대구광역시 교통사고 위험지역 군집분석 및 분류 모델 개발

선문대학교 K-D 프로그램

팀장 : 김대양

팀원 : 태정수

팀원 : 서동윤

목차

1 데이터 분석

주제 사고위험도(ECLO)를 토대로 군집분석

ECLO (인명피해 심각도)

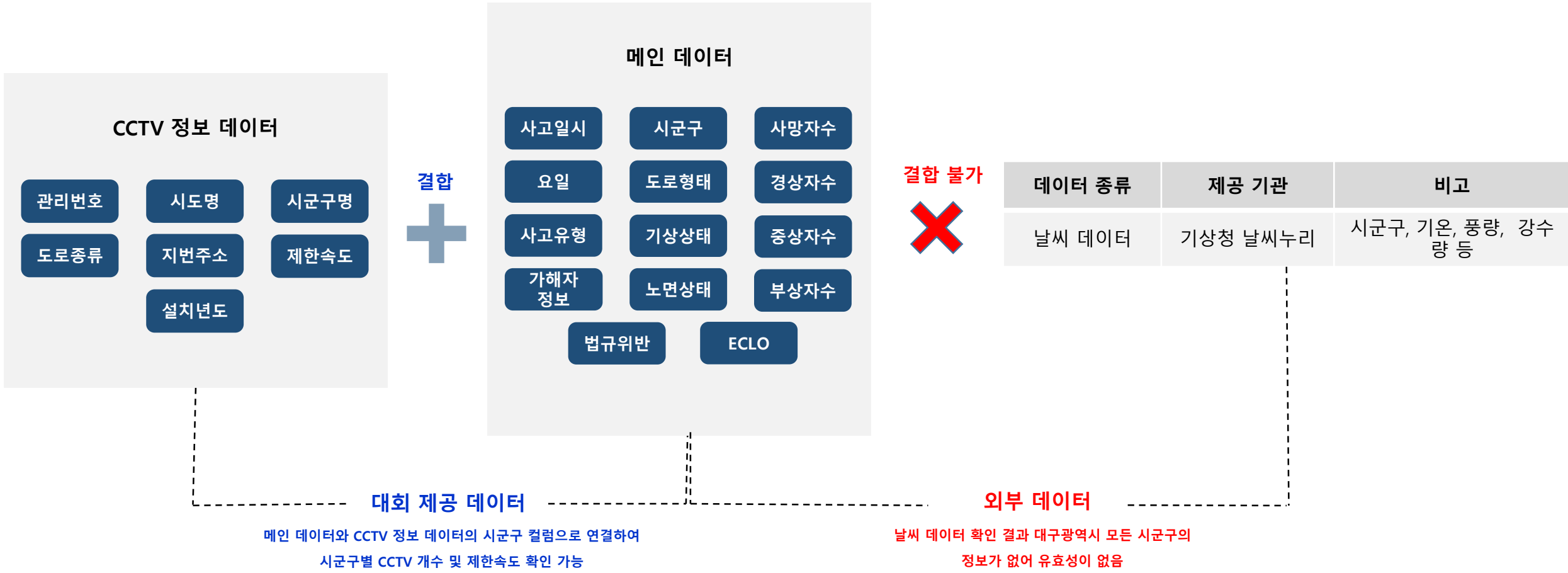
ECLO = 사망자수 * 10 + 중상자수 * 5 + 경상자수 * 3 + 부상자수 * 1

군집분석

ECLO 바탕으로 군집분석 진행

목표 특성을 분석하여 모형 개발 및 사고 위험 요인 탐색

1 데이터 분석


CCTV 정보 데이터
관리번호
시도명
시군구명
도로종류
지번주소
제한속도
설치년도
결합
메인 데이터
사고일시
시군구
사망자수
요일
도로형태
경상자수
사고유형
기상상태
중상자수
가해자 정보
노면상태
부상자수
법규위반
ECLO
결합 불가
데이터 종류
제공 기관
비고
날씨 데이터
기상청 날씨누리
시군구, 기온, 풍량, 강수량 등
대회 제공 데이터
메인 데이터와 CCTV 정보 데이터의 시군구 컬럼으로 연결하여
시군구별 CCTV 개수 및 제한속도 확인 가능
외부 데이터
날씨 데이터 확인 결과 대구광역시 모든 시군구의
정보가 없어 유효성이 없음

1 데이터 분석

1) 메인 데이터(train) 분석

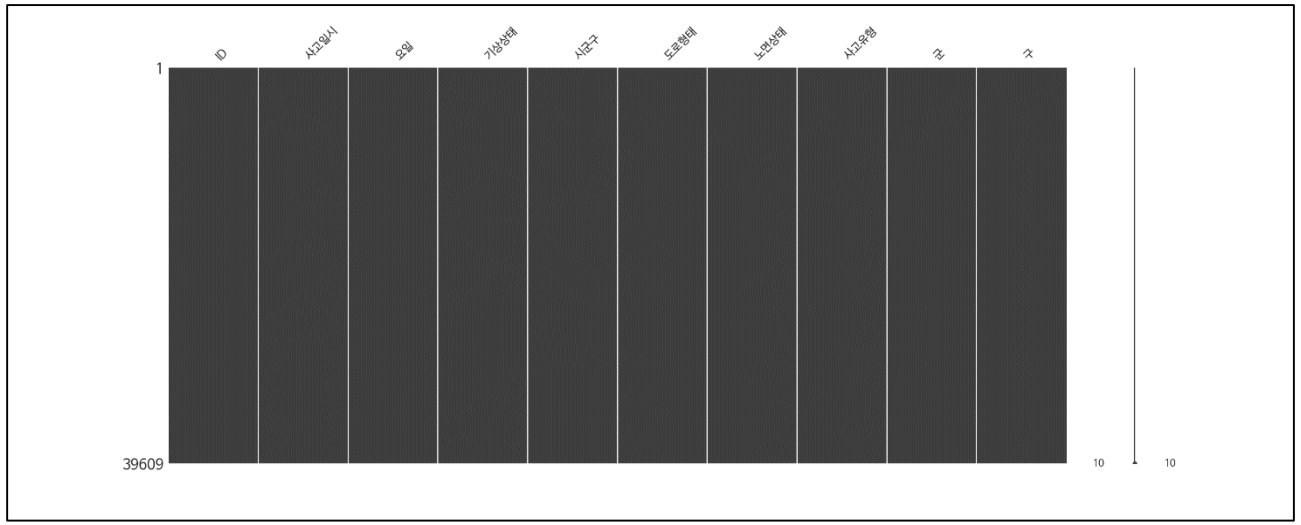
```
# information

train.info()
```

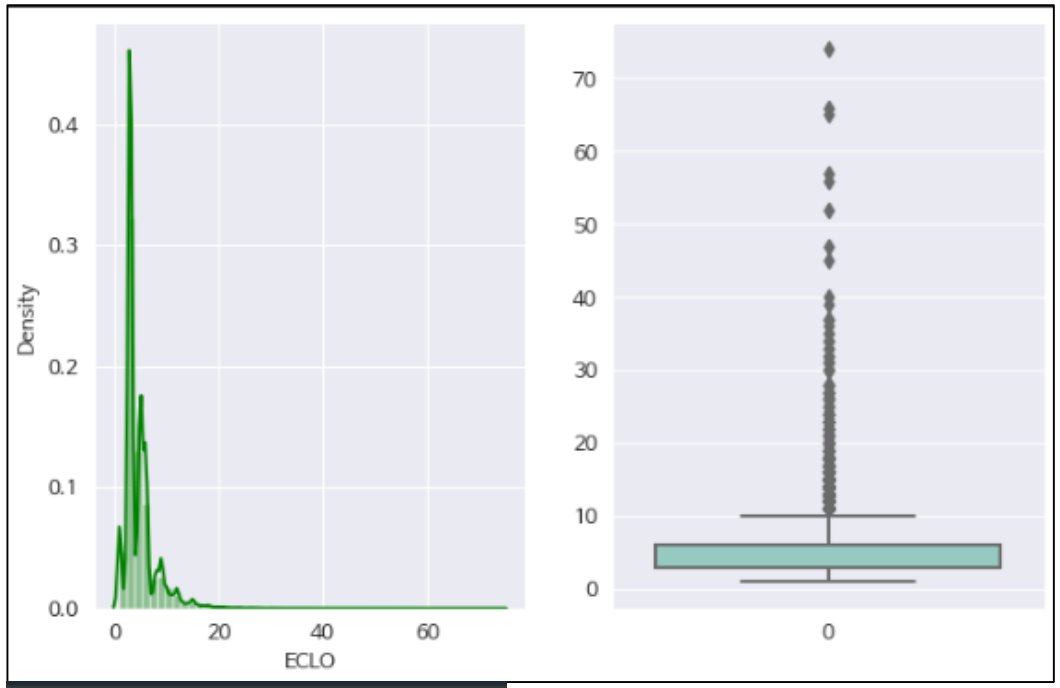
✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39609 entries, 0 to 39608
Data columns (total 23 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   ID           39609 non-null  object
 1   사고일시         39609 non-null  object
 2   요일           39609 non-null  object
 3   기상상태         39609 non-null  object
 4   시군구          39609 non-null  object
 5   도로형태         39609 non-null  object
 6   노면상태         39609 non-null  object
 7   사고유형         39609 non-null  object
 8   사고유형 - 세부분류  39609 non-null  object
 9   법규위반         39609 non-null  object
 10  가해운전자 차종     39609 non-null  object
 11  가해운전자 성별     39609 non-null  object
 12  가해운전자 연령     39609 non-null  object
 13  가해운전자 상해정도   39609 non-null  object
 14  피해운전자 차종     38618 non-null  object
 15  피해운전자 성별     38618 non-null  object
 16  피해운전자 연령     38618 non-null  object
 17  피해운전자 상해정도   38618 non-null  object
 18  사망자수         39609 non-null  int64
 19  중상자수         39609 non-null  int64
 20  경상자수         39609 non-null  int64
 21  부상자수         39609 non-null  int64
 22  ECLO         39609 non-null  int64
dtypes: int64(5), object(18)
memory usage: 7.0+ MB
```

1 데이터 분석

1.1) 메인 데이터(train) 분석 결측치 및 이상치 확인

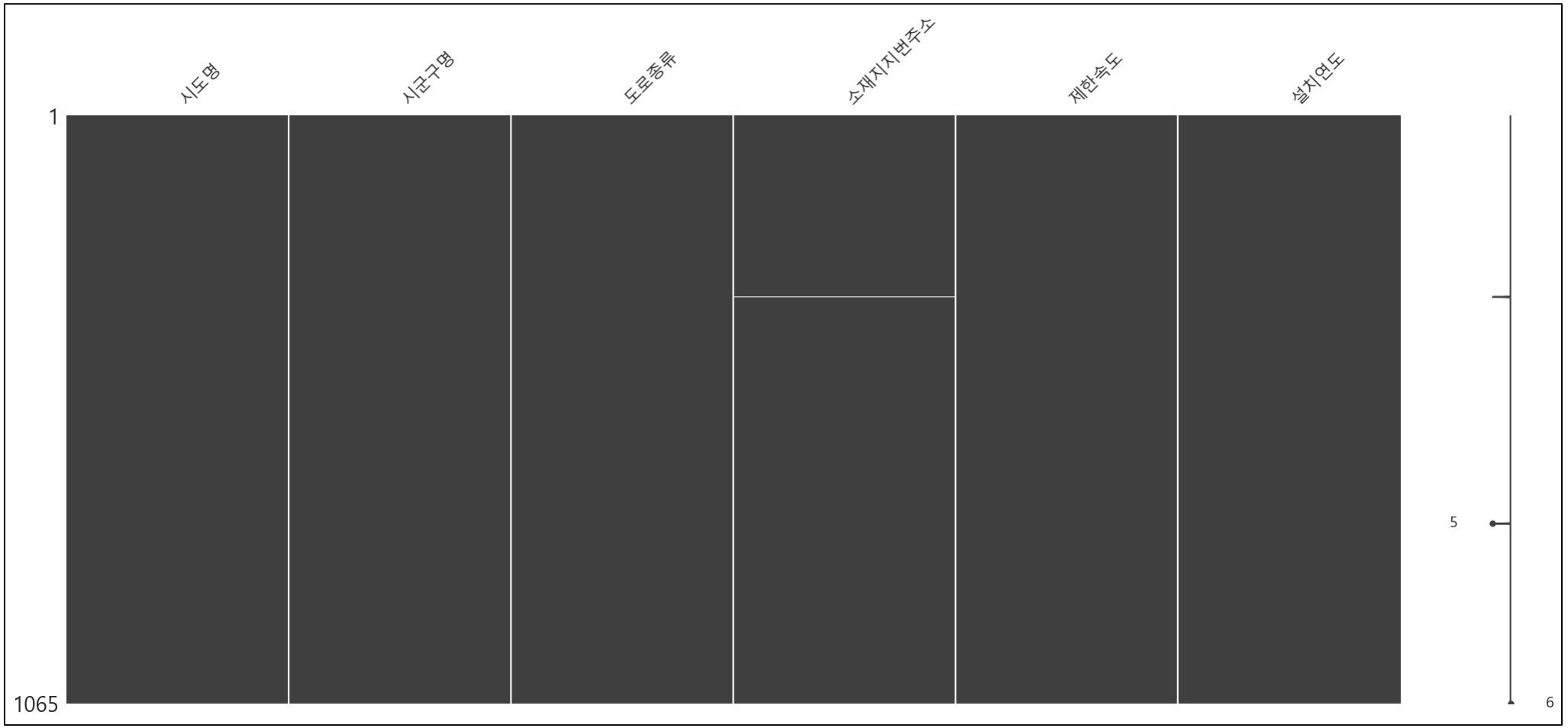




이상치 데이터의 비율 : 5.36%

1 데이터 분석

1.2) CCTV 데이터 분석 결측치 확인



2 데이터 결합 및 파생변수 생성

1) 공간적 데이터 파생변수 생성

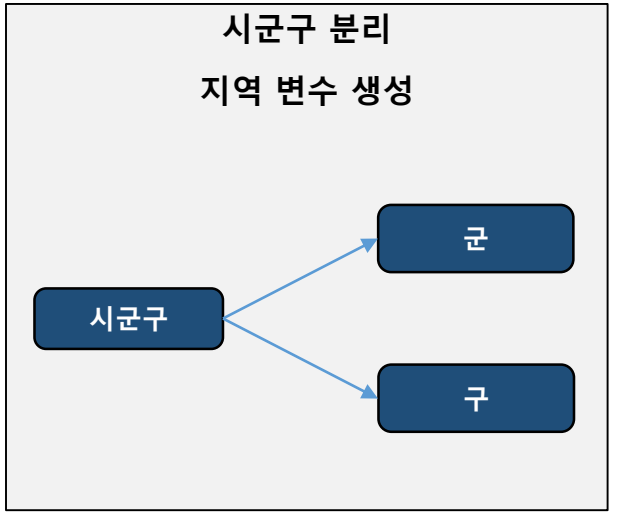
```
# 시군구 분리
# 시 , 군 , 구로 바꾼다
# 시의 경우 대구광역시밖에 없기에 삭제

gun = []
for i in range(len(train)):
    gun.append(train['시군구'].iloc[i].split()[1])

gu = []
for i in range(len(train)):
    gu.append(train['시군구'].iloc[i].split()[2])

train['군'] = gun
train['구'] = gu

gun = []
for i in range(len(test)):
    gun.append(test['시군구'].iloc[i].split()[1])

gu = []
for i in range(len(test)):
    gu.append(test['시군구'].iloc[i].split()[2])

test['군'] = gun
test['구'] = gu
```

```
# train을 test와 연동

target = train['ECLO']
train = train[test.columns]
```

```
# information

train.info()
```

✓ 0.0s

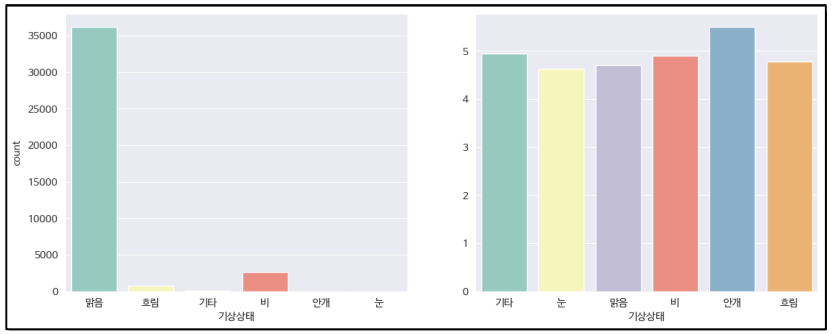
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 39609 entries, 0 to 39608
Data columns (total 10 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      39609 non-null  object
 1   사고일시    39609 non-null  datetime64[ns]
 2   요일      39609 non-null  object
 3   기상상태    39609 non-null  object
 4   시군구     39609 non-null  object
 5   도로형태    39609 non-null  object
 6   노면상태    39609 non-null  object
 7   사고유형    39609 non-null  object
 8   군       39609 non-null  object
 9   구       39609 non-null  object
dtypes: datetime64[ns](1), object(9)
memory usage: 3.0+ MB
```



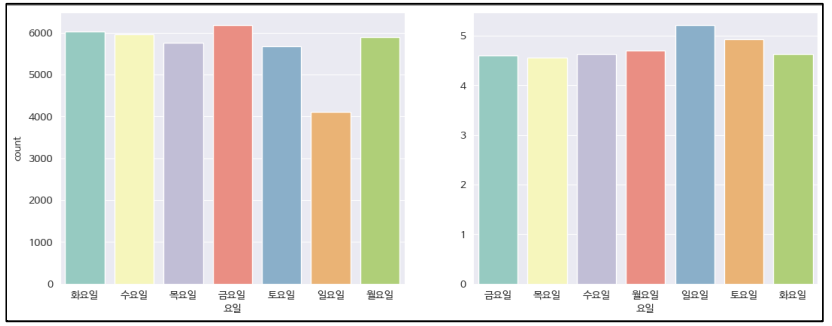
3 EDA

1) 각 컬럼 별 ECLO 연관성 확인

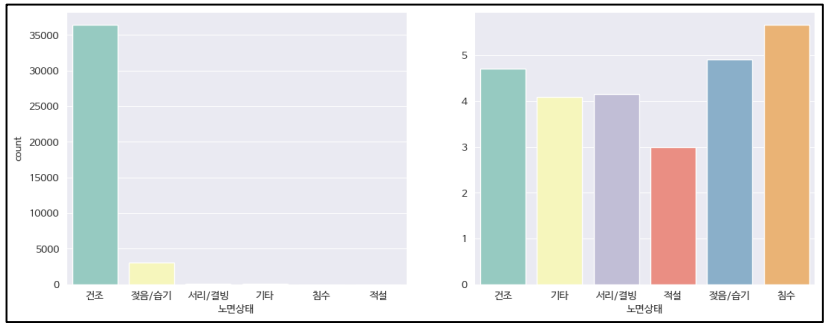
기상상태별 ECLO 평균



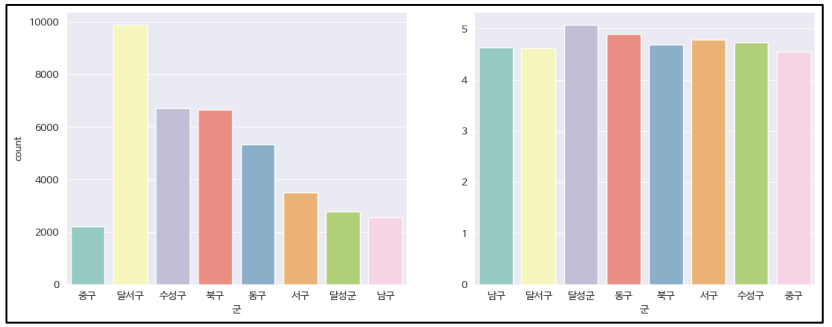
요일별 ECLO 평균



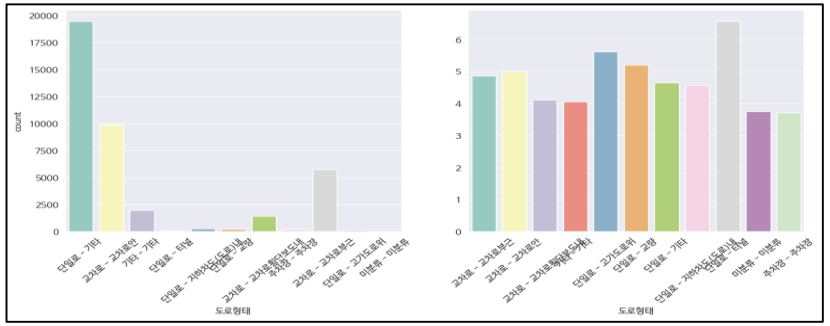
노면상태별 ECLO 평균



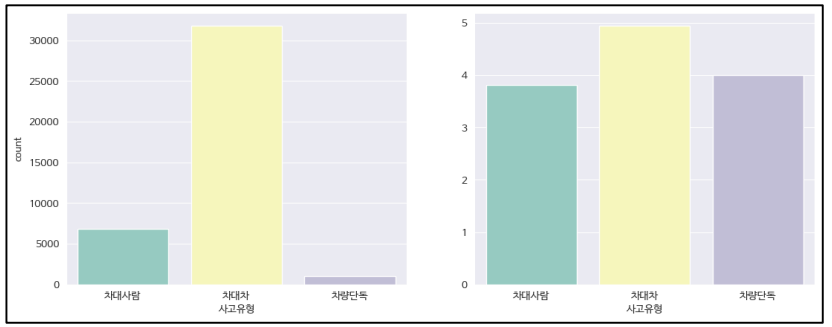
시군별 ECLO 평균



도로형태별 ECLO 평균

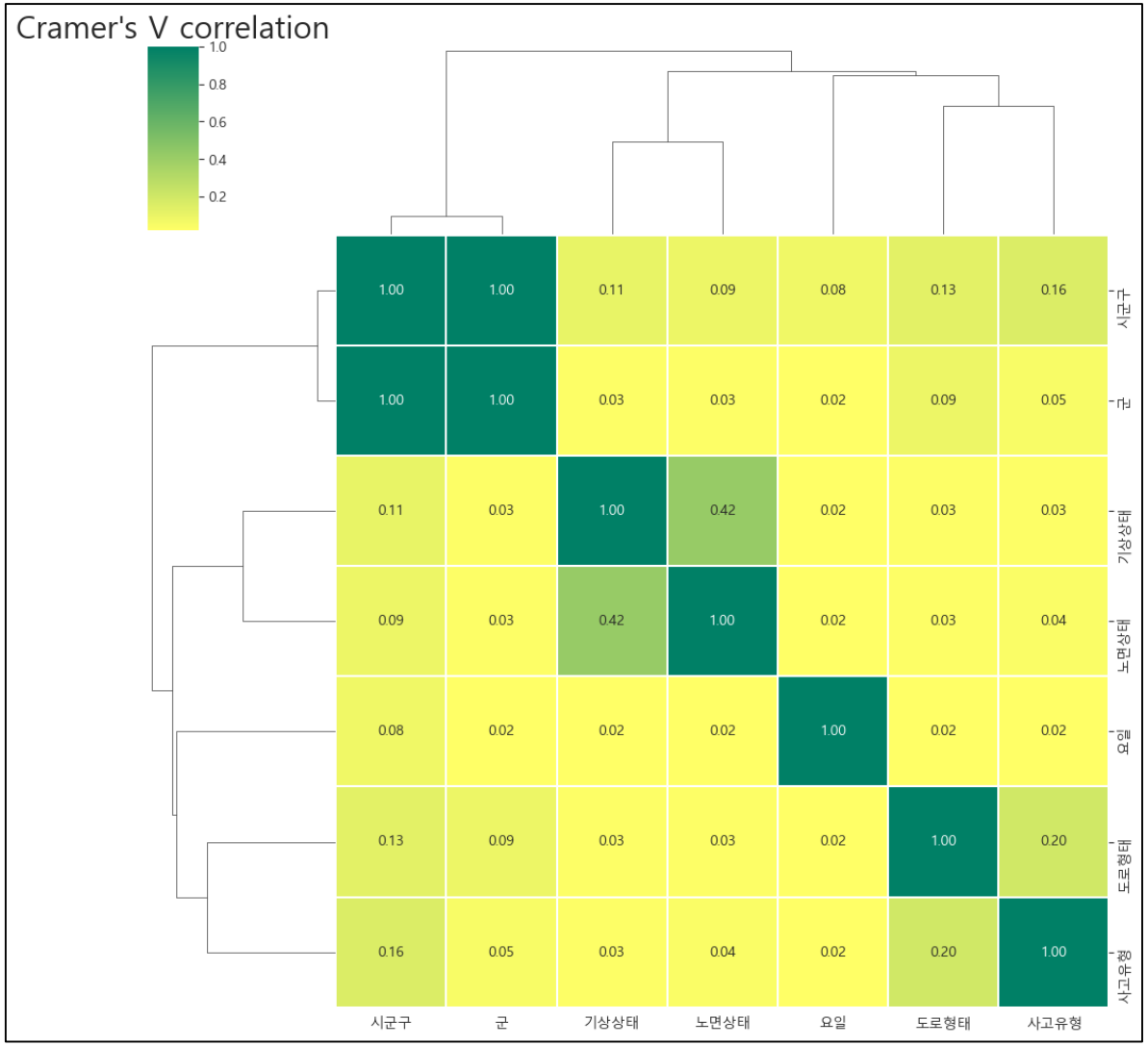


사고유형별 ECLO 평균



3 EDA

2) 각 항목별 상관 분석 결과



Cramer’s V

3개 이상의 범주형 변수간의 상관관계를 파악할 때 사용 하는 방법

3 EDA

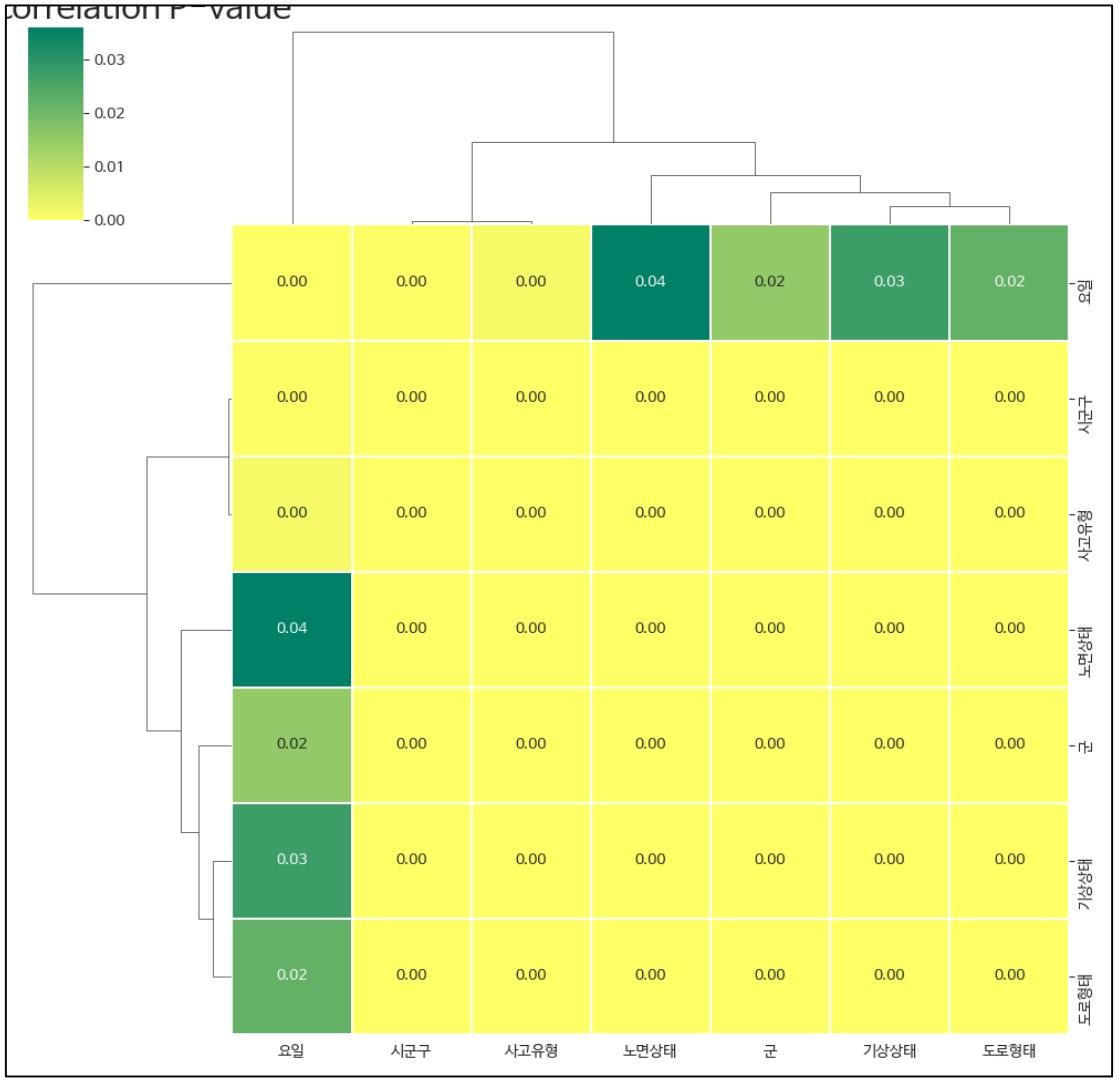
3) 각 항목별 상관 분석 결과

P-value < 0.05

믿을 만한 상관계수

P-value > 0.05

상관계수가 틀릴 가능성도 있다



상관관계의 P-value

3 EDA

4) 카이제곱 독립성 검정

P-value < 0.05

변수 간 유의한 관련성이 있다고 판단

P-value > 0.05

변수 간 유의한 관련성이 없다고 판단

```
p_vals = pd.Series(
    p_vals ,
    index = train.columns[2:8]
)

index = p_vals.index

for i in range(len(p_vals)):

    print(p_vals[i], index[i])
    if p_vals[i] < 0.05:
        print(f"{index[i]} : 관련이 있다.")
        #print(p_vals[i])
        print("------------------------")
    else:
        print(f"{index[i]} : 독립이다.")
        #print(p_vals[i])
        print("------------------------")
```

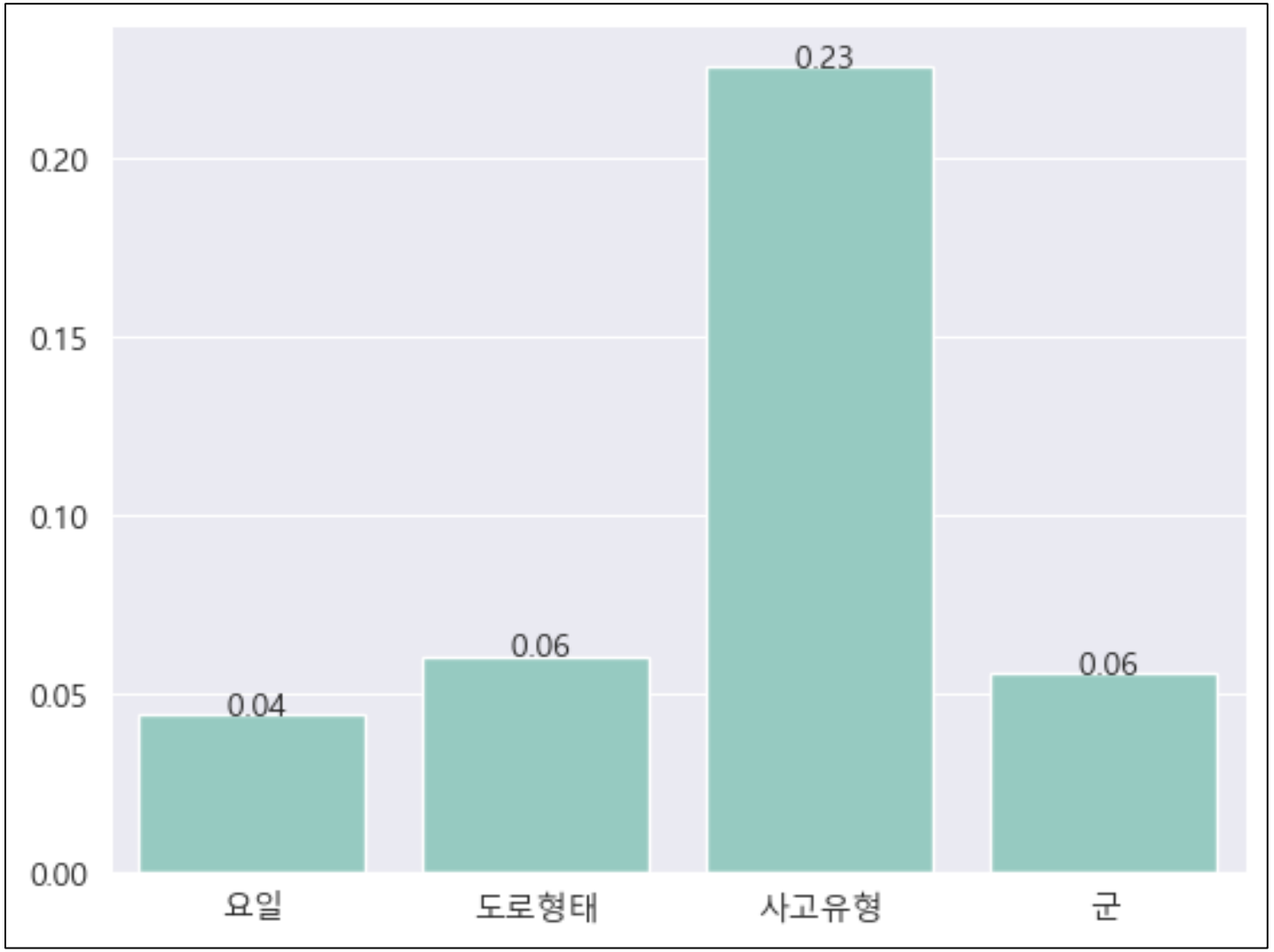
✓ 0.0s

```
2.7889563820901824e-12 요일
요일 : 관련이 있다.
------------------------
0.9945128888640956 기상상태
기상상태 : 독립이다.
------------------------
0.0 시군구
시군구 : 관련이 있다.
------------------------
2.4977920985114386e-105 도로형태
도로형태 : 관련이 있다.
------------------------
0.994682480770459 노면상태
노면상태 : 독립이다.
------------------------
0.0 사고유형
사고유형 : 관련이 있다.
------------------------
```

3 EDA

5) 종속변수와의 상관관계측정

변수간 관련이 있는 데이터를 종속변수와의 상관관계 측정



3 EDA

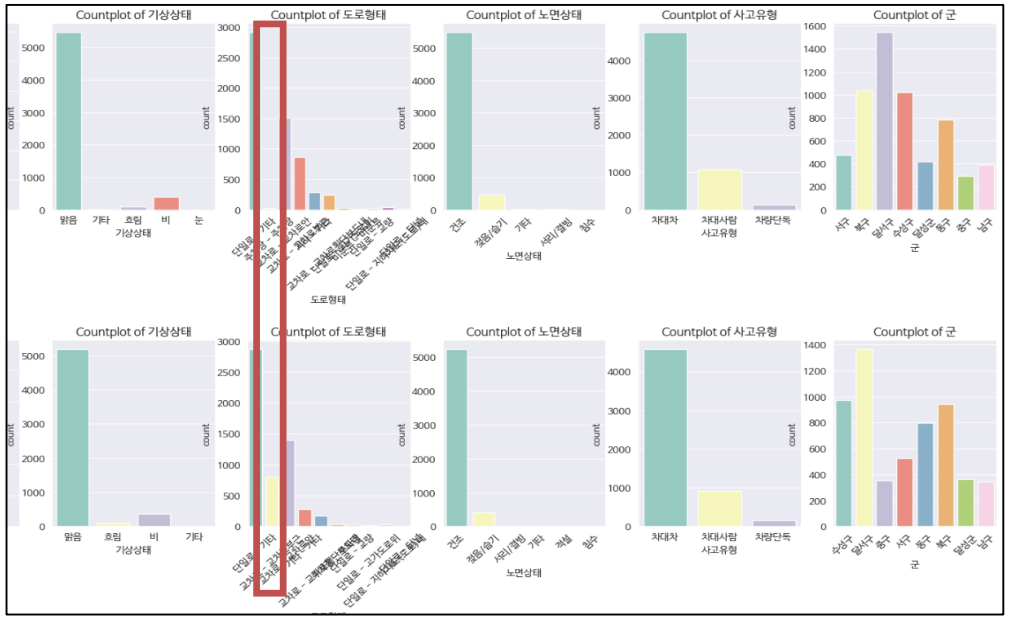
6) 요일별 사고유형 분석

```python
# 주말과 주중에는 통계적으로 의미있는 차이가 있는가

data_weekday = train[(train['요일'] != '토요일') & (train['요일'] != '일요일')]['ECLO']
data_weekend = train[(train['요일'] == '토요일') | (train['요일'] == '일요일')]['ECLO']

def test(var1 , var2):
    _ , p_value_var , = scipy.stats.levene(var1 , var2)

    if p_value_var > 0.05:
        _ , p_value = scipy.stats.ttest_ind(var1 , var2 , equal_var = True)
    else:
        _ , p_value = scipy.stats.ttest_ind(var1 , var2 , equal_var = False)

    if p_value < 0.05:
        print("주중과 주말의 종속변수의 평균 차이는 통계적으로 의미있다.")
    else:
        print("아니다")

test(data_weekday , data_weekend)
```

```
주중과 주말의 종속변수의 평균 차이는 통계적으로 의미있다.
```

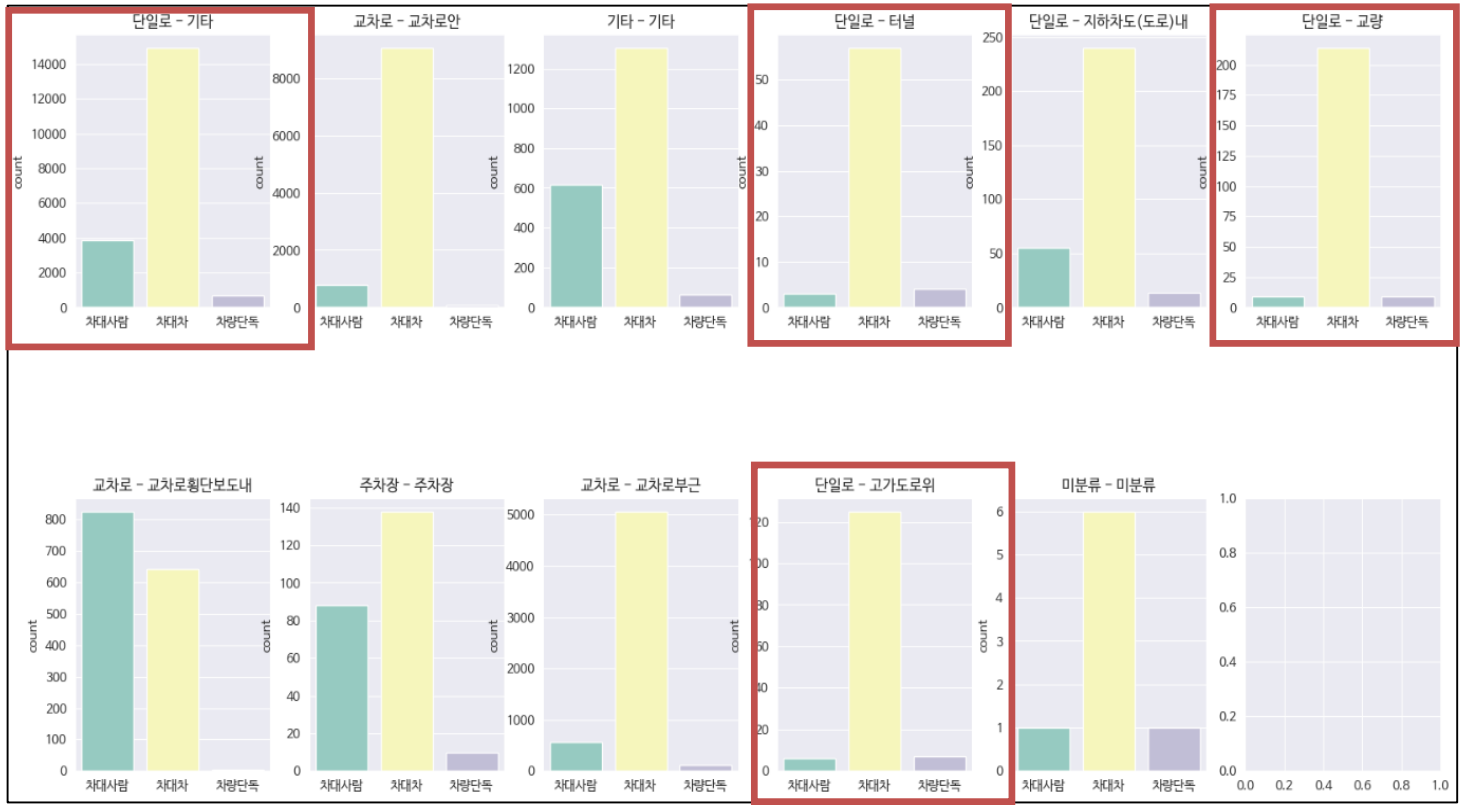
주말과 주중에는 통계적 의미가 있는지 검토

```python
# 수요일 <-> 토요일 차이는 무엇이 있나

data1 = train[train['요일'] == '수요일']
data2 = train[train['요일'] == '토요일']

fig , ax = plt.subplots(2 , 6 , figsize = (20 , 10))
plt.subplots_adjust(hspace = 0.7)

cols = ['요일' , '기상상태' , '도로형태' , '노면상태' , '사고유형' , '군']

# data1 시각화 (수요일)
for i , col in enumerate(cols):

    if col == '도로형태' or col == '노면상태' or col == '군':
        sns.countplot(data = data1 , x = col , ax = ax[0][i]) ,
        ax[0][i].set_title(f"Countplot of {col}")
        ax[0][i].set_xticklabels(ax[0][i].get_xticklabels() , rotation = 45)
    else:
        sns.countplot(data = data1 , x = col , ax = ax[0][i]) ,
        ax[0][i].set_title(f"Countplot of {col}")

# data2 시각화 (토요일)
for i , col in enumerate(cols):

    if col == '도로형태' or col == '노면상태' or col == '군':
        sns.countplot(data = data2 , x = col , ax = ax[1][i]) ,
        ax[1][i].set_title(f"Countplot of {col}")
        ax[1][i].set_xticklabels(ax[1][i].get_xticklabels() , rotation = 45)
    else:
        sns.countplot(data = data2 , x = col , ax = ax[1][i]) ,
        ax[1][i].set_title(f"Countplot of {col}")
```



3 EDA

7) 도로형태별 사고건수 분석



3 EDA

8) Tukey Outlier

```
# tukey 방식으로 outlier 색출

q1 = np.percentile(train['ECLO'] , 25)
q3 = np.percentile(train['ECLO'] , 75)

IQR = q3 - q1

upper_fence = q3 + 1.5 * IQR
lower_fence = q1 - 1.5 * IQR

outlier_data = train[(train['ECLO'] < lower_fence) | (train['ECLO'] > upper_fence)]
```

✓ 0.0s

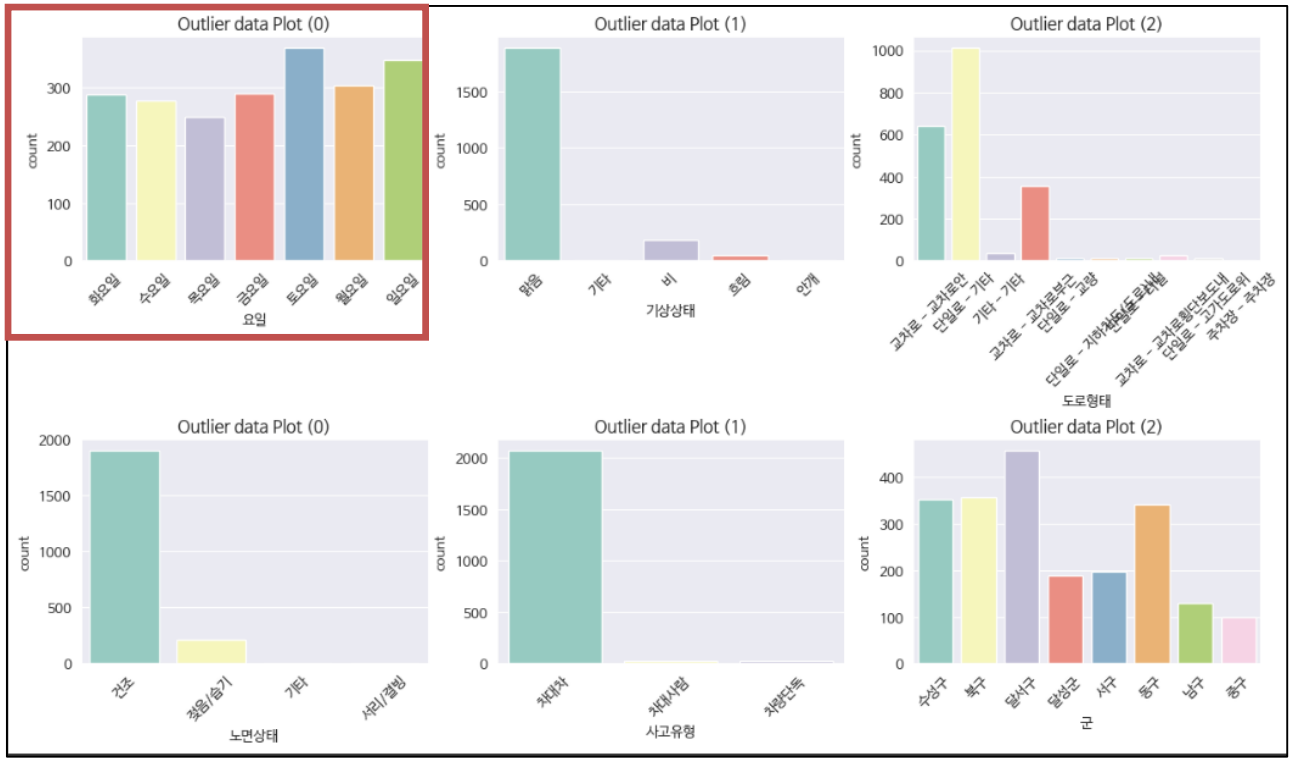
```
# 비율은 어느정도 인가?

print(f"이상치 데이터의 비율 : {round(len(outlier_data) / len(train) * 100 , 2)}%")
```

✓ 0.0s

```
이상치 데이터의 비율 : 5.36%
```

제 1 사분위수 - 1.5 * IQR 보다 작거나 제 3사분위수 + 1.5 * IQR 보다 큰 값은 이상치로 판별

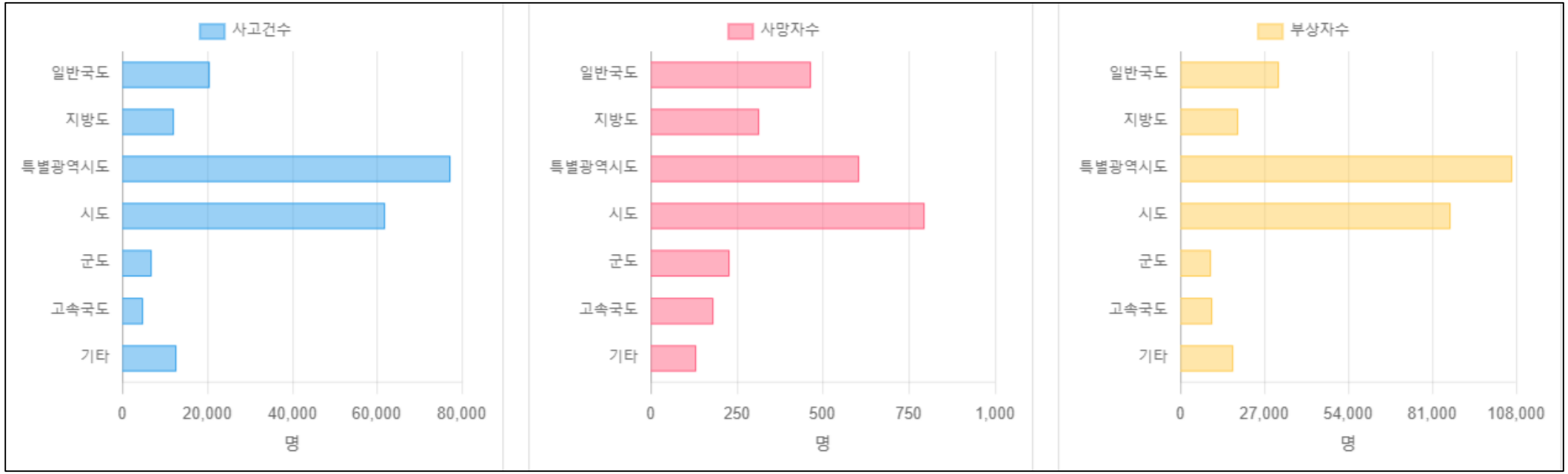
```
cols = ['요일' , '기상상태' , '도로형태' , '노면상태' , '사고유형' , '군']

fig , ax = plt.subplots(2 , 3 , figsize = (15 , 8))
plt.subplots_adjust(hspace = 0.8)

for i in range(6):
    row = i // 3
    col = i % 3

    sns.countplot(
        data = outlier_data ,
        x = cols[i] ,
        ax = ax[row][col]
    )

    ax[row][col].set_xticklabels(ax[row][col].get_xticklabels() , rotation = 45)
    ax[row][col].set_title(f"Outlier data Plot ({col})")
```



이상치 대체 → 최빈값

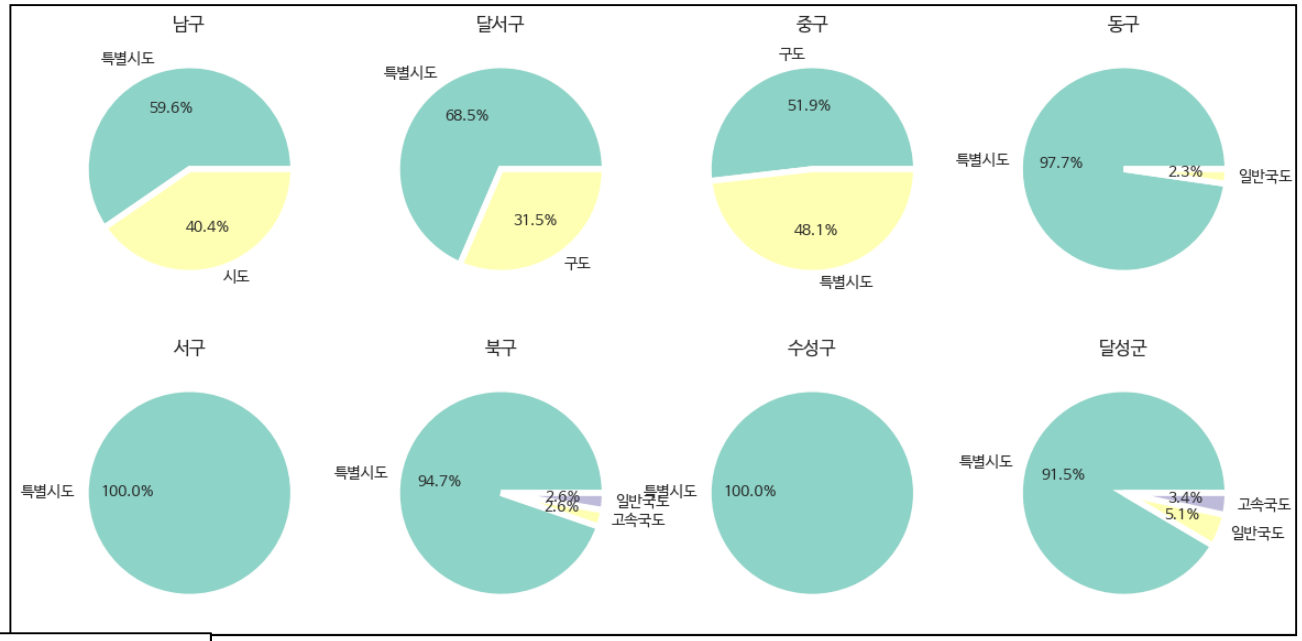
3 EDA

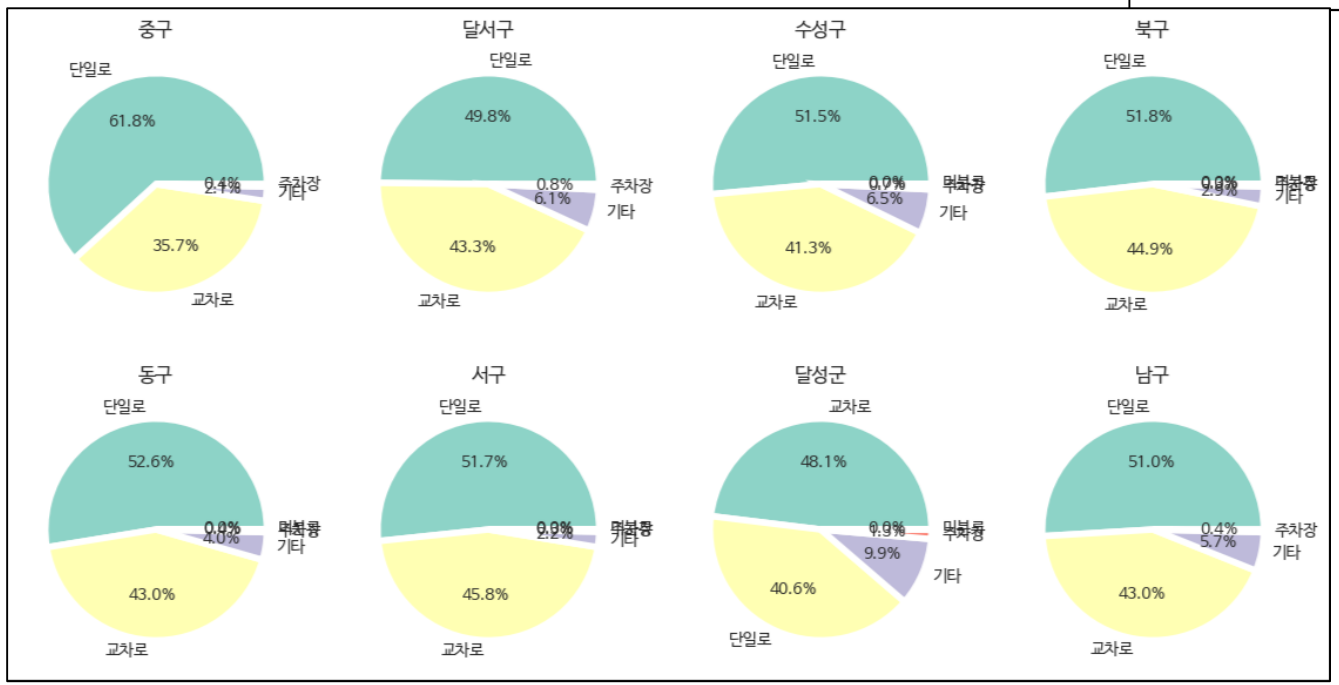
9.1) 도로별 사고건수 분석



3 EDA

9.2) 도로별 사고건수 분석





3 EDA

10) CCTV 데이터분석

2019년 ~ 2021년 사이 데이터 분석

3 EDA

11) train + cctv 데이터 결합

```
print(f"CCTV 시군구 데이터 : {len(cctv['시군구'].unique())}")
print(f"TRAIN 시군구 데이터 : {len(train['시군구'].unique())}")
print(f"차이 : {len(train['시군구'].unique()) - len(cctv['시군구'].unique())}")
```

✓ 0.0s

```
CCTV 시군구 데이터 : 116
TRAIN 시군구 데이터 : 199
차이 : 83
```

시군구 기준으로 병합

```
merged_data = pd.merge(train , cctv , on = '시군구' , how = 'left')
```

✓ 0.0s

```
# NA 값 비율

for col in merged_data.columns:
    print(f"{col} NA Percentage : {merged_data[col].isna().sum() / len(merged_data) * 100}%")
```

✓ 0.0s

```
ID NA Percentage : 0.0%
사고일시 NA Percentage : 0.0%
요일 NA Percentage : 0.0%
기상상태 NA Percentage : 0.0%
시군구 NA Percentage : 0.0%
도로형태 NA Percentage : 0.0%
노면상태 NA Percentage : 0.0%
사고유형 NA Percentage : 0.0%
군 NA Percentage : 0.0%
구 NA Percentage : 0.0%
ECLO NA Percentage : 0.0%
도로형태(대분류) NA Percentage : 0.0%
시도명 NA Percentage : 1.752734064475299%
시군구명 NA Percentage : 1.752734064475299%
도로종류 NA Percentage : 1.752734064475299%
소재지지번주소 NA Percentage : 1.752734064475299%
```

결측치 데이터

3 EDA

12) 변수 선택

ID -> 기본키와 같은 데이터이기 때문에 학습에 의미는 없음.

사고일시 -> 시간데이터이기에 모델 학습에는 사용하지 않는다.

기상상태 -> 상관분석에서 노면상태와의 상관성을 고려하여 도로의 상태에 더 직접적인 데이터를가지고 있는 노면상태를 채택하여 , 기상상태는 제거하기로 함.

시군구 -> 데이터의 결합에 쓴 후에는 , 라벨의 정답 데이터가 되기 때문에 뺀다.

구 -> 시군구와 마찬가지이다.

군 -> 시군구와 마찬가지이다.

도로형태(대분류) -> 조금 더 상세한 데이터인 도로형태가 존재하기에 삭제.

시도명 -> 시군구와 마찬가지이다.

시군구명 -> 시군구와 마찬가지이다.

소재지지번주소 -> 시군구와 마찬가지이다.

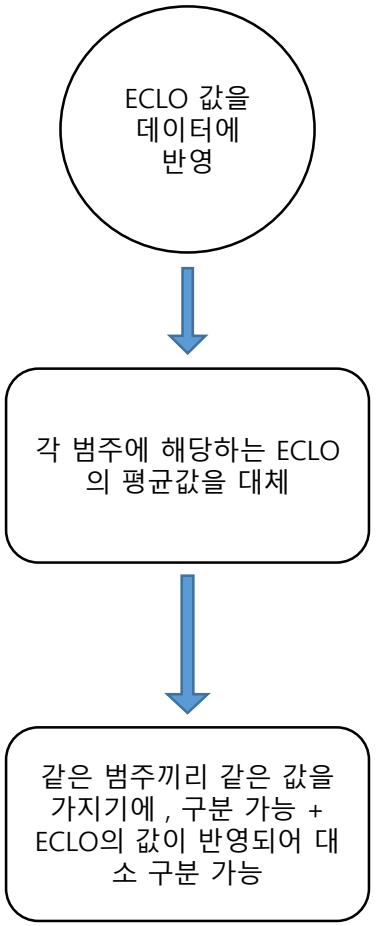
3 EDA

13) 이상치 처리

```
# 이상치 처리

# 상한 대체
def replace_outlier(train , col):
    q1 = np.percentile(train[col] , 25)
    q3 = np.percentile(train[col] , 75)
    IQR = q3 - q1
    lower_fecne = q1 - 1.5 * IQR
    upper_fence = q3 + 1.5 * IQR

    return train[col].apply(lambda x : train[col].mode().iloc[0] if x < lower_fence or x > upper_fence else x)

train['ECLO'] = replace_outlier(train , 'ECLO')
```

3 EDA

14) 수치형 데이터 변경



범주형 변수의 각 항목별 변수의 평균값으로 대체

범주형 데이터를 수치형으로 변환

```
from category_encoders.target_encoder import TargetEncoder

categorical_features = ['요일', '기상상태', '도로형태', '노면상태', '사고유형', '군', '도로종류', '제한속도']

for feature in categorical_features:
    TR = TargetEncoder(cols = [feature])
    merged_data[feature] = TR.fit_transform(merged_data[feature], target)
```

```
merged_data['구'] = merged_data['시군구'].str.split().str[1]

merged_data
```

✓ 0.3s

	요일	기상상태	시군구	도로형태	노면상태	사고유형	군	도로종류	제한속도	구
0	4.554885	4.615564	대구광역시 중구 대신동	4.554166	4.617512	3.757712	4.576527	4.576497	4.565163	중구
1	4.554885	4.615564	대구광역시 중구 대신동	4.554166	4.617512	3.757712	4.576527	4.640177	4.597668	중구
2	4.554885	4.631723	대구광역시 달서구 감삼동	4.554166	4.617512	3.757712	4.562907	4.576497	4.565163	달서구
3	4.554885	4.631723	대구광역시 달서구 감삼동	4.554166	4.617512	3.757712	4.562907	4.640177	4.779189	달서구
4	4.554885	4.631723	대구광역시 달서구 감삼동	4.554166	4.617512	3.757712	4.562907	4.640177	4.659467	달서구
...	...	...	...	...	...	...	...	...	...	...
259133	4.535246	4.615564	대구광역시 서구 비산동	4.560219	4.617512	4.859885	4.643969	4.640177	4.602463	서구
259134	4.535246	4.615564	대구광역시 서구 비산동	4.560219	4.617512	4.859885	4.643969	4.640177	4.602463	서구
259135	4.535246	4.615564	대구광역시 서구 비산동	4.560219	4.617512	4.859885	4.643969	4.640177	4.602463	서구
259136	4.535246	4.615564	대구광역시 서구 비산동	4.560219	4.617512	4.859885	4.643969	4.640177	4.602463	서구
259137	4.535246	4.615564	대구광역시 서구 비산동	4.560219	4.617512	4.859885	4.643969	4.640177	4.597668	서구

254684 rows × 10 columns

3 EDA

15) 수치형 데이터 변경 후 평균값 추출

```
merged_data.drop(['시군구'] , axis = 1 , inplace = True)

merged_data.groupby(['구'])['요일' , '기상상태' , '도로형태' , '노면상태' ,'사고유형' , '군' , '도로종류' , '제한속도'].mean()
merged_data_group = merged_data.groupby(['구'])[['요일', '기상상태', '도로형태', '노면상태', '사고유형', '군', '도로종류', '제한속도']].mean()
```

✓ 0.0s

```
merged_data_group.drop(['군'] , axis = 1 , inplace = True)
```

```
merged_data_group
```

✓ 0.0s

구	요일	기상상태	도로형태	노면상태	사고유형	도로종류	제한속도
남구	4.635854	4.636801	4.634726	4.638663	4.601966	4.605646	4.595892
달서구	4.634883	4.634360	4.614097	4.634130	4.640459	4.620651	4.621577
달성군	4.633634	4.633527	4.632008	4.624920	4.670697	4.670658	4.706926
동구	4.639866	4.634247	4.648083	4.634581	4.643457	4.665961	4.660934
북구	4.632803	4.637319	4.673097	4.639085	4.660574	4.660113	4.643909
서구	4.637186	4.631348	4.669819	4.632438	4.621109	4.641056	4.637007
수성구	4.631261	4.633946	4.619678	4.635098	4.643926	4.649320	4.663341
중구	4.641848	4.634772	4.631149	4.636571	4.588615	4.682514	4.681099

3 EDA

16) 엘보우 메소드 , 실루엣

```
merged_data_group
✓ 0.0s
```

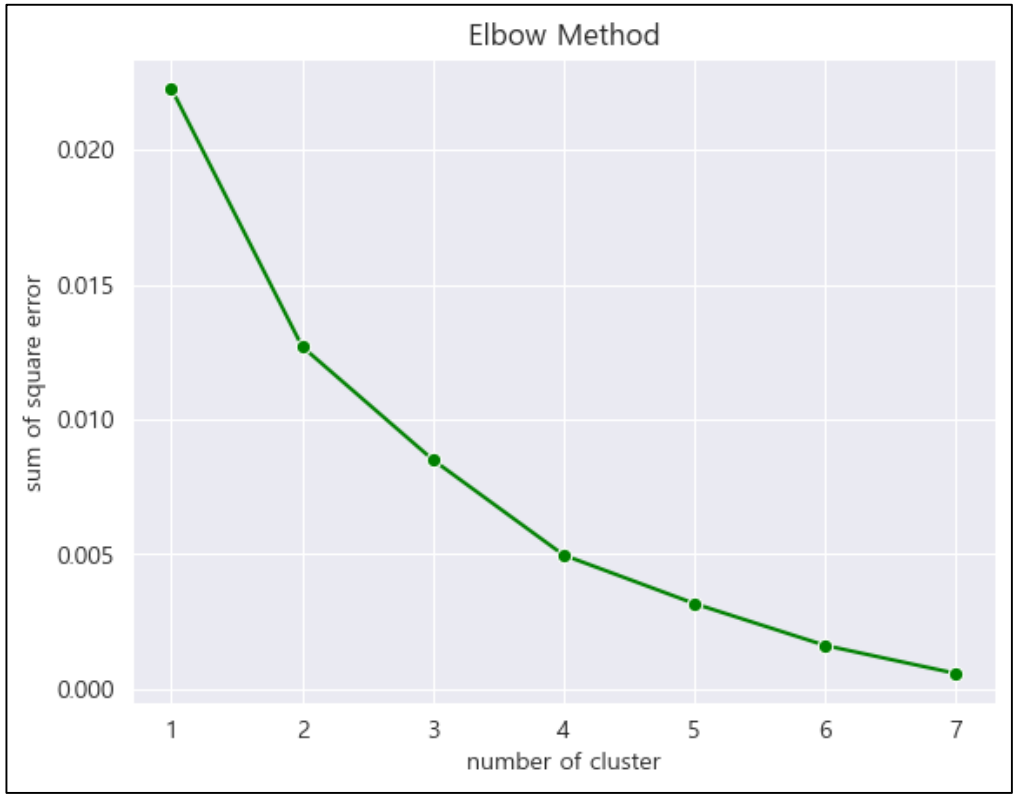
구	요일	기상상태	도로형태	노면상태	사고유형	도로종류	제한속도
남구	4.635854	4.636801	4.634726	4.638663	4.601966	4.605646	4.595892
달서구	4.634883	4.634360	4.614097	4.634130	4.640459	4.620651	4.621577
달성군	4.633634	4.633527	4.632008	4.624920	4.670697	4.670658	4.706926
동구	4.639866	4.634247	4.648083	4.634581	4.643457	4.665961	4.660934
북구	4.632803	4.637319	4.673097	4.639085	4.660574	4.660113	4.643909
서구	4.637186	4.631348	4.669819	4.632438	4.621109	4.641056	4.637007
수성구	4.631261	4.633946	4.619678	4.635098	4.643926	4.649320	4.663341
중구	4.641848	4.634772	4.631149	4.636571	4.588615	4.682514	4.681099

해당 데이터는 엘보우 메소드와 실루엣 메소드를 사용하기 위해 구를 기준으로 group으로 묶어 평균값을 계산

3 EDA

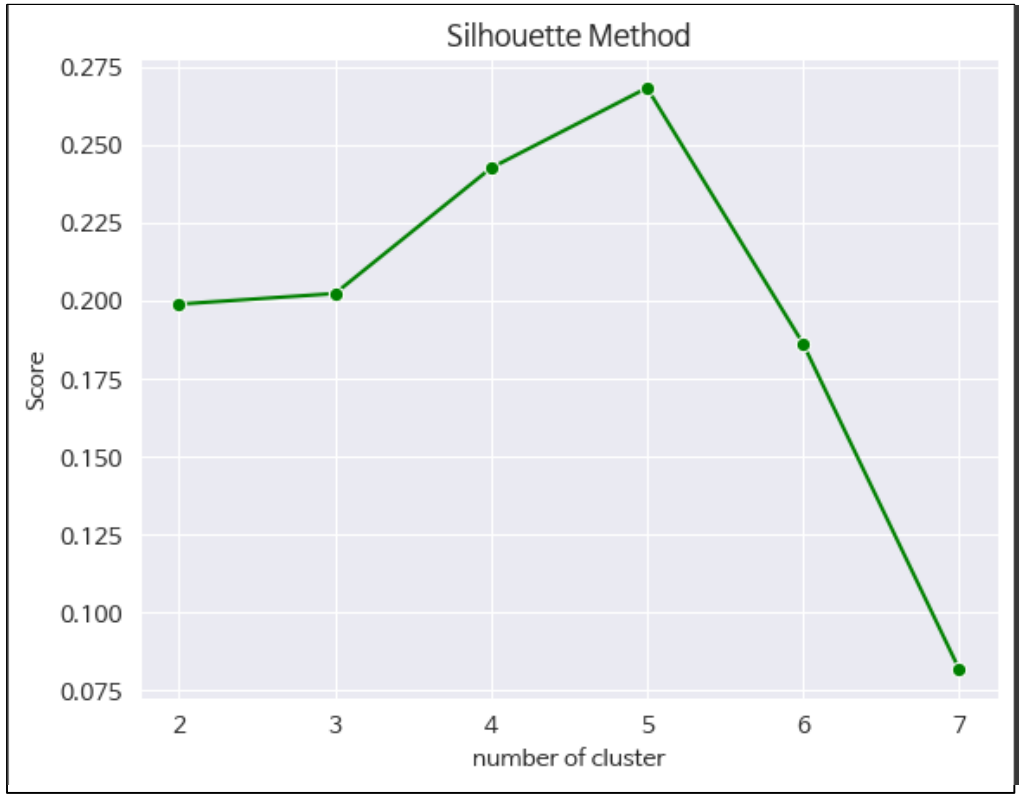
16.2) 엘보우 메소드 , 실루엣

Elbow Method



Y축은 클러스터의 중심으로부터 데이터들이 얼마나 떨어져 있는지에 대한 오차의 제곱이어서 , 이 값이 낮아지는 적절한 부분을 고른다.

Silhouette Method

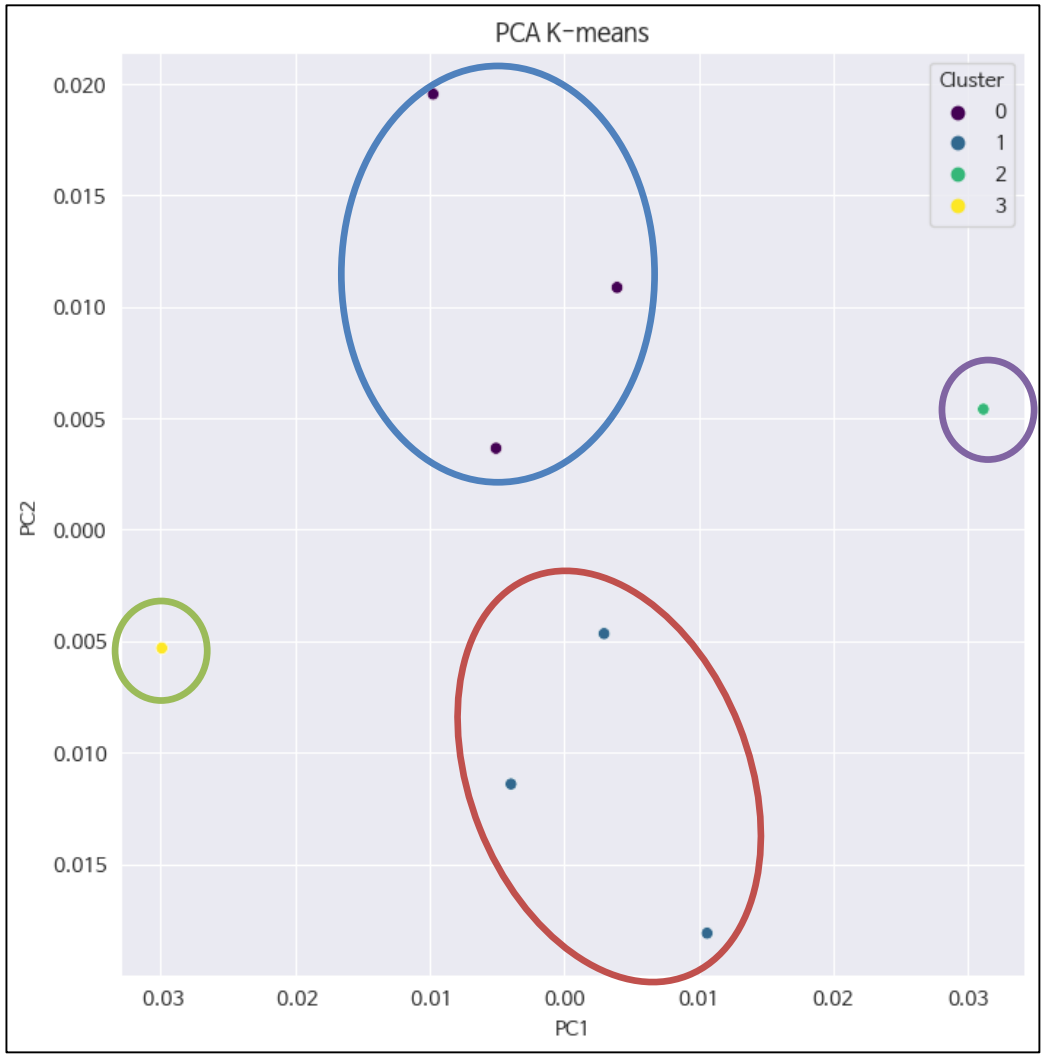


실루엣 스코어를 기준으로 적절한 수를 찾는다 , 데이터가 클러스터의 중점에 잘 모여 있을 수록 스코어가 올라간다.

3 EDA

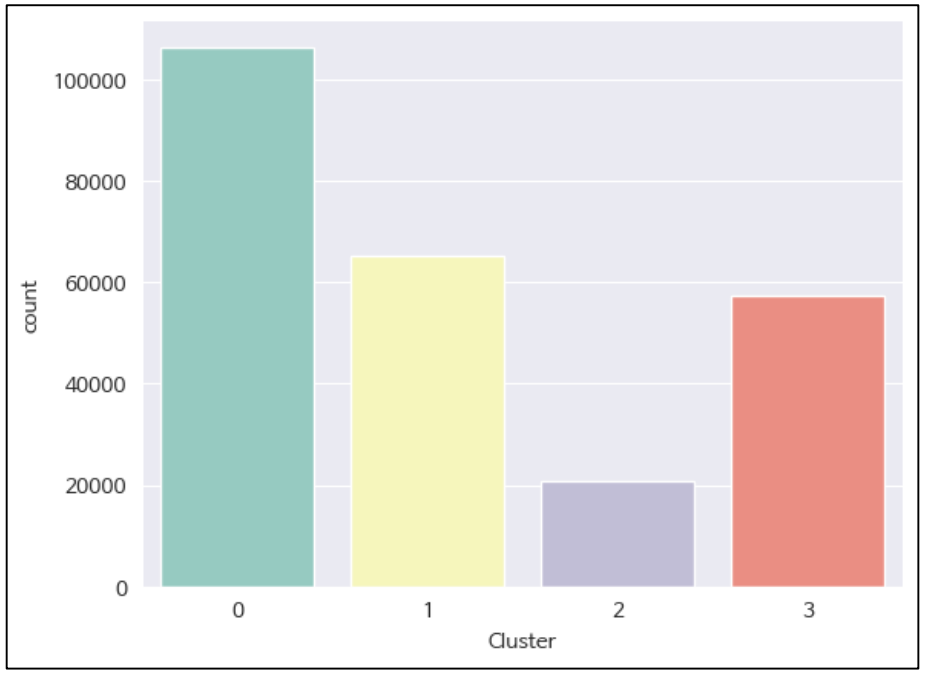
17) 군집 시각화

0번 군집 : 중구 , 달서구 , 수성구
1번 군집 : 북구 , 동구 , 서구
2번 군집 : 달성군
3번 군집 : 남구

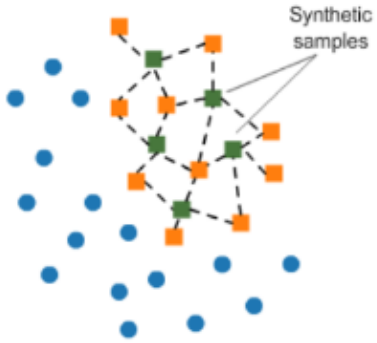


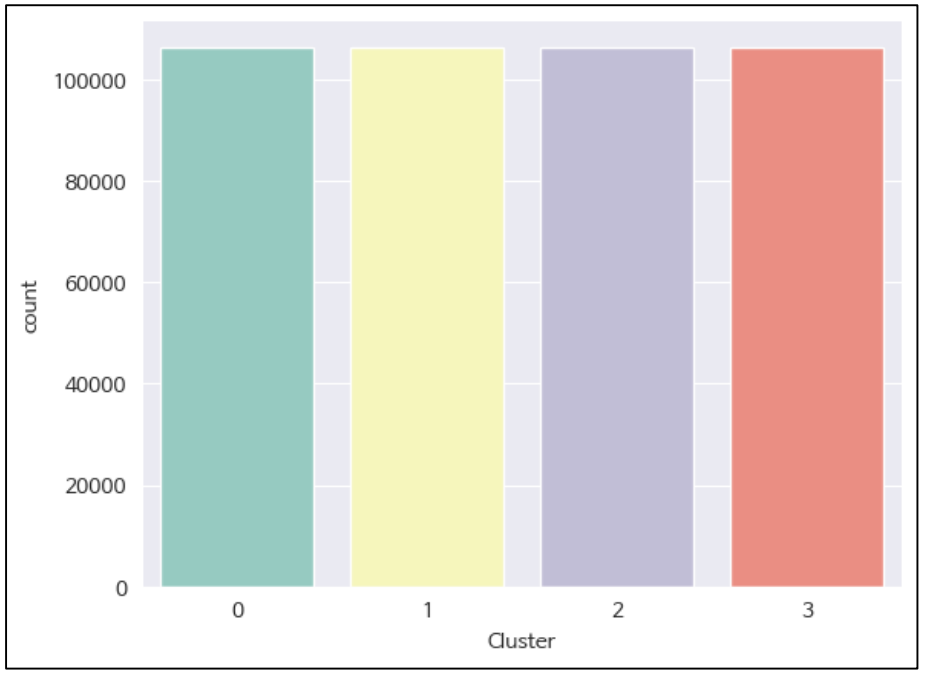
3 EDA

3.7) 불균형 데이터 처리



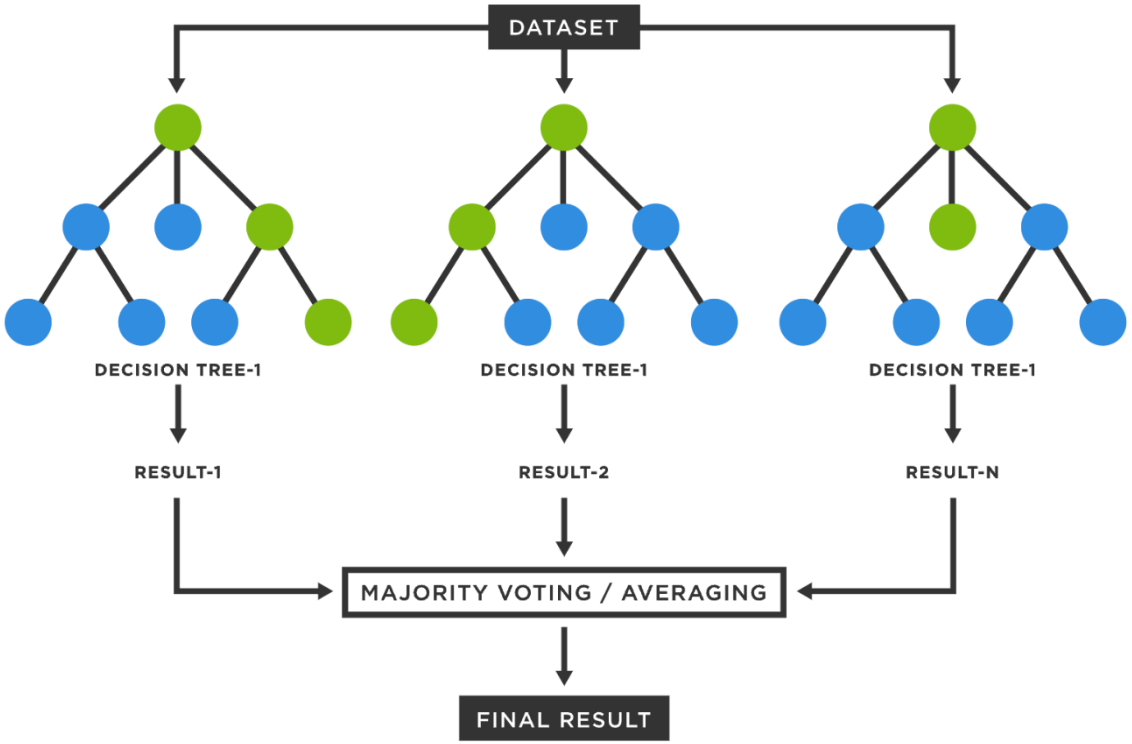
SMOTE 알고리즘 사용



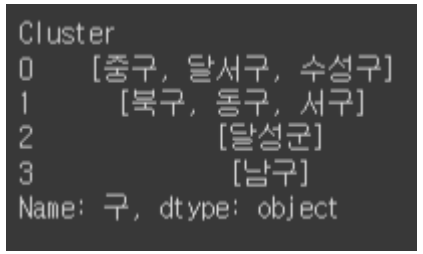


4 모델 학습

1. 과적합이 잘 일어나지 않는다.
2. 의사결정나무 알고리즘 기반이기 때문에 스케일링을 할 필요 없음
3. 데이터가 범주형 데이터가 많고 비선형적이기 때문에 의사결정나무 기반의 알고리즘을 채택



5 분석 결과



1) 0번 군집

0번 군집의 경우 , 데이터가 각각 특별시도 , 구도 이렇게 두 종류를 띄고 있으며 , 제한속도는 중간정도이다. 0번 군집의 경우 , 사고율이 높은 특별시도 , 구도가 자리를 잡고 있으며 다른 지역에 비해서 , 관리 면적이 크면서 도심이기 때문에 사고의 위험이 큰 지역이라고 판단이 가능하다 , 달서구 , 수성구는 사고의 빈도도 높고 , 중구 같은 경우에는 단일로의 비중이 높아서 사고 위험도가 높다.

1) 1번 군집

1번 군집의 경우 , 다른 지역에 비해서 특별시도의 비중이 크다 , 하지만 다른 지역에 비해서 면적이 그렇게 크지 않으며 , 그리고 제한속도도 중간정도에 위치하기 때문에 0번 군집에 비해서 사고율은 낮을 것으로 판단된다.

1) 2번 군집

2번 군집의 경우 , 다른 지역에 비해 면적이 매우 크지만 , 고속국도가 자리를 잡고 있으며 , 그리고 도심에서 조금 떨어진 지역이기 때문에 , 사고의 위험은 크지 않다고 판단됨.

1) 3번 군집

3번 군집의 경우 , 면적이 넓지도 않고 , 빈도수 자체도 적기 때문에 크게 위험하지 않다고 판단됨.

5 분석 결과

	precision	recall	f1-score	support
0	0.65	0.38	0.48	21298
1	0.45	0.52	0.48	21319
2	0.53	0.78	0.63	21154
3	0.73	0.59	0.65	21402
accuracy			0.57	85173
macro avg	0.59	0.57	0.56	85173
weighted avg	0.59	0.57	0.56	85173

1. 하이퍼패러미터 튜닝
2. 모델 선택 및 학습 곡선
3. 범주형 수치 데이터 binning

성능향상

6 시사점 및 개선점

1. 시사점

1) 교통사고의 발생 시기는 주말이 많다
2) 교통사고의 주요 발생 지역은 도로교통법상 과속 단속 구역
3) 교통사고의 위험이 높은 도로의 종류가 존재
4) 교통사고의 위험도는 차대차 사고가 높다

2. 개선점

1) 교통사고 데이터의 품질 향상
2) 교통사고 분석의 심층성 강화
3) 교통사고 예방 및 감소 방안의 실효성 검토
4) 기계학습 모델의 성능 향상 필요

END