 분석 결과 정리

요일별 데이터 정리

남태령역	혜화&동대문역
요일에 상관없이 4번 칸이 가장 혼잡도가 높게 나타남	요일에 상관없이 7번 칸이 가장 혼잡도가 높게 나타남

노선별 데이터 정리

남태령역	혜화&동대문역
오이도 방면: 2,3,4호칸 혼잡도 높음 오이도발 열차: 6,9호칸 혼잡도 높음 평일, 주말 상관없이 당고개행 혼잡도가 높음	노선에 상관없이 7번 칸이 가장 혼잡도가 높게 나타남 평일에는 사당행, 주말에는 당고개행 혼잡도 높음

1. 평일은 출퇴근시간, 주말은 남태령 역 기준 18~19시, 혜화 및 동대문역 기준 17~18시가 가장 혼잡함.
2. 요일에 상관없이 평균적으로 남태령역은 4호칸이, 혜화 및 동대문역은 7호칸이 꾸준하게 혼잡함.
3. 같은 역이라도 각 지하철 노선에 따라 혼잡도가 다르게 나타남
4. 어느 역이든, 어느 노선이든 오이도발 열차의 혼잡도는 항상 높게 나옴.

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

3.1 시사점 및 한계점

시사점

시사점

1. 각 독립변수별로 일관되게 혼잡한 열차 칸을 특정할 수 있었다.
2. 이러한 분석 결과를 토대로 증차나 배차간격 조정, 신형 전동차로의 교체 이전에 적은 비용을 들여 승객들의 열차 이용 만족도를 높일 수 있을 것
3. 일정한 패턴을 보이는 만큼, 실시간 데이터가 아니더라도 붐비는 시간에 간단한 안내만 해도 혼잡도 개선에 기여할 수 있을 것

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

3.1 시사점 및 한계점

한계점 및 향후 연구 과제

한계점

1. 데이터 양이 지나치게 방대하여 한정된 역 중에서 특정 시간대만 따로 뽑아서 진행할 수밖에 없었음
2. 동시에 9호선에 대한 데이터는 한정적이어서 우리와 가장 밀접한 관련이 있는 9호선에 대한 분석을 하기는 힘들었다
3. 실시간 API 데이터는 최근 3개월간의 데이터만 보여주기 때문에 해당 예측이 틀릴 수도 있음
4. 각 역에 대해 가장 혼잡한 한 시간 분량의 데이터로만 진행하였기 때문에, 시간대별로 또 결과가 더 다양하게 나올 수 있을 것이다.
5. 칸 별 혼잡도가 모두 다 혼잡할 경우에는 의미가 없는 데이터분석이 되어버린다는 한계점
   -> 증차와 배차간격 조정 이전에 임시방편으로 사용하거나 참고자료로 사용하면 좋을 것 같음

향후 연구 과제

1. 실시간 데이터가 아니어서 어느 정도의 예측도를 확실하게 가지지 못하면 사람들으로부터 신뢰를 잃어버림.

   즉, 안내에 따르는 승객이 적어져 데이터 분석 자체의 본질을 잃어버리게 될 수 있음.

2. 더욱 장기적으로 데이터를 분석하여 역 별, 시간대 별, 노선 별 등 세분화된 자료를 바탕으로
   칸 별 혼잡도를 예측할 필요가 있음.

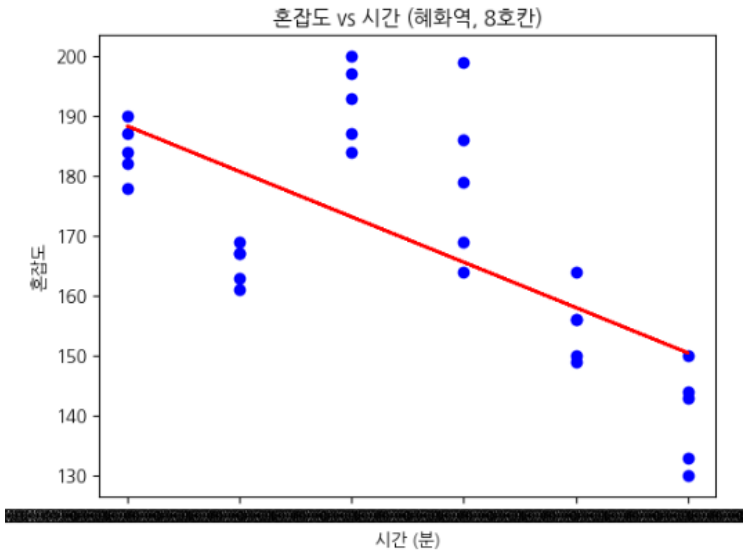
Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

3.2 데이터분석 실패사례

선형회귀분석의 실패

각 역별로 이전역을 구분하여 시간 흐름에 따른 칸별 혼잡도 예측을 선형회귀분석을 이용하여 알아봄

```python
# 필요한 라이브러리를 불러옵니다
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# 데이터를 불러옵니다
# df = pd.read_csv('파일 경로 또는 URL')
df = result_df_동대문8시

# 이전역별로 그룹화합니다
grouped = df.groupby('이전역')

# 각 이전역별로 선형 회귀를 진행합니다
for name, group in grouped:
  print(f"₩n이전역: {name}")

  # X 데이터와 y 데이터를 정의합니다
  X = group[['hh', 'mm']]
  for i in range(1, 11):  # 각각의 호칸에 대해 분석을 진행
    print(f"호칸: {i}")
    y = group[str(i) + '호칸']

    # 선형 회귀 모델을 학습합니다
    model = LinearRegression()
    model.fit(X, y)

    # 모델의 계수(coefficient)와 절편(intercept)를 출력합니다
    print("계수: ", model.coef_)
    print("절편: ", model.intercept_)

    # 그래프를 그립니다
    plt.scatter(X['hh']*60 + X['mm'], y, color='blue')  # 실제 데이터
    plt.plot(X['hh']*60 + X['mm'], model.predict(X), color='red')  # 예측 데이터
    plt.title(f"혼잡도 vs 시간 ({name}, {i}호칸)")
    plt.xlabel("시간 (분)")
    plt.ylabel("혼잡도")
    plt.show()
```



그래프 분석

그래프는 우하향 하는 형태를 띠고 있는데, 정작 30분까지는 데이터들이 높게 나오는 경향이 있음

출근 시간대가 지난 후 갑자기 인원이 줄어드는 것을 반영해 우하향 하는 듯 싶었으나, 정작 출근시간 피크 시간인 8시~8시30분까지는 유지되거나 높아지는 경향이 있어 신뢰할 만한 선형회귀분석은 아닌 듯 함

Copyrightⓒ. Saebyeol Yu. All Rights Reserved.

The End

경청해주셔서 감사합니다