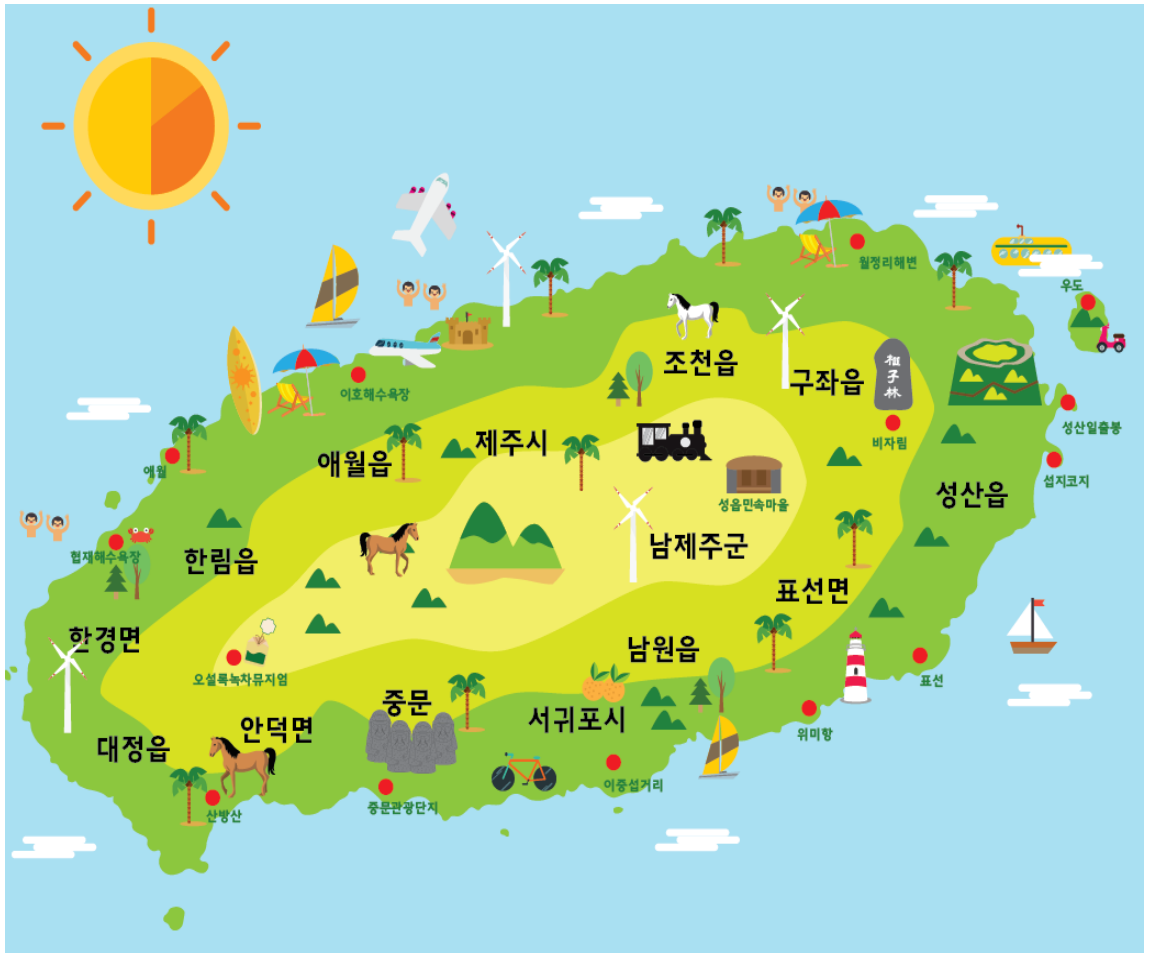
제주도 도로 교통량 데이터 분석 및 시각화

2022.10.14

박소영

✓ 데이터 분석 목적



제주도 인구 연간 1.3% 증가

✚

주민등록인구 68만명 + 외국인/관광객
합산 시 총 90만명 이상 예상

제주도의 교통체중 문제 심화

"제주도 도로 교통량 예측 AI 경진대회"

Jeju 제주특별자치도 Jeju Special Self-Governing Province
JTP 제주테크노파크 JEJU TECHNOPARK

✓ 데이터 개요

Train.csv

4,701,217개의 데이터

train.shape
(4701217, 24)

- id : 샘플 별 고유 id
- 날짜, 시간, 교통 및 도로구간 등 정보
- target : 도로의 차량 평균 속도(km)

Test.csv

291,241개의 데이터

test.shape
(291241, 23)

- id : 샘플 별 고유 id
- 날짜, 시간, 교통 및 도로구간 등 정보

data_info.csv

데이터의 각 Column 별 설명

data_info.shape
(24, 2)

```
print("Train dataset shape is",train.shape," and Test dataset shape is ",test.shape)
```

>> Train dataset shape is (4701217, 24) and Test dataset shape is (291241, 23)

```
print("Train data has " + str(train.isnull().sum().sum()) + " null values and Test data has " + str(test.isnull().sum().sum()) + " null values")
```

>> Train data has 0 null values and Test data has 0 null values (결측치 없음)

✓ data_info.csv

	변수명	변수 설명
0	id	아이디
1	base_date	날짜
2	day_of_week	요일
3	base_hour	시간대
4	road_in_use	도로사용여부
5	lane_count	차로수
6	road_rating	도로등급
7	multi_linked	중용구간 여부
8	connect_code	연결로 코드
9	maximum_speed_limit	최고속도제한
10	weight_restricted	통과제한하중
11	hight_restricted	통과제한높이
12	road_type	도로유형
13	start_latitude	시작지점의 위도
14	start_longitude	시작지점의 경도
15	start_turn_restricted	시작 지점의 회전제한 유무
16	end_latitude	도착지점의 위도
17	end_longitude	도착지점의 경도
18	end_turn_restricted	도작지점의 회전제한 유무
19	road_name	도로명
20	start_node_name	시작지점명
21	end_node_name	도착지점명
22	vehicle_restricted	통과제한차량
23	target	평균속도(km)

✓ train.csv

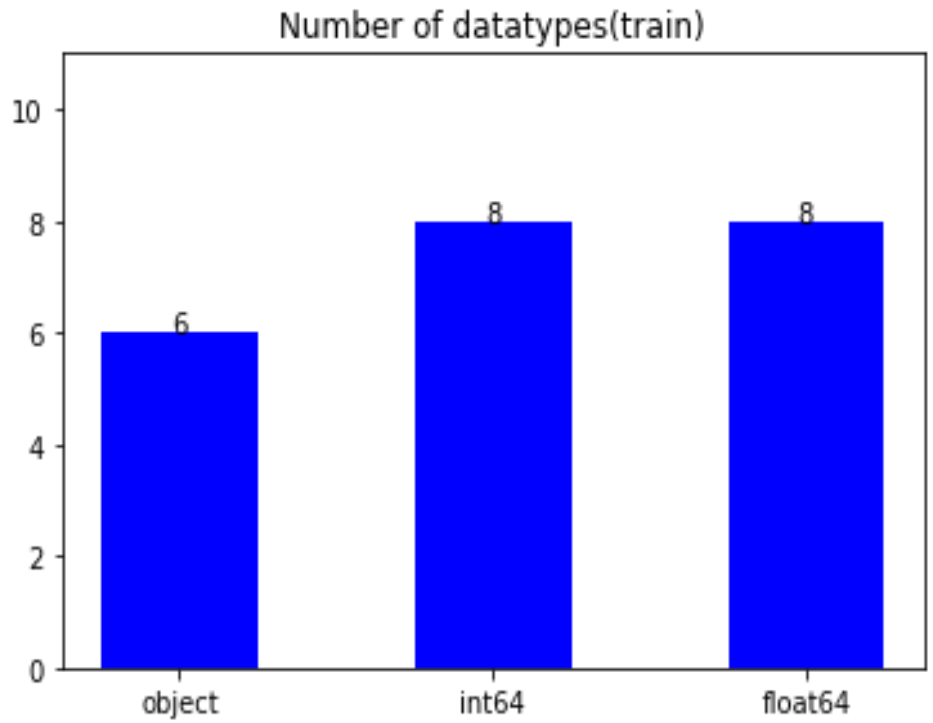
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4701217 entries, 0 to 4701216
Data columns (total 24 columns):
 #   Column                 Dtype
---  ------                 -----
 0   id                     object
 1   base_date              int64
 2   day_of_week            object
 3   base_hour              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  object
 19  end_node_name          object
 20  end_latitude           float64
 21  end_longitude          float64
 22  end_turn_restricted    object
 23  target                 float64
dtypes: float64(9), int64(8), object(7)
memory usage: 860.8+ MB
```

Train과 Test 데이터의 피처 차이탐색

```
set(train.columns)-
set(test.columns)
>>{'target'}
```

Target 데이터를 제외하고 train과 test데이터는 동일한 column을 보유함

✓ 데이터 타입별 Column 개수 비교




```
fig, ax = plt.subplots()
x = ['object','int64','float64']
y = [object_col.shape[1]-1,int_col.shape[1], float_col.shape[1]-1]
plt.bar(x,y,width=0.5,color='b')
#exclude 'id' and 'target

def addlabels(x,y):
  for i in range(len(x)):
    plt.text(i,y[i],y[i],ha='center')


plt.ylim(0,11)
plt.title("Number of datatypes(train)")
addlabels(x,y)
plt.show()
```

✓ 데이터 타입별 Column 분류

1. 데이터 타입별 group by 를 통해 Column 리스트 반환

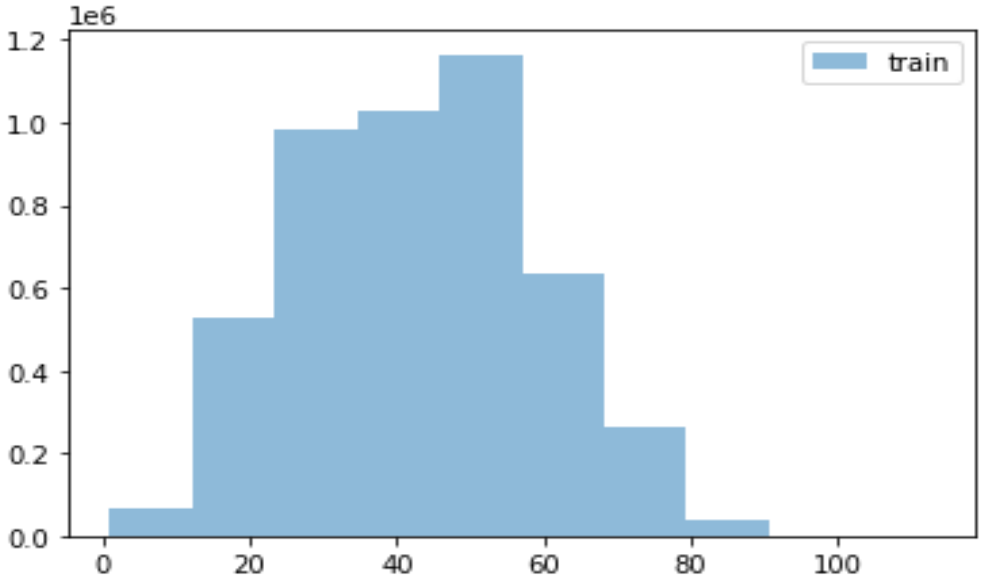
```
type_summary = {str(i):list(j) for i,j in train.groupby(train.dtypes,axis=1)}
type_summary
```

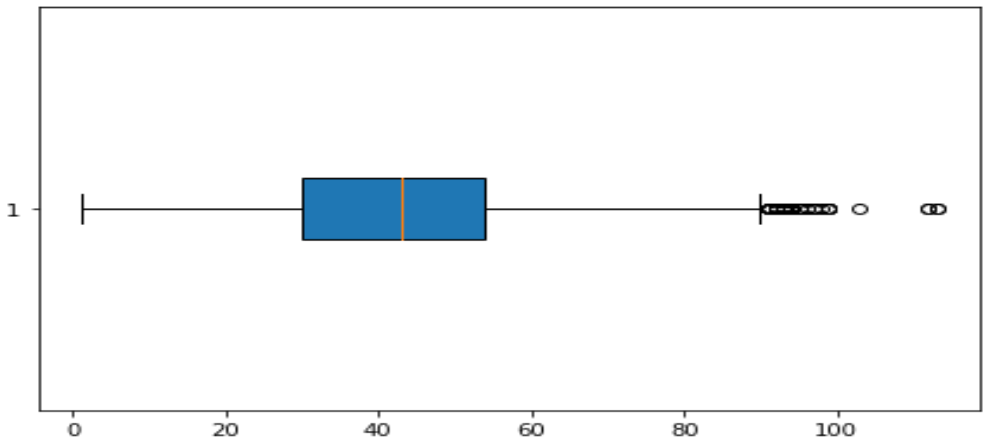
```
{'int64': ['base_date', 'base_hour', 'road_in_use', 'lane_count', 'road_rating', 'multi_linked', 'connect_code',
'road_type'], 'float64': ['maximum_speed_limit', 'vehicle_restricted', 'weight_restricted', 'height_restricted',
'start_latitude', 'start_longitude', 'end_latitude', 'end_longitude', 'target'], 'object': ['id', 'day_of_week',
'road_name', 'start_node_name', 'start_turn_restricted', 'end_node_name', 'end_turn_restricted']}
```

2. 효과적인 전처리를 위해 select_dtypes 를 이용한 데이터 프레임 분리

```
int_col = train.select_dtypes(include=['int64'])
float_col = train.select_dtypes(include=['float64'])
object_col = train.select_dtypes(include=['object'])
```

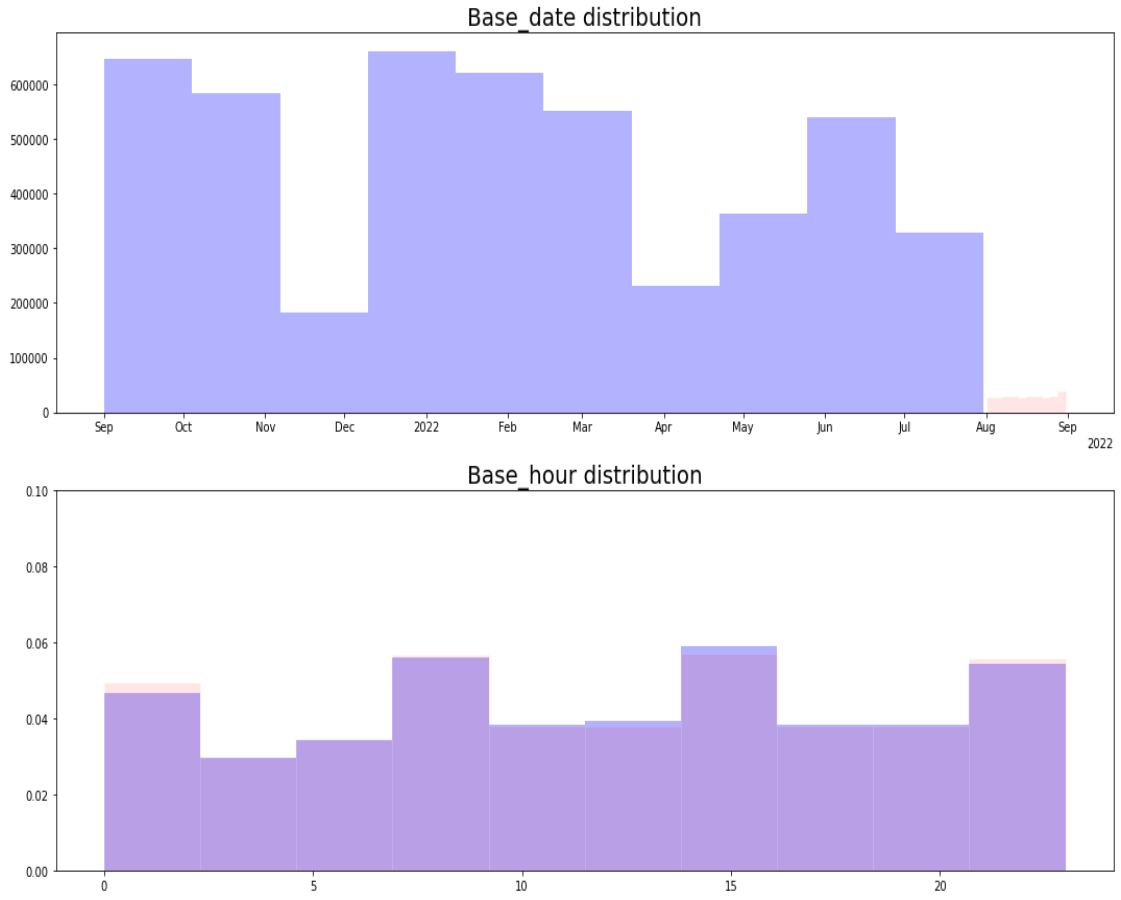
✓ Target 데이터 분포 파악






```
fig, ax = plt.subplots()
plt.hist(train.target,label='train',alpha=.5)
plt.legend()
plt.show()

train.target.describe().apply("{:.1f}".format)

count    4701217.0
mean          42.8
std           16.0
min            1.0
25%           30.0
50%           43.0
75%           54.0
max          113.0


plt.boxplot(train.target,vert=False,patch_artist=True)
plt.tight_layout()
```

✓ int 타입 데이터 분석 #1



- 월/시간대에 따라 교통량 변화 존재
- Train과 test 데이터의 base_hour 변화 유사

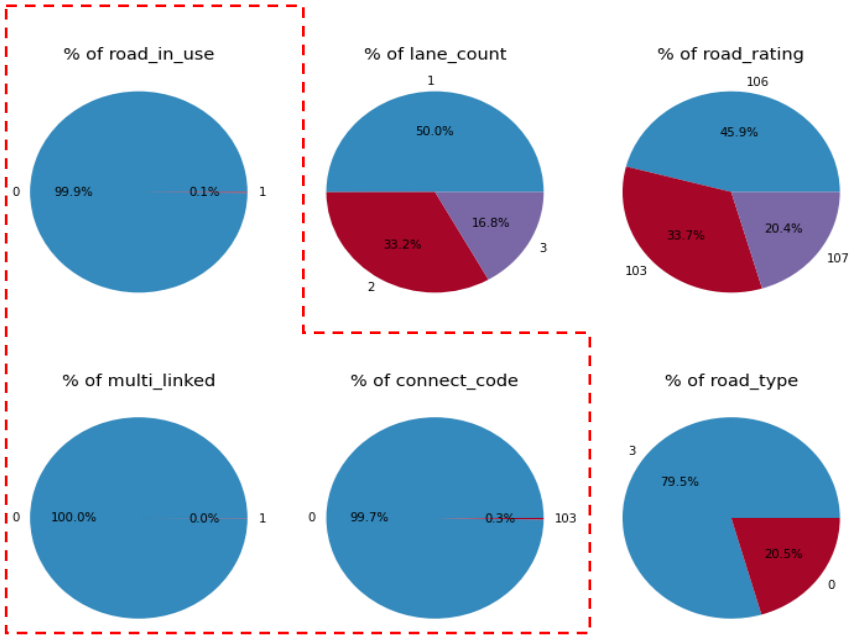
```
#datetime 으로 데이터 타입 변환
import matplotlib.dates as mdates
train['base_date'] = pd.to_datetime(train['base_date'],format="%Y%m%d")
test['base_date'] = pd.to_datetime(test['base_date'],format="%Y%m%d")
# fig, ax 생성
fig, ax = plt.subplots(2,figsize=(15,10))
ax[0].hist(train['base_date'],alpha=.3,color='b')
ax[0].hist(test['base_date'],alpha=.1,color='r',edgecolor='white')
# x 축 간격, format 지정
ax[0].xaxis.set_major_locator(mdates.MonthLocator())
ax[0].xaxis.set_minor_locator(mdates.MonthLocator())
ax[0].xaxis.set_major_formatter(mdates.ConciseDateFormatter(ax[0].xaxis.get_major_locator()))
ax[0].set_title("Base_date distribution",fontsize=20)

ax[1].hist(train['base_hour'],density=True,alpha=.3,color='b')
ax[1].hist(test['base_hour'],density=True,alpha=.1,color='r',edgecolor='white')
ax[1].set_title("Base_hour distribution",fontsize=20)
ax[1].set_ylim(0,.1)

plt.tight_layout()
plt.show()
```

✓ int 타입 데이터 분석 #2

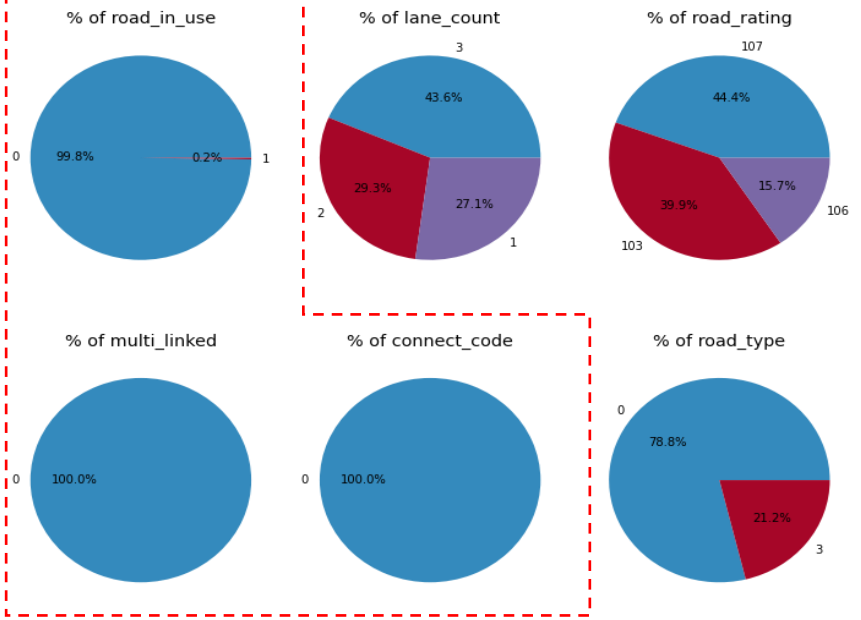
```
plt.figure(figsize=(10,12))
plt.style.use('bmh')

for i, col in enumerate(list(int_col.columns[2:])):
  plt.subplot(3,3,i+1)
  plt.pie(test[col].value_counts(),labels = test[col].u
nique(),autopct="%.1f%%")
  plt.title(f"% of {col}")

plt.tight_layout(h_pad = 2.5)
```

- Road_in_use, connect_code, 'multi linked' 데이터는 99-100% 한가지 값을 가지는 편향된 데이터로 무의미함
- 그외 Train과 Test 데이터 사이의 데이터 비율은 유사함

Train 데이터



Test 데이터



✓ Float 타입 데이터 분석 #1

```
float_col.describe().round(1)
```

	maximum_ speed_limit	vehicle_ restricted	weight_ restricted	height_ restricted	start_latitude	start_ longitude	end_latitude	end_ longitude
count	4701217.0	4701217.0	4701217.0	4701217.0	4701217.0	4701217.0	4701217.0	4701217.0
mean	61.3	0.0	5618.7	0.0	33.4	126.5	33.4	126.5
std	12.1	0.0	13953.4	0.0	0.1	0.2	0.1	0.2
min	30.0	0.0	0.0	0.0	33.2	126.2	33.2	126.2
25%	50.0	0.0	0.0	0.0	33.3	126.4	33.3	126.4
50%	60.0	0.0	0.0	0.0	33.4	126.5	33.4	126.5
75%	70.0	0.0	0.0	0.0	33.5	126.6	33.5	126.6
max	80.0	0.0	50000.0	0.0	33.6	126.9	33.6	126.9

`float_col.nunique()`

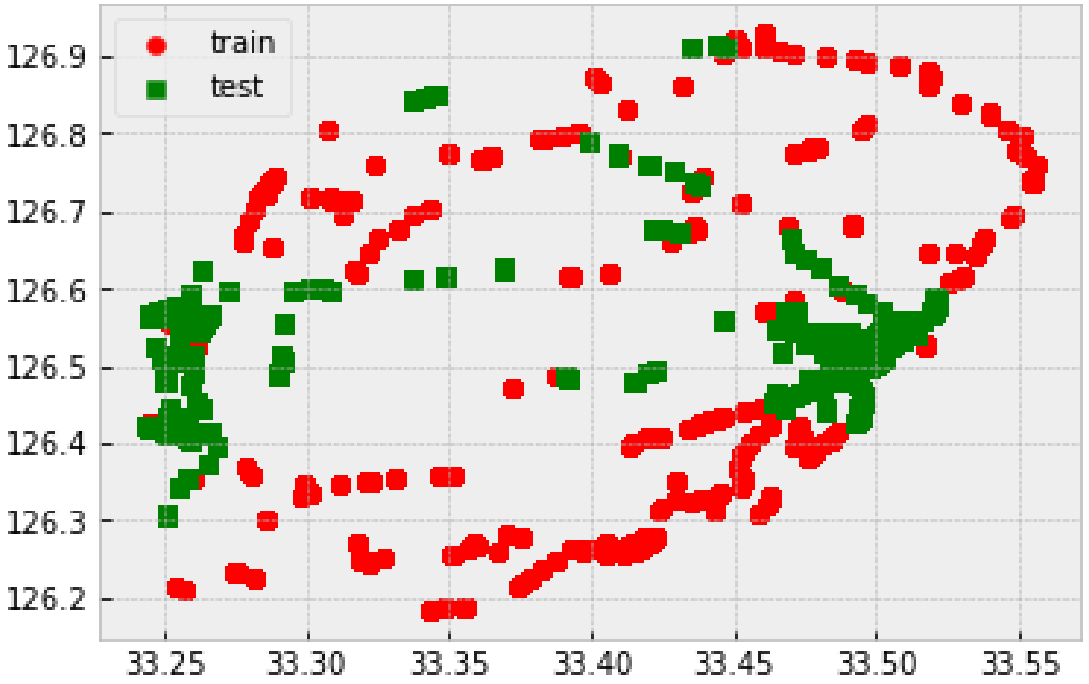
```
maximum_speed_limit 6
vehicle_restricted 1
weight_restricted 4
height_restricted 1
start_latitude 586
start_longitude 586
end_latitude 586
end_longitude 586
```

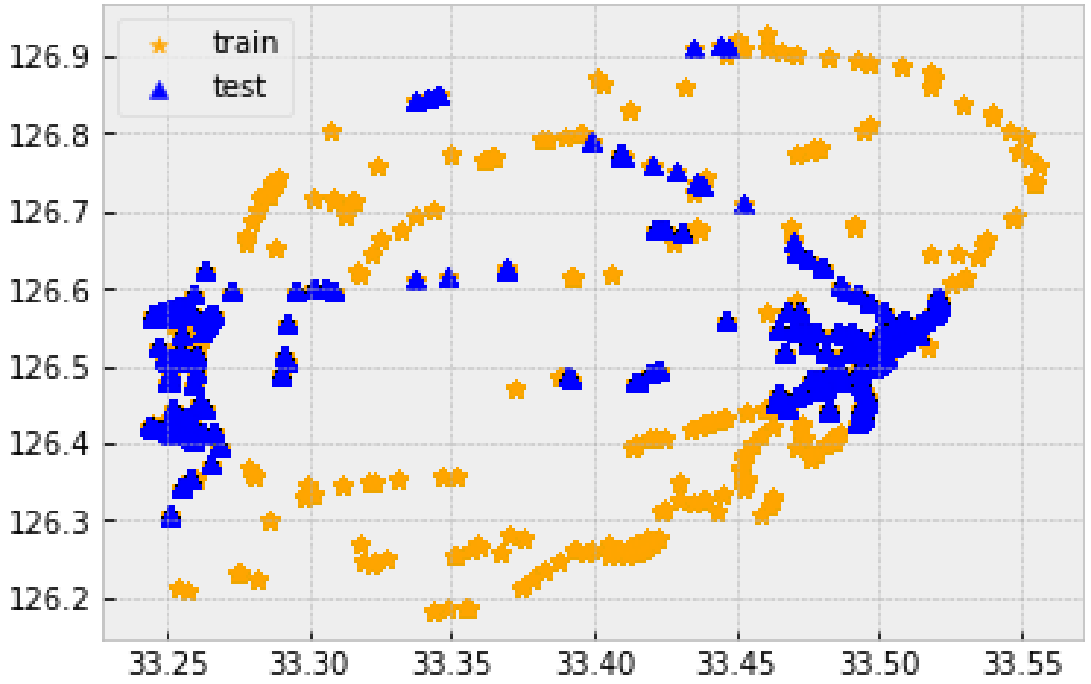
- Vehicle_retricted와 height_restricted 는 값이 0으로 통일되어 있어 무의미함
- Maximum speed limit은 속도 제한의 간격에 따라 카테고리화 되어 있음
- 위도와 경도는 min max 범위가 크지 않으며 start과 end 위도와 경도가 상당히 유사함

✓ Float 타입 데이터 분석 #2 (산점도 분석)

```
fig = plt.figure()
ax1 = fig.add_subplot(111)
ax1.scatter(train['start_latitude'],train['start_longitude
'],c='r',marker='o',label='train')
ax1.scatter(test['start_latitude'],test['start_longitude']
,c='g',marker='s',label='test')
plt.legend(loc='upper left')
plt.show()
```

- Train 데이터의 위도 경도 데이터가 훨씬 분산되어 있음
- Start 과 end 의 위도와 경도가 거의 동일한 분포를 보임



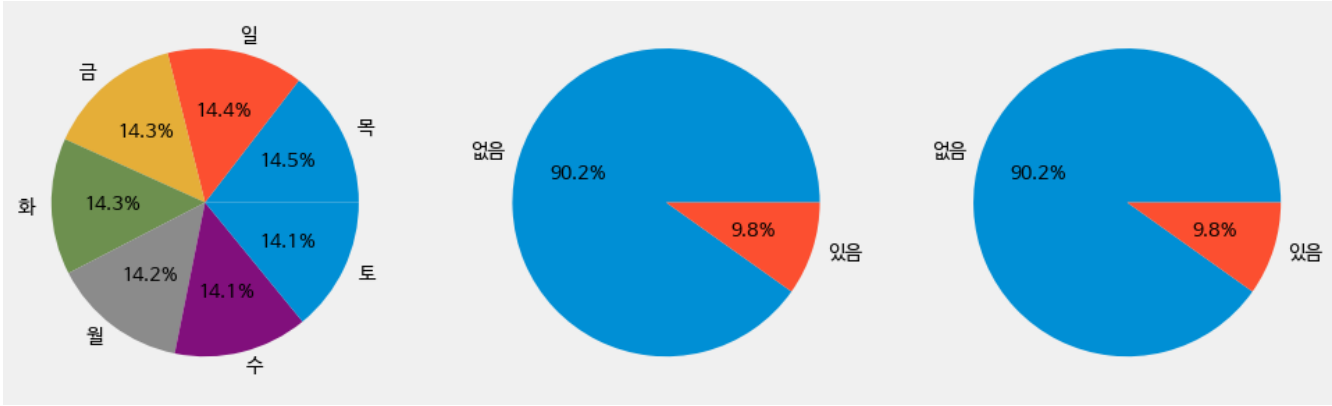


✓ Object 타입 데이터 분석 #1

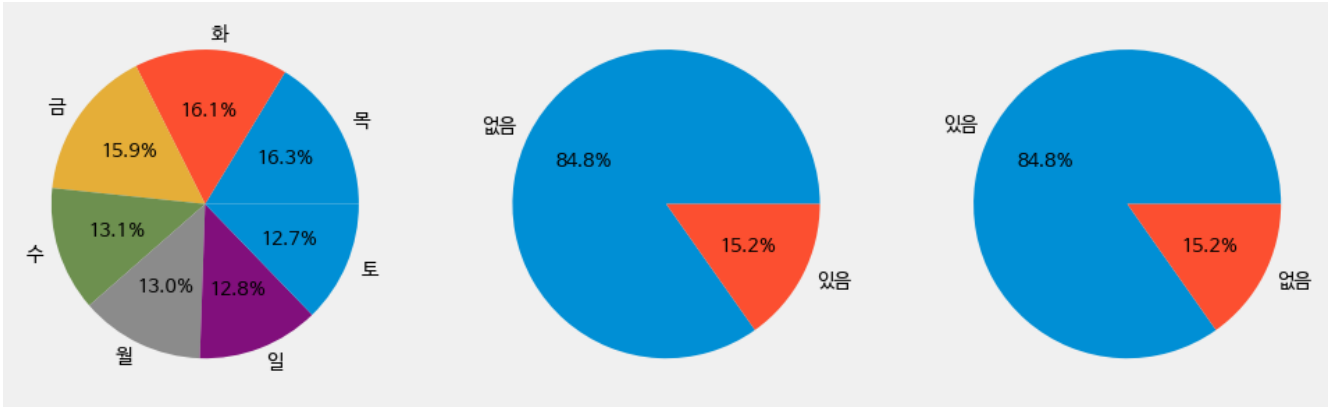
```
object_col.nunique()

id 4701217
day_of_week 7
road_name 61
start_node_name 487
start_turn_restricted 2
end_node_name 487
end_turn_restricted 2
```

```
plt.rc('font',family='NanumBarunGothic')
f = plt.figure(figsize=(15,10))
ax1, ax2, ax3 = f.subplots(1,3)

plt.style.use('fivethirtyeight')
ax1.pie(train['day_of_week'].value_counts(
),labels = train['day_of_week'].unique(),a
utopct="%.1f%%")
ax2.pie(train['start_turn_restricted'].val
ue_counts(),labels = train['start_turn_res
tricted'].unique(),autopct="%.1f%%")
ax3.pie(train['end_turn_restricted'].value
_counts(),labels = train['end_turn_restric
ted'].unique(),autopct="%.1f%%")
```

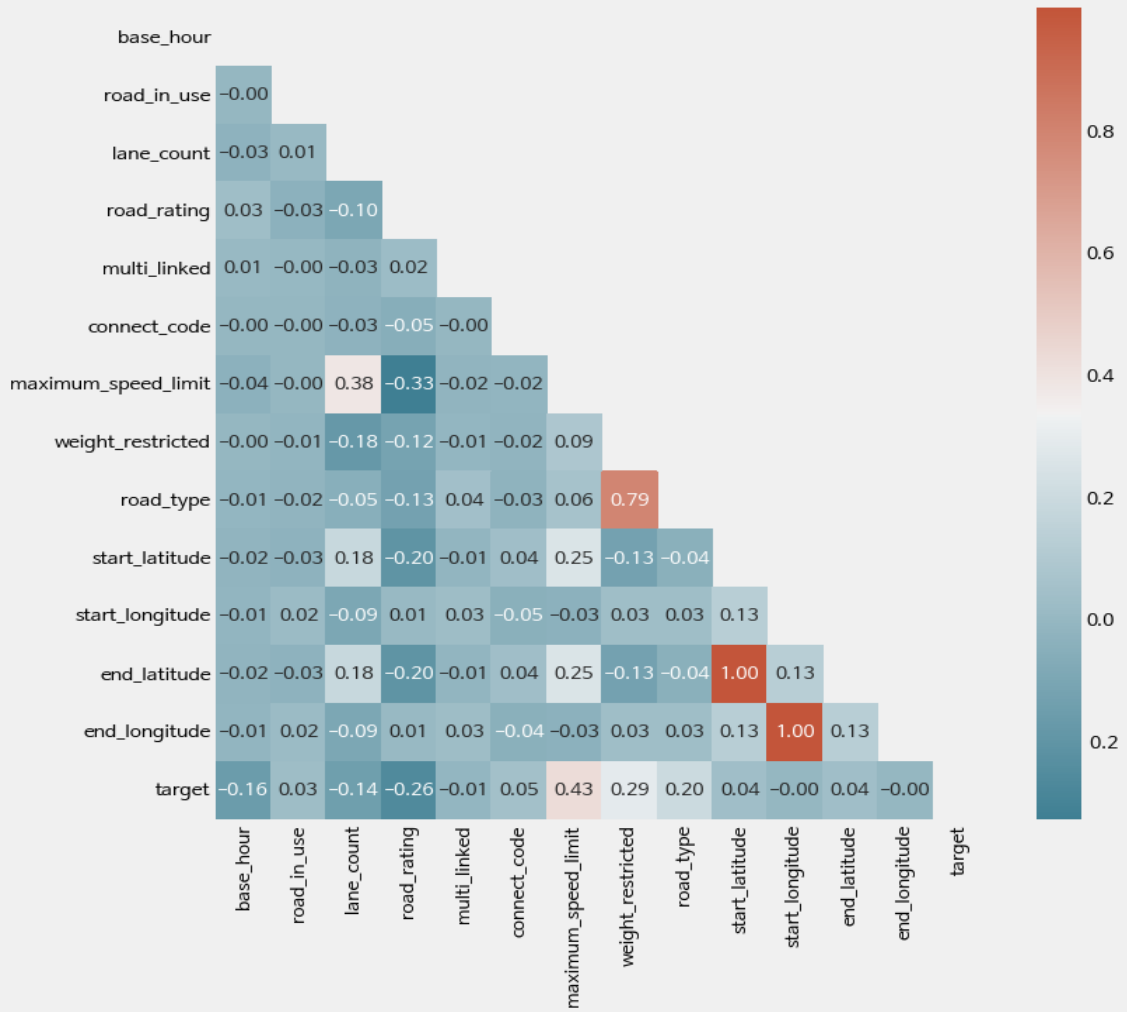
Train 데이터



Test 데이터



✓ Train 데이터 전체 상관관계 분석




```
corr = train.corr()

mask = np.triu(np.ones_like(corr,dtype=bool))
f, ax = plt.subplots(figsize=(11,11))
cmap = sns.diverging_palette(220, 20, as_cmap=True)
sns.heatmap(corr,cmap=cmap,mask=mask,annot=True,fmt=".2f")
```

- 산점도 분석에서 추론했듯이, 시작과 종점의 위도와 경도는 동일하여 start과 end 데이터를 하나로 통일해서 데이터의 중복을 없애야 함

- 그 외에 변수간에 상관관계가 높은 피처는 다음과 같음
  - road_type & weight_restricted (0.79)
  - Maximum speed limit & target (0.43)
  - weight_restricted & target (0.29)
  - road_rating & target

감사합니다