파이썬기반 빅데이터

파이썬을 이용한 데이터 분석 프로젝트

21011928 전주혁

Contents

Intro

분석 데이터 선정

DACON 제주도 도로 교통량 데이터

* 제주 테크노파크 제주도 도로 교통량 예측 AI 경진대회

* 기간 : 2022.10.03 ~ 2022.11.14

* 정형 데이터 / 회귀

분석의 목표

분석의 목표

제주도 교통량 데이터 분석의 목표

1. **제주도 도로 교통량 예측 알고리즘 개발**

   -> 교통 체증이 일어나는 구간을 미리 예측

2. **새로운 비즈니스 가치 창출**

   -> 심야 버스 노선 개발

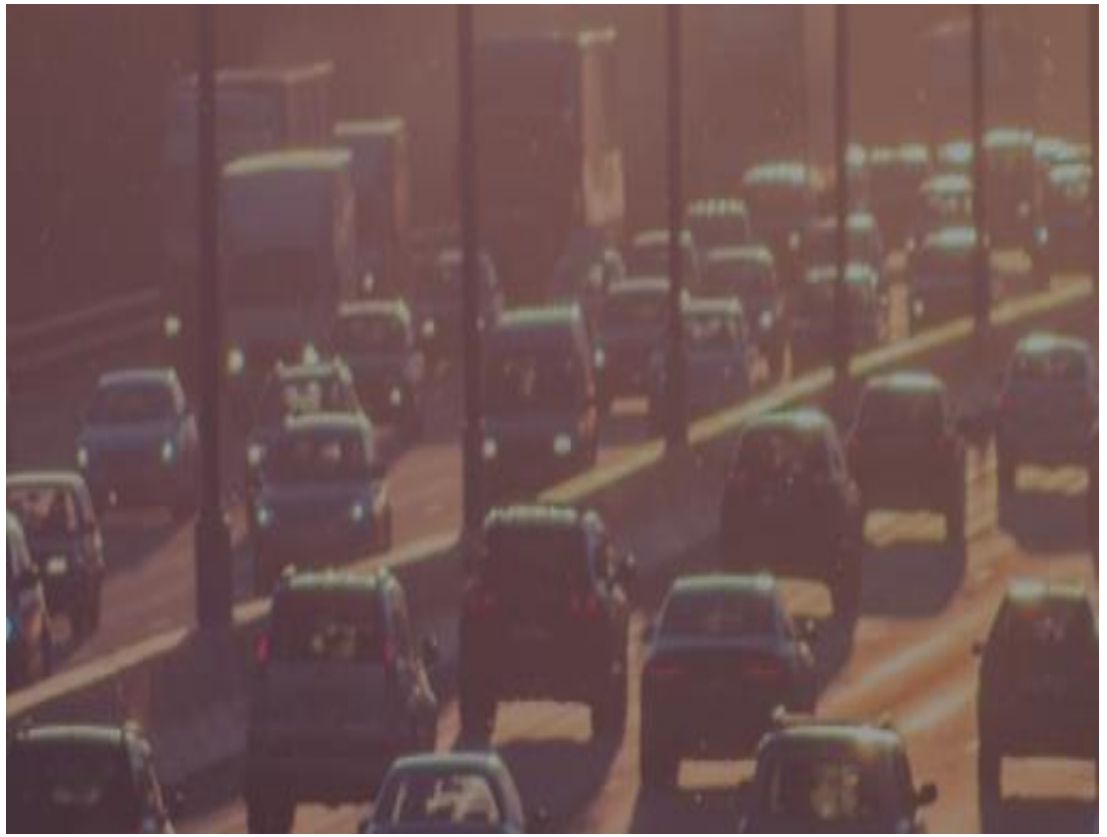
   -> 대중교통 서비스 개설

   -> 환승 거점 센터 선정

3. **새로운 정책 개발**

   -> 교통 운영 계획

   -> 공공 분야와 정책 개발에 활용



분석 데이터

분석 데이터

1. Train Data

- 2022년 8월 이전 데이터
- 4,701,217 개의 데이터
- 25 Columns

2. Test Data

- 2022년 8월 데이터
- 291,241 개의 데이터
- 24 Columns

Data Info

1. Id : 아이디
2. Base_date : 날짜
3. day_of_week : 요일
4. Base_hour : 시간대
5. Road_in_use : 도로사용여부
6. Lane_count : 차로수
7. Road_rating : 도로등급
8. Mulit_linked : 중용구간 여부
9. Connect_code : 연결로 코드
10. Maximum_speed_limit : 속도제한
11. Weight_restricted : 통과제한하중
12. Height_restricted : 통과제한높이
13. geight_restricted : 통과제한높이
14. road_type : 도로유형
15. start_latitude : 시작지점의 위도
16. start_longitude : 시작지점의 경도
17. start_turn_restricted:시작지점 회전제한 유무
18. end_latitude : 도착지점의 위도
19. end_longitude : 도착지점의 경도
20. end_turn_restricted:도착지점 회전제한 유무
21. road_name : 도로명
22. Start_node_name : 시작지점명
23. end_node_name : 도착지점명
24. vehicle_restricted : 통과제한 차량
25. Target : 평균속도(Test Data에는 제외)

분석 데이터

	id	base_date	day_of_week	base_hour	road_in_use	lane_count	road_rating	road_name	multi_linked	connect_code	...	road_type	start_node_name	start_latitude	start
0	TRAIN_0000000	20220623	목	17	0	1	106	지방도 1112호선	0	0	...	3	제3교래교	33.427747	
1	TRAIN_0000001	20220728	목	21	0	2	103	일반국도11 호선	0	0	...	0	광양사거리	33.500730	
2	TRAIN_0000002	20211010	일	7	0	2	103	일반국도16 호선	0	0	...	0	창고천교	33.279145	
3	TRAIN_0000003	20220311	금	13	0	2	107	태평로	0	0	...	0	남양리조트	33.246081	
4	TRAIN_0000004	20211005	화	8	0	2	103	일반국도12 호선	0	0	...	0	애월샷시	33.462214	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4701212	TRAIN_4701212	20211104	목	16	0	1	107	-	0	0	...	0	대림사거리	33.422145	
4701213	TRAIN_4701213	20220331	목	2	0	2	107	-	0	0	...	3	광삼교	33.472505	
4701214	TRAIN_4701214	20220613	월	22	0	2	103	일반국도12 호선	0	0	...	0	고성교차로	33.447183	
4701215	TRAIN_4701215	20211020	수	2	0	2	103	일반국도95 호선	0	0	...	0	제6광령교	33.443596	
4701216	TRAIN_4701216	20211019	화	6	0	2	107	경찰로	0	0	...	0	서귀포경찰서	33.256785	

4701217 rows × 24 columns

Train 데이터의 구조

분석 데이터

Train 데이터 info
- 9개의 float 형
- 10개의 int 형
- 5개의 object 형

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4701217 entries, 0 to 4701216
Data columns (total 24 columns):
 #   Column                 Dtype
---  ------                 -----
 0   id                     object
 1   base_date              int64
 2   day_of_week            object
 3   base_hour              int64
 4   road_in_use            int64
 5   lane_count             int64
 6   road_rating            int64
 7   road_name              object
 8   multi_linked           int64
 9   connect_code           int64
 10  maximum_speed_limit    float64
 11  vehicle_restricted     float64
 12  weight_restricted      float64
 13  height_restricted      float64
 14  road_type              int64
 15  start_node_name        object
 16  start_latitude         float64
 17  start_longitude        float64
 18  start_turn_restricted  int64
 19  end_node_name          object
 20  end_latitude           float64
 21  end_longitude          float64
 22  end_turn_restricted    int64
 23  target                 float64
dtypes: float64(9), int64(10), object(5)
```

Test 데이터 info
- 8개의 float 형
- 8개의 int 형
- 7개의 object 형

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 291241 entries, 0 to 291240
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   id                     291241 non-null  object
 1   base_date              291241 non-null  int64
 2   day_of_week            291241 non-null  object
 3   base_hour              291241 non-null  int64
 4   road_in_use            291241 non-null  int64
 5   lane_count             291241 non-null  int64
 6   road_rating            291241 non-null  int64
 7   road_name              291241 non-null  object
 8   multi_linked           291241 non-null  int64
 9   connect_code           291241 non-null  int64
 10  maximum_speed_limit    291241 non-null  float64
 11  vehicle_restricted     291241 non-null  float64
 12  weight_restricted      291241 non-null  float64
 13  height_restricted      291241 non-null  float64
 14  road_type              291241 non-null  int64
 15  start_node_name        291241 non-null  object
 16  start_latitude         291241 non-null  float64
 17  start_longitude        291241 non-null  float64
 18  start_turn_restricted  291241 non-null  object
 19  end_node_name          291241 non-null  object
 20  end_latitude           291241 non-null  float64
 21  end_longitude          291241 non-null  float64
 22  end_turn_restricted    291241 non-null  object
dtypes: float64(8), int64(8), object(7)
```

Part 2,

분석 데이터

Train 데이터의 피쳐별 분포도

- **node_in_use**
- **mulit_linked**
- **connect_code**
- **vehicle_restricted**
- **height_restricted**

그래프만 확인 시 위 피쳐들의 value 값이 한 종류로 파악됨

Part 2,

분석 데이터

value_counts()로 확인하기

True columns -> Drop

```
train['vehicle_restricted'].value_counts()
```

```
0.0    4701217
Name: vehicle_restricted, dtype: int64
```

```
train['height_restricted'].value_counts()
```

```
0.0    4701217
Name: height_restricted, dtype: int64
```

```
train['road_in_use'].value_counts()
```

```
0    4694812
1       6405
Name: road_in_use, dtype: int64
```

```
train['connect_code'].value_counts()
```

```
0      4689075
103      12142
Name: connect_code, dtype: int64
```

```
train['multi_linked'].value_counts()
```

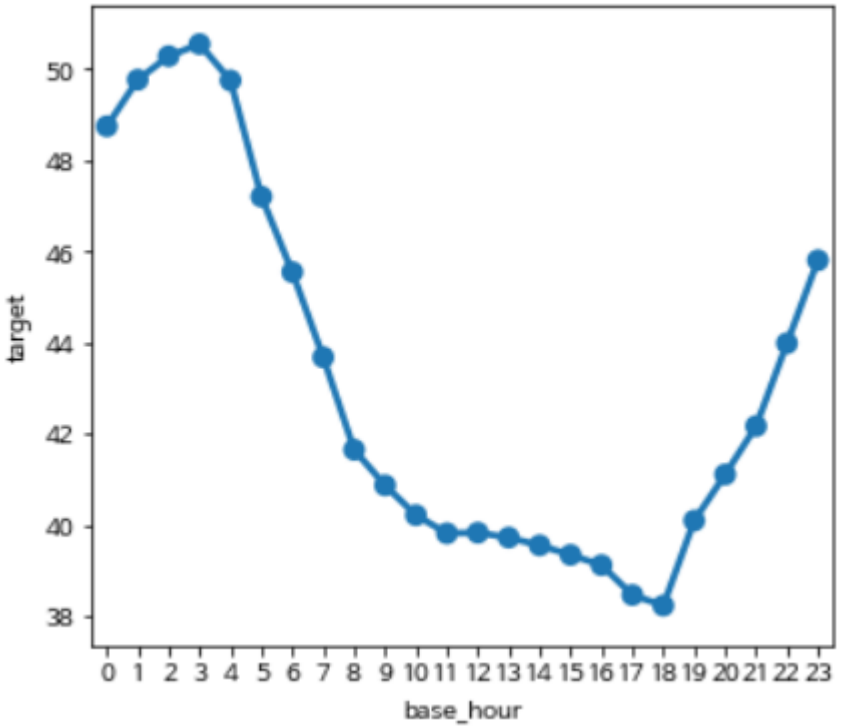
```
0    4698978
1       2239
Name: multi_linked, dtype: int64
```

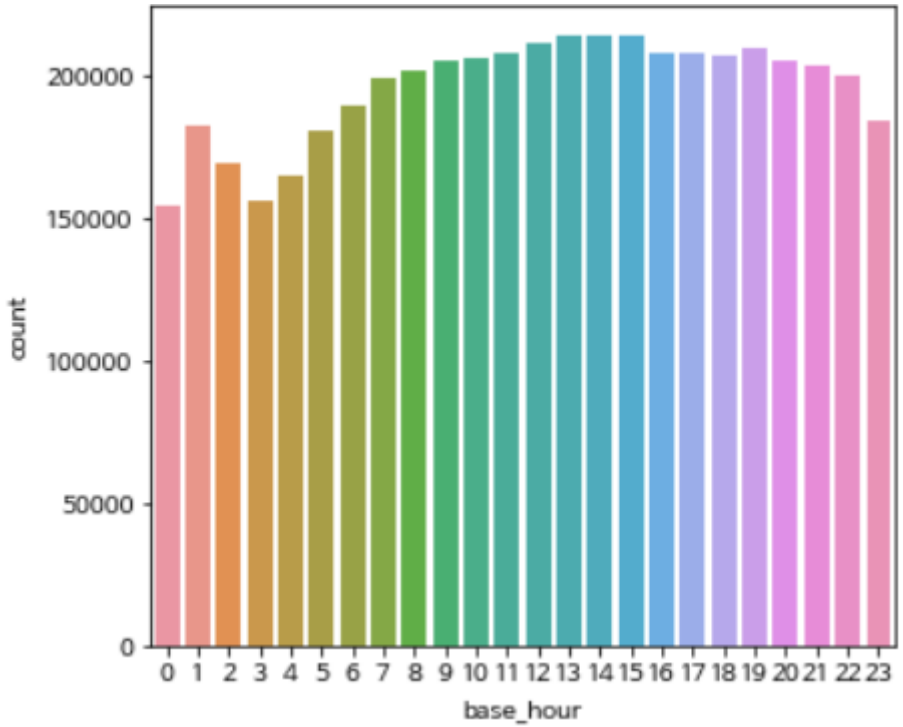
분석 결과 + 예상

Part 2,

분석 데이터

1. 출퇴근 시간의 반전



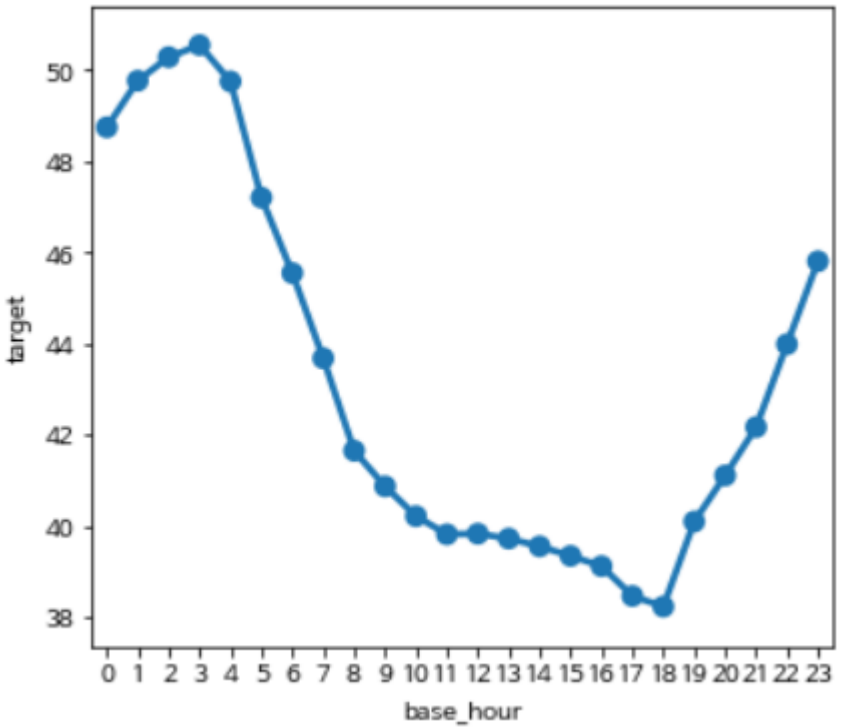


일반적으로 하루 중 출,퇴근시간이 교통체증이 제일 심할 것 이라 생각하지만 틀린 가설
-> 데이터의 불균형도 확인

Part 2,

분석 데이터

1. 출퇴근 시간의 활용 방안



파생변수 생성

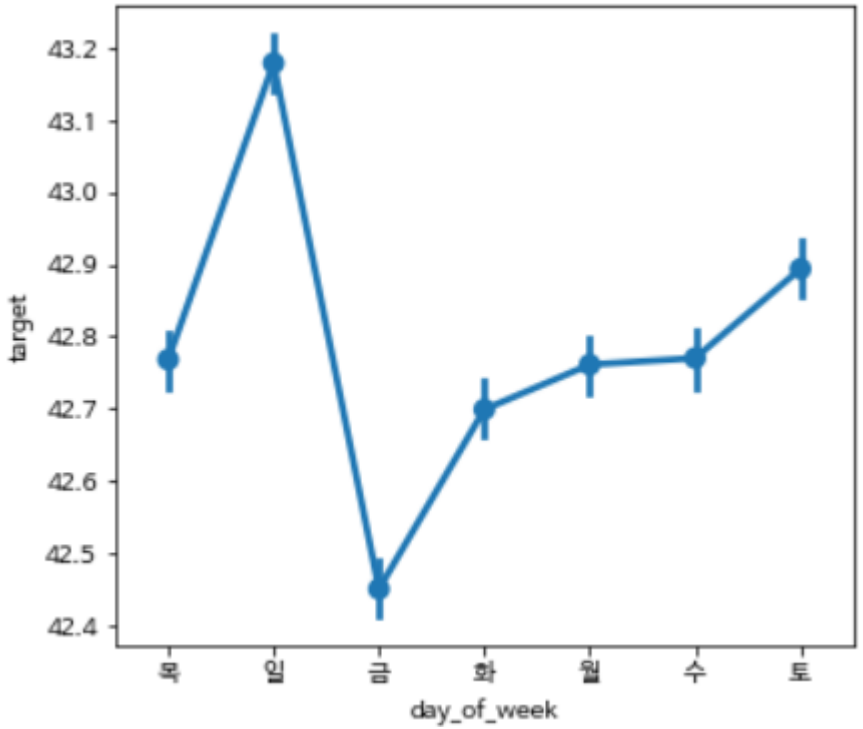
- **dummi**e 화

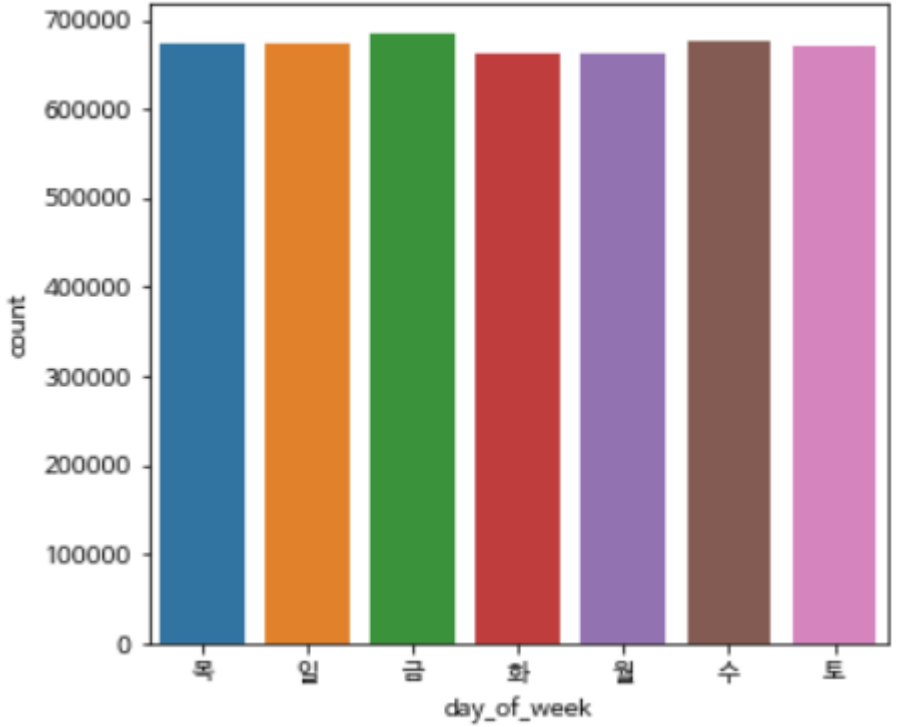
- 출근시간부터 퇴근시간까지 평균 속도가 점차 줄어들다가 퇴근시간이 끝난 후 부터 평균속도가 증가함
  -> 8시~20시를 기준으로 새로운 피쳐 생성
  → 8시~20시인 시간대는 1로 그 외 시간대는 0으로

Part 2,

분석 데이터

2. 요일의 반전

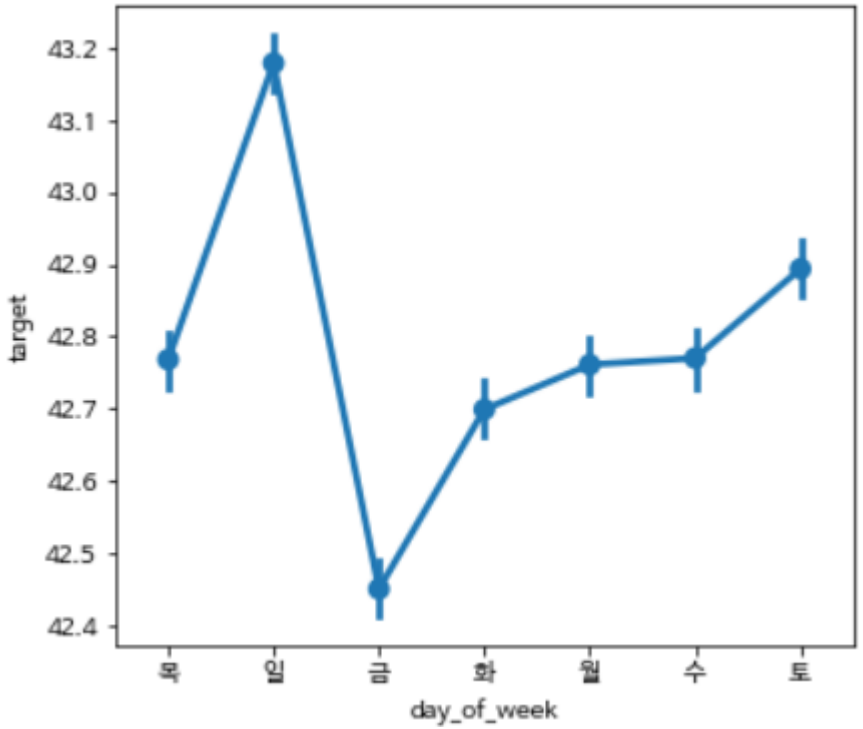




일반적으로 금,토,일 교통체증이 제일 심할 것 이라 생각했지만 틀린 가설
-> 데이터의 불균형도 확인

분석 데이터

2. 요일 활용 방안



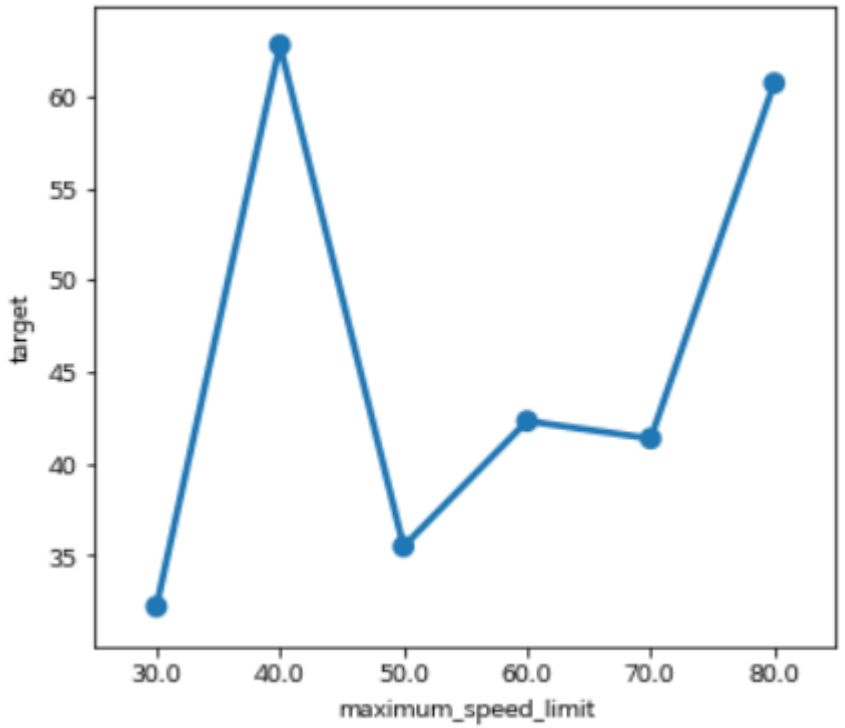
Solution : **파생변수** 생성

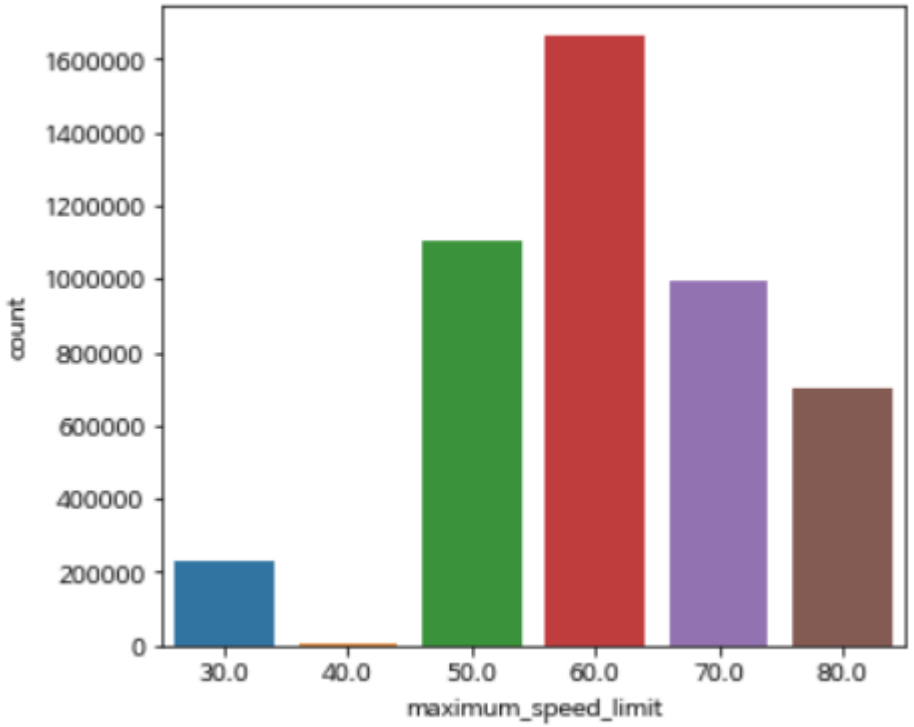
- 비교적 평균속도가 비슷한 목,화,월,수,토 피쳐 생성 후 해당되는 인덱스는 1, 그 외는 0
- 평균속도가 제일 낮은(교통체증이X) 금요일 피쳐 생성 후 해당되는 인덱스는 1, 그 외는 0
- 평균속도가 제일 높은(교통체증이X) 일요일 피쳐 생성 후 해당되는 인덱스는 1, 그외는 0

Part 2,

분석 데이터

3. 상식적으로 이해가 되지 않는 그래프

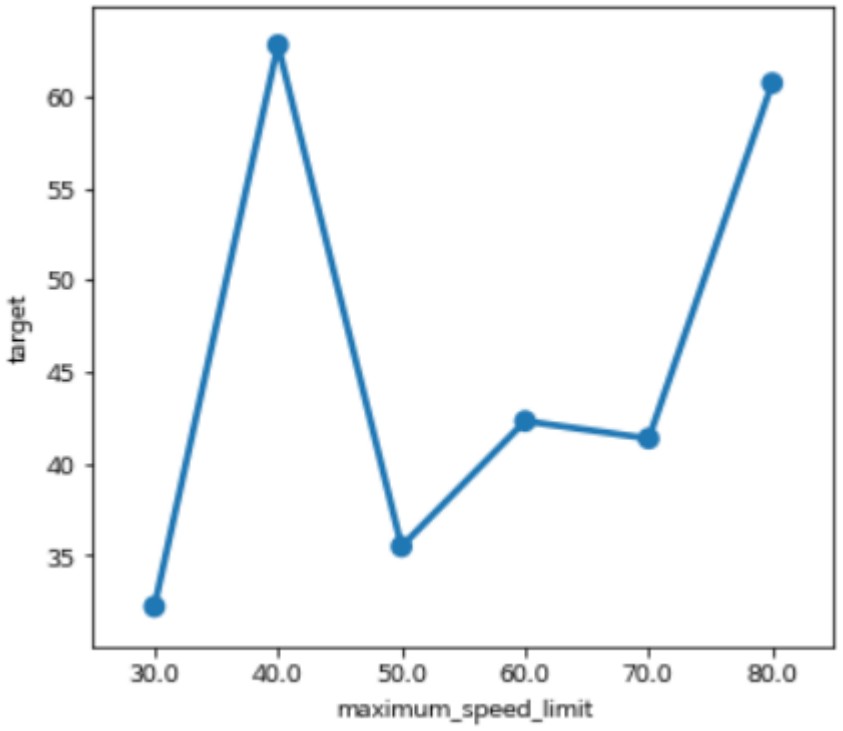




최고 제한 속도가 40인 도로가 평균 속도가 약 70으로 제일 높음
-> 데이터 불균형 존재
-> 불균형과 관계 有?

분석 데이터

3. 최고속도제한 활용 방안




```
(train['maximum_speed_limit'] == 40).value_counts()
```

```
False    4694427
True        6790
Name: maximum_speed_limit, dtype: int64
```

최고제한속도가 40인 데이터의 개수는
약 **470만개 중 6790개 존재**

Solution : **Drop**
-> 데이터가 약 0.15% 존재
-> 최고제한속도가 40인 데이터들은 잘못 측정 or 잘못 입력된 데이터로 생각
-> 또한 데이터가 40인 index들은 Drop을 시켜줘도 영향 X

Outro

Part 2,

분석 데이터

Why? : 새로운 피쳐를 여러개 만들어 주는 이유

- column 한 개에 값을 0, 1, 2, 3 이렇게 구분을 해 주는게 더 편하지 않나?

-> LabelEncoder vs OneHotEncoder

1. LabelEncoder : 한 column에 피쳐의 종류가 3개 존재한다면 한 column에 0,1,2로 변환

2. OneHotEncoder : 한 column에 피쳐의 종류가 3개 존재한다면 세개의 column을 생성해 해당하는 index를 1로 해당하지 않는 index를 0으로 변환

Ex)

	강의명
0	파이썬빅데이터
1	세계사
2	동고강

LabelEncoder

	강의명
0	2
1	1
2	0

OneHotEncoder

	0	1	2
0	0.0	0.0	1.0
1	0.0	1.0	0.0
2	1.0	0.0	0.0

Part 2,

분석 데이터

그렇다면 LabelEncoder가 메모리 용량이 적으니 LabelEncoder를 사용하는게 좋지 않나?

-> 순서가 없는 한 column 안에 사과 = 1, 배 = 2, 딸기 = 3으로 Label Encoding 한다면 1 + 2 = 3
즉, 사과 + 배 = 딸기 같이 상관관계가 있다고
생각하면서 머신러닝 모델이 학습할 수 있다.

결론

1. Label Encoding
 -> 순서의 의미가 있을 때
 -> 고유값의 개수가 너무 많을 때

2. One-Hot-Encoding
 -> 순서가 없을 때
 -> 고유값의 개수가 많지 않을 때

“ *감사합니다.* ”