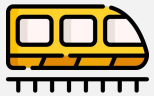
지하철 승하차 인원 예측 시스템

제로베이스 머신러닝 프로젝트 1조

김양우, 강민주, 원수진

목 차



주제 데이터 EDA 전처리 모델 결과 결론

1 주제

매일이 지옥인 '지옥철', 내가 이용할 시간의 승/하차 이용객 수를 예측할 수 있으면 이런 문제를 해결할 수 있지 않을까?

녹색경제신문 | 12시간 전

[유통가레이더] "지옥철 안타도 되나"...하림산업, 유연근무제 도입...

종합식품기업 하림산업이 판교 근무 임직원을 대상으로 출퇴근 시간을 자율적으로 조정하는 유연근무제를 도입했습니다. 하림산업은 포스트코로나 시대에 발맞...

한국경제 | A34면 1단 | 1일 전 | 네이버뉴스

[데스크 칼럼] 수도권 지옥철 이대로 방치할건가

'김포 지옥철'로 불리는 까닭이다. 이 노선의 명성(?)은 국내 1위 혼잡률(285%)로도 확인된다. 서울 최대... 김포도시철도 혼잡률 '국내 1위' 역대 최악의 교통 행정 사...

서울신문 PiCK | 2023.02.09. | 네이버뉴스

"야근·지옥철 싫어 2시간 일찍 출근하는데...물 흐린다네요"

통근지옥과 야근이 싫어 일찍 출근하는 직원이 직장 동료로부터 "분위기 흐리지 말고 정시 출근하라"는 말을 들었다며 억울함을 토로했다. 최근 '일찍 출근하는 ...

1칸에 300명 빽빽...출퇴근 '지옥철' 위험한 일상

머니투데이 | 김지현 기자, 기성훈 기자, 정진우 기자 2022.11.03 05:00

https://news.mt.co.kr/mtview.php?no=2022110223121684163&type=1 기사주소 복사

1 주제

- 지하철 승하차 인원 예측 시스템

1

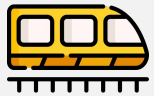
지하철 역명 / 시간 /
평일·주말 / 호선 /
승·하차 입력

서울역 / 1호선 / 15 / 평일 / 승차

2

해당 호선 / 해당 역 / 해당 시간 /
평일·주말 / 해당 호선의
승·하차 인원 수 예측

1호선 서울역 15시 평일 승차
이용객 수는 00명 입니다



주제 데이터 EDA 전처리 모델 결과 결론

2 데이터

http://data.seoul.go.kr/dataList/OA-12252/S/1/datasetView.do

데이터 정보

공개일자	2015.02.17.	최신수정일자	2023.02.05.
갱신주기	매월 5일	분류	교통
원본시스템	교통카드 정산시스템	저작권자	서울특별시
제공기관	서울특별시	제공부서	도시교통실 교통기획관 교통정책과
담당자	조하람 (02-2133-2237)		
원본형태	DB	제3저작권자	없음
라이선스	저작권자표시(BY) : 이용이나 변경 및 2차적 저작물의 작성을 포함한 자유이용을 허락합니다.		
관련태그	교통, 통계, 교통카드, 지하철역, 승차, 하차, 인원		

- 서울시 열린데이터 광장 공공데이터 활용
- 서울시 지하철 호선별 역별 시간대별 승/하차 인원
- 2022.01.01~2022.12.31 간의 데이터 활용

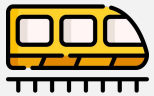
  - 2022년 1년 간의 데이터

2 데이터

	연번	수송일자	호선	고유역번호(외부역코드)	역명	승하차구분	06시이전	06-07시간대	07-08시간대	08-09시간대	...	15-16시간대	16-17시간대	17-18시간대	18-19시간대	19-20시간대	20-21시간대	21-22시간대	22-23시간대	23-24시간대	24시이후
0	1	2022-01-01	1	150	서울역	승차	120	137	211	439	...	1566	1686	1591	1358	1062	899	1327	814	234	NaN
1	2	2022-01-01	1	150	서울역	하차	113	560	617	910	...	1329	1251	1126	884	764	654	728	416	131	NaN
2	3	2022-01-01	1	151	시청	승차	38	66	101	139	...	474	550	672	528	420	434	491	232	38	NaN
3	4	2022-01-01	1	151	시청	하차	31	195	224	380	...	408	377	354	213	131	98	137	61	24	NaN
4	5	2022-01-01	1	152	종각	승차	44	71	86	158	...	889	964	1024	803	855	1099	1209	255	62	NaN

- 총 25개의 칼럼, 서울시 내의 총 583개의 지하철 역에 대한 데이터
- 수송일자, 호선, 고유역번호(외부역코드), 승하차구분, 시간
- 같은 역명 - 다른 호선이면 고유역번호가 상이함

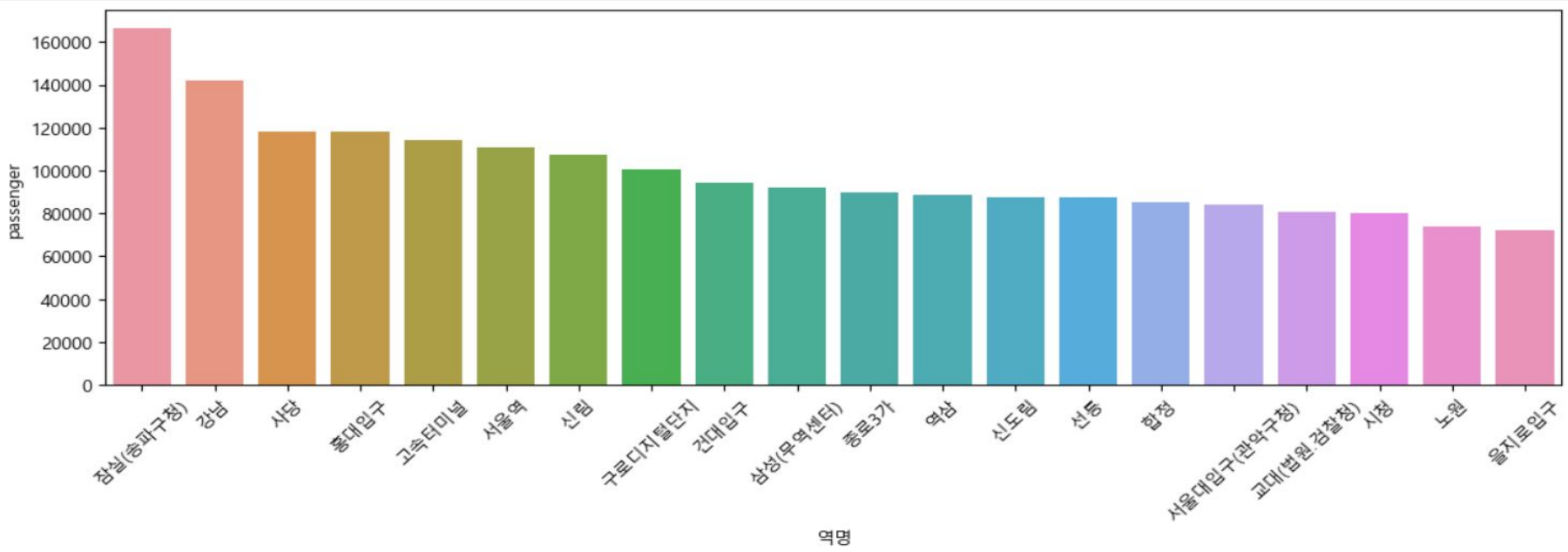
  예) 서울역 1호선 고유역번호(150) / 서울역 4호선 고유역번호(426)



주제 데이터 EDA 전처리 모델 결과 결론

3 EDA

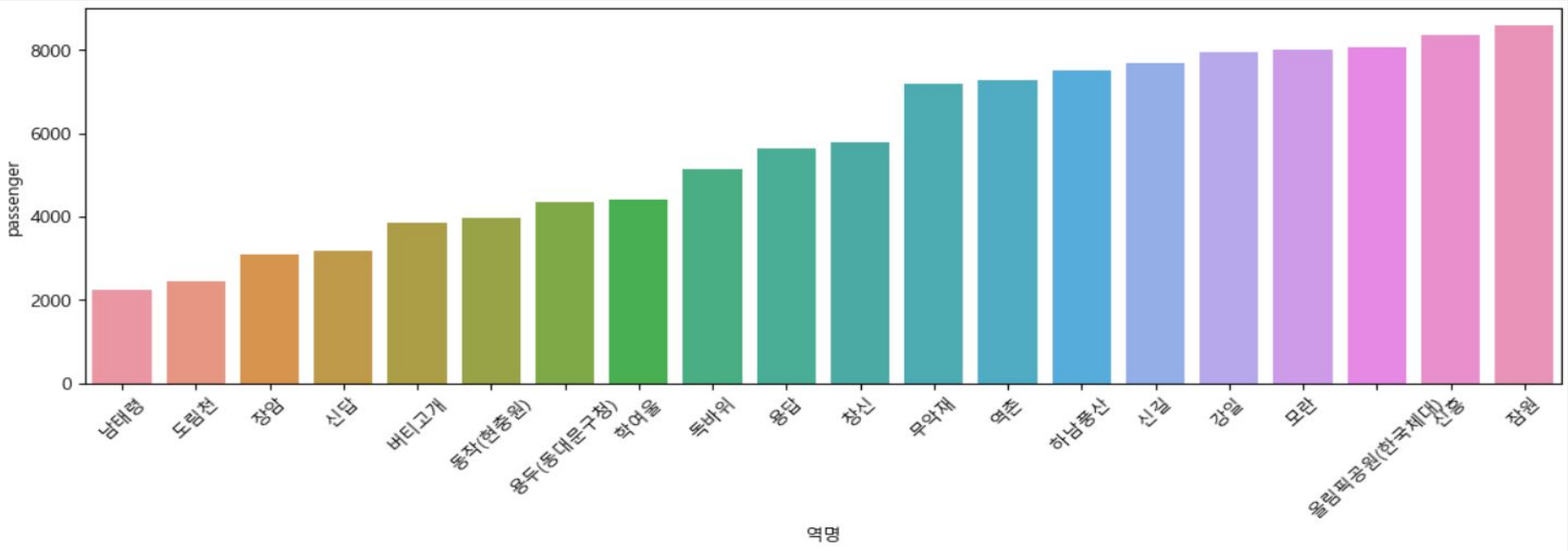
- 하루 평균 승/하차 인원 합 상위 20개 역



- 2022년 하루 평균 역별 지하철 승/하차 인원은 '잠실(송파구청)'역이 166,540명으로 가장 많음
- 표에 해당하는 상위 7개 역인 잠실역(166,540명),강남역(141,769명),사당역(118,390명),홍대입구역(118,222명), 고속터미널역(114,357명),서울역(110,822명),신림(107,236명) 에서 신림역을 제외한 모든 역들은 2개 이상의 역들이 교차하는 환승역으로 서울의 주요 이동 허브이다.

3 EDA

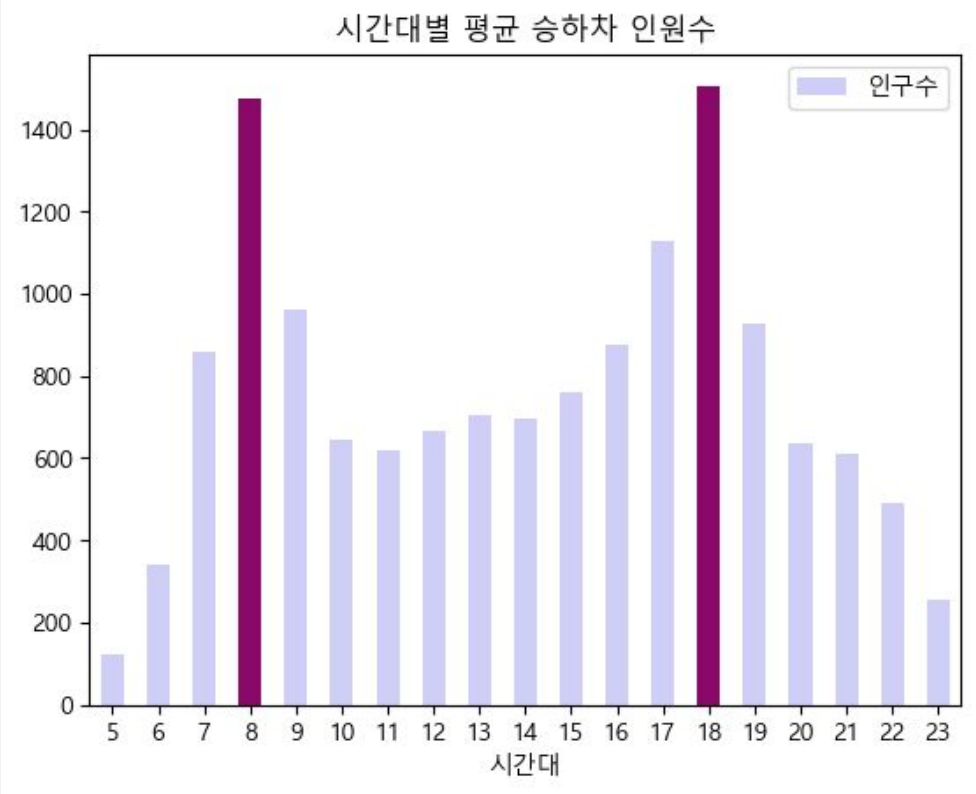
- 하루 평균 승/하차 인원 합 하위 20개 역



- 2022년 하루 평균 역별 지하철 승/하차 인원은 '남태령'역이 2,236명으로 가장 적음
- 표에 해당하는 하위 7개 역인 남태령역(2,236명),도림천역(2,451명),장암역(3,103명),신답역(3,172명), 버티고개역(3,844명),동작(현충원)역(3,981명),용두(동대문구청)역(4,337명) 에서 동작(현충원)역을 제외한 모든 역이 환승역이 아님.

3 EDA

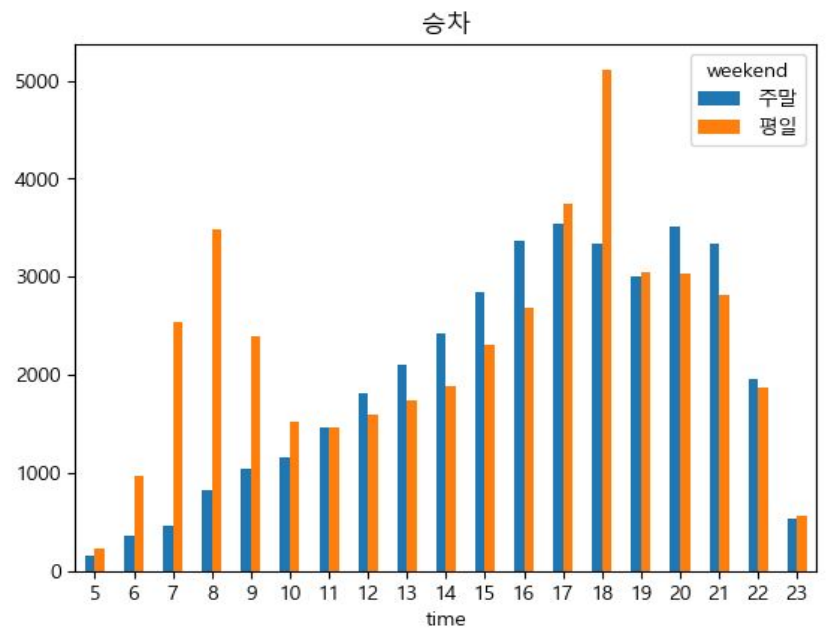
- 2022년 시간대별 평균 승/하차 인원 수

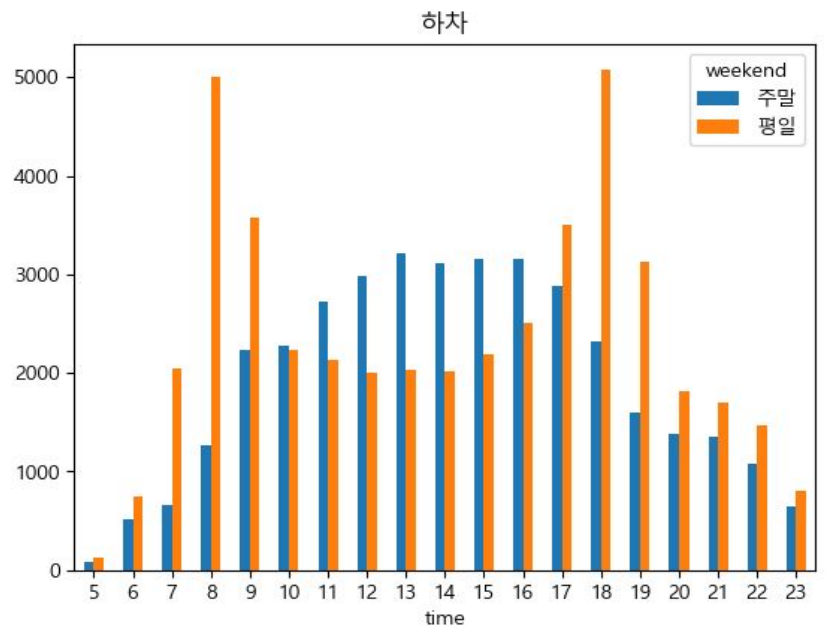


- 2022년 1년간의 평균 승/하차 인원 수를 시간대별로 평균을 낸 차트
- 차트를 보면 출/퇴근 시간대인 8시와 18시에 평균 승/하차 인원 수가 많은 것을 확인할 수 있음
- 첫차시간 5시의 이용객보다 막차시간 23시의 이용객이 상대적으로 많음

3 EDA

- 잠실역기준 승/하차 구분 별 주말/평일 시간대별 평균 이용객 수

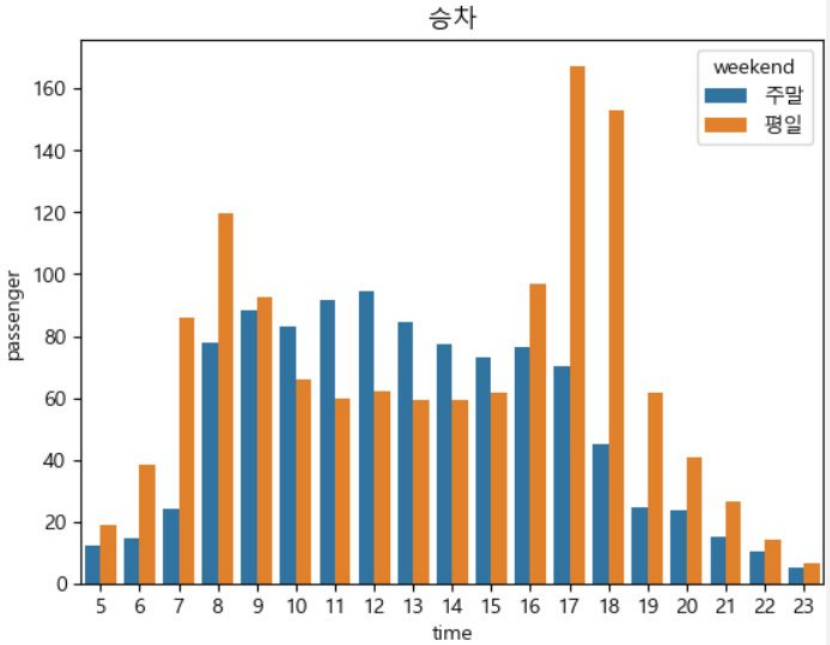


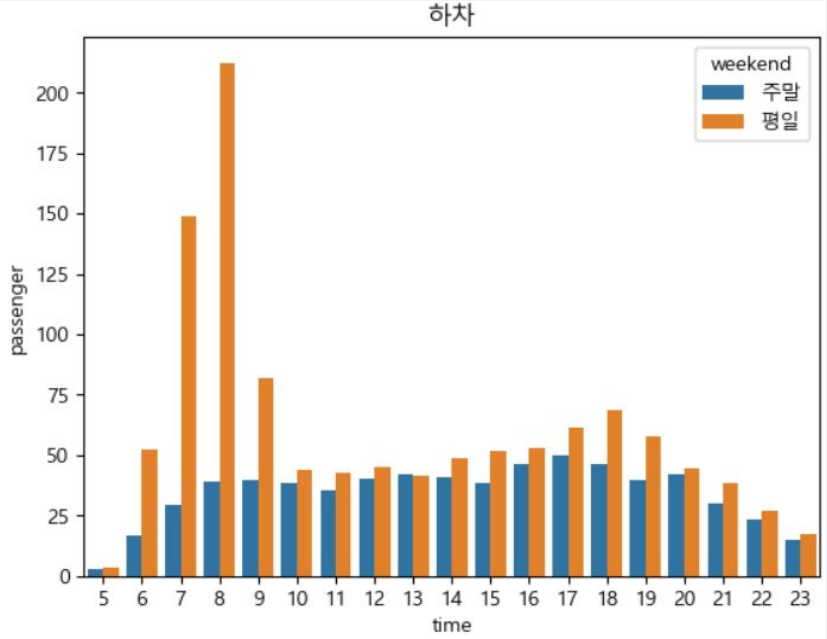


- 잠실역을 기준으로 승차객수와 하차객수를 구분해서 주말/평일 시간대별 평균 이용객수를 나타낸 차트
- 평일기준 출/퇴근시간에 승차인원과 하차인원이 모두 많은 것을 알 수 있음
- 주말 승차시에는 저녁시간대까지(5시-17시) 승차 인원 차트가 상승곡선을 이루다가 저녁시간(16시-21시)에 대체로 높은 수준을 보임(저녁에 승차하는 인원이 많음)
- 주말 하차시에는 낮시간대(11시-17시)에 주로 높은 수준을 보임(낮시간에 하차하는 인원이 많음)
- 따라서 주말에 잠실역을 이용하는 이용객은 주로 낮에 하차 후 저녁에 승차하는 경우가 많음

3 EDA

- 남태령역기준 승/하차 구분 별 주말/평일 시간대별 평균 이용객 수





- 남태령역을 기준으로 승차객수와 하차객수를 구분해서 주말/평일 시간대별 평균 이용객수를 나타낸 차트
- 평일기준 남태령역의 퇴근시간(17시-19시)에서 승차 객수가 많음
- 평일기준 남태령역의 출근시간(7시-9시)에서 하차 객수가 높은 수준을 보임
- 출근시간 하차인원 / 퇴근시간 승차인원이 비교적 많은 것으로 보아 남태령역은 출퇴근지역임을 알 수 있음
- 주말은 승/하차 모두 낮시간에 이용객이 고르게 분포해 있음


주제
데이터
EDA
전처리
모델
결과
결론

4 전처리

- 데이터에 대한 이해

	연번	수송일자	호선	고유역번호(외부역코드)	역명	승하차구분	06시이전	06-07시간대	07-08시간대	08-09시간대	...	15-16시간대	16-17시간대	17-18시간대	18-19시간대	19-20시간대	20-21시간대	21-22시간대	22-23시간대	23-24시간대	24시이후
0	1	2022-01-01	1	150	서울역	승차	120	137	211	439	...	1566	1686	1591	1358	1062	899	1327	814	234	NaN
1	2	2022-01-01	1	150	서울역	하차	113	560	617	910	...	1329	1251	1126	884	764	654	728	416	131	NaN
2	3	2022-01-01	1	151	시청	승차	38	66	101	139	...	474	550	672	528	420	434	491	232	38	NaN
3	4	2022-01-01	1	151	시청	하차	31	195	224	380	...	408	377	354	213	131	98	137	61	24	NaN
4	5	2022-01-01	1	152	종각	승차	44	71	86	158	...	889	964	1024	803	855	1099	1209	255	62	NaN

- ‘연번’ 칼럼 삭제 - index값과 동일한 의미
- ‘수송일자’ 칼럼 데이터 타입 변경(pd.to_datetime)
- NaN 데이터 처리 필요

4 전처리

- 결측치(NaN값) 제거

	연번	수송일자	호선	고유역번호(외부역코드)	역명	승하차구분	06시이전	06-07시간대	07-08시간대	08-09시간대	...	15-16시간대	16-17시간대	17-18시간대	18-19시간대	19-20시간대	20-21시간대	21-22시간대	22-23시간대	23-24시간대	24시이후
0	1	2022-01-01	1	150	서울역	승차	120	137	211	439	...	1566	1686	1591	1358	1062	899	1327	814	234	NaN
1	2	2022-01-01	1	150	서울역	하차	113	560	617	910	...	1329	1251	1126	884	764	654	728	416	131	NaN
2	3	2022-01-01	1	151	시청	승차	38	66	101	139	...	474	550	672	528	420	434	491	232	38	NaN
3	4	2022-01-01	1	151	시청	하차	31	195	224	380	...	408	377	354	213	131	98	137	61	24	NaN
4	5	2022-01-01	1	152	종각	승차	44	71	86	158	...	889	964	1024	803	855	1099	1209	255	62	NaN

- '24시이후' 칼럼은 모든 값이 NaN 으로 칼럼을 삭제

4 전처리

- 결측치 제거

```
matro_df_raw[matro_df_raw['고유역번호(외부역코드)']==2758]
```

✓ 0.1s Python

	수송일자	호선	고유역번호(외부역코드)	역명	승하차구분	06시이전	06-07시간대	07-08시간대	08-09시간대	09-10시간대	...	14-15시간대	15-16시간대	16-17시간대	17-18시간대	18-19시간대	19-20시간대	20-21시간대	21-22시간대	22-23시간대	23-24시간대
123200	2022-08-14	7	2758	상동	승차	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
123201	2022-08-14	7	2758	상동	하차	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

2 rows × 24 columns

```
metro_df[metro_df['code']==2758]
```

✓ 0.2s Python

	date	code	station	in_out	weekend	time	passenger
2340800	2022-08-14	2758	상동	승차	주말	5	0
2340801	2022-08-14	2758	상동	승차	주말	6	0
2340802	2022-08-14	2758	상동	승차	주말	7	0
2340803	2022-08-14	2758	상동	승차	주말	8	0

- 이외에도 데이터의 값이 전부 0값인 행들(결측치)이 존재
- 해당 데이터는 출근시간 '08-09시간대'과 퇴근시간 '18-19시간대'의 값이 0인 행을 추출하여 NaN값으로 변환 후 해당 행을 삭제
  (모든 값이 0인 데이터(결측값)와 실제 데이터 값이 0인 데이터를 구분하기 위해 이와 같은 방법을 활용)
- 결측치 제거 후 총 583개의 역 데이터 ▶ 545개의 역 데이터 (총 38개의 역의 데이터가 결측치)

4 전처리

- 평일/주말을 데이터 추가

```
seoul_station_line1['요일']=seoul_station_line1['수송일자'].dt.dayofweek
seoul_station_line1.loc[seoul_station_line1['요일']>=5, 'weekend' ]='주말'
seoul_station_line1.loc[(seoul_station_line1['요일']<=4) & (seoul_station_line1['요일']>=0),'weekend' ]='평일'
```

	수송일자	호선	고유역번호(외부역코드)	역명	승하차구분	06시이전	06-07시간대	07-08시간대	08-09시간대	09-10시간대	...	16-17시간대	17-18시간대	18-19시간대	19-20시간대	20-21시간대	21-22시간대	22-23시간대	23-24시간대	요일	weekend
0	2022-12-01	1	150	서울역	승차	302	412	1567	2339	2051	...	3707	5962	8807	4035	2725	2759	1705	845	3	평일
1	2022-12-01	1	150	서울역	하차	254	1472	4108	9435	6001	...	2462	2955	3134	1831	1311	1182	821	408	3	평일
2	2022-12-01	1	151	시청	승차	89	109	285	309	365	...	1845	3318	6765	2402	2181	2200	1429	434	3	평일

- ‘수송일자’ 칼럼을 활용하여 ‘요일’ 칼럼 추가
- ‘요일’ 칼럼을 바탕으로 평일/주말을 구분하는 ‘weekend’ 칼럼 추가

4 전처리

- 시간 칼럼명 변경

	수송일자	호선	고유역번호(외부역코드)	역명	승하차구분	5	6	7	8	9	...	16	17	18	19	20	21	22	23	요일	weekend
0	2022-12-01	1	150	서울역	승차	302	412	1567	2339	2051	...	3707	5962	8807	4035	2725	2759	1705	845	3	평일
1	2022-12-01	1	150	서울역	하차	254	1472	4108	9435	6001	...	2462	2955	3134	1831	1311	1182	821	408	3	평일
2	2022-12-01	1	151	시청	승차	89	109	285	309	365	...	1845	3318	6765	2402	2181	2200	1429	434	3	평일

- 1시간 단위의 '시간'칼럼을 시작 시간으로 변경

- 주요 칼럼 추출

```
matro_pd_last=matro_pd[['수송일자','고유역번호(외부역코드)','역명','승하차구분','weekend']]
```

	수송일자	고유역번호(외부역코드)	역명	승하차구분	weekend
0	2022-01-01	150	서울역	승차	주말
1	2022-01-01	150	서울역	하차	주말
2	2022-01-01	151	시청	승차	주말
3	2022-01-01	151	시청	하차	주말

- 데이터를 이용하여 머신러닝을 하기 위해서는 시간칼럼이 추가되어야한다고 생각(시간을 피처로 활용하기 위해)
- 이를 위해 시간 칼럼 외의 데이터(수송일자,고유역번호,역명,승하차구분,weekend)를 따로 추출

4 전처리

- 시간 칼럼 추가

	수송일자	고유역번호(외부역코드)	역명	승하차구분	weekend
0	2022-01-01	150	서울역	승차	주말
1	2022-01-01	150	서울역	하차	주말

▶

	수송일자	고유역번호(외부역코드)	역명	승하차구분	weekend	time
0	2022-01-01	150	서울역	승차	주말	5
0	2022-01-01	150	서울역	승차	주말	6
0	2022-01-01	150	서울역	승차	주말	7
0	2022-01-01	150	서울역	승차	주말	8
0	2022-01-01	150	서울역	승차	주말	9

- 칼럼명으로 나뉘어있던 시간 데이터를 칼럼으로 추가
- 각 역별로 승/하차 2행으로 구분되던 데이터가 승차/하차 별로 19개씩 추가(5시~23시)

4 전처리

- 인원 칼럼 추가

5	6	7	8	9	...	16	17	18	19	20	21	22	23
120	137	211	439	592	...	1686	1591	1358	1062	899	1327	814	234
113	560	617	910	1232	...	1251	1126	884	764	654	728	416	131
97	137	206	507	683	...	2029	2316	2131	2226	1803	2014	1593	472
53	343	489	791	1013	...	1744	1624	1540	1296	1127	928	442	170
312	433	1782	2545	1708	...	2533	4364	7854	3031	2163	2277	813	248

- 시간 별로 행으로 묶인 이용객 데이터를 transpose()를 활용하여 세로로 변경

	0	1	2	3
5	120	113	97	53
6	137	560	137	343
7	211	617	206	489
8	439	910	507	791
9	592	1232	683	1013
10	878	1260	1045	1048
11	1278	1028	1592	1187
12	1508	1555	1883	1537
13	1502	1264	2104	1448
14	1323	1265	1439	1642
15	1566	1329	2308	1724
16	1686	1251	2029	1744
17	1591	1126	2316	1624
18	1358	884	2131	1540
19	1062	764	2226	1296
20	899	654	1803	1127
21	1327	728	2014	928
22	814	416	1593	442
23	234	131	472	170

4 전처리

- 인원 칼럼 추가

	수송일자	역명	호선	고유역번호(외부역코드)	승하차구분	weekend	time	passenger
0	2022-01-01	서울역	1.0	150	승차	주말	5	120.0
1	2022-01-01	서울역	1.0	150	승차	주말	6	137.0
2	2022-01-01	서울역	1.0	150	승차	주말	7	211.0
3	2022-01-01	서울역	1.0	150	승차	주말	8	439.0
4	2022-01-01	서울역	1.0	150	승차	주말	9	592.0
...	...	...	...	...	...	...	...	...
3772635	2022-12-31	남위례	8.0	2828	하차	주말	19	240.0
3772636	2022-12-31	남위례	8.0	2828	하차	주말	20	175.0
3772637	2022-12-31	남위례	8.0	2828	하차	주말	21	196.0
3772638	2022-12-31	남위례	8.0	2828	하차	주말	22	178.0
3772639	2022-12-31	남위례	8.0	2828	하차	주말	23	118.0

- ‘passenger’ 칼럼 추가

	0	1	2	3
5	120	113	97	53
6	137	560	137	343
7	211	617	206	489
8	439	910	507	791
9	592	1232	683	1013
10	878	1260	1045	1048
11	1278	1028	1592	1187
12	1508	1555	1883	1537
13	1502	1264	2104	1448
14	1323	1265	1439	1642
15	1566	1329	2308	1724
16	1686	1251	2029	1744
17	1591	1126	2316	1624
18	1358	884	2131	1540
19	1062	764	2226	1296
20	899	654	1803	1127
21	1327	728	2014	928
22	814	416	1593	442
23	234	131	472	170

4 전처리

- 고유역번호(code) 확인을 위한 딕셔너리 만들기

```
code_df=matro_df_raw[['역명','호선','고유역번호(외부역코드)']]
code_df.drop_duplicates(inplace=True)
```

	역명	호선	고유역번호(외부역코드)
0	서울역	1	150
2	시청	1	151
4	종각	1	152
6	종로3가	1	153
8	종로5가	1	154

▶

```
{('서울역', 1): '150',
 ('시청', 1): '151',
 ('종각', 1): '152',
 ('종로3가', 1): '153',
 ('종로5가', 1): '154',
 ('동대문', 1): '155',
 ('신설동', 1): '156',
 ('제기동', 1): '157',
 ('청량리(서울시립대입구)', 1): '158',
 ('동묘앞', 1): '159',
```

- '역명', '호선', '고유역번호(code)' 칼럼을 가져옴
- 딕셔너리는 {key:value}로 구성(키 값은 중복 X)
- 우리가 필요한 정보는 '역명', '호선'을 알때, 해당역의 고유역번호(code)
- 따라서 (역명, 호선) 튜플데이터를 key값으로, 이때의 고유역번호(code)를 value값으로 구성

4 전처리

- 인코딩(encoder)

	date	station	line	code	in_out	weekend	time	passenger
0	2022-01-01	서울역	1.0	150	승차	주말	5	120.0
1	2022-01-01	서울역	1.0	150	승차	주말	6	137.0
2	2022-01-01	서울역	1.0	150	승차	주말	7	211.0

  - 본 연구의 데이터에는 다수의 범주형 데이터 존재
    'code(고유역번호)', 'in_out(승/하차구분)', 'weekend(평일/주말구분)', 'time(이용시간)'
  - 이러한 데이터들을 분석모델에 활용하기 위해서는 인코딩이 필수
  - 인코딩 방법으로는 OneHotEncoder를 활용

4 전처리

- 원핫인코더(OneHotEncoder)

  - 데이터의 특성이 '크다', '중간', '작다'와 같이 수치적으로 차이가 있는 데이터가 아닐 때,
    수치적 연관이 없는 데이터를 벡터형으로 인코딩하는 방법

  - 'code(고유역번호)', 'in_out(승/하차구분)', 'weekend(평일/주말구분), time(이용시간)'

승/하차 구분
승차
승차
하차

▶

승/하차구분	승차	하차
승차	1	0
승차	1	0
하차	0	1


주제
데이터
EDA
전처리
모델
결과
결론

5 모델

- 초기 모델링

  - 데이터 양이 많다 보니, 임의로 12월의 데이터를 따로 분리하여 초기 모델링을 시도
  - code(고유역번호), in_out(승/하차 구분), weekend(평일/주말) 데이터는 OneHotEncoder를 적용 후 모델링

- 변수 선택 (Feature Selection)

  - 초기 모델링이기 때문에 모든 Feature를 넣고 모델링을 진행
    x = code(고유역번호), in_out(승/하차 구분), weekend(평일/주말), time(시간)
    y = passenger(이용객수)

- Training / Test Data 나누기

  - Training 데이터와 Test 데이터는 0.8, 0.2의 비율로 나눔

5 모델

- **회귀문제를 풀기 위해 4종류의 회귀모델을 활용**

  - 선형회귀분석, 의사결정나무, 랜덤포레스트, 서포트벡터회귀 총 4개의 회귀모델을 활용하여 모델링 시도

- **선형회귀모델**

  - 선형회귀분석(Linear regression analysis)은 종속변수에 유의한 영향을 미칠 것으로 기대되는 독립변수를 고려하여 종속변수를 통계적 방법에 의해 추정하는 방법
  - 선형회귀모형은 선형회귀분석을 통해 종속변수와 독립변수간의 관계를 선형식으로 나타낸 모형이다(Kim, Hong, et al., 2015; Choi, Kim, Kim, et al., 2017).

5 모델

- 의사결정나무
  - 의사결정나무(Decision tree)는 변수들의 규칙 혹은 조건문을 토대로 자료를 나무(tree) 형태의 그래프로 표현하며, 분리규칙에 의해 서로 유사한 데이터들로 세분화하고 최종 분류 기준을 만족할 때까지 분류하여 전체 나무를 구성한다.(Breiman et al., 1984; Bae, 2014)

- 랜덤포레스트
  - 랜덤포레스트(Random forest)는 말 그대로 (의사결정)나무들이 많이 있는 모형으로, 의사결정나무 모형을 다수 만들어 예측력을 높이는 방법이다.(Yoo, 2015).
  - 랜덤포레스트는 하나의 데이터 집합에서 여러개의 훈련용 데이터를 만들어 여러개의 의사결정 나무를 생성하여 이들을 결합한 결과를 통해 예측력을 향상시키는 방법

5 모델

- 서포트 벡터 머신(Support Vector Machine, SVM)
  - 서포트 벡터 머신(Support Vector Machine, SVM)은 Vapnik(1995)에 의해 제시된 이후 문서분류, 고객분류 등 여러가지 문제에서 분류결과가 매우 정확하고, 다양한 형태의 자료에 적용이 가능한 방법으로 알려져있다(Choi et al., 2013).
  - SVM은 선형으로 서로 다른 클래스의 벡터들을 그 사이의 거리에 대한 최대의 마진으로 분류할 수 있는 서포트 벡터들로 이루어진 초평면을 찾는 방법이다(Lee, Chung, et al., 2016)

- 서포트 벡터 회귀(Support Vector Regression, SVR)
  - SVM은 주로 분류(Classification)문제의 예측에 사용되며, SVM에 ε-무감도 손실함수(ε-insensitive loss function)를 도입하여 회귀 분석을 할 수 있도록 확장한 방법을 Support Vector Regression(SVR)이라 한다.(Kim et al.,,2012).
  - SVM은 분류문제에서 사용되지만, 이를 통해 임의의 실수값을 예측할 수 있도록 SVM을 일반화한 방법이 SVR이다.(C.K. Park, 2006).
  - 여러가지 커널 함수를 활용하여 회귀문제에 SVM을 적용

- Scikit-learn 라이브러리 활용
  - 본 프로젝트에서는 Scikit-learn 라이브러리를 중심으로 프로젝트를 진행하였음

5 모델

- 회귀분석의 성능 평가 지표
  - 회귀분석에서는 예측값과 실제값의 차이(오차)에 기반하여 성능을 평가
- MAE(Mean Absolute Error)
  - 실제 값과 예측 값의 차이를 절대값으로 변환한 값의 평균값
- MSE(Mean Squared Error)
  - 실제 값과 예측 값의 차이를 제곱한 값의 평균 값
- RMSE(Root Mean Squared Error) ★
  - MSE는 오차를 제곱한 값의 평균이므로, 실제 오차보다 큰 값
  - 따라서 RMSE는 MSE에 루트를 씌운 값
- RMSLE(Root Mean Squared Log Error) ★
  - RMSE에 로그를 적용해준 지표

5 모델

- 회귀분석의 성능 평가 지표

- RMSLE(Root Mean Squared Log Error)의 장점

  - 아웃라이어에 강함

    아웃라이어가 있더라도 변동폭이 크지 않음

  - 상대적 Error를 측정 가능 ★

    예측값과 실제값에 로그를 취해줌으로 상대적 비율을 구할 수 있음

    예) 예측값=100, 실제값=90일 때, RMSLE=0.1053, RMSE=10

    예측값 = 10000, 실제값=9000일 때, RMSLE=0.1053, RMSE=1000

  - 서로 다른 역에 대한 모델의 상대적 Error을 측정하는 지표로 RMSLE를 활용

5 모델

time	code_150	code_151	code_152	code_153	code_154	...	code_2824	code_2825	code_2826	code_2827	code_2828	in_out_승차	in_out_하차	weekend_주말	weekend_평일
5	1	0	0	0	0	...	0	0	0	0	0	1	0	0	1
6	1	0	0	0	0	...	0	0	0	0	0	1	0	0	1
7	1	0	0	0	0	...	0	0	0	0	0	1	0	0	1
8	1	0	0	0	0	...	0	0	0	0	0	1	0	0	1
9	1	0	0	0	0	...	0	0	0	0	0	1	0	0	1

- 첫 모델링 시도
  - 모든 피처를 모델링에 활용
  - 위의 데이터는 데이터를 인코딩한 후 x 데이터를 나타냄
  - code(고유역번호), in_out(승/하차구분), weekend(평일/주말)이 인코딩 된 것을 확인할 수 있음
  - 시간데이터는 인코딩 하지 않음

5 모델

- 선형회귀분석
  - RMSE = 826.676

```
826.6756944500596
```

- 의사결정나무
  - RMSE = 177.277

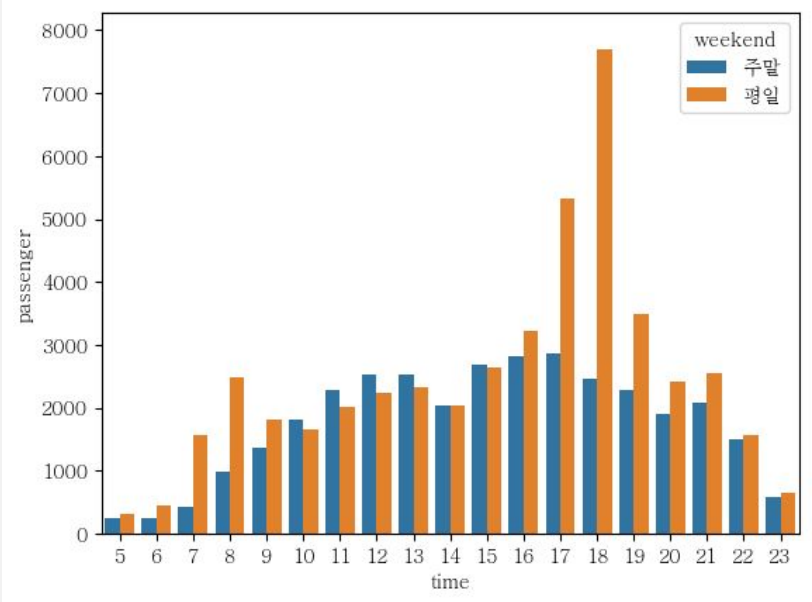
```
177.2768852923641
```

- 결과
  - 랜덤포레스트 / SVR의 경우 모델링이 제대로 되지 않았음
  - 선형회귀분석의 RMSE값은 826으로 높고, 의사결정나무는 177으로 낮은 값을 기록
  - 두 모델의 성능 간극이 매우 큼
  - 선형회귀분석의 성능이 매우 낮게 모델링되어 이를 향상시킬 방법에 대해 고민

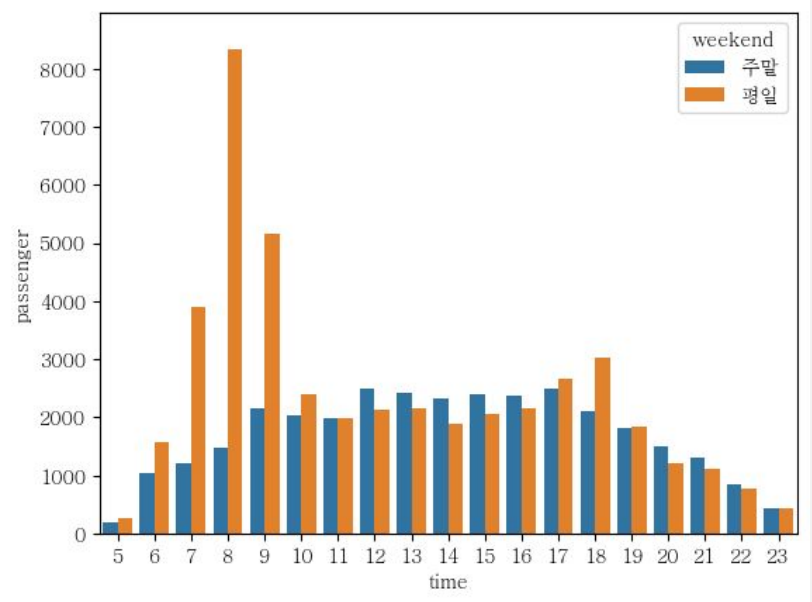
5 모델

- 서울역에 시간 별 평균 승/하차 인원 수

서울역 시간 별 평균 승차 인원 수



서울역 시간 별 평균 하차 인원 수

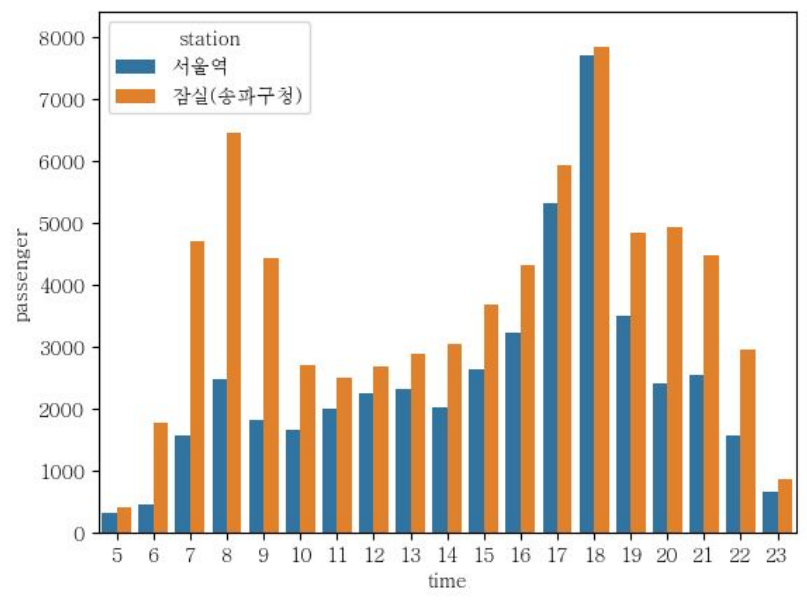


- 서울역의 평균 승차 하차 인원 수 그래프를 통해 상차/하차, 평일/주말에 따라 이용객 패턴이 다른 것을 확인
- 서울역은 서울의 대표적인 출/퇴근지역으로 출근시간(7시-9시)에는 하차인원이 눈에 띄게 많고, 퇴근시간(17시~19시)에는 승차 인원이 많았음
- 앞에서 살펴봤던 잠실역과는 확연히 다른 패턴이라고 생각하여 두 역을 비교해보기로 함

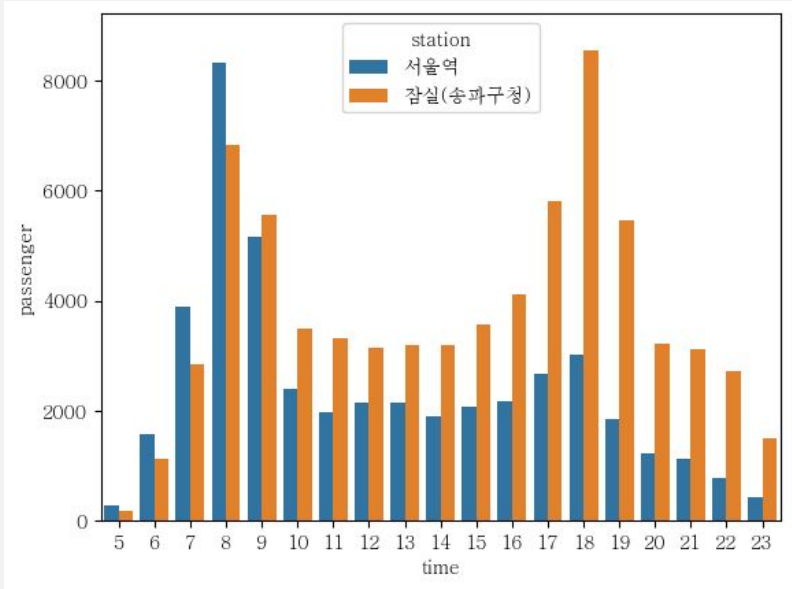
5 모델

- 서울역과 잠실역에 시간 별 평일 평균 승/하차 인원 수

서울역과 잠실역의 시간 별 평균 승차 인원 수



서울역과 잠실역 시간 별 평균 하차 인원 수



- 두 역의 이용객의 패턴의 차이가 확연히 드러나는 평일을 기준으로 두 역의 시간 별 평균 승/하차 인원수를 비교
- 출/퇴근시간 모두 승/하차 인원이 많은 잠실역과는 달리 서울역은 출근시간에만 하차인원이, 퇴근시간에만 승차인원이 많음
- 역 별로 데이터를 나누어 회귀 모델을 만드는 것이 모델을 더 잘 예측할 수 있을 것이라 생각하게됨

5 모델

- **지하철 역별 개별 모델링**
  - 잠실(송파구청)역과 서울역의 지하철 이용객의 평균 추이가 서로 달랐던 것 처럼 역마다 서로 다른 패턴이 존재할 것이라 생각
  - 따라서 'code(고유역번호)'를 기준으로 역별로 회귀모델을 개별로 모델링 기획

- **'time(시간)'피처의 인코딩**
  - 시간이 회귀모델에서 우리가 생각하는 시간의 역할을 하지못한다는 것을 알게됨
    ※ 5시와 23시는 수치적인 차이가 있기때문에 이대로 사용하면 가중치가 다르게 적용됨
  - 따라서 'time(시간)'을 어떻게 인코딩할지 고민
    1. 시간을 출근시간 전 / 출근시간 / 출근시간 후 / 퇴근시간 / 퇴근시간 후 등의 범위를 지정하여 인코딩
    2. 시간을 시간 그대로 OneHotEncoder를 적용하여 인코딩

5 모델

- 실험1. 서울역에 대한 'time(시간)'을 범위로 나누어 인코딩한 후 모델링
  - 서울역 1호선의 1년치 데이터에서 'time(시간)' 피처를 출근 시간 전(5시-7시) / 출근 시간(7시-10시) / 출근 시간 후(10시-17시) / 퇴근 시간(17시-20시) / 퇴근 시간 후(20시-24시) 로 분류하여 모델링

승하차구분_0	승하차구분_1	weekend_0	weekend_1	time_출근시간	time_출근시간전	time_출근시간후	time_퇴근시간	time_퇴근시간후
0	1	0	1	0	1	0	0	0
0	1	0	1	0	1	0	0	0
0	1	0	1	1	0	0	0	0
0	1	0	1	1	0	0	0	0
0	1	0	1	1	0	0	0	0

5 모델

- 실험2. 서울역에 대한 'time(시간)'을 OneHotEncoder로 인코딩한 후 모델링
  - 앞의 모델과 비교를 위해 서울역 1호선의 1년치 데이터의 시간 피처를 OneHotEncoder로 인코딩한 후 모델링

	in_out_승차	in_out_하차	weekend_주말	weekend_평일	time_5	time_6	time_7	time_8	time_9	time_10	...	time_14	time_15	time_16	time_17	time_18	time_19	time_20	time_21	time_22	time_23
0	1	0	1	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	1	0	1	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	1	0	1	0	0	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
4	1	0	1	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3762337	0	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
3762338	0	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
3762339	0	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
3762340	0	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
3762341	0	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1

5 모델

- 서울역에 대한 시간 인코딩 전/후 예측 모델 비교 상승 하락

- **선형회귀분석**
  - RMSE =1020.228
  - RMSE = 1266.678

- **의사결정나무**
  - RMSE = 911.287
  - RMSE = 617.132

- **랜덤포레스트**
  - RMSE=911.101
  - RMSE =617.043

- 선형회귀분석의 RMSE 값이 시간을 인코딩 하기 전보다 높아졌음
- 의사결정나무, 랜덤포레스트의 RMSE 값은 시간을 분류하여 인코딩한 것보다 낮아짐
- 전체적으로 RMSE값이 시간을 분류하여 인코딩한것보다 낮아짐 ▶ 모델의 성능이 향상됨

5 모델

- SVR 커널 함수 선택
  - Scikit-learn 라이브러리의 SVR의 모델의 경우, 총 3개의 커널 함수를 제공('RBF', 'Polynomial', 'sigmoid')
  - 커널함수를 선택하기 위해 총 10개의 랜덤 역을 샘플링하여 실험을 진행
    np.random.choice를 사용
  - 커널 함수 별 성능 평가는 RMSE값을 기준으로 전체 샘플 10개의 데이터 중 가장 낮은 RMSE 값을 가지는 함수를 선택

	time	in_out	weekend	code	station_name	line
0	9	하차	평일	2649	신내	6
1	7	승차	평일	2719	태릉입구	7
2	11	승차	주말	2729	건대입구	7
3	14	승차	평일	2628	효창공원앞	6
4	5	하차	주말	336	학여울	3
5	21	승차	평일	414	수유(강북구청)	4
6	10	하차	평일	411	노원	4
7	11	하차	주말	2543	답십리	5
8	17	승차	평일	2817	송파	8
9	6	승차	평일	2738	이수	7

5 모델

- SVR 커널 함수 선택

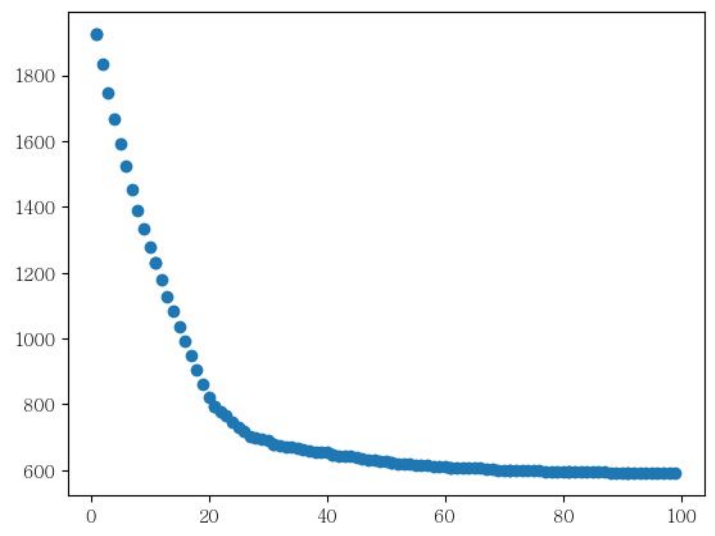
	0	1	2	3	4	5	6	7	8	9	average
RBF	40.800771	111.217696	199.386185	93.951723	192.575416	534.497184	237.940294	110.127534	124.27028	204.997365	184.712368
Polynomial	40.887167	111.07021	196.992454	93.975037	191.327996	401.122561	222.494759	108.756598	122.113078	205.45074	170.569290
Sigmoid	40.771143	111.361577	200.407943	94.034532	192.893898	584.604297	252.635459	110.796981	125.763883	204.923082	190.933126
0	Sigmoid	Polynomial	Polynomial	RBF	Polynomial	Polynomial	Polynomial	Polynomial	Polynomial	Sigmoid	NaN

- 전체 10개의 샘플데이터의 RMSE의 값을 나타낸 데이터
- 총 10개의 데이터 중 6개의 데이터에서 ‘Polynomial’함수가 가장 낮은 값은 RMSE값을 가짐
- 전체 샘플데이터의 RMSE 평균값도 ‘Polynomial’함수가 170.57로 가장 낮은 값을 가짐
- 따라서 SVR모델의 커널함수는 ‘Polynomial’함수로 선정함

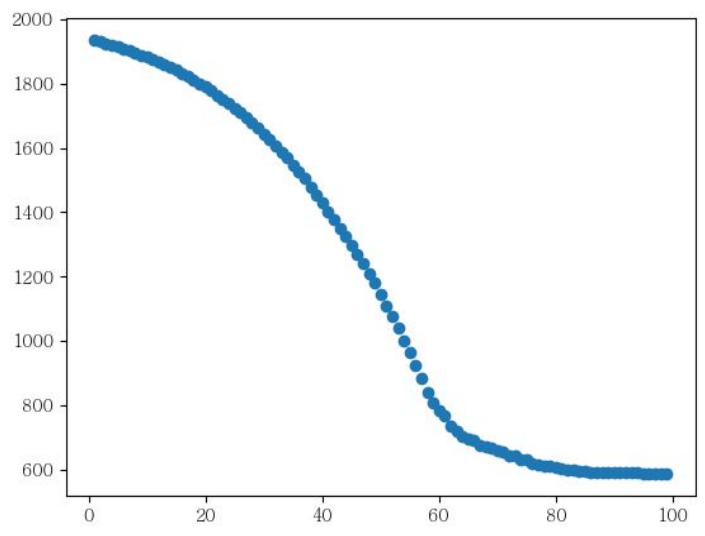
5 모델

- SVR모델 C값 / Degree값 설정

C(오차허용율)



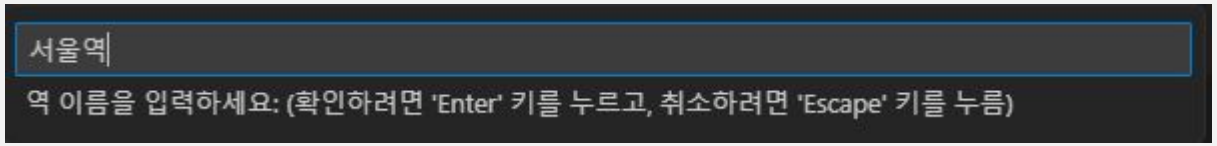
Degree(차원)



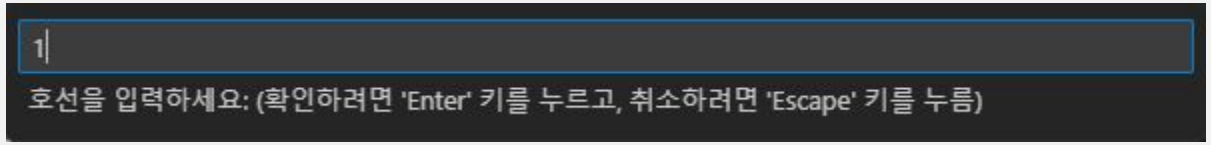
- 선정한 커널함수인 'Polynomial'함수로 모델링을 할 때, C값과 Degree값을 어떻게 지정할지 고려하기 위해 C값/Degree에 따른 모델의 RMSE값의 변화를 scatter 차트로 그림
- C값의 경우 40 이후에서 대체로 일정한 값으로 수렴한다고 생각하여 C=40으로 선정
- Degree값의 경우 80 이후에서 대체로 일정한 값으로 수렴한다고 생각하여 Degree=80으로 선정

5 모델

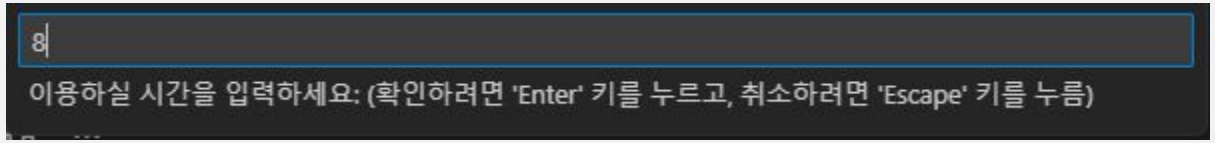
- 모델 테스트 – 1호선 서울역 평일 8시 승차 데이터 입력



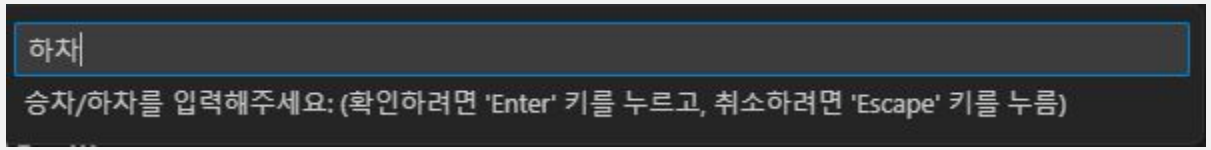
- ‘역명’ 입력



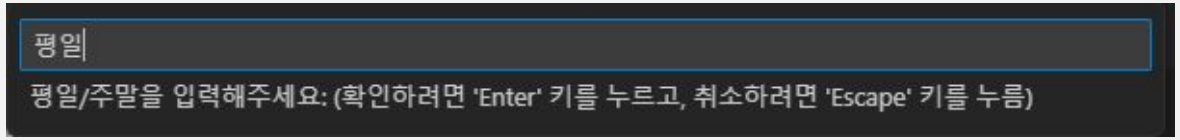
- ‘호선’ 입력



- ‘시간’ 입력



- ‘승/하차’ 입력



- ‘평일/주말’ 입력

5 모델

- 회귀모델 결과 – 1호선 서울역 평일 8시 승차

- 선형회귀분석
  - RMSE = 1252.857
  - RMSLE = 계산할 수 없음(음수값 존재)

- 의사결정나무
  - RMSE = 617.70
  - RMSLE=0.30271

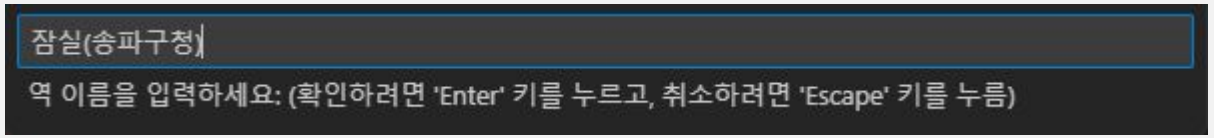
★ 가장 좋은 성능을 내는 모델

- 랜덤포레스트
  - RMSE = 617.72
  - RMSLE =0.30275

- SVR
  - RMSE = 631.6
  - RMSLE =0.304

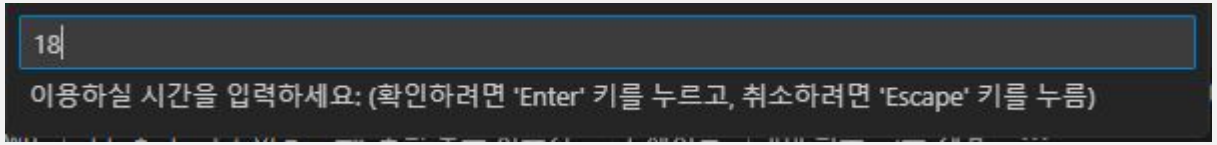
5 모델

- 모델 테스트 – 2호선 잠실(송파구청) 평일 18시 승차 데이터 입력



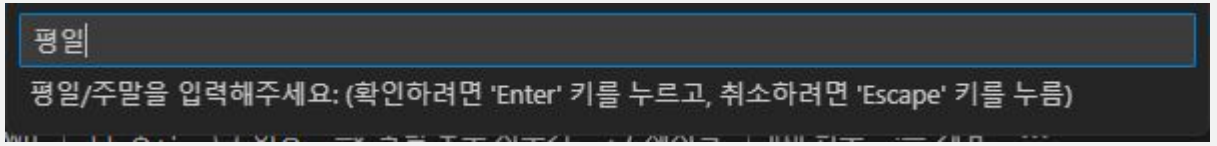
- ‘역명’ 입력

- ‘호선’ 입력



- ‘시간’ 입력

- ‘승/하차’ 입력



- ‘평일/주말’ 입력

5 모델

- 회귀모델 결과 – 2호선 잠실(송파구청) 평일 18시 승차

- 선형회귀분석
  - RMSE = 1278.577
  - RMSLE = 계산할 수 없음(음수값 존재)

- 의사결정나무
  - RMSE = 909.365
  - RMSLE = 0.280295

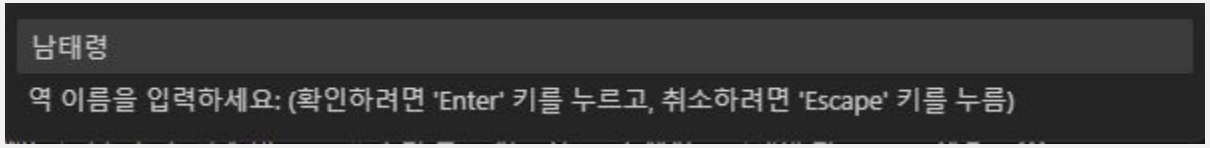
- 랜덤포레스트
  - RMSE = 908.725
  - RMSLE = 0.280200

★ 가장 좋은 성능을 내는 모델

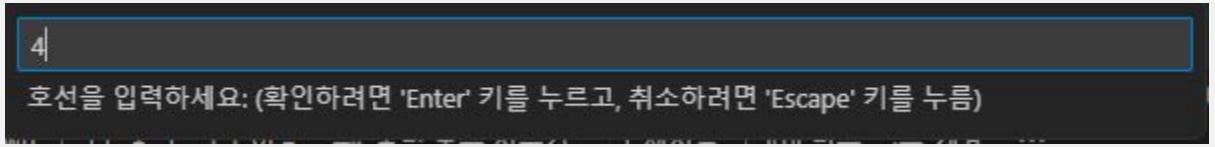
- SVR
  - RMSE = 922.611
  - RMSLE = 0.281

5 모델

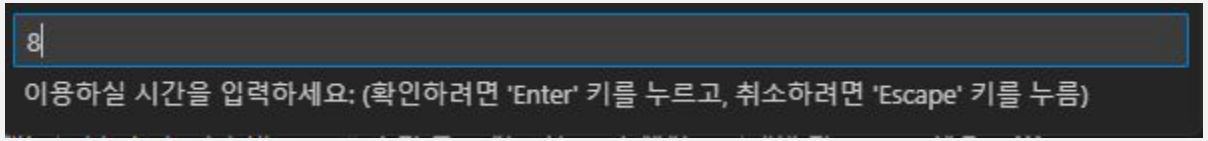
- 모델 테스트 – 4호선 남태령역 평일 8시 하차 데이터 입력



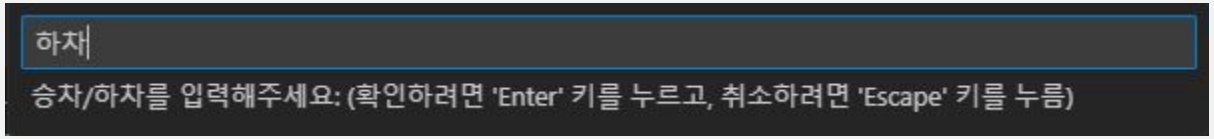
- ‘역명’ 입력



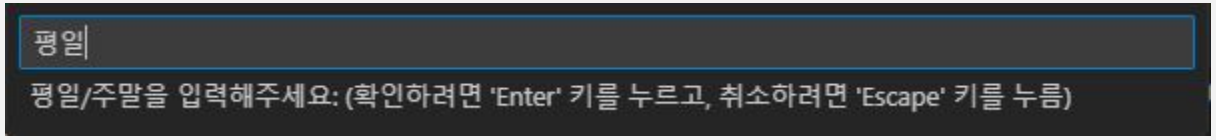
- ‘호선’ 입력



- ‘시간’ 입력



- ‘승/하차’ 입력



- ‘평일/주말’ 입력

5 모델

- 회귀모델 결과 – 4호선 남태령역 평일 8시 하차

- 선형회귀분석
  - RMSE = 38.259
  - RMSLE = 계산할 수 없음(음수값 존재)

- 의사결정나무
  - RMSE = 29.243
  - RMSLE = 0.34892

★ 가장 좋은 성능을 내는 모델

- 랜덤포레스트
  - RMSE = 29.247
  - RMSLE = 0.34915

- SVR
  - RMSE = 29.507
  - RMSLE = 0.34478

5 모델

- 3개의 모델 테스트 RMSLE 값 비교

1호선 서울역 평일 8시 승차 모델

- 의사결정나무
  - RMSLE=0.30271
- 랜덤포레스트
  - RMSLE =0.30275
- SVR
  - RMSLE =0.304

2호선 잠실(송파구청) 평일 18시 승차 모델

- 의사결정나무
  - RMSLE = 0.280295
- 랜덤포레스트
  - RMSLE = 0.280200
- SVR
  - RMSLE = 0.281

4호선 남태령역 평일 8시 하차 모델

- 의사결정나무
  - RMSLE = 0.34892
- 랜덤포레스트
  - RMSLE = 0.34915
- SVR
  - RMSLE = 0.34478

- 3개의 모델의 RMSLE 값은 **2호선 잠실역모델 < 1호선 서울역 모델 < 4호선 남태령역** 모델 순으로 증가함
  잠실역 모델이 상대적 error가 가장 적은 모델로 모델링 되었음
- 이런 모델의 상대적 순서는 지하철의 이용객 수와도 일치함
  - 이용객 수가 많은 역이 상대적 error가 더 적은 회귀모델을 만든다면, 우리의 연구 목적과도 일치

5 모델

- 최종 모델
  - 최종적으로 피처는 ‘time(시간)’, ‘in_out(승/하차)’, ‘weekend(주말/평일)’을 선택
  - 종속변수는 ‘passenger(이용객 수)’
- 인코딩
  - 모든 피처는 OneHotEncoder로 인코딩 후 사용

	code	passenger	in_out_승차	in_out_하차	weekend_주말	weekend_평일	time_5	time_6	time_7	time_8	...	time_14	time_15	time_16	time_17	time_18	time_19	time_20	time_21	time_22	time_23
4484	434	11.0	1	0	1	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4485	434	13.0	1	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
4486	434	17.0	1	0	1	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
4487	434	33.0	1	0	1	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
4488	434	34.0	1	0	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3766821	434	44.0	0	1	1	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
3766822	434	43.0	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	1	0	0	0
3766823	434	25.0	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
3766824	434	16.0	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
3766825	434	15.0	0	1	1	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1

- 회귀모델은 지하철 역별 개별 모델링

5 모델

- 총 2개의 회귀모델(의사결정나무, 랜덤포레스트)를 활용

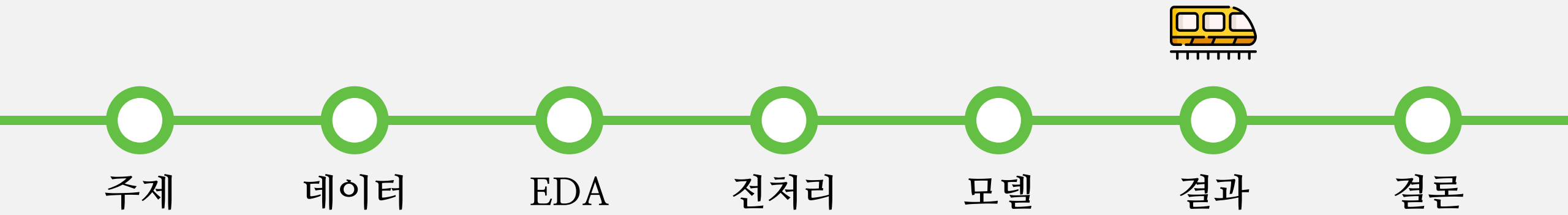
- 선형회귀모델의 경우 모든 모델 테스트에서 다른 3개의 회귀 모델에 비해 현저히 낮은 성능을 보임

  따라서 선형회귀모델을 제외

- SVR 모델 또한 의사결정나무, 랜덤포레스트에 비해 3번의 모델테스트에서 모두 낮은 성능을 보임

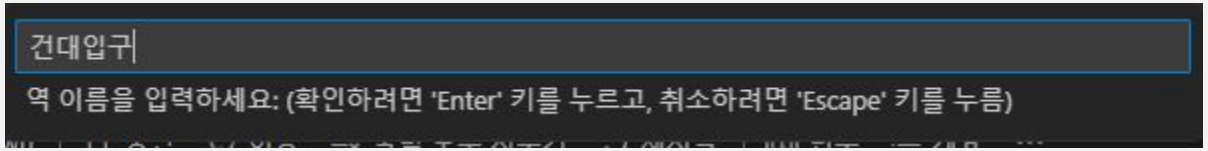
  SVR 모델 또한 제외

- 선정된 2개의 모델에 대해서는 두 모델의 예측값의 평균 값으로 결과를 도출

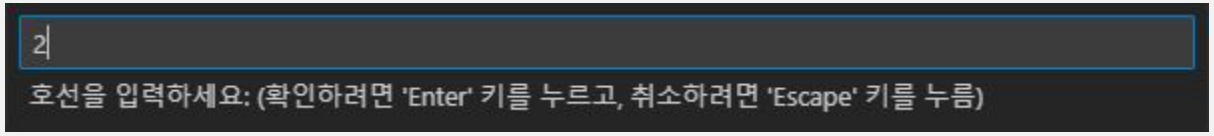

주제
데이터
EDA
전처리
모델
결과
결론

6 결과

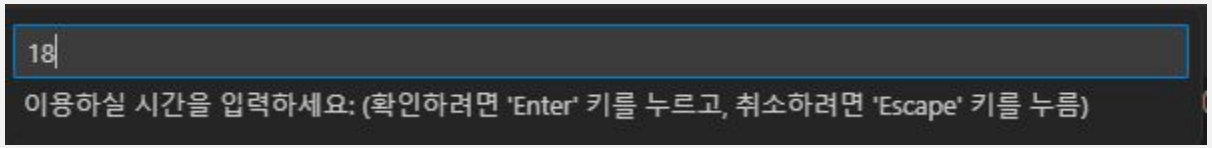
- 조건 데이터 입력



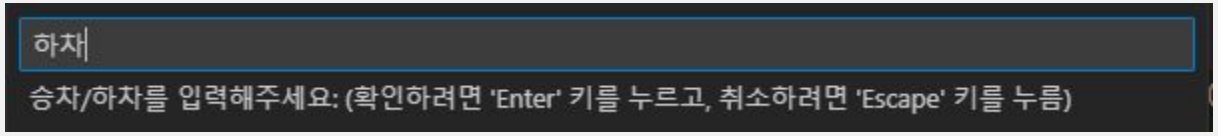
- ‘역명’ 입력



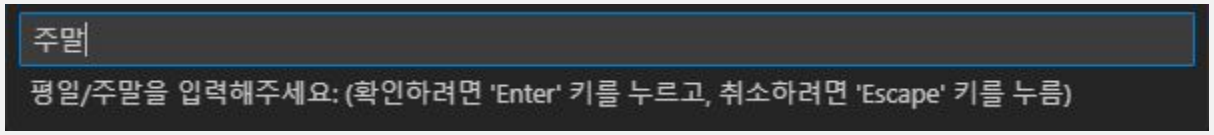
- ‘호선’ 입력



- ‘시간’ 입력



- ‘승/하차’ 입력



- ‘평일/주말’ 입력

6 결과

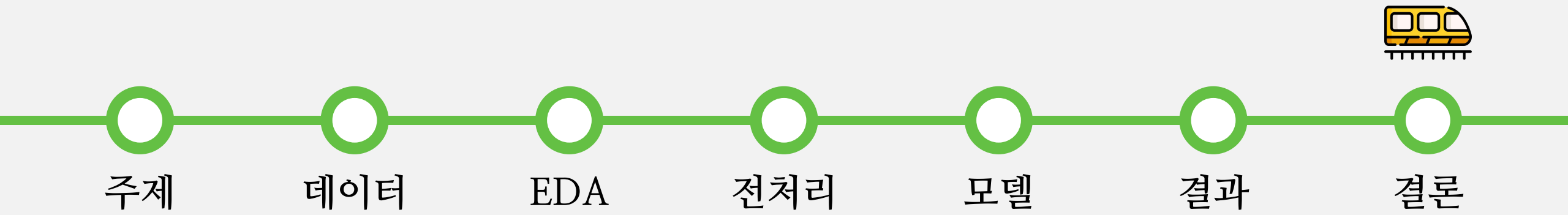
- **모델 선택 및 결과 도출**

  - 의사결정나무의 예측값
    3177.432명 / RMSE= 397.345
  - 랜덤포레스트의 예측값
    3181.084명 / RMSE = 397.439

  - 최종 결과

```
print(f'{line}호선 {station_name}역 {time}시 {week} {in_out} 이용객 수는 {num_of_cus}명 입니다')
✓ 0.0s
2호선 건대입구역 18시 주말 하차 이용객 수는 3179명 입니다
```

  - 예측 결과(두 모델의 예측값의 평균값) 2호선 건대입구역 18시 주말 하차 이용객 수는 3179명
    평균 RMSE = 397.392


주제
데이터
EDA
전처리
모델
결과
결론

7 결론

- 본 연구의 한계점

  - 회귀분석을 기반으로 승/하차 인원을 예측했으나, 이러한 승/하차 인원을 고객이 보고 이를 이용할 근거가 필요
    예) 1호선 '서울역'역 8시 평일 하차 이용객 수는 8406명 입니다. ▶ 8406명이 많은 것인가? 얼마나 많은 것인가?

  - 데이터가 1시간 단위의 데이터이기 때문에 결과도 1시간 동안의 이용객 수를 예측
    예) 7시 01분에 승차 이용객 수 = 7시 59분 승차 이용객 수
    총 58분 = 약 1시간의 시간 간극이 있지만, 데이터 단위가 1시간이기때문에 결과값의 이용 시간은 같은 7시로 예측

  - 본 연구에서는 공휴일은 따로 고려하지 않았음
    ※ 공휴일은 주말이지만, 본 연구에서는 요일만을 고려하여 평일/주말을 분류하였기 때문에 공휴일이 평일로 분류된 경우가 많음

  - 초기 모델링에서 모델링 방법에 대한 이해도가 부족하여 제대로 모델링을 하지 못했음
    예) 첫 모델링, 서울역 모델링 등에서 SVR에 대해 제대로 이해하지 못해서 모델의 결과를 얻지 못함

  - 모델 테스트를 다양하게 하지 못했던 점
    · 시간 피처의 인코딩에 대한 실험에서는 서울역만을 기준으로 비교
    · 모델 테스트에서는 3개의 역(서울역, 잠실(송파구청), 남태령)의 결과만을 비교하여 최종 모델을 선정

7 결론

- 본 연구의 개선방안

  - 회귀분석을 통해 예측한 인원 수를 바탕으로 승차/하차 및 역의 혼잡도를 분류할 수 있는 분류 모델을 개발
    예) 1호선 서울역 8시 예상 승차 인원 00명 (매우 한산/ 한산 / 보통 / 혼잡 / 매우 혼잡)
    ※ '혼잡한 정도를 어떻게 분류할 것인가?'에 대한 연구 필요

  - 피처에 대한 더 구체적인 분석을 통해 연구 개선
    · 각 피처의 중요도를 평가하는 주성분 분석(PCA)등의 방법 등을 통해 모델의 성능을 향상하는 좋은 피처를 선택할 수 있음
    · 이러한 추가적인 분석에서는 날씨, 명절 등 더 다양한 피처들에 대해서도 고려해볼 수 있음

  - '지하철 이용객이 더 많은 역은 더 정확한 회귀모델을 만드는가?' 에 대한 구체적인 실험이 더해지면,
    본 연구의 첫 목적이었던 지옥철에 대한 문제 해결에 더 유의미한 값을 가진다고 할 수 있음

8 참고문헌

- 성진홍(2018), 기계학습을 이용한 서울 지하철 승하차 인원예측, 한양대학교 공학대학원 석사학위논문

- 최창현,김종성,김동현,이준형,김덕환(2018), 머신러닝 기법을 이용한 수도권 지역의 호우피해 예측함수 개발,한국방재학회논문집, vol.18, no.7, 통권 95호 pp. 435-447 (13 pages)

- 민미경(2018), 기계학습 클러스터링을 이용한 승하차 패턴에 따른 서울시 지하철역 분류,한국인터넷방송통신학회 논문지, vol.18, no.4, pp. 13-18 (6 pages)

- 김진수(2016), 빅데이터 분석을 이용한 지하철 혼잡도 예측 및 추천시스템,디지털융복합연구,한국디지털정책학회, vol.14, no.11, pp. 289-295 (7 pages)

감사합니다