서울시 지하철 승/하차 데이터분석 프로젝트

by 이민철

프로젝트 배경

문제 상황 :

본인은 평일에 숭실대학교 중앙도서관에서 공부하기 위해 지하철을 이용하는데, 출퇴근 시간대에 지하철에 사람이 너무 많아 큰 불편함을 겪음.

→

해결 방안 :

이러한 불편함을 해소하고자 객관적인 데이터를 기반으로 서울시 지하철의 시간대별 승/하차 승객 수를 파악하고 사람이 적은 시간대를 찾아 이동하고자 함.

이동경로

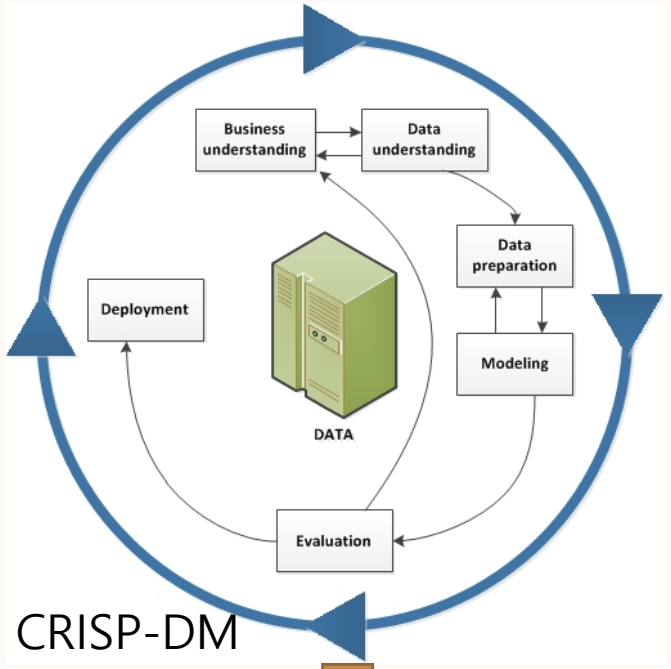
- 1. 상일동역에서 승차
- 2. 군자역에서 환승
- 3. 숭실대입구역에서 하차

=> 상일동역에서 군자역, 군자역에서
숭실대입구역까지의 시간대별 승객 수 데이터를 확인하자.

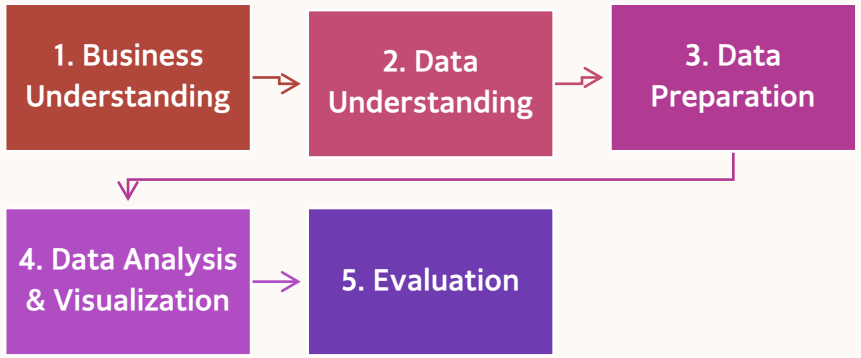
출 처 : 카 카 오 지 하 철 앱

데이터 분석 절차

- CRISP-DM (cross industry standard process for data mining) 방식을 본 프로젝트에 맞게 변형
- 변형 방법
  1. Modeling 삭제
  2. Data Analysis & Visualization 추가
  3. Deployment 삭제



변형



CRISP-DM 모델 출처 : https://www.ibm.com/docs/ko/spss-modeler/SaaS?topic=dm-crisp-help-overview

데이터 분석 절차 상세

1. Business Understanding : 프로젝트 목적과 요구사항을 이해

2. Data Understanding : 분석을 위한 데이터를 수집하고 데이터 속성을 이해

3. Data Preparation : 데이터 전처리를 통해 분석용 데이터셋 확보

4. Data Analysis & Visualization : 데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

5. Evaluation : 분석 결과 데이터에 대한 객관적인 검증과 평가

1. Business Understanding :

프로젝트 목적과 요구사항을 이해

프로젝트의 목적 :

* 본인이 지하철을 이용할 때 좀 더 쾌적하게 이동할 수 있는 시간대를 찾는 것.

프로젝트의 요구사항 :

* 서울시 지하철의 시간대별, 역별 승/하차 승객 수 관련 데이터가 필요

* 본인의 이동경로를 고려한 역(상일동~ 군자~숭실대입구)만을 대상으로 분석

2. Data Understanding :
분석을 위한 데이터를 수집하고 데이터 속성을 이해

```
# 데이터 수집
import pandas as pd
from google.colab import drive
drive.mount('/content/gdrive')

df = pd.read_csv(
'/content/gdrive/My Drive/data_analysis/data/seoul-metro-time-population-data/
서울시 지하철 호선별 역별 시간대별 승하차 인원 정보.csv',encoding='cp949')
df
```

데이터 수집 : 서울시 지하철 호선별, 역별, 시간대별 승/하차 인원 정보 활용 (서울 열린데이터 광장)

데이터 속성 이해

```
# 전체 정보의 속성 확인

df.info()
```

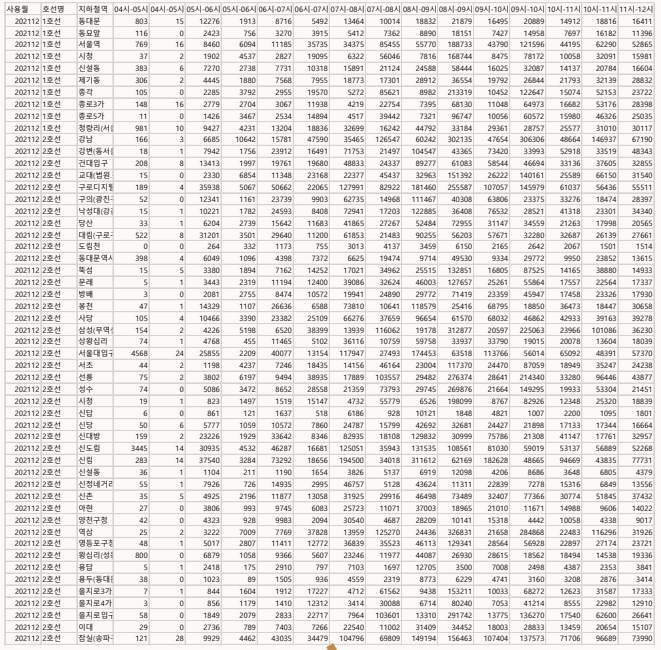
데이터 출처 : https://data.seoul.go.kr/dataList/OA-12252/S/1/datasetView.do

colab 링크 : 이 프로젝트의 모든 파이썬 소스코드는 이 링크에서 확인하실 수 있습니다.
또한 해당 결과물이 모두 출력된 상태이므로 코드 셀을 따로 실행시킬 필요 없습니다 :)

2-1. Data Understanding :

Colab 출력 결과 (Colab 링크)

(아래에 출력결과물을 포함하긴 했으나 위의 링크로 들어가서 보시는 것이 더 편합니다)



사용월	호선명	지하철역	04시-05시	04시-05시	05시-06시	05시-06시	06시-07시	06시-07시	07시-08시	07시-08시	08시-09시	08시-09시	09시-10시	09시-10시	10시-11시	10시-11시	11시-12시
202112	1호선	동대문	803	15	12276	1913	8716	5492	13464	10014	18832	21879	16495	20889	14912	18816	16411
202112	1호선	동묘앞	116	0	2423	756	3270	3915	5412	7362	8890	18151	7427	14958	7697	16182	11396
202112	1호선	서울역	769	16	8460	6094	11185	35735	34375	85455	55770	188733	43790	121596	44195	62290	52865
202112	1호선	시청	37	2	1902	4537	2827	19095	6322	56046	7816	168744	8475	78172	10058	32091	15981
202112	1호선	신설동	383	6	7270	2738	7731	10318	15891	21124	24588	58444	16025	32087	14137	20784	16604
202112	1호선	제기동	306	2	4445	1880	7568	7955	18773	17301	28912	36554	19792	26844	21793	32139	28832
202112	1호선	종각	105	0	2285	3792	2955	19570	5272	85621	8982	213319	10452	122647	15074	52153	23722
202112	1호선	종로3가	148	16	2779	2704	3067	11938	4219	22754	7395	68130	11048	64973	16682	53176	28398
202112	1호선	종로5가	11	0	1426	3467	2534	14894	4517	39442	7321	96747	10056	60572	15980	46326	25035
202112	1호선	청량리(서	981	10	9427	4231	13204	18836	32699	16242	44792	33184	29361	28757	25577	31010	30117
202112	2호선	강남	166	3	6685	10642	15781	47590	35465	126547	60242	302135	47654	306306	48664	146937	67190
202112	2호선	강변(동서	18	1	7942	1756	23912	16491	71753	21497	104547	43365	73420	33993	52918	33519	48343
202112	2호선	건대입구	208	8	13413	1997	19761	19680	48833	24337	89277	61083	58544	46694	33136	37605	32855
202112	2호선	교대(법원	15	0	2330	6854	11348	23168	22377	45437	32963	151392	26222	140161	25589	66150	31540
202112	2호선	구로디지털	189	4	35938	5067	50662	22065	127991	82922	181460	255587	107057	145979	61037	56436	55511
202112	2호선	구의(광진	52	0	12341	1161	23739	9903	62735	14968	111467	40308	63806	23375	33276	18474	28397
202112	2호선	낙성대(강	15	1	10221	1782	24593	8408	72941	17203	122885	36408	76532	28521	41318	23301	34340
202112	2호선	당산	33	1	6204	2739	15642	11683	41865	27267	52484	72955	31147	34559	21263	17998	20565
202112	2호선	대림(구로	522	8	31201	3501	29640	11200	61853	21483	90255	56203	57671	32280	32687	26139	27661
202112	2호선	도림천	0	0	264	332	1173	755	3013	4137	3459	6150	2165	2642	2067	1501	1514
202112	2호선	동대문역사	398	4	6049	1096	4398	7372	6625	19474	9714	49530	9334	29772	9950	23852	13615
202112	2호선	뚝섬	15	5	3380	1894	7162	14252	17021	34962	25515	132851	16805	87525	14165	38880	14933
202112	2호선	문래	5	1	3443	2319	11194	12400	39086	32624	46003	127657	25261	55864	17557	22564	17337
202112	2호선	방배	3	0	2081	2755	8474	10572	19941	24890	29772	71419	23359	45947	17458	23326	17930
202112	2호선	봉천	47	1	14329	1107	26636	6588	73810	10641	118579	25416	68795	18850	36473	18447	30658
202112	2호선	사당	105	4	10466	3390	23382	25109	66276	37659	96654	61570	68032	46852	42933	39163	39278
202112	2호선	삼성(무역(	154	2	4226	5198	6520	38399	13939	116062	19178	312877	20597	225063	23966	101086	36230
202112	2호선	상왕십리	74	1	4768	455	11465	5102	36116	10759	59758	33937	33790	19015	20078	13604	18039
202112	2호선	서울대입구	4568	24	25855	2209	40077	13154	117947	27493	174453	63518	113766	56014	65092	48391	57370
202112	2호선	서초	44	2	1198	4237	7246	18435	14156	46164	23004	117370	24470	87059	18949	35247	24238
202112	2호선	선릉	75	2	3802	6197	9494	38935	17889	103557	29482	276374	28641	214340	33280	96446	43877
202112	2호선	성수	74	0	5086	3472	8652	28558	21359	73793	29745	269876	21664	149295	19933	53304	21451
202112	2호선	시청	19	1	823	1497	1519	15147	4732	55779	6526	198099	8767	82926	12348	25320	18839
202112	2호선	신답	6	0	861	121	1637	518	6186	928	10121	1848	4821	1007	2200	1095	1801
202112	2호선	신당	50	6	5777	1059	10572	7860	24787	15799	42692	32681	24427	21898	17133	17344	16664
202112	2호선	신대방	159	2	23226	1929	33642	8346	82935	18108	129832	30999	75786	21308	41147	17761	32957
202112	2호선	신도림	3445	14	30935	4532	46287	16681	125051	35943	131535	108561	81030	59019	53137	56889	52268
202112	2호선	신림	283	14	37540	3284	73292	18656	194500	34018	311612	62169	182628	48665	94669	43835	77731
202112	2호선	신설동	36	1	1104	211	1190	1654	3826	5137	6919	12098	4206	8686	3648	6805	4379
202112	2호선	신정네거리	55	1	7926	726	14935	2995	46757	5128	43624	11311	22839	7278	15316	6849	13556
202112	2호선	신촌	35	5	4925	2196	11877	13058	31925	29916	46498	73489	32407	77366	30774	51845	37432
202112	2호선	아현	27	0	3806	993	9745	6083	25723	11071	37003	18965	21010	11671	14988	9606	14022
202112	2호선	양천구청	42	0	4323	928	9983	2094	30540	4687	28209	10141	15318	4442	10058	4338	9017
202112	2호선	역삼	25	2	3222	7009	7769	37828	13959	125270	24436	326831	21658	284868	22483	116296	31926
202112	2호선	영등포구청	48	1	5017	2807	11411	12772	36839	35523	46113	129341	28564	56928	22897	27174	23721
202112	2호선	왕십리(성(	800	0	6879	1058	9366	5607	23246	11977	44087	26930	28615	18562	18494	14538	19336
202112	2호선	용답	5	1	2418	175	2910	797	7103	1697	12705	3500	7008	2498	4387	2353	3841
202112	2호선	용두(동대	38	0	1023	89	1505	936	4559	2319	8773	6229	4741	3160	3208	2876	3414
202112	2호선	을지로3가	7	1	844	1604	1912	17227	4712	61562	9438	153211	10033	68272	12623	31587	17333
202112	2호선	을지로4가	3	0	856	1179	1410	12312	3414	30088	6714	80240	7053	41214	8555	22982	12910
202112	2호선	을지로입구	58	0	1849	2079	2833	22717	7964	103601	13310	291742	13775	136270	17540	62600	26641
202112	2호선	이대	29	0	2736	789	7403	7266	22540	11002	31409	34452	18003	28833	13459	20654	15107
202112	2호선	잠실(송파	121	28	9929	4462	43035	34479	104796	69809	149194	156463	107404	137573	71706	96689	73990

데이터 수집의 결과 (마우스 우클릭> Macro-Enabled Worksheet개체 > 열기를 통해 엑셀로 열 수 있습니다.)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48986 entries, 0 to 48985
Data columns (total 52 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   사용월           48986 non-null  int64
 1   호선명           48986 non-null  object
 2   지하철역          48986 non-null  object
 3   04시-05시 승차인원  48986 non-null  int64
 4   04시-05시 하차인원  48986 non-null  int64
 5   05시-06시 승차인원  48986 non-null  int64
 6   05시-06시 하차인원  48986 non-null  int64
 7   06시-07시 승차인원  48986 non-null  int64
 8   06시-07시 하차인원  48986 non-null  int64
 9   07시-08시 승차인원  48986 non-null  int64
 10  07시-08시 하차인원  48986 non-null  int64
 11  08시-09시 승차인원  48986 non-null  int64
 12  08시-09시 하차인원  48986 non-null  int64
 13  09시-10시 승차인원  48986 non-null  int64
 14  09시-10시 하차인원  48986 non-null  int64
중략
 ...
 51 작업일자 48986 non-null int64 dtypes:
int64(50), object(2)
 ...
 후략
```

column에는 사용월, 호선명, 시간대별 승차인원, 작업일자가 포함되어 있음을 파악

이외 각 column의 데이터 타입(int64, object) 파악

non-null 정보를 통해 결측치가 없음을 확인

3. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

전처리 조건 :

1. 작업일자 column은 따로 쓰일 일이 없으니 제거
2. 전체 데이터에서 실제 이동 경로에 해당되는 역만 추출
3. 이상치 처리 (실제 승객이 이용하는 시간대만을 따진다)
4. 2020년 1월부터 2021년 12월까지 2년치의 데이터만 추출 (코로나 19로 인한 변동가능성 -> 코로나 19와 지하철 이용의 변화 논문)
5. 추출된 지하철역과 사용월을 기준으로 오름차순으로 정렬

3-1. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

전처리 조건 :

1. 작업일자 column은 따로 쓰일 일이 없으니 제거

```
# 1. 작업일자 column은 따로 쓰일 일이 없으니 제거
df.drop('작업일자', inplace=True, axis=1)
df
```

3-2. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

전처리 조건

- 2. 전체 데이터에서 실제 이동 경로에 해당되는 역만 추출
  (상일동역(출발)~군자역까지 10개 역,
  군자역(환승)~숭실대입구역(도착)까지 13개 역)

2016년 10월부터 이름이 바뀐 역 존재 (역명을 병기하는 역들이 생김)

- 데이터셋에 병기하는 역이 반영된 역이 있고 그렇지 않은 역이 있음
- 따라서 23개 역을 엑셀로 필터링 후 일일히 검색해서 전처리 수행
- 병기하는 역들 :
  - 굽은다리(강동구민회관앞), 군자(능동), 천호(풍납토성), 광나루(장신대), 아차산(어린이대공원후문), 어린이대공원(세종대), 총신대입구(이수), 숭실대입구(살피재)

2개 이상 호선이 겹치는 역의 경우, 실제 이동경로가 아닌 호선의 경우 제거해야함.

- 실제 이동하지 않는 경로
  - 천호(풍납토성)역 8호선
  - 건대입구역 2호선
  - 강남구청역 분당선
  - 고속터미널역 3호선,9호선
  - '총신대입구(이수)'역 4호선
    - 단, 이 경우 7호선 역명인 '이수'와 다르기 때문에 제거할 필요가 없다.

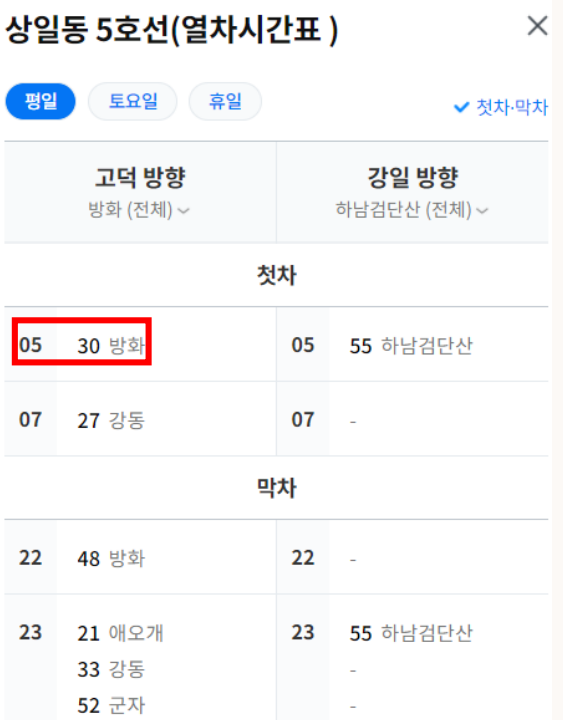
전처리 파이썬 코드

```
# 2. 실제 이동 경로에 해당되는 역만 추출

station = ['상일동','고덕','명일','굽은다리(강동구민회관앞)','길동','강동','천호(풍납토성)','광나루(장신대)','아차산(어린이대공원후문)','군자(능동)', '어린이대공원(세종대)','건대입구','뚝섬유원지','청담','강남구청','학동','논현','반포','고속터미널','내방','이수','남성','숭실대입구(살피재)']
df_station = df[df['지하철역'].isin(station)]

# 단, 2개 이상 호선이 겹치는 역의 경우, 실제 이동경로가 아닌 호선의 경우 제거해야한다.

cheonho = df_station[(df_station['지하철역']=='천호(풍납토성)')&(df_station['호선명']=='8호선')].index
df_station = df_station.drop(cheonho)

konkuk = df_station[(df_station['지하철역']=='건대입구')&(df_station['호선명']=='2호선')].index
df_station=df_station.drop(konkuk)

gang_nam_gu_office = df_station[(df_station['지하철역']=='강남구청')&(df_station['호선명']=='분당선')].index
df_station=df_station.drop(gang_nam_gu_office)

express_bus_terminal_3 = df_station[(df_station['지하철역']=='고속터미널')&(df_station['호선명']=='3호선')].index
df_station = df_station.drop(express_bus_terminal_3)

express_bus_terminal_9 = df_station[(df_station['지하철역']=='고속터미널')&(df_station['호선명']=='9호선')].index
df_station = df_station.drop(express_bus_terminal_9)

df_station

# 현재 dataFrame을 엑셀파일로 변환
df_station.to_excel("path-station.xlsx")
```

3-3. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

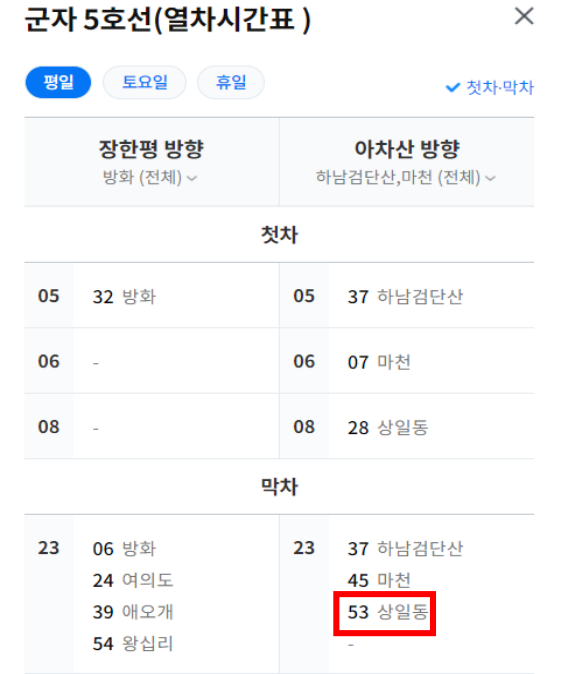
전처리 조건 :

- 3. 이상치 처리 (실제 승객이 이용하는 시간대만을 따진다)
  - 데이터 중 첫차시간과 막차시간 사이의 시간대가 아닌데도 승/하차 인원이 있다.
  - 이는 승객이 아닌 사람(직원 등)일 가능성이 높다.
  - 상일동역에서 처음 승차하므로 상일동역의 첫차시간을 파악, 돌아올 때는 환승역인 군자(능동)역에서 상일동행 열차를 타야하므로, 군자역의 막차시간을 파악.
  - 파악 결과, 오전 5시 30분(첫차), 오후 11시 53분(막차)
- 다만, 군자역에서 상일동역까지 시간(약 20분)을 고려하여 그 시간대의 승객 수는 포함해야함.
  - 따라서 00시-01시 승/하차 인원까지 포함한다.
  - **결론 : 오전 5시이전과 오전1시 이후의 데이터는 제외한다.**


상일동 5호선(열차시간표)

평일 토요일 휴일 첫차·막차

고덕 방향 (방화 (전체))		강일 방향 (하남검단산 (전체))	
첫차			
05	30 방화	05	55 하남검단산
07	27 강동	07	-
막차			
22	48 방화	22	-
23	21 애오개 33 강동 52 군자	23	55 하남검단산 - -


군자 5호선(열차시간표)

평일 토요일 휴일 첫차·막차

장한평 방향 (방화 (전체))		아차산 방향 (하남검단산,마천 (전체))	
첫차			
05	32 방화	05	37 하남검단산
06	-	06	07 마천
08	-	08	28 상일동
막차			
23	06 방화 24 여의도 39 애오개 54 왕십리	23	37 하남검단산 45 마천 53 상일동 -

3-3. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

파이썬 코드

```
# 3. 이상치 처리 (실제 승객이 이용하는 시간대만을 따진다.)
outlier = ['04시-05시 승차인원', '04시-05시 하차인원',
           '01시-02시 승차인원','01시-02시 하차인원',
           '02시-03시 승차인원','02시-03시 하차인원',
           '03시-04시 승차인원','03시-04시 하차인원'
           ]
df_station.drop(columns=outlier, inplace=True, axis=1)
df_station
```

3-4. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

전처리 조건 :

- 4. 2020년 1월부터 2021년 12월까지 2년치의 데이터만 추출 (코로나 19로 인한 승객 수 변동이 발생해서 전체 데이터를 보는 것이 현재 관점에서는 불필요하다고 판단. 따라서 코로나 19 발생시기부터 추출진행 -> 근거자료 : https://www.korea.kr/news/policyNewsView.do?newsId=148885347)

파이썬 코드

```python
#4 2020년 1월부터 2021년 12월까지 2년치의 데이터만 추출
date=[i for i in range(202001,202012+1)] + [i for i in range(202101,202112+1)]
df_station = df_station[df_station['사용월'].isin(date)]

# 사용월을 오름차순으로 정렬
df_station.sort_values('사용월', ascending=True, inplace=True)
df_station
```

3-5. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

전처리 조건 :

5. 추출된 지하철역과 사용월을 기준으로 오름차순으로 정렬

파이썬 코드

```
# 5. 추출된 지하철역과 사용월을 기준으로 오름차순으로 정렬
df_station = df_station.sort_values(by=['지하철역','사용월'])
df_station
```

3-6. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

최종 결과물

- 1) 마우스 우클릭 > 2) Worksheet 개체 > 3) 열기 하시면 엑셀로 결과물 확인 가능합니다.혹은 더블 클릭하시면 열어볼 수 있습니다.

	사용월	호선명	지하철역	-06시 승차	-06시 하차	-07시 승차	-07시 하차	-08시 승차	-08시 하차	-09시 승차	-09시 하차	-10시 승차	-10시 하차	-11시 승차	-11시 하차	-12시 승차	-12시 하차
14097	202001	7호선	강남구청	1849	1730	3984	10053	6351	27422	9368	108720	8816	95027	8931	34929	11154	23889
13500	202002	7호선	강남구청	1701	1808	3534	10176	5808	27444	8616	105429	7607	92662	7569	34353	8796	21253
12903	202003	7호선	강남구청	1562	1852	3256	10160	5234	27936	7094	91268	6079	83245	5835	31037	6730	17217
12304	202004	7호선	강남구청	1563	1774	3371	9801	5405	28017	7432	90674	6197	85080	6249	30230	7483	17849
11707	202005	7호선	강남구청	1653	1627	3620	10380	5485	28072	7990	94118	7014	84895	7280	31109	8964	20472
11107	202006	7호선	강남구청	1649	1752	3895	11737	6330	31922	9003	107521	7687	94121	7292	33115	9144	20335
10509	202007	7호선	강남구청	1720	1900	4035	12119	6831	33014	9718	112324	7803	96893	7557	34306	9495	21637
9914	202008	7호선	강남구청	1561	1575	3350	10723	5499	27950	8096	90015	6630	80802	6812	30884	8550	19019
9311	202009	7호선	강남구청	1311	1658	3229	10564	5795	28369	7416	91069	6175	79862	6259	28974	7922	17697
8706	202010	7호선	강남구청	1565	1515	3476	10937	6435	28470	7945	93568	7078	85841	7229	31709	9405	20312
8103	202011	7호선	강남구청	1461	1732	3535	11521	6375	29918	8495	100102	7363	91614	7372	33871	9555	21270
7500	202012	7호선	강남구청	987	1630	2856	10585	5221	26723	7090	87384	6319	80107	6728	30788	8229	18144
6896	202101	7호선	강남구청	993	1576	2908	9875	5070	26783	7561	87208	6331	81028	6688	32051	8096	18659
6291	202102	7호선	강남구청	944	1451	2546	9172	4883	24776	7023	83036	5977	75390	6472	29548	8198	19008
5688	202103	7호선	강남구청	1267	1789	3424	12277	6587	33080	8847	105527	7961	97295	8523	36685	10062	22434
5080	202104	7호선	강남구청	1352	1802	3306	12739	6705	33238	8831	106626	8127	95093	8427	35573	10131	22070
4472	202105	7호선	강남구청	1309	1744	3413	12843	6000	29658	8319	93988	7865	88736	8326	35201	10232	21742
3864	202106	7호선	강남구청	1301	1828	3536	13349	6589	33024	8985	107445	8154	96843	8208	36735	10080	22242
3257	202107	7호선	강남구청	1457	1814	3354	12927	5798	31276	8221	98573	7779	87136	8165	34167	9940	20417
2650	202108	7호선	강남구청	1354	1816	3307	12167	5738	27891	7730	89074	7469	84227	7761	33547	9561	19965
2043	202109	7호선	강남구청	1304	1678	3279	11579	5737	27061	7660	84812	7025	80952	8004	32038	9657	19868
1434	202110	7호선	강남구청	1352	1702	3659	12649	5919	28536	8329	91235	7705	88309	8434	36711	10471	23224
825	202111	7호선	강남구청	1689	1760	3586	13077	6477	30189	9217	107228	8417	100213	9054	38822	10373	23458
216	202112	7호선	강남구청	1457	1712	3399	12578	6090	29307	8814	105492	7980	96955	9054	39173	10630	23147
14006	202001	5호선	강동	11238	1121	20525	11657	60239	24271	79515	40206	43140	25992	30795	19116	29199	18311
13409	202002	5호선	강동	10301	1033	19162	11466	57254	23640	74265	39498	38161	23013	24338	16670	21804	14632
12812	202003	5호선	강동	10014	1131	18468	11328	51918	22478	64403	37997	34489	21351	18985	12836	17114	10982
12213	202004	5호선	강동	9870	1088	19064	11036	53455	23513	66793	41670	35757	22907	21603	14427	19728	12385
11616	202005	5호선	강동	10845	1148	19973	11943	56679	24803	73669	44682	39132	24666	25673	16984	24172	15540
11016	202006	5호선	강동	11372	1138	22577	12195	67132	28914	81681	50163	40992	26434	25882	16999	23306	15641
10418	202007	5호선	강동	11615	1156	23818	12524	69599	29703	85050	52295	42484	27718	27845	17827	24995	16611
9821	202008	5호선	강동	10327	1161	19808	10767	51658	23450	64219	42883	33735	23323	22166	14808	21002	13343
9218	202009	5호선	강동	10295	1193	19045	10511	50064	23714	60903	42399	31648	22437	19478	13358	17841	12755
8613	202010	5호선	강동	10919	1222	19395	11034	52500	25821	65172	44565	35042	24541	24280	15919	22214	15196
8010	202011	5호선	강동	11211	1202	20104	11053	55357	27891	70473	50201	36199	26691	23872	17207	21714	15104
7407	202012	5호선	강동	10298	1083	18330	10664	47034	23800	60623	44268	31901	24488	19757	14773	17859	12891
6803	202101	5호선	강동	9346	1084	17558	9675	46144	23590	59210	42993	31683	23989	20282	15035	18951	13386
6198	202102	5호선	강동	8779	1096	16299	8575	43435	21655	56361	39796	30175	22836	20202	14686	19156	13262
5592	202103	5호선	강동	11006	1405	21187	11280	59586	29156	71105	50814	36582	28194	24948	17833	22846	15842
4984	202104	5호선	강동	10617	1521	20050	11406	56077	28371	67190	50356	34161	27273	23095	17692	21294	15224
4376	202105	5호선	강동	10460	1443	18855	11341	50508	25185	61993	45097	33436	24725	23116	17285	21864	15934
3768	202106	5호선	강동	10874	1589	20211	11957	56427	28884	68484	50582	33851	25766	22997	17247	21434	15482
3161	202107	5호선	강동	11203	1439	19171	11864	50502	26690	61882	47046	32306	24737	21271	17038	20015	14496
2554	202108	5호선	강동	10709	1244	17370	10734	46223	25073	57627	42422	30248	23779	20689	16061	20169	14384
1947	202109	5호선	강동	9940	1133	16658	9974	45915	24287	55348	40733	29405	22778	20600	15882	19851	14397
1338	202110	5호선	강동	10504	1320	17900	10942	48875	25296	60164	44306	32614	24998	23347	18252	22661	16525
729	202111	5호선	강동	10498	1357	18925	11039	54347	27664	70532	49273	34957	25815	23901	18513	22555	16023
120	202112	5호선	강동	9998	1352	18879	10755	52202	26694	69239	48724	34807	25850	23389	18251	22519	15799
14098	202001	7호선	건대입구	4252	944	4807	7607	11584	17922	20568	32457	16007	32556	13685	23251	15179	23732
13501	202002	7호선	건대입구	3751	810	4347	6848	10565	15188	19256	29493	13335	25994	10510	17738	11413	16229
12904	202003	7호선	건대입구	3754	827	4139	6257	9493	14459	15879	24901	12251	18694	8391	16132	8608	11625
12305	202004	7호선	건대입구	3825	853	4363	6006	9999	14344	16311	24922	12482	20591	9275	14536	9565	12818
11708	202005	7호선	건대입구	4043	894	4554	6360	10832	14590	17472	27382	13552	23184	10838	17213	11049	15744

4. Data Analysis & Visualization :
데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

- 데이터 분석 유형 중 예측 분석에 해당된다.
- 예측 분석은 과거의 데이터를 기반으로 향후 추세를 예측하기 위해 사용되는 방법
- 이 프로젝트의 경우 시간별 지하철 이용 승객 수를 바탕으로 향후 지하철 이용 승객 수의 추세를 예측하고 본인의 지하철 이용에 참고하기 위함.

4. Data Analysis & Visualization :
데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

어떻게 분석 할 것인가?

- 본인이 이동하는 경로 상 모든 역의 승차 승객 수를 각 시간대별로 모두 더한 값(A),
- 본인이 이동하는 경로 상 모든 역의 하차 승객 수를 각 시간대별로 더한 값(B)
- 우선 (A-B)/(전체 시간별, 이동경로 상에 있는 역별 행의 개수)의 값을 구해본다.

예상 결과

- 각 시간대별로 승객이 얼마나 집중되는 지를 파악할 수 있을 것이라고 생각한다.

4. Data Analysis & Visualization :
데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

(A-B)/(전체 시간별 역별 행의 개수)를 구하는 파이썬 코드

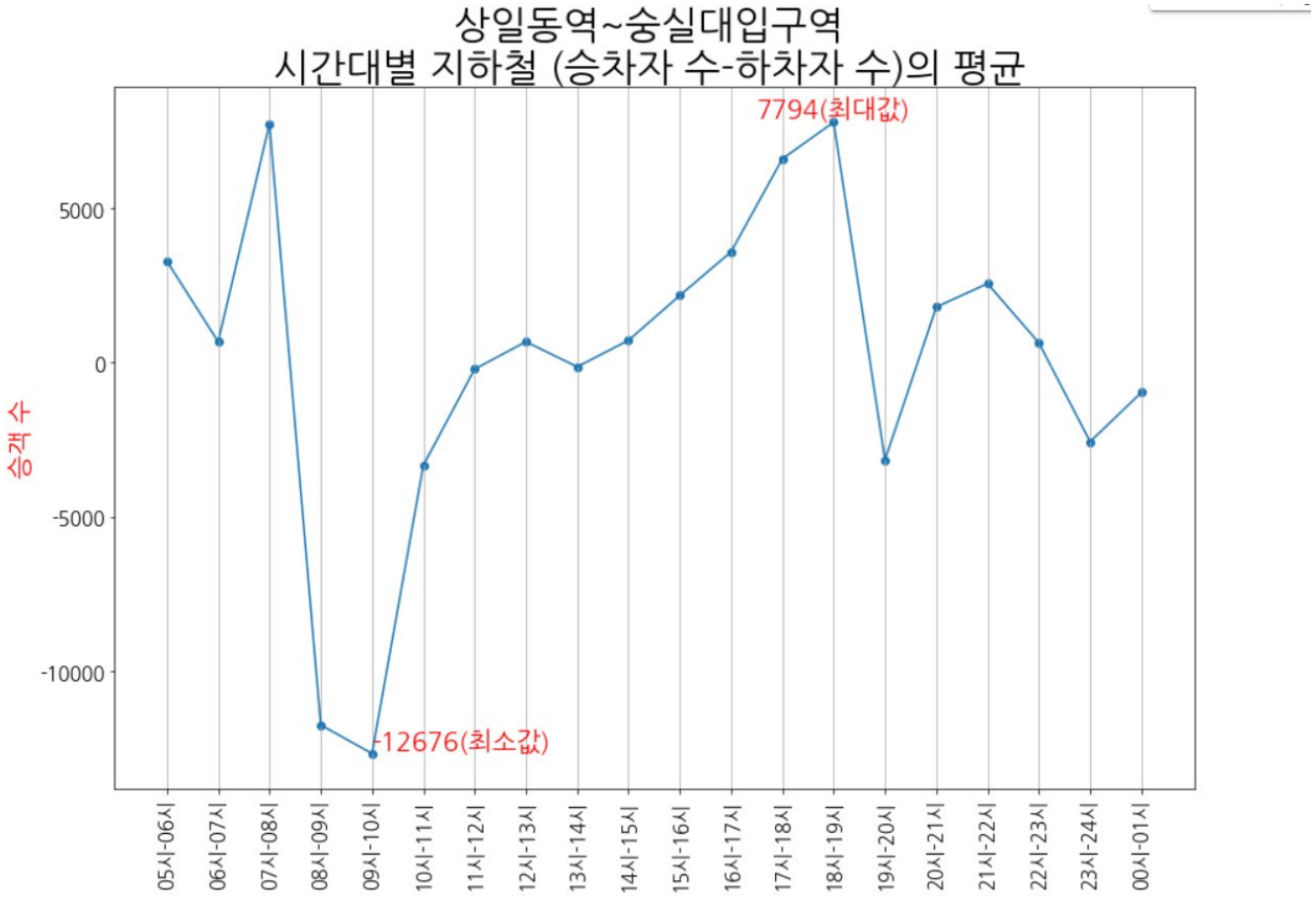
```
# 딕셔너리 자료형 활용
sum_dict={}

# 전체 행의 개수 (각 시간대별 인원 값을 전체 행의 개수로 나누기 위함)
num_of_rows=int(df_station.shape[0])

# 반복문을 활용하여 각 시간대 (승차인원 - 하차인원)의 값을 정수형으로 반환 후 딕셔너리에 저장
for i in range(5,24):
  if i<=8:
    sum_dict[f'0{i}시-0{i+1}시']=int(int(df_station[f'0{i}시-0{i+1}시 승차인원'].sum() - df_station[f'0{i}시-0{i+1}시 하차인원'].sum()) / num_of_rows)
  elif i==9:
    sum_dict[f'0{i}시-{i+1}시']=int(int(df_station[f'0{i}시-{i+1}시 승차인원'].sum() - df_station[f'0{i}시-{i+1}시 하차인원'].sum()) / num_of_rows)
  else:
    sum_dict[f'{i}시-{i+1}시'] = int(int(df_station[f'{i}시-{i+1}시 승차인원'].sum() - df_station[f'{i}시-{i+1}시 하차인원'].sum()) / num_of_rows)
sum_dict['00시-01시'] = int(int(df_station['00시-01시 승차인원'].sum() - df_station['00시-01시 하차인원'].sum()) / num_of_rows)
sum_dict
```

```
{'00시-01시': -967,
 '05시-06시': 3291,
 '06시-07시': 679,
 '07시-08시': 7735,
 '08시-09시': -11750,
 '09시-10시': -12676,
 '10시-11시': -3332,
 '11시-12시': -211,
 '12시-13시': 676,
 '13시-14시': -136,
 '14시-15시': 723,
 '15시-16시': 2176,
 '16시-17시': 3592,
 '17시-18시': 6608,
 '18시-19시': 7794,
 '19시-20시': -3158,
 '20시-21시': 1802,
 '21시-22시': 2563,
 '22시-23시': 642,
 '23시-24시': -2565}
```

4. Data Analysis & Visualization :

데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

이전 슬라이드 파이썬 코드 실행결과



4. Data Analysis & Visualization :

데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

이전 슬라이드 실행결과를 시각화한 그래프

```python
# 시각화
import matplotlib.pyplot as plt
plt.rcParams['font.family']=['NanumGothic','sans-serif']
plt.rcParams['axes.unicode_minus'] = False

# 그래프 그리는 코드
plt.figure(figsize=(15,10))
plt.title('''상일동역~숭실대입구역
이동경로 시간대별 지하철 승차자 수-하차자 수의 평균 ''',fontsize=30)
plt.xticks(rotation=90, fontsize=15)
plt.yticks(fontsize=15)
plt.grid(True,axis='x')
plt.plot(timezone, num_of_get_on_minus_get_off)
plt.scatter(timezone,num_of_get_on_minus_get_off)

# 그래프에 텍스트를 나타내기 위해 기존의 딕셔너리 key,value값을 서로 맞바꾼다.
sum_graph=dict(zip(sum_dict.values(),sum_dict.keys()))

# 최솟값 표시
plt.text(sum_graph[int(min(num_of_get_on_minus_get_off))],
         int(min(num_of_get_on_minus_get_off)),
         str(int(min(num_of_get_on_minus_get_off)))+'(최소값)',
         color='r',
         verticalalignment='bottom',
         fontsize=20)

# 최댓값 표시
plt.text(sum_graph[int(max(num_of_get_on_minus_get_off))],
         int(max(num_of_get_on_minus_get_off)),
         str(int(max(num_of_get_on_minus_get_off)))+'(최대값)',
         color='r',
         horizontalalignment='center',
         verticalalignment='bottom',
         fontsize=20)
```

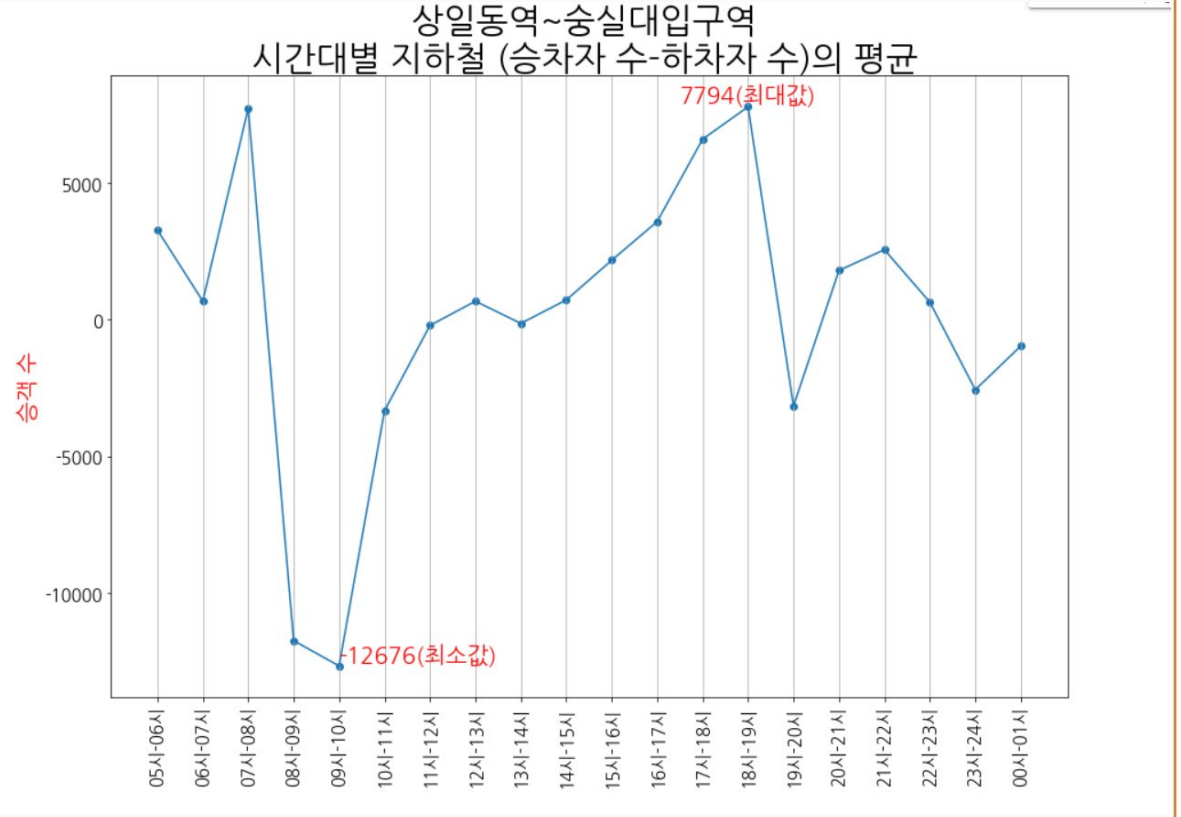
4. Data Analysis & Visualization :

데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

앞 슬라이드 그래프 시각화 파이썬 코드

4. Data Analysis & Visualization :
데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

- 상일동역에서 출발해서 숭실대입구역에 도착하기까지 약 50분~ 60분이 걸린다.
- 값의 최대는 18시-19시이고 최소는 09시-10시이다.
- 08시에서 10시까지는 승차자 수에 비해 하차자 수의 평균이 더 많다.
- 19시-20시까지의 값이 급격히 감소한다.
- 결론적으로 08시-10시에 승차 후 19시나 23시 이후에 하차하면 된다.



4. Data Analysis & Visualization :
데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

추가 분석

- 앞에서 도출한 결과에는 오류가 있음
- 그 이유는 1시간 단위로 승/하차자 수를 구한 것이 이산적인 값이기 때문 (연속성 x)
- 지하철 이용 특성상, 시간의 흐름은 연속적인 값이므로 이에 따른 승객 수 변화를 고려하는 것이 필요함.
- 따라서, 이렇게 도출한 결과를 분석의 결과로 활용하기보다는 **누적되는 값**을 구하는 것이 더 좋을 것이라고 판단함

4. Data Analysis & Visualization :
데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

누적값 구하기

```
from matplotlib import pyplot as plt
import itertools

# 누적합 (0번 인덱스 값 : 오전 05시-오전06시, 마지막 인덱스 값 : 오전 00시-오전01시)
result = list(itertools.accumulate(sum_dict.values()))
result
```

결과 (첫번째 값인 05시-06시의 값부터 마지막 값인 00시-01시 값까지 계속 더한 값)

- [3291, 3970, 11705, -45, -12721, -16053, -16264, -15588, -15724, -15001, -12825, -9233, -2625, 5169, 2011, 3813, 6376, 7018, 4453, 3486]

결과를 바탕으로 그래프를 그려보겠다.

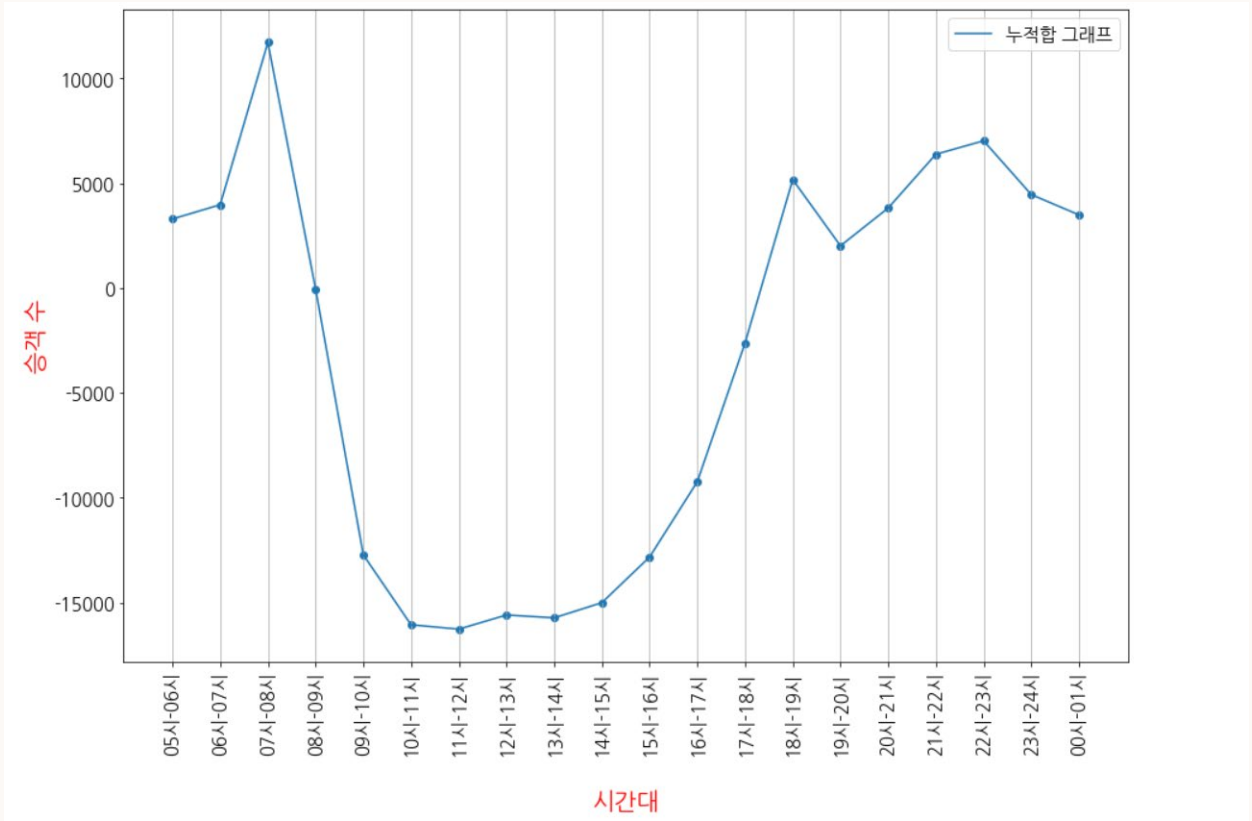
4. Data Analysis & Visualization :
데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

분석결과

- 그래프는 시간대별로 누적되는 승객 수의 값을 그래프로 표현한 것이다.
- 09시부터 17시까지의 승객 수는 하위 8개 값에 속한다.

결론

- 위 그래프를 통해 해당 시간대에 지하철을 이용하는 것이 혼잡도가 덜 할 것임을 예상해볼 수 있다.



4. Data Analysis :
데이터 분석 유형을 고려한 분석

앞 슬라이드의 그래프를 그리는 파이썬 코드

```
from matplotlib import pyplot as plt

# 그래프 그리는 코드
plt.figure(figsize=(15,10))
plt.xticks(rotation=90, fontsize=15)
plt.yticks(fontsize=15)
plt.grid(True,axis='x')
plt.plot(timezone,result, label='누적합 그래프')
plt.scatter(timezone,result)
plt.xlabel('시간대',fontsize=20, labelpad=30, color='r')
plt.ylabel('승객 수',fontsize=20, color='r')
plt.legend(fontsize=15)
```

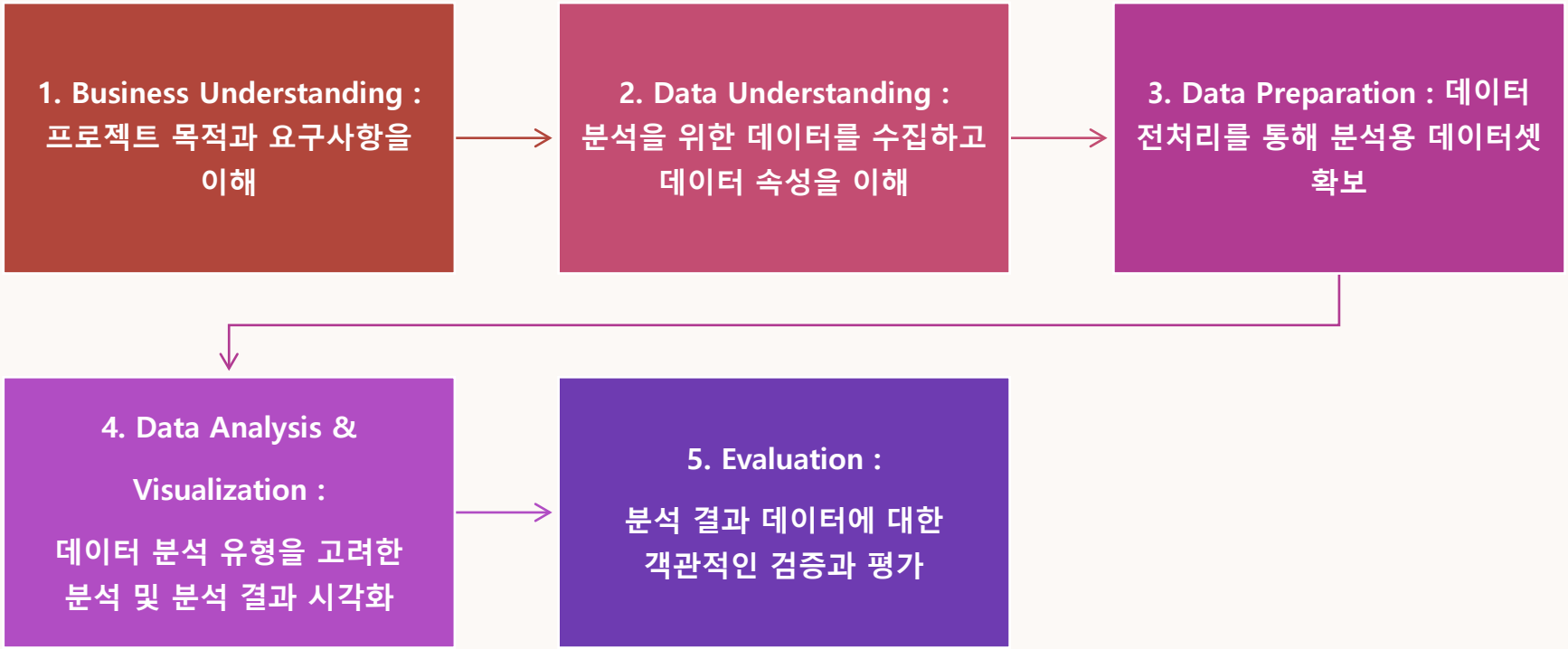
5. Evaluation :
분석 결과 데이터에 대한 객관적인 검증과 평가

- **데이터 평가**
  - 분석 대상 데이터의 오염(Data corruption)이 있었나? : 대상 데이터에 오염이 있지는 않았다. info()함수를 활용해보았는데 결측치 (null)값이 없었다.
  - 승객 데이터 자체에는 문제가 없었지만 실제 지하철 운행시간에 맞춰서 시간대를 정하고 이동경로에 맞는 역을 추출했다.
- **분석 프로세스 평가**
  - 분석 과정에서 계획했던 모든 단계를 제대로 실행하였는가? : 모든 단계를 제대로 실행했다.
- **분석 결과 평가**
  - 상식과 일반적인 관점에서 수용 가능한 결과인가? : 상식선에서 출퇴근 시간대에 사람이 많이 몰리는 것으로 예측가능하며, 이는 수용 가능하다.
  - 분석 결과가 프로젝트 초기에 세웠던 목표인 쾌적한 지하철 이용에 조금이나마 도움이 될 것같다.

추가

- 본인이 서울시 지하철 호선별, 역별, 시간대별 승/하차 인원 정보를 활용하여 얻은 분석결과와 비교하기 위해 아래의 데이터로 추가 분석을 진행해 봄
- 서울교통공사_지하철혼잡도정보 데이터 : https://www.data.go.kr/data/15071311/fileData.do
- 서울교통공사 1-8호선 30분 단위 평균 혼잡도로 30분간 지나는 열차들의 평균 혼잡도 (단위 : %, 단위기준 : 열차 1량의 승차인원 = 160명 = 100%)
- 2년 단위 업데이트 자료이며 최신 수정일이 2021년 12월 22일

(앞서 살펴봤던) 데이터 분석 절차 상세



(추가)1. Business Understanding :

프로젝트 목적과 요구사항을 이해

프로젝트의 목적 :

* 본인이 지하철을 이용할 때 좀 더 쾌적하게 이동할 수 있는 시간대를 찾는 것.

프로젝트의 요구사항 :

* ~~서울시 지하철의 시간대별, 역별 승/하차 승객 수 관련 데이터~~ -> 지하철 혼잡도 데이터

* 본인의 이동경로를 고려한 역(상일동~군자~숭실대입구)만을 대상으로 분석

(추가)2. Data Understanding :
분석을 위한 데이터를 수집하고 데이터 속성을 이해

데이터 수집 : 서울교통공사 지하철혼잡도 정보 데이터

```
# 데이터 수집
import pandas as pd
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)

df_plus = pd.read_csv('/content/gdrive/My Drive/data_analysis/data/seoul-metro-complexity-data/서울교통공사_혼잡도_20191231.csv',encoding='cp949')
df_plus
```

데이터 속성 이해

```
# 전체 데이터의 구조 파악
df_plus.info()
```

colab 링크 : 이 프로젝트의 모든 파이썬 소스코드는 이 링크의 '추가' 아래부분부터 보시면 확인하실 수 있습니다.

또한 해당 결과물이 모두 출력된 상태이므로 코드 셀을 따로 실행시킬 필요 없습니다 :)

(추가)3. Data Preparation :
데이터 전처리를 통해 분석용 데이터셋 확보

```
# 실제 이동 경로에 해당되는 역만 추출
station = ['상일동','고덕','명일','굽은다리','길동','강동',
           '천호','광나루','아차산','군자',
           '어린이대공원','건대입구','뚝섬유원지',
           '청담','강남구청','학동','논현','반포','고속터미널',
           '내방','총신대입구','남성','숭실대입구']
df_plus = df_plus[df_plus['역명'].isin(station)]

# 평일만 포함시킨다.
saturday = df_plus[(df_plus['조사일자']=='토요일')].index
sunday = df_plus[(df_plus['조사일자']=='일요일')].index
df_plus = df_plus.drop(saturday)
df_plus = df_plus.drop(sunday)

# 단, 2개 이상 호선이 겹치는 역의 경우, 실제 이동경로가 아닌 호선의 경우 제거해야한다.

cheonho = df_plus[(df_plus['역번호']==2812)].index
df_plus = df_plus.drop(cheonho)

konkuk = df_plus[(df_plus['역번호']==212)].index
df_plus=df_plus.drop(konkuk)

express_bus_terminal_3 = df_plus[(df_plus['역번호']==329)].index
df_plus = df_plus.drop(express_bus_terminal_3)

chong_sin = df_plus[(df_plus['역번호']==432)].index
df_plus = df_plus.drop(chong_sin)

df_plus
```

- 실제 이동경로에 해당되는 역만 추출
- 이전의 분석과 달리 역 데이터에 병기하는 경우는 없었음
- 데이터에 평일, 토요일, 일요일 정보가 있어서 해당 데이터 중 본인이 평일에만 이동하므로 평일만 활용하고 나머지 데이터는 제거
- 이전 분석에서 진행했던 것처럼 2개 이상 호선이 겹치는 경우 실제 이동경로가 아닌 호선의 경우 제거해야 함

(추가) 4. Data Analysis & Visualization :

데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

```python
# 혼잡도 평균
# 딕셔너리 자료형 활용
sum_plus_dict={}

# 전체 행의 개수 (각 시간대별 값을 전체 행의 개수로 나누기 위함)
num_of_rows=int(df_plus.shape[0])

# 반복문 활용
for i in range(5,24+1):
  zero=0
  three=3
  if i== 5:
    df_plus[f'{i}시{three}0분'] = df_plus[f'{i}시{three}0분'].astype('float')
    sum_plus_dict[f'{i}시 {three}0분 합계 평균']=int(df_plus[f'{i}시{three}0분'].sum() / num_of_rows)
    continue
  df_plus[f'{i}시{zero}0분'] = df_plus[f'{i}시{zero}0분'].astype('float')
  sum_plus_dict[f'{i}시 {zero}0분 합계 평균']=int(df_plus[f'{i}시{zero}0분'].sum() / num_of_rows)

  df_plus[f'{i}시{three}0분'] = df_plus[f'{i}시{three}0분'].astype('float')
  sum_plus_dict[f'{i}시 {three}0분 합계 평균']=int(df_plus[f'{i}시{three}0분'].sum() / num_of_rows)

print(sum_plus_dict)
```

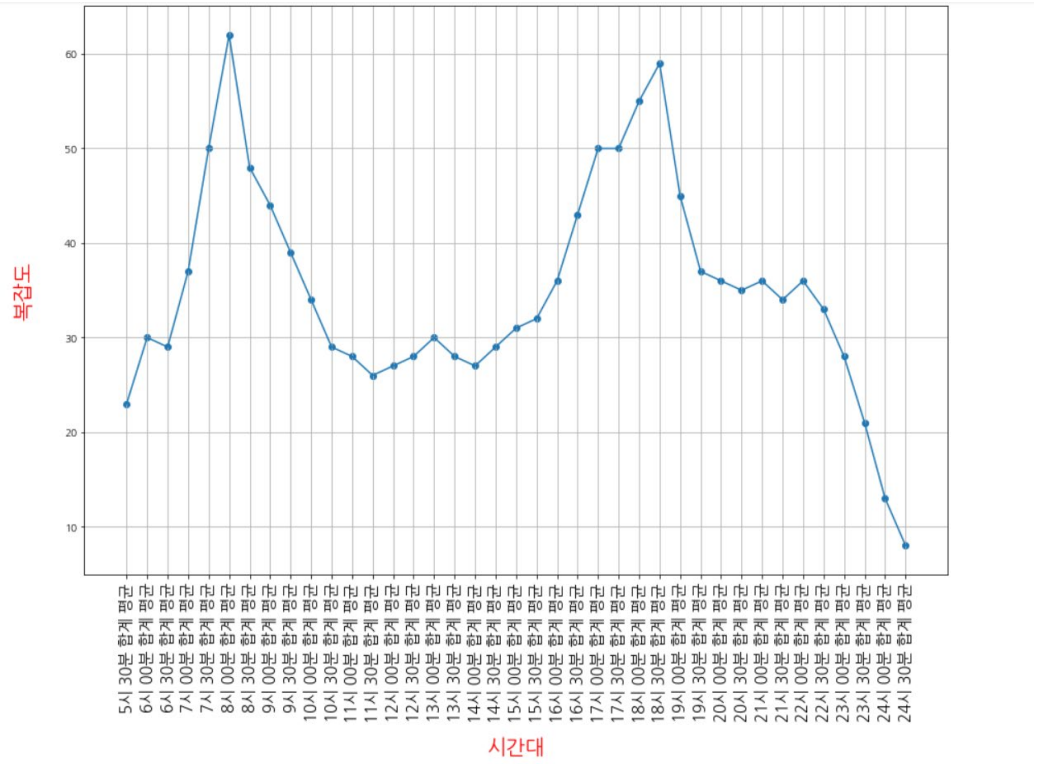
코드 실행 결과 :

{'5시 30분 합계 평균': 23, '6시 00분 합계 평균': 30,
'6시 30분 합계 평균': 29, '7시 00분 합계 평균': 37,
'7시 30분 합계 평균': 50, '8시 00분 합계 평균': 62,
'8시 30분 합계 평균': 48, '9시 00분 합계 평균': 44,
'9시 30분 합계 평균': 39, '10시 00분 합계 평균': 34,
'10시 30분 합계 평균': 29, '11시 00분 합계 평균': 28,
'11시 30분 합계 *평균': 26, '12시 00분 합계 평균': 27,*
'12시 30분 합계 평균': 28, '13시 00분 합계 평균': 30,
*'13시 30분 합계 평*균': 28, '14시 00분 합계 평균': 27,
'14시 30분 합계 평균': 29, '15시 00분 합계 평균': 31,
'15시 30분 합계 평균': 32, '16시 00분 합계 평균': 36,
'16시 30분 합계 평균': 43, '17시 00분 합계 평균': 50,
'17시 30분 합계 평균': 50, '18시 00분 합계 평균': 55,
'18시 30분 합계 평균': 59, '19시 00분 합계 평균': 45,
'19시 30분 합계 평균': 37, '20시 00분 합계 평균': 36,
'20시 30분 합계 평균': 35, '21시 00분 합계 평균': 36,
'21시 30분 합계 평균': 34, '22시 00분 합계 평균': 36,
'22시 30분 합계 평균': 33, '23시 00분 합계 평균': 28,
'23시 30분 합계 평균': 21, '24시 00분 합계 평균': 13,
'24시 30분 합계 평균': 8}

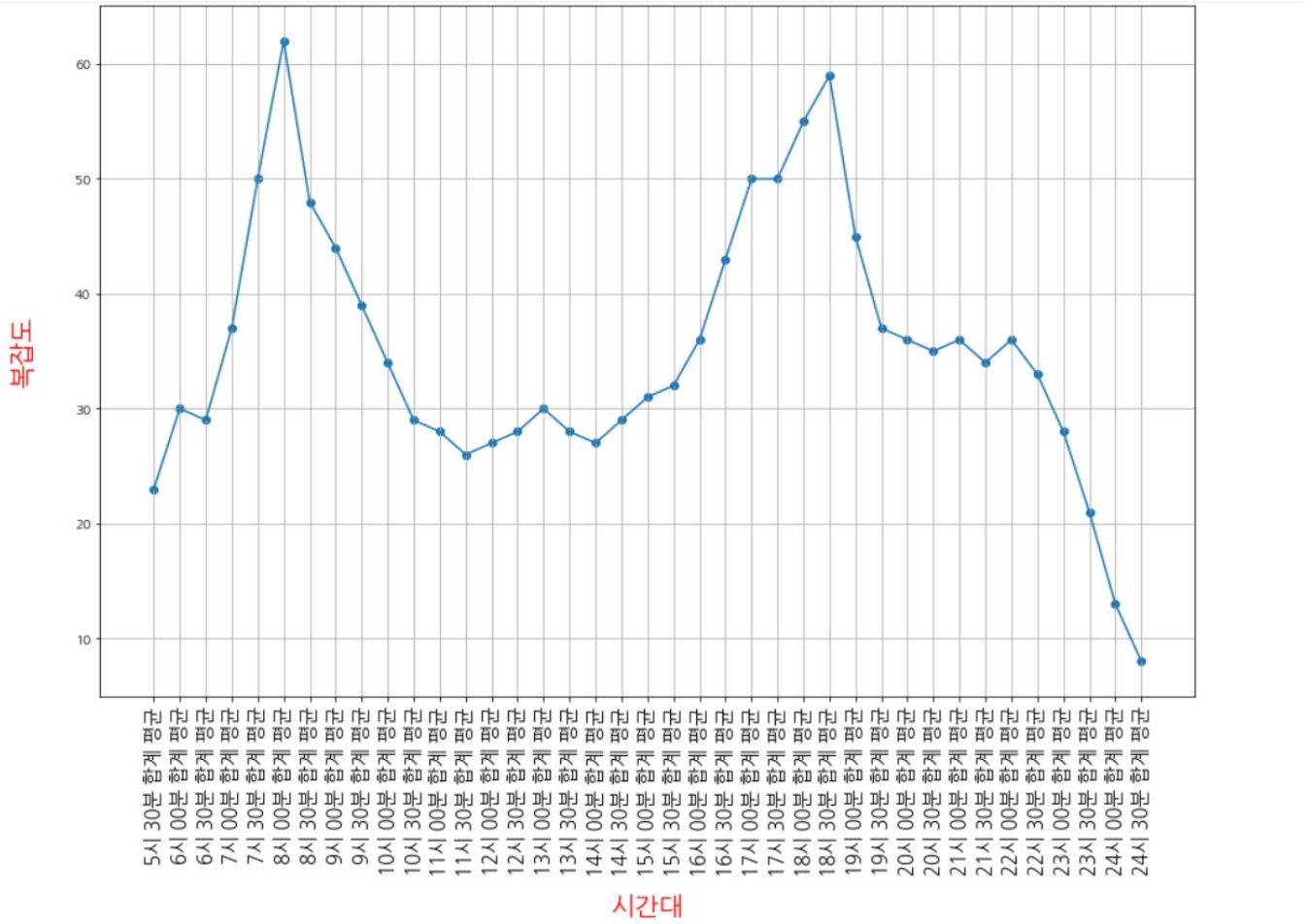
혼잡도의 평균을 구하기 위한 코드로, 딕셔너리 자료형으로 반환함.

(추가) 4. Data Analysis & Visualization :
데이터 분석 유형을 고려한 분석 및 분석 결과 시각화

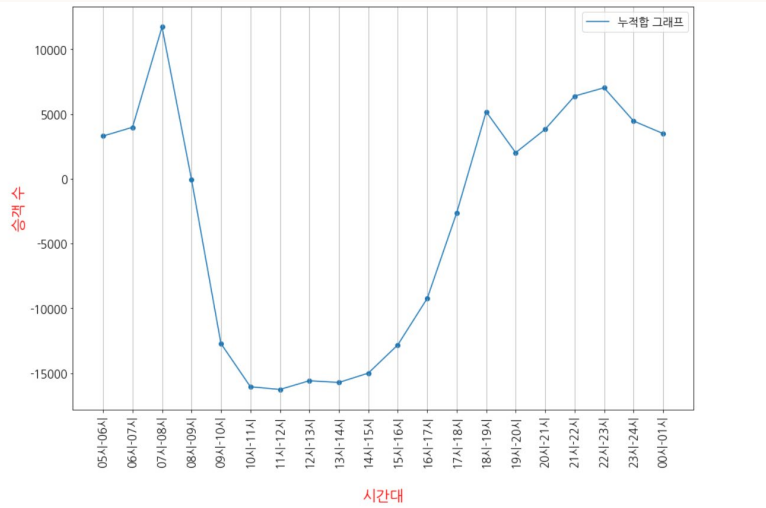
```python
# 데이터 시각화 위해 import
%matplotlib inline
import matplotlib.pyplot as plt

# 한글 폰트 적용
plt.rcParams['font.family']=['NanumGothic','sans-serif']
plt.rcParams['axes.unicode_minus'] = False

# 그래프 그리는 코드
timezone_complexity=list(sum_plus_dict.keys())
complexity_percentage=list(sum_plus_dict.values())

plt.figure(figsize=(15,10))
plt.xticks(rotation=90,fontsize=15)
plt.grid(True,axis='x')
plt.grid(True,axis='y')
plt.xlabel('시간대',fontsize=20, color='r', labelpad=15)
plt.ylabel('복잡도',fontsize=20, color='r', labelpad=30)
plt.plot(timezone_complexity, complexity_percentage)
```



(추가) 4. Data Analysis & Visualization :
분석 결과 시각화



(추가) 4. Data Analysis & Visualization :
두 그래프 해석

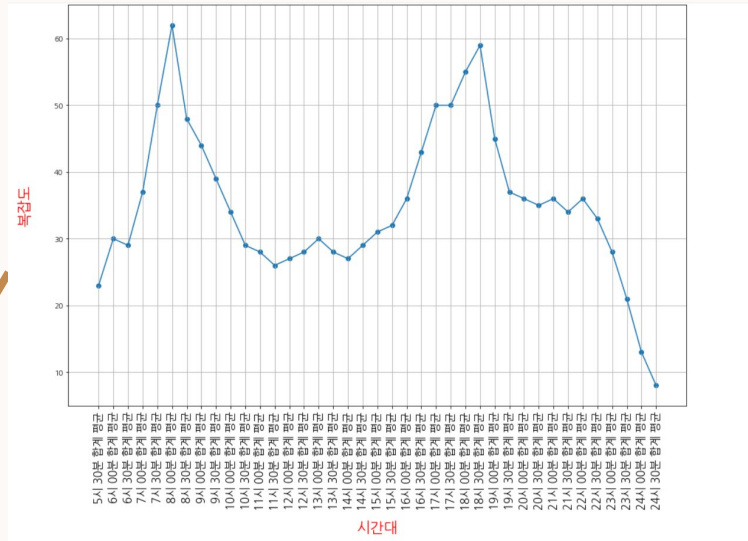


서울시 지하철 호선별, 역별, 시간대별 승/하차 인원 정보 데이터기반 누적합 그래프

왼쪽 그래프 :
1. 07시-08시 승객 수가 제일 많음
2. 09시-16시까지 승객수 누적합이 하위값에 해당
3. 18시-19시 급격히 증가

오른쪽 그래프 :
1. 8시00분 합계 혼잡도가 급격히 높아짐.
2. 10시 00분-16시 00분 혼잡도 평균의 값이 하위권
3. 18시 00분 혼잡도 합계 평균과 18시 30분 혼잡도 합계평균이 상위 3위와 2위에 해당

- 비록 완전히 똑같지는 않지만 **유사한 경향**을 보여줌.



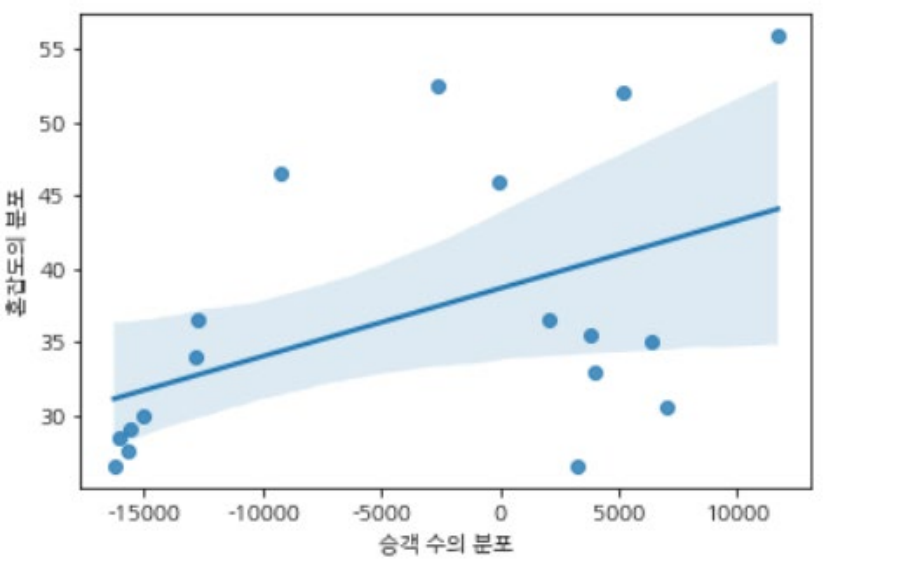
서울교통공사_지하철혼잡도정보
데이터기반 그래프

(추가) 4. Data Analysis & Visualization :
두 그래프 상관관계 및 p-value 분석

- 앞서 두 그래프의 모양만 보고 해석했지만, 통계적 검증을 활용하는 것이 더 확실한 검증방법
- 따라서, 가설 검정 기법과 피어슨 상관계수 및 p-value(유의 확률)와 같은 통계적 검증기법을 활용한다.
- (참고) 가설 검정 기법은 귀무가설 – 대립가설 - 검정 – 결론의 방식을 활용할 것이다.

(추가) 4. Data Analysis & Visualization :
두 그래프 상관관계 및 p-value 분석

- 관련 코드를 첨부했으나 colab에서 열어 보실 것을 권장

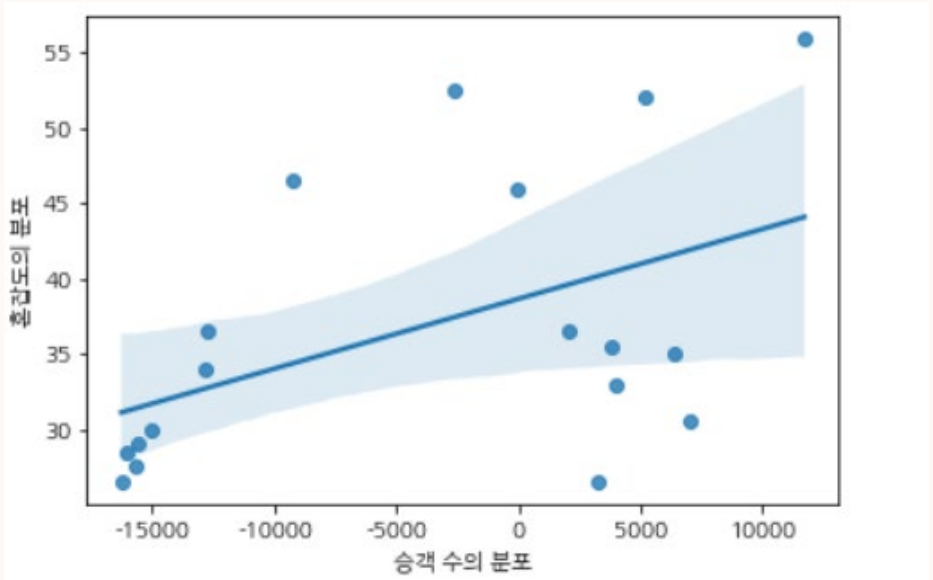


원형 점으로 표기된 것은 각 분포의 값을 의미한다.
선으로 표기된 것은 추세선으로 점들의 분포에 따라 기울기가 결정된다.
하늘색으로 옅게 표시된 부분은 추세선의 95% 신뢰구간을 의미한다.

```
# 상관관계 분석
# https://mindscale.kr/course/basic-stat-python/6/
# https://ordo.tistory.com/100
# https://angeloyeo.github.io/2020/03/29/p_value.html
import numpy as np

# 두 리스트의 길이가 달라서 상관계수 분석을 위해서는 두 리스트의 길이를 맞추는 것이 필요.
(result:20,list(sum_plus_dict.values()):39)
# list(sum_plus_dict_values())의 값은 30분 간격이므로 이를 1시간 간격으로 맞추기 위해 x시 00분과
x시 30분의 값을 더한 후 평균(mean)을 구한다.
# 따라서 두 리스트에서 제일 마지막 값을 하나씩 제거하기로 한다.(19:38)
print(len(result), len(list(sum_plus_dict.values())))

sum_plus_on_time_idx=list()
sum_plus_thirty_idx=list()

# 반복문을 실행해가며 list(sum_plus_dict.values())를 정각과 30분단위의 두개로 나눠준다.
for i in range(len(list(sum_plus_dict.values()))):
  if i==0:
    sum_plus_thirty_idx.append(list(sum_plus_dict.values())[i])
    continue
  if i % 2 == 0 and i != 0:
   sum_plus_thirty_idx.append(list(sum_plus_dict.values())[i])
  elif i%2 ==1 and i!=0:
    sum_plus_on_time_idx.append(list(sum_plus_dict.values())[i])

# 제일 마지막에 있는 값을 하나씩 빼자, 단 실행할때마다 값이 하나씩 없어지니 특정 값으로 지워보자.
sum_plus_thirty_idx.remove(8)

print('result:',result)
print('len(result):', len(result))
print(list(sum_plus_dict.values()))

print(sum_plus_on_time_idx)
print(sum_plus_thirty_idx)

# sum_plus_on_time의 값은 정각의 값만 들어와있고, sum_plus_thirty_idx값은
# 30분의 값만 들어와 있으므로
# 이를 1시간 간격으로 맞추기 위해 x시 00분과 x시 30분의 값을 더한 후 평균(mean)을 구한다.
# 이렇게하면 간격의 조정이 된다. (result는 05시-06시부터 시작, mean은 5시30분+06시00분이 합쳐진 값
부터 시작)
mean=list()
for j in range(len(sum_plus_on_time_idx)):
  var1 = sum_plus_on_time_idx[j]
  var2 = sum_plus_thirty_idx[j]
  var3 = (var1 + var2)/2
  mean.append(var3)
  var3=0
print('mean :',mean)


# 23시-24시에 이동하지는 않을 것이므로 이를 제거한다.
mean.remove(17.0)
# result.remove(4453)

print(len(result),len(mean))

# from matplotlib import pyplot as plt
# plt.scatter(result, mean, alpha=0.7)
# plt.xlabel('승객수의 분포')
# plt.ylabel('혼잡도의 분포')
# plt.figure()
# plt.show()
import seaborn as sns
plot = sns.regplot(result,mean)
plot.set(xlabel='승객 수의 분포', ylabel='혼잡도의 분포')

# 출력값 : 상관계수, p-value : 0.05보다 작으면 귀무가설을 기각하고 대립가설을 채택한다.
# 피어슨 상관계수에 따르면 0.46의 값은 뚜렷한 양적 선형관계를 가진다.
# p-value는 0.05보다 작은 0.049를 보이므로 귀무가설(상관관계가 없다)를 기각하고 대립가설(상관관계가
있다)를 채택한다.
import scipy.stats as stats
print('(상관계수, p-value) :',stats.pearsonr(result,mean))
```

(추가) 4. Data Analysis & Visualization :
두 그래프 상관관계 및 p-value 분석

- 귀무가설 : 두 변수(승객 수의 분포(x축), 혼잡도의 분포(y축)) 사이의 상관관계가 없다.
- 대립가설 : 두 변수 사이의 상관관계가 있다.
- 검정 : 피어슨 상관계수를 구해본 결과 약 0.47에 해당하는 값이 도출되었다. 이는 뚜렷한 양적 선형관계를 보인다고 할 수 있다. (0.3과 0.7 이내에 값이 있으면 이와 같이 칭한다.)
- 또한 p-value를 구한 결과 0.05보다 작은 약 0.045이므로 귀무가설을 기각하고 대립가설(상관관계가 있다)을 채택한다.

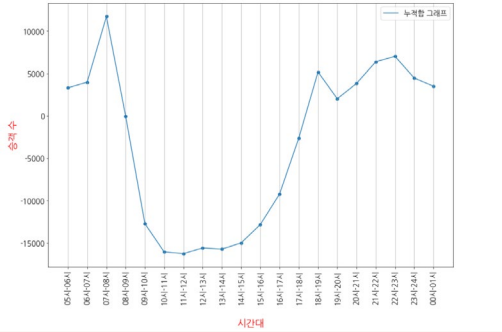


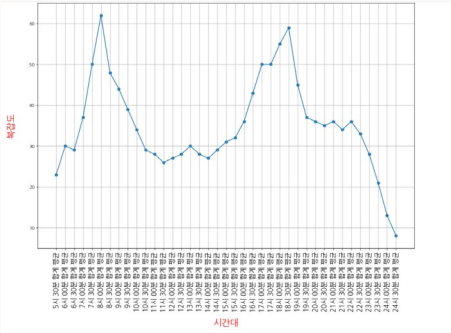

```
import scipy.stats as stats
print('(상관계수, p-value) :',stats.pearsonr(result,mean))
```

(상관계수, p-value) : (0.4777126309213842, 0.044963149310342405)

(추가) 4. Data Analysis & Visualization : 결론

- 앞 슬라이드의 통계적 검증으로부터 승객의 누적합 그래프와 혼잡도 그래프가 상관관계가 있다는 것을 알 수 있다. 즉, 두 그래프의 결과의 유사성을 알 수 있다.
- 본인이 구한 승객의 누적합 그래프와 혼잡도 그래프를 종합해보면 오전 8시 이전에 출발한 후 오후 6시 전후 1시간 간격을 제외한 시간대에 지하철을 타고 돌아오면 혼잡도가 덜하다는 결론을 낼 수 있다.





5. Evaluation :
분석 결과 데이터에 대한 객관적인 검증과 평가

- **데이터 평가**
  - 분석 대상 데이터의 오염(Data corruption)이 있었나? : 대상 데이터에 오염이 있지는 않았다. info()함수를 활용해보았는데 결측치 (null)값이 없었다.
  - 다만 0시에서 1시의 값은 본인이 이동하지 않는 시간대라 전처리 과정에서 제외했다.
- **분석 프로세스 평가**
  - 분석 과정에서 계획했던 모든 단계를 제대로 실행하였는가? : 모든 단계를 제대로 실행했다.
- **분석 결과 평가**
  - 상식과 일반적인 관점에서 수용 가능한 결과인가? : 상식선에서 출퇴근 시간대에 사람이 많이 몰리는 것으로 예측가능하며, 이는 수용 가능하다.
  - 분석 결과가 프로젝트 초기에 세웠던 목표인 쾌적한 지하철 이용에 조금이나마 도움이 될 것같다.

느낀점

- 간단한 데이터분석 기법과 통계적 검정을 통해 실생활의 문제에 대한 인사이트를 얻을 수 있어서 뜻깊었다. 비록 분석 결과가 어느 정도 예상 가능하긴 했으나(출/퇴근 시간대 승객수 증가) 전체 지하철 노선의 데이터가 아닌 본인의 이동경로를 고려한 분석이라 더 의미있었다.
- 데이터 자체는 아무 것도 말해주지 않는다. 즉, 데이터를 어떻게 이해하고 해석하는 지에 따라 분석의 결과가 다르게 나타난다는 것을 깨달았다.
- 데이터 분석 과정에서 데이터의 속성에 따라서 데이터 분석의 방법이 달라질 수도 있다는 것을 깨달았다. 처음의 방식대로 활용하여 (승차자 수 – 하차자 수)의 평균을 구하면 지하철 이용 특성상 시간에 따라 누적되는 승객 수를 분석할 수 없게 된다. 따라서 분석의 결과로 적절하지 않을 것이다.