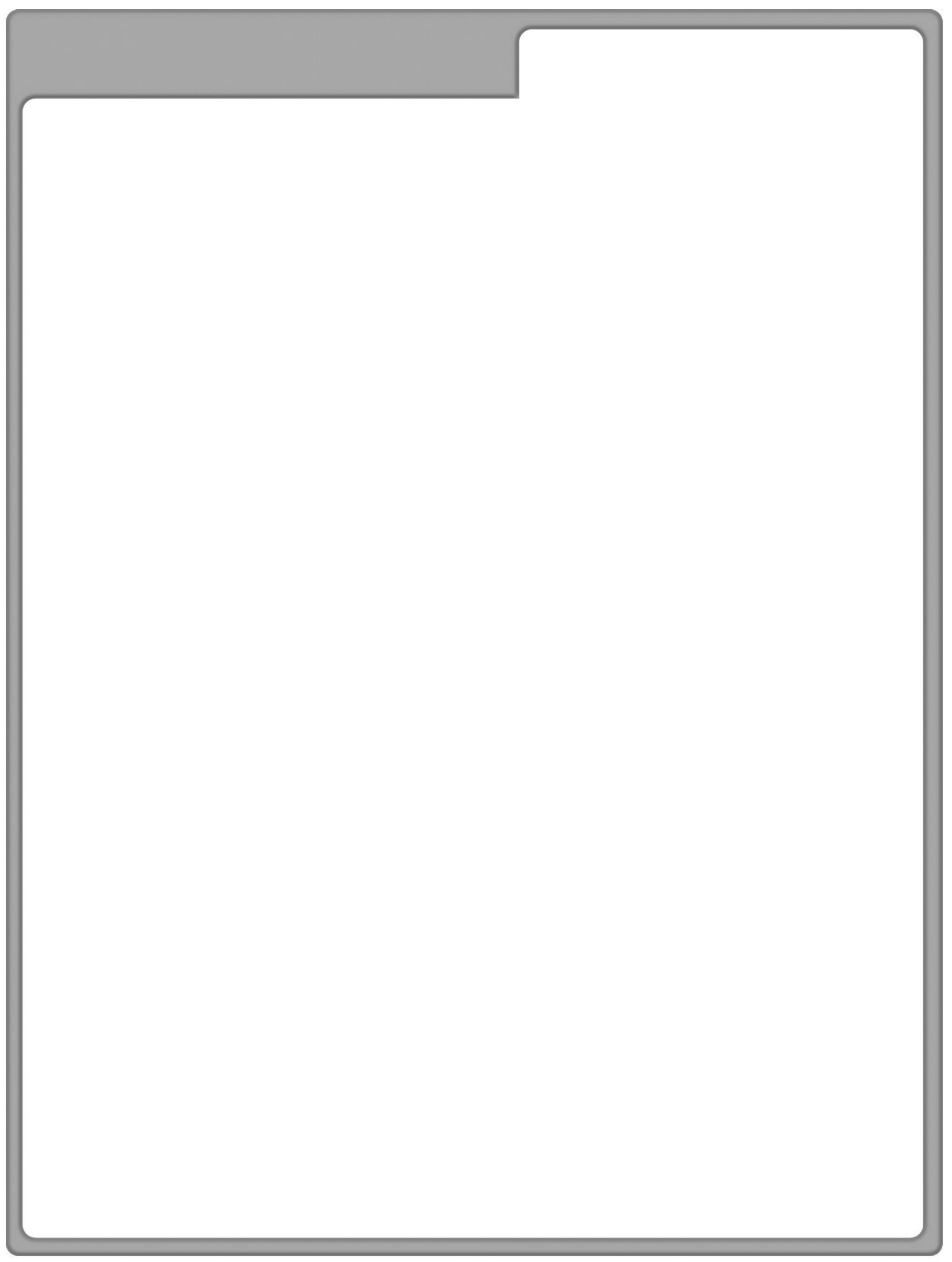
|  |
| --- |
| 프로젝트기반 빅데이터 분석 과정 8기  대구 교통사고 분석 결과 보고서 |

2023. 12.18

|  |  |  |
| --- | --- | --- |
| **프로젝트 명** | **시공간 정보로부터 사고위험도 예측 AI 모델 개발** | |
| **개발 기간** | **2023.11.27.~2023.12.22** | |
| **팀 장** | **김대양** | **kenken2021320012@gmail.com** |
| **팀 원** | **태정수** | **xowjdtn@gmail.com** |
| **서동윤** | **sdy9716@gmail.com** |



**Ⅰ. 분석 결과 요약**

1. 대구 교통사고 분석 결과 요약 1

1) 분석 결과 개요 1

2) 분석 결과 요약 2

**Ⅱ. 서론**

1. 분석 개요 5

2. 분석 개발 현황 5

3. 데이터 수집 개요 6

**Ⅲ. 데이터 수집 및 전처리**

1. 데이터 수집 8

2. 데이터 전처리 8

**Ⅳ. 모델 구축 및 검증**

1. 모델 구축 27

1) 변수 선택

2) 모델 구축

2. 모델 검증 30

1) 모델 검증 방법론

2) 모델 평가 과정 및 방법

**V. 결론**

1. 분석 결과

2. 시사점 및 개선점

**Ⅵ. 산출물 목록**

목 차

|  |  |  |
| --- | --- | --- |
| **Ⅰ** |  | **분석 결과 요약** |

|  |  |
| --- | --- |
| 1 | 분석 결과 개요 |

❍ 프로젝트 명: 대구 교통사고 분석

❍ 개발 기간: **2023.11.27 ~ 2023.12.22**

❍ 개발 목표: 대구 교통사고 피해 예측 AI 알고리즘을 개발하여, 사고 위험도가 높은 지역과 시간을 파악하고, 교통사고 예방 할 수 있다.

❍ 주요 분석 방법

1. 사고 위험도와 관련있는 변수와 , 독립변수 간의 관계성을 파 악하기 위한 상관분석

2. 탐색적 데이터 분석에서 얻은 통찰이나 , 가설이 유의한지 파 악하기 위한 통계적 데이터 분석

3. 기준을 넘는 이상치 데이터들의 패턴을 파악하기 위한 이상치 분석

4. 가공 데이터의 적합성 및 지역별 위험도의 그룹핑을 위한 Kmeans를 통한 군집분석

|  |  |
| --- | --- |
| 2 | 분석 결과 및 시사점 |

1. 교통사고의 주요 발생 지역 및 발생 시기 분석.

- 주말과 야간에 사고 발생률이 높음.

2. 교통사고발생요인을 고려한 안전 우선고려지역 선정

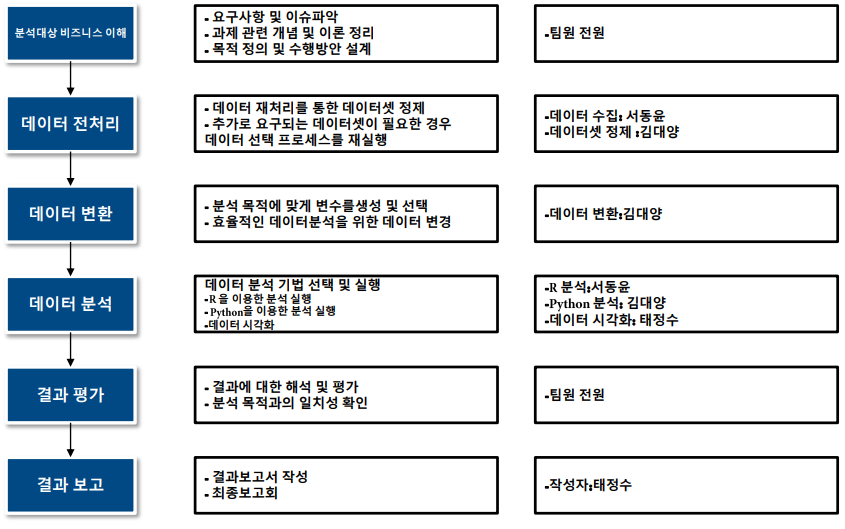
- 과속 단속 강화가 필요

|  |  |  |
| --- | --- | --- |
| **Ⅱ** |  | **서 론** |

|  |  |
| --- | --- |
| 1 | 분석 개요 |

대구광역시 교통사고 예방 및 해결 방안을 탐색하기 위해 2019년~2021년 공공데이터를 이용해, 사고 발생 지역, 노면상태 등을 파악함.

|  |  |
| --- | --- |
| 2 | 분석 개발 현황 |

❍ 업무 분장

❍ 개발 일정

| **추진과제** | **11월 24일 ~ 12월 22일** | | | | | | | | | | | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1주** | | | | **2주** | | | | | **3주** | | | | | **4주** | | | | |
| **1. 모델링 과정** | | | | | | | | | | | | | | | | | | | |
| ▪ 데이터 파악 및 정리 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ▪ 데이터 분석 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ▪ 데이터 추가 확보 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ▪ 특성 엔지니어링 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ▪ 모델링 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **2. 결론 도출 및 정리** | | | | | | | | | | | | | | | | | | | |
| ▪ 모델 해석 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ▪ 정리 자료 제작 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **3. 보고회 등** | | | | | | | | | | | | | | | | | | | |
| ▪ 계획서 보고/중간/최종보고회 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ▪ 검수 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

|  |  |
| --- | --- |
| 3 | 데이터 수집 개요 |

- 활용 데이터

| **데이터 명** | **출처** | **주요 항목 및 특징** |
| --- | --- | --- |
| 대구교통사고통계(19~21년) | KOSIS 국가 통계포털 | 대구 각 시군구 내 교통사고 건수 |
| 대구 CCTV 정보 |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **Ⅲ** |  | **데이터 수집 및 전처리** |

|  |  |
| --- | --- |
| 1 | 데이터 수집 |

1. **데이터 수집**

**■ 원시 데이터 수집**

**1)** 대구 교통사고 통계(2019년 ~ 2021년) (Dacon 대구 교통사고 피해 예측 AI 경진대회)

https://dacon.io/competitions/official/236193/overview/description  
2**)** 대구 CCTV 정보 (Dacon 대구 교통사고 피해 예측 AI 경진대회)

https://dacon.io/competitions/official/236193/overview/description

**2. 데이터 불러오기**

판다스의 read\_csv() 함수를 사용하여 데이터를 불러온다.

#####################################################################

import pandas as pd

path = ‘our\_directory\_path’

train = pd.read\_csv(path + ‘train.csv’)

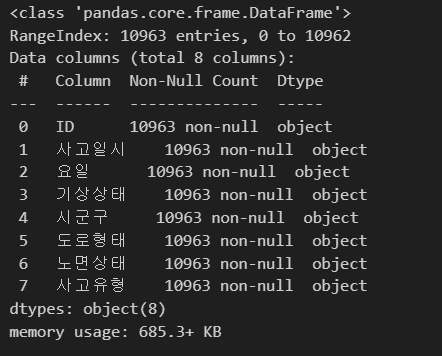
cctv = pd.read\_csv(path + ‘cctv.csv’)

train.info()

cctv.info()

#####################################################################

**■ 데이터를 불러온 뒤 구조 확인하기**

 **>** 대구 교통사고 통계(2019년 ~ 2021년)



> 대구 CCTV 정보

- 사용 컬럼 정보 정리

컬럼의 수가 많기 때문에 분석을 통해서 얻은 성찰을 통하여 중요한 컬럼들을 선택함.

자세한 분석 과정은 이후의 과정에서 소개.

> 대구 교통사고 통계 데이터

|  |  |  |  |
| --- | --- | --- | --- |
| 데이터 | 컬럼명 | 컬럼 설명 | 데이터 타입 |
| 대구 교통사고 통계  데이터 | 요일 | 요일 데이터 | Object |
| 기상상태 | 사고당시 기상상태 | Object |
| 시군구 | 대구광역시 시군구 데이터 | Object |
| 도로형태 | 사고당시 도로형태 | Object |
| 노면상태 | 사고당시 노면상태 | Object |
| 사고유형 | 사고의 유형 | Object |
| ECLO | 인명피해 심각도 | Int |

> 대구 CCTV 정보 데이터

|  |  |  |  |
| --- | --- | --- | --- |
| 데이터 | 컬럼명 | 컬럼 설명 | 데이터 타입 |
| 대구 CCTV 정보 데이터 | 도로종류 | 각 시군구별 도로의 종류 | Object |
| 소재지지번주소 | 상세 소재지지번주소 | Object |
| 제한속도 | 각 시군구별 도로의 제한속도 | Int |

**3. 데이터 결합하기**

**■ 데이터 결합하기**

- 대구 교통사고 통계 데이터와 대구 CCTV 정보 데이터를 “시군구” 기준으로 결합시킨다.

대구 교통사고 통계 데이터의 경우 , 시군구 컬럼이 있지만 , 대구 CCTV 정보 데이터의 시군구 데이터가 없기 때문에 대구 CCTV 정보 데이터의 “소재지지번주소” 데이터의 시군구 부분을 추출해서 새롭게 시군구 컬럼을 만들어 준다.

#############################################################

# 소재지지번주소의 시군구 부분을 새롭게 리스트로 만들고 시군구 컬럼을 생성

data = list(cctv['소재지지번주소'].str.split().str[0:3])

detail = []

for i in range(len(data)):

detail.append(" ".join(data[i]))

cctv[‘시군구’] = detail

merged\_data = pd.merge(train , cctv , on = ‘시군구’ , how = ‘left’)

#############################################################



|  |  |
| --- | --- |
| 2 | 데이터 전처리 |

**1. 데이터 탐색**

**■ 대구 교통사고 통계 데이터**

**■ 결측값 , 이상치 확인**

데이터의 결측값과 이상치를 확인한다.

###############################################################

# 결측값 확인 라이브러리

import missingno as msno

msno.matrix(train)

# 이상치 확인

fig , ax = plt.subplots(1 , 2 , figsize = (8 , 5))

sns.distplot( # 밀도함수

target ,

color = 'g' ,

ax = ax[0]

)

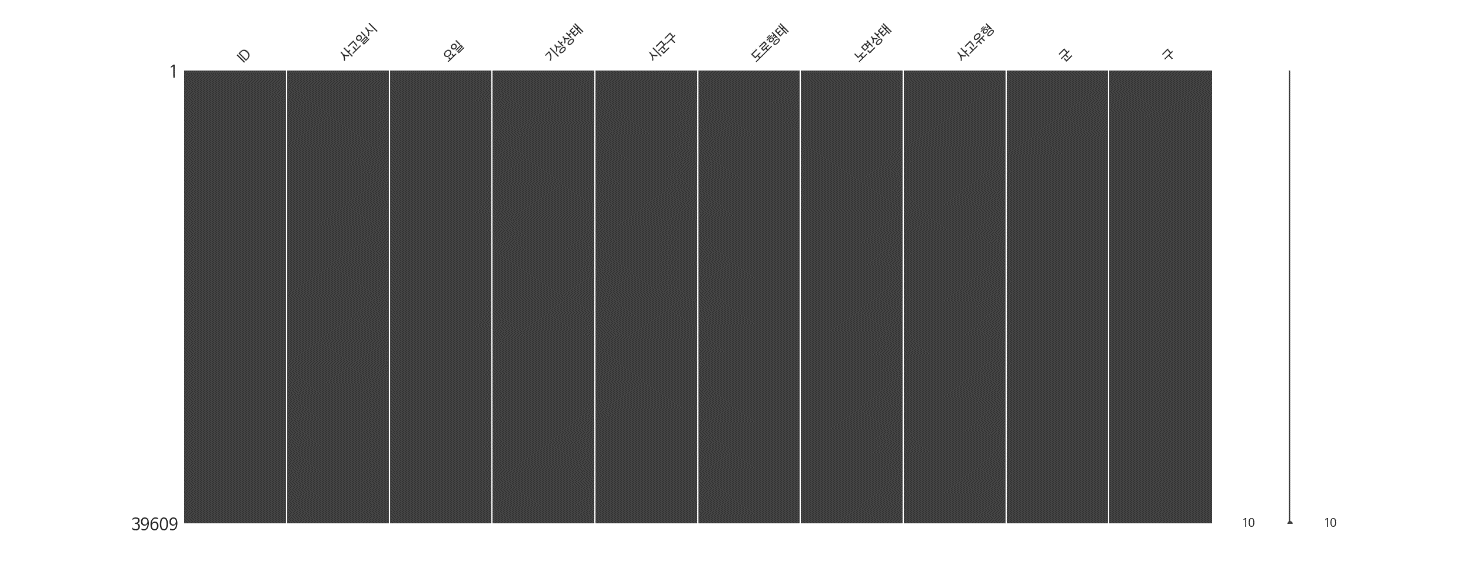
sns.boxplot( # 상자수염그림

target ,

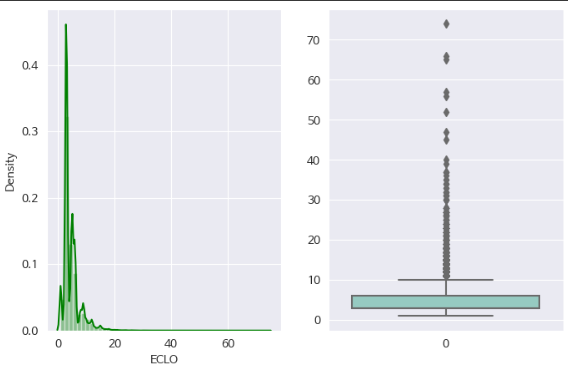
ax = ax[1]

)

###############################################################

****

위의 그래프를 토대로 , NA의 값을 판단할 수 있는데 만약 NA값이 있다면 해당 막대에 빈 줄이 생기기 때문에 NA값은 없다고 판단이 가능하다.



위의 그래프에서 , 종속변수의 분포인데 여기에서 오른꼬리분포를 가지는 것을 알 수 있다 , 상자수염그림을 그려서 보면 상한을 넘는 값들이 많이 있다는 것을 알 수 있다.

위의 그림을 토대로 , 전체 데이터중 이상치의 비율을 계산하면 ,



이상치의 비율은 5% 정도이다.

-성찰

ECLO의 데이터 타입이 , 이산형 데이터이기 때문에 로그변환과 같은 방식으로 치우침을 해결하는 것은 이산형 데이터가 가지고 있는 값들의 의미가 왜곡될 수도 있기 때문에 , 분석을 진행 후에 적절한 방법으로 처리를 해야 한다.

**■ 데이터 빈도 분석**

- 데이터의 대부분이 범주형 데이터이기 때문에 , 빈도 분석을 시행함으로서 데이터가 어떤 성향을 띄고 있는지를 파악한다.

###############################################################

# 그래프 작성 함수 정의

def graph(df , col , condi = False):

data = pd.concat([df[col] , target] , axis = 1)

fig , ax = plt.subplots(1 , 2 , figsize = (13 , 5))

sns.countplot(

x = data[col] ,

ax = ax[0]

)

if condi:

ax[0].set\_xticklabels(ax[0].get\_xticklabels() , rotation = 45)

g\_data = data.groupby([col])[‘ECLO’].mean()

sns.barplot(

x = g\_data.index ,

y = g\_data.values

ax = ax[1]

)

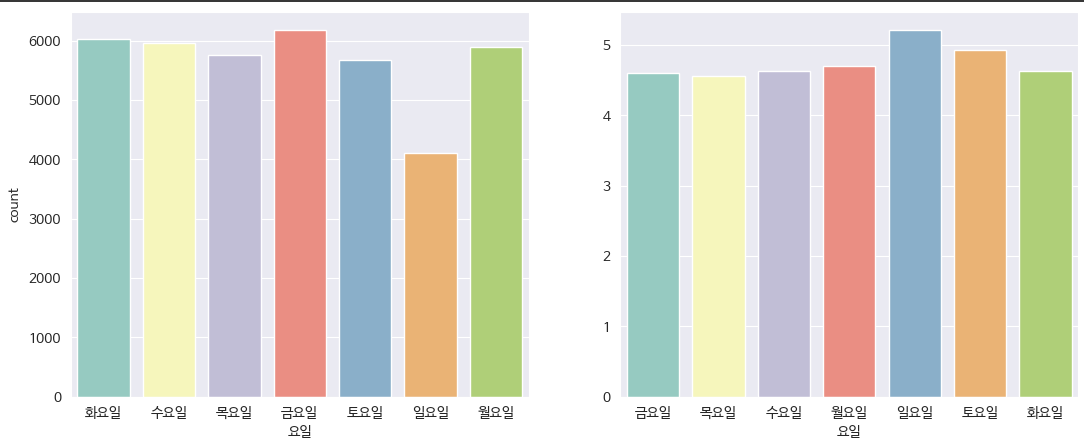
if condi:

ax[1].set\_xticklabels(ax[1].get\_xticklabels() , rotation = 45)

###############################################################

위의 함수를 토대로 trian데이터의 각 범주형 데이터의 빈도와 위험도를 식별할 수 있는 컬럼인 ECLO의 평균을 구하여 시각화함.

1) 요일



성찰

- 주말(토 , 일)은 평일에 비해서 사고 건수가 적고 , 특히 일요일이 적다

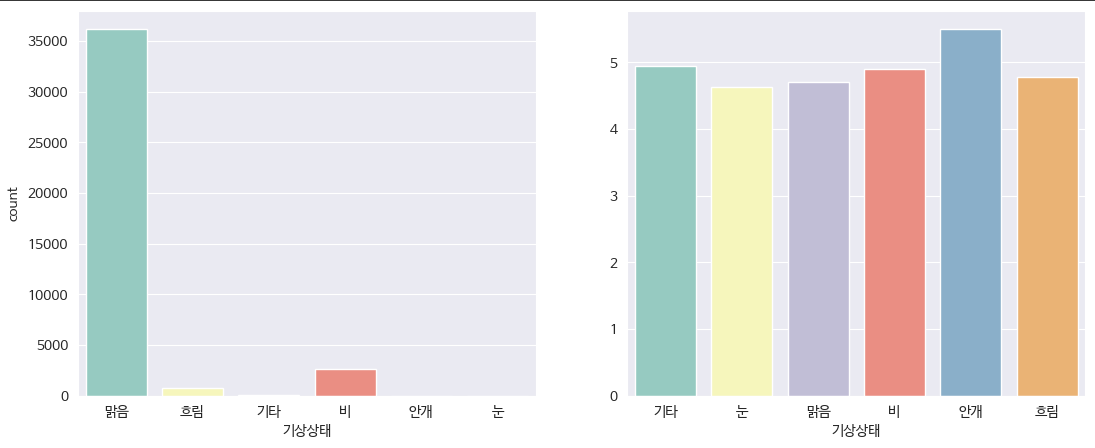
- 인명피해 심각도(ECLO)의 경우 주말에 가장 올라간다

- 평일은 대부분은 빈도수가 큰 차이를 보이지 않는다

가설

- 주말과 , 평일은 유의미한 차이가 있으며 , 주말에 더 사고의 위험성이 있을 것이다.

2) 기상상태



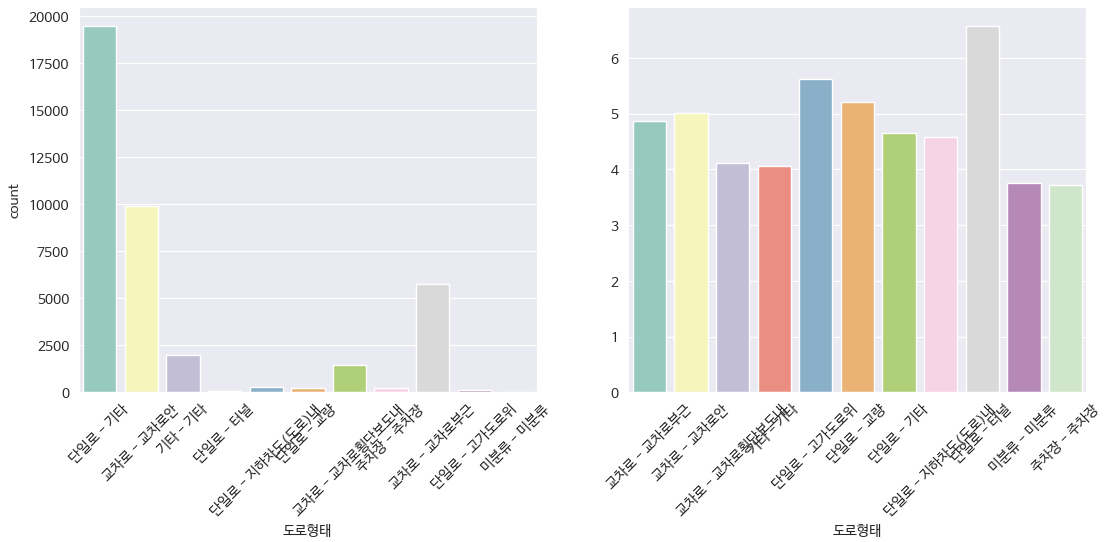
성찰

- 빈도수는 맑음이 압도적으로 많은 빈도수를 보이고 있으며 , 때때로 비와 흐림이 있다.

- ECLO의 평균을 각각 구할 경우 , 안개에서 가장 큰 인명피해 심각도가 나옴.

- 해당 컬럼이 ECLO와 서로 영향을 주고 받는지 , 통계적 검정 필요

3) 도로형태



성찰

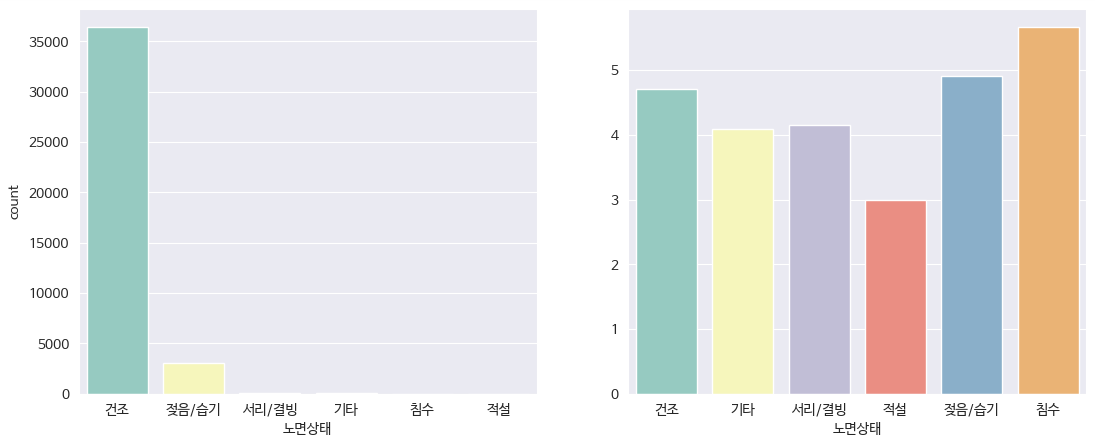
- 도로의 종류별로 그룹으로 묶어서 범주를 줄이려고 했었지만 , 같은 분류 (단일로 , 교차로 등) 안에서도 종류에 따라서 ECLO의 값이 매우 다르기 때문에 해당 방법을 기각.

- 해당 컬럼과 ECLO사이의 영향력에 대해서 통계적 검정 필요

가설

- 단일로와 교차로 사이에 ECLO에 대한 통계적인 차이가 있을 것이다.

4) 노면상태



성찰

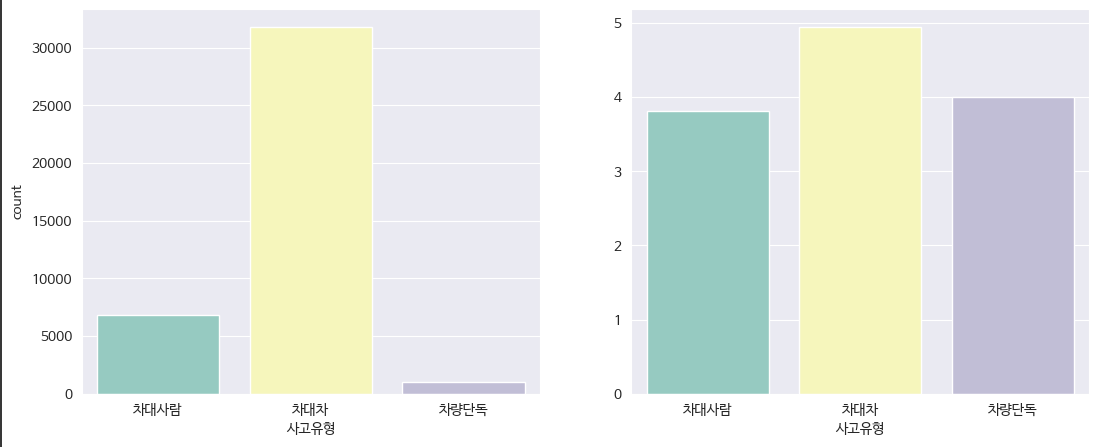
- 대부분이 건조이고 , 때때로 젖음/습기가 있다 , 그 외에는 거의 보이지 않을 정도이다.

- ECLO의 평균은 침수 , 젖음/습기때 가장 많았다. 하지만 서리/결빙 , 침수 , 적설등은 매번 있는 것은 아니기 때문에 빈도수도 적어서 ECLO와 연관이 있는지 통계적 검정 필요.

가설

- 노면상태와 기상상태 사이에는 상관성이 존재할 것이다.

5) 사고유형



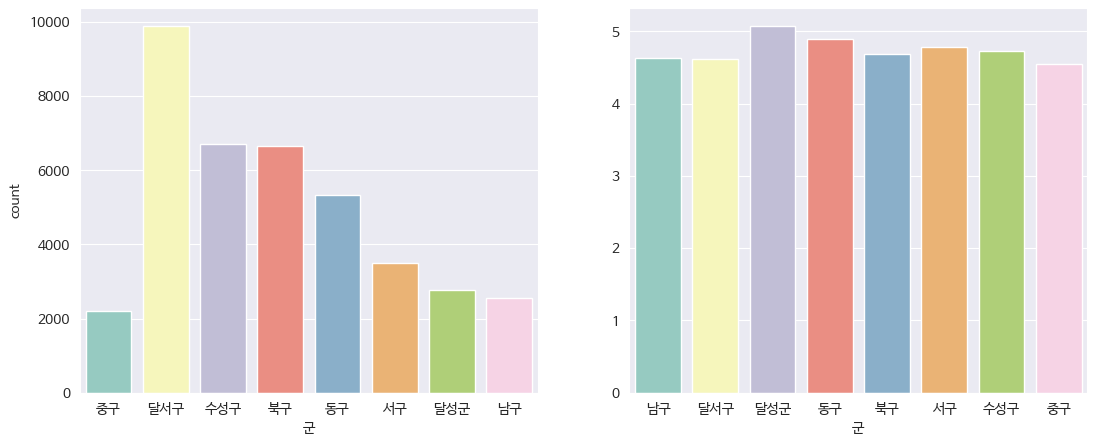
성찰

- 차대차가 가장 많고 , 차량단독은 적은 편에 속하고 , 차대사람 , 차량 단독 이렇게 두가지는 ECLO 값이 거의 차이가 없으며 , 차대차 같은 경우는 조금 더 높다는 것을 알 수 있다.

가설

- 사고에 해당하는 컬럼인 만큼 , ECLO에 가장 크고 , 직접적인 영향을 주는 컬럼일 것이다.

6) 군



성찰

- 지역별로 교통 사고 건수가 다르기 때문에 , 각 구별 특징에 대해서 살펴볼 필요가 존재.

- ECLO의 값은 전체적으로 차이가 크게 나지 않는다.

**# summary**

- 각 변수별 상관관계를 파악하기 위해서 , 상관분석을 진행.

- 각 변수가 종속변수와 서로 영향을 주고받는지 , 독립검사를 진행.

- 위에서 얻은 몇가지 가설을 검정한다.

**■ 상관분석**

- 모든 데이터가 범주형 데이터이기 때문에 , 보통 사용하는 방법으로는 불가능하며 , 3개 이상의 범주형 컬럼 사이의 상관성을 분석하기 위해서 , Cramer’s V방법을 통해서 상관성 분석을 진행하였다.

Cramer’s V란 , 3개 이상의 범주형 변수들 사사이의 상관성을 분석하는데 사용하는 기법으로 , 0~1.0 사이의 값을 가지며 0.5를 넘는 시점에서 강한 상관관계로 볼 수 있다.

###############################################################

# Cramer’s V 계산 함수

def cramers\_V(var1 , var2):

crosstab = np.array(pd.crosstab(var1 , var2 , rownames = None , colnames = None))

stat = chi2\_contingency(crosstab)[0]

p\_value = chi2\_contingency(crosstab)[1]

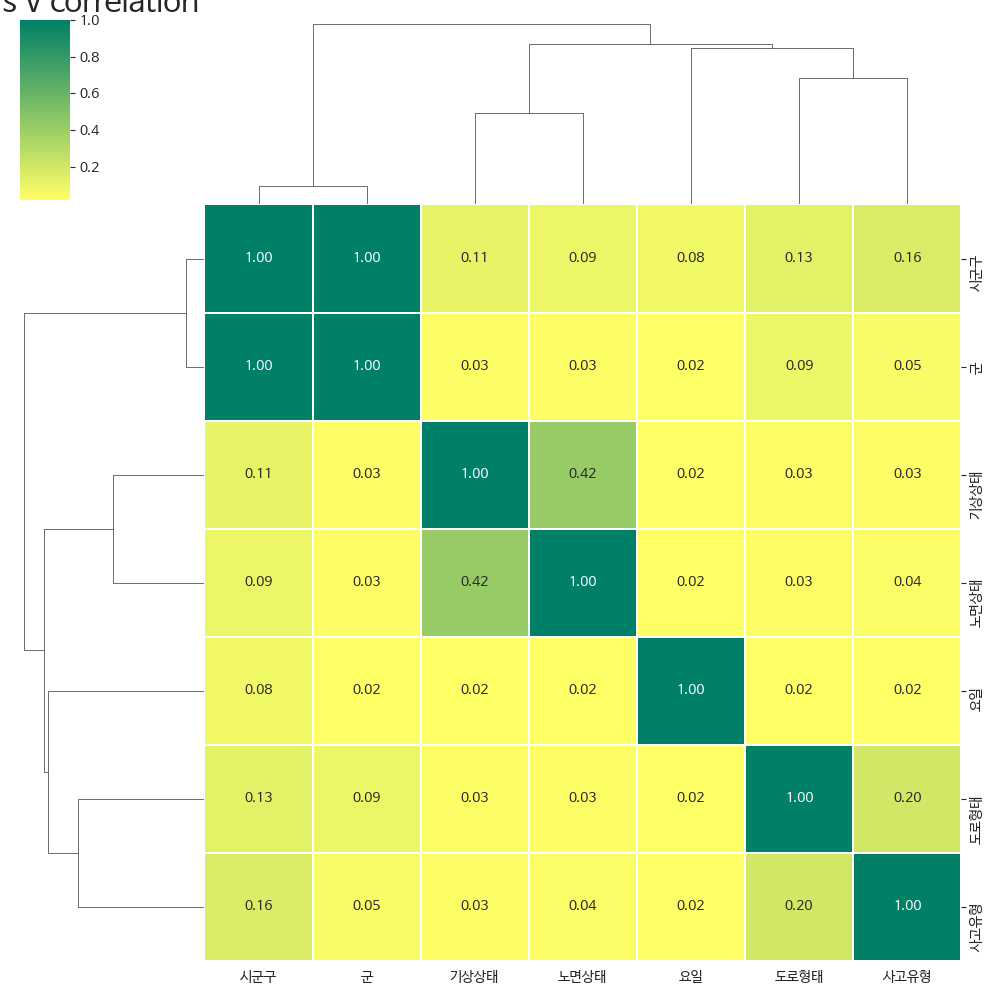
obs = np.sum(crosstab)

mini = min(crosstab.shape) - 1

return np.sqrt(stat / (obs \* mini)) , p\_value

###############################################################

1) 독립변수 간 상관관계



성찰

- 위의 사진에서 각 컬럼별 상관계수를 볼 수 있다.

1) 시군구 <-> 군

해당 데이터는 지역을 나타내고 있으며 , 시군구 안에 이미 군이 들어있기 때 문에 완전히 같은 값을 가지기 때문에 이후에 사용할 시군구를 남긴다.

2) 기상상태 <-> 노면상태

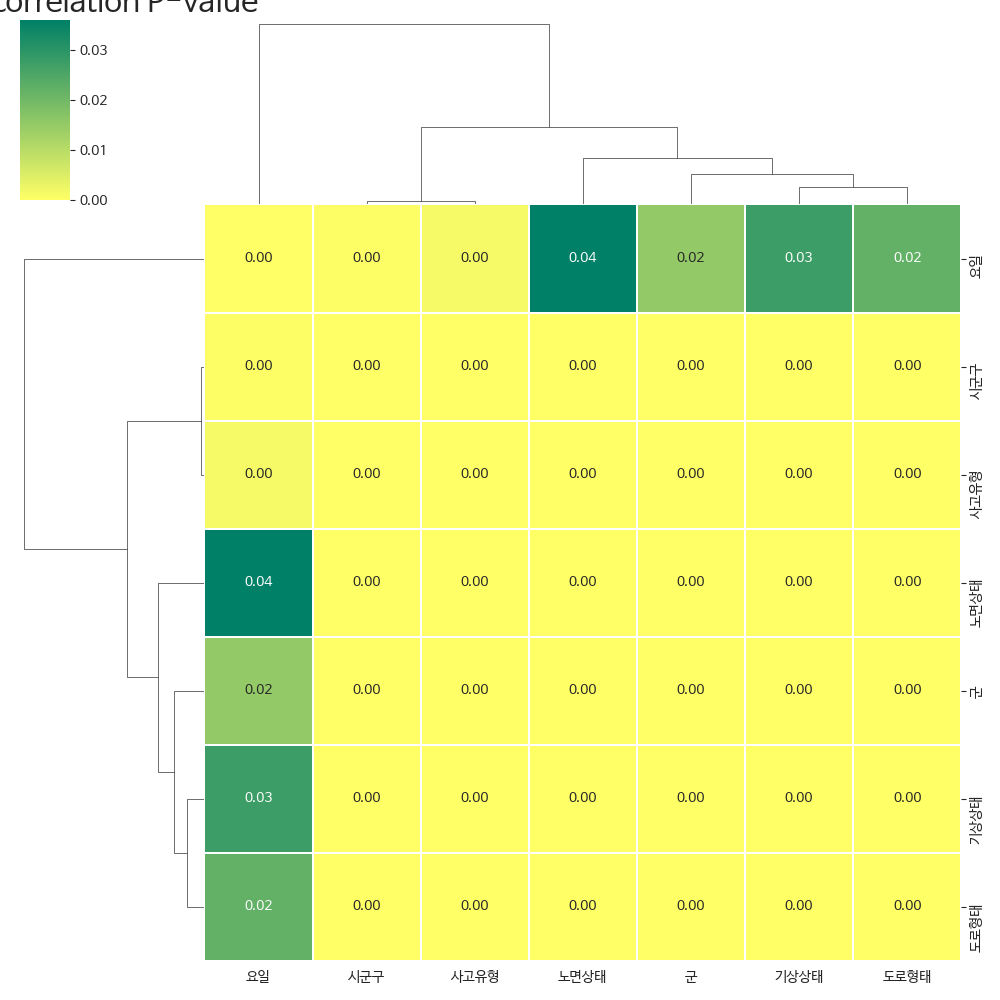
기상상태와 노면상태는 위에서 세운 가설로 , 기상상태와 노면상태 사이에는 0.42로 중간 이상으로 높은 상관관계를 가지고 있다고 볼 수 있다.

3) 도로형태 <-> 사고유형

두 컬럼 사이에는 , 약한 상관관계가 있다고 볼 수 있다.

2) 상관관계의 유의성

- 상관관계를 계산할 경우 , p-value도 함께 계산이 된다. 해당 상관계수가 의미가 있는지 , 통계적으로 유의한지에 대해서 살펴보기 위해서 p-value도 함께 봐야한다.



- 위의 분석결과로 모든 값이 0.05보다 낮기 때문에 , 통계적으로 의미가 있으며 위에서 분석한 상관분석의 결과는 믿을만 하다는 결론이 나온다.

3) 독립성 검정

- 독립 변수들과 종속 변수 사이에 유의미한 관계가 있는지 , 서로 영향을 주고 받는지 여부에 대해서 파악하기 위해 , 독립성 검정을 수행.

< 가설 설정 >

H0 : 두 변수 간에는 통계적으로 유의한 관계가 없다. (독립)

H1 : 두 변수 간에는 통계적으로 유의한 관계가 있다. (종속)

#############################################################################

# 독립성 검정

p\_vals = []

for var in train.columns[2:8]:

cross = np.array(pd.crosstab(train[var] , target))

\_ , p\_value , \_ , \_ = chi2\_contingency(cross)

p\_vals.append(p\_value)

p\_vals = pd.Series(

p\_vals ,

index = train.columns[2:8]

)

index = p\_vals.index

for i in range(len(p\_vals)):

if p\_vals[i] < 0.05:

print(f"{index[i]} : 관련이 있다.")

else:

print(f"{index[i]} : 독립이다.")

#############################################################################

|  |  |
| --- | --- |
| 컬럼명 | 검정 결과 |
| 요일 | 관련이 있다 |
| 기상상태 | 독립이다 |
| 시군구 | 관련이 있다 |
| 도로형태 | 관련이 있다 |
| 노면상태 | 독립이다 |
| 사고유형 | 관련이 있다 |

위의 검정을 통해서 , 종속 변수와 종속적인 관계를 가지고 있는 데이터에 대해서 가설검정을 진행하면서 , 추가적인 관계성 시각화 분석을 진행한다.

텍스트, 스크린샷, 도표, 디자인이(가) 표시된 사진

자동 생성된 설명

종속적인 관계를 가지고 있었던 , 4가지 컬럼에 대해서 종속변수와의 상관관계를 시각화 하면 ,사고유형에서 중간정도의 상관성을 가지는 것을 볼 수 있으며 , 그리고 다른 컬럼에서도 미약하지만 , 상관성이 있다는 것을 알 수 있음.

4) 추가 데이터 분석

4-1) 요일

######################################################################

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")

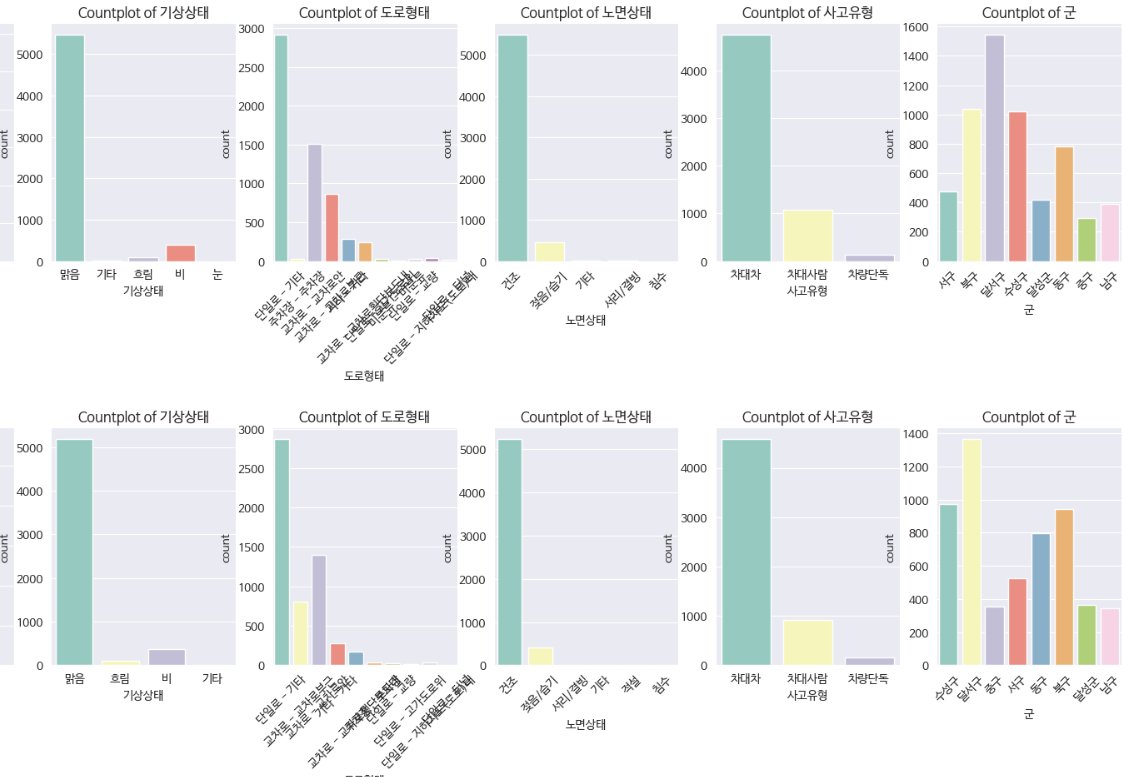
test(data\_weekday , data\_weekend)

######################################################################

통계적 검정을 통해서 주중과 주일 사이에 통계적인 차이가 있는지에 대해서 파악을 하였더니 , 통계적인 차이가 있다는 것을 파악.



주중과 주말의 차이를 컬럼의 분포에 따라서 시각화 할 경우 (위 : 주중 , 아래 : 주말)



성찰

* 전체적으로 큰 차이를 보지는 못하지만 , “도로형태” 컬럼의 교차로 부근에서 분포에 차이를 볼 수 있다.
* 즉 , ECLO의 차이를 통한 검정에서 차이가 있는 부분은 도로형태 컬럼에서 교차로 부분에서 차이를 볼 수 있다.

4-2) 도로형태

######################################################################

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("단일로와 교차로의 종속변수의 평균 차이는 통계적으로 의미있다.")

else:

print("아니다")

######################################################################

위의 검정을 통해서 , 단일로와 교차로로 분류된 범주 사이에 통계적인 차이가 있다는 것을 파악.



단일로와 교차로의 차이를 파악하기 위한 분석을 할 경우 먼저 , ECLO의 평균을 시각화 할 경우 ,

텍스트, 스크린샷, 도표, 다채로움이(가) 표시된 사진

자동 생성된 설명

대부분의 단일로가 평균을 넘고 있고 , 전체적으로 단일로가 교차로보다 더 높은 인명피해 심각도를 가지고 있다..

이것의 원인을 살펴보기 위해서 , 위에서 얻은 성찰을 토대로 종속변수인 ECLO와 가장 연관이 있는 사고유형에 대해서 분포를 살펴보면

텍스트, 스크린샷, 도표, 평면도이(가) 표시된 사진

자동 생성된 설명

성찰

* 단일로의 경우 , 다른 범주에 비해서 차대차의 비율이 높다 , 교차로의 경우 사람들이 걷는 인도가 가까이 있는 경우도 있기 때문에 , 차대사람의 피해도 있을테지만 , 단일로 특히 터널 같은 경우에는 사람과의 사고가 많지 않고 드물다.
* ECLO의 경우 , 차대차가 높기 때문에 비율이 높은 곳은 더 종속변수의 값에 큰 영향을 미칠 수도 있다는 성찰을 얻을 수 있다.

**# Summary**

* 상관분석을 통해서 , 변수의 선택이 가능해짐.
* 독립성 검정을 통해서 , 종속 변수에 영향을 주는 데이터를 기준으로 해당 데이터의 정보를 보완해줄 수 있는 데이터를 찾는다.
* 종속 변수에 영향을 주는 데이터를 중점으로 데이터 분석을 진행한다.

**■ 이상치 분석**

**-** tukey방식으로 상한과 하한 외에 있는 데이터를 이상치로 보고 , 이상치 데이터를 추출한다.

##############################################################

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

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

IQR = q3 - q1

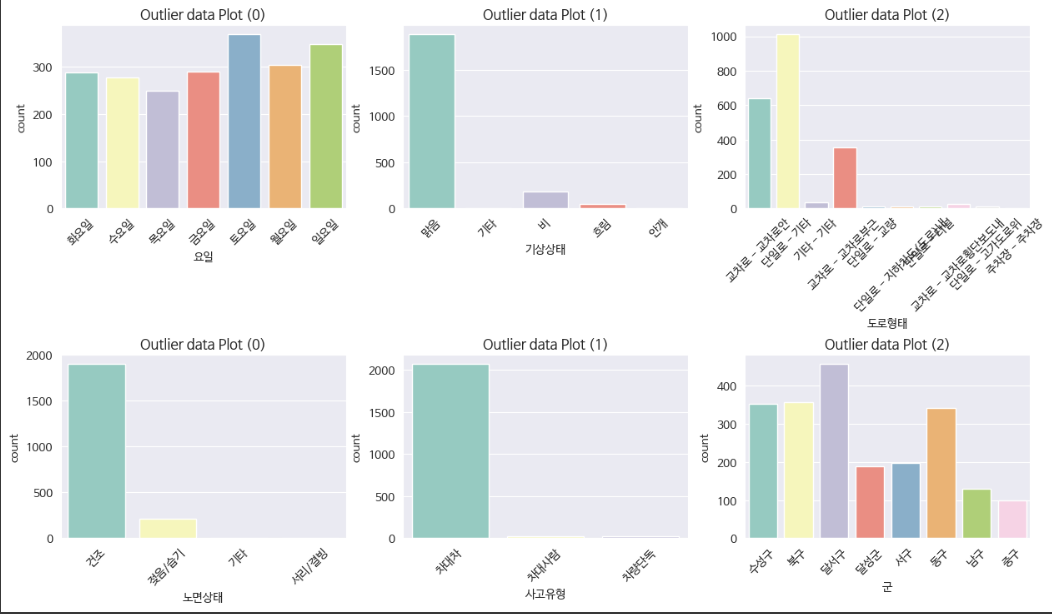
upper\_fence = q3 + 1.5 \* IQR

lower\_fence = q1 - 1.5 \* IQR

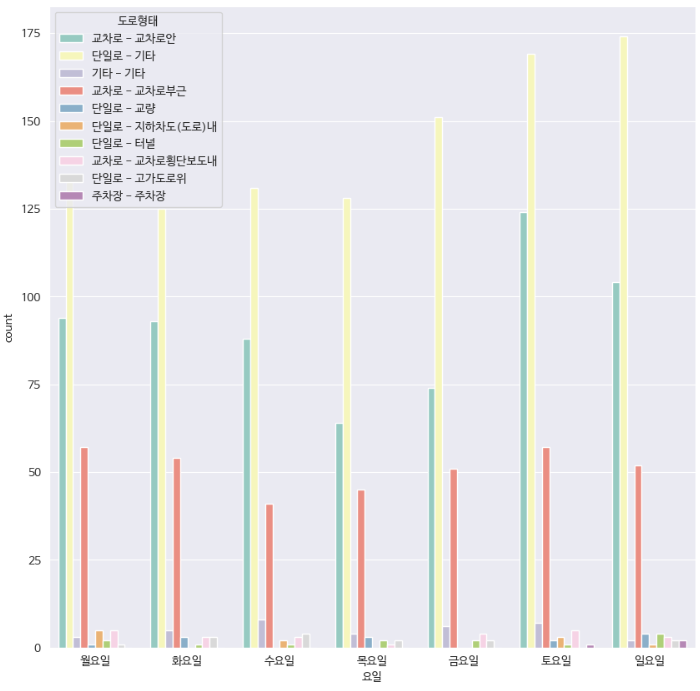
outlier\_data = train[(train['ECLO'] < lower\_fence) | (train['ECLO'] > upper\_fence)]

##############################################################

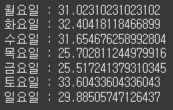
위의 방식으로 얻은 데이터를 토대로 , 빈도 분석을 진행하면



대부분이 원천데이터에서의 분포와 유사하게 있다는 것을 볼 수 있지만 , 요일 부분에서는 원래의 분포와는 다르게 주말이 더 많이 있다는 것을 알 수 있다.



위의 시각화를 통해서 , 이상치는 금요일~일요일에 많이 발생한 것으로 보이고 그 안에서도 요일에 따른 도로형태의 변화에서 큰 변화를 볼 수 있다.



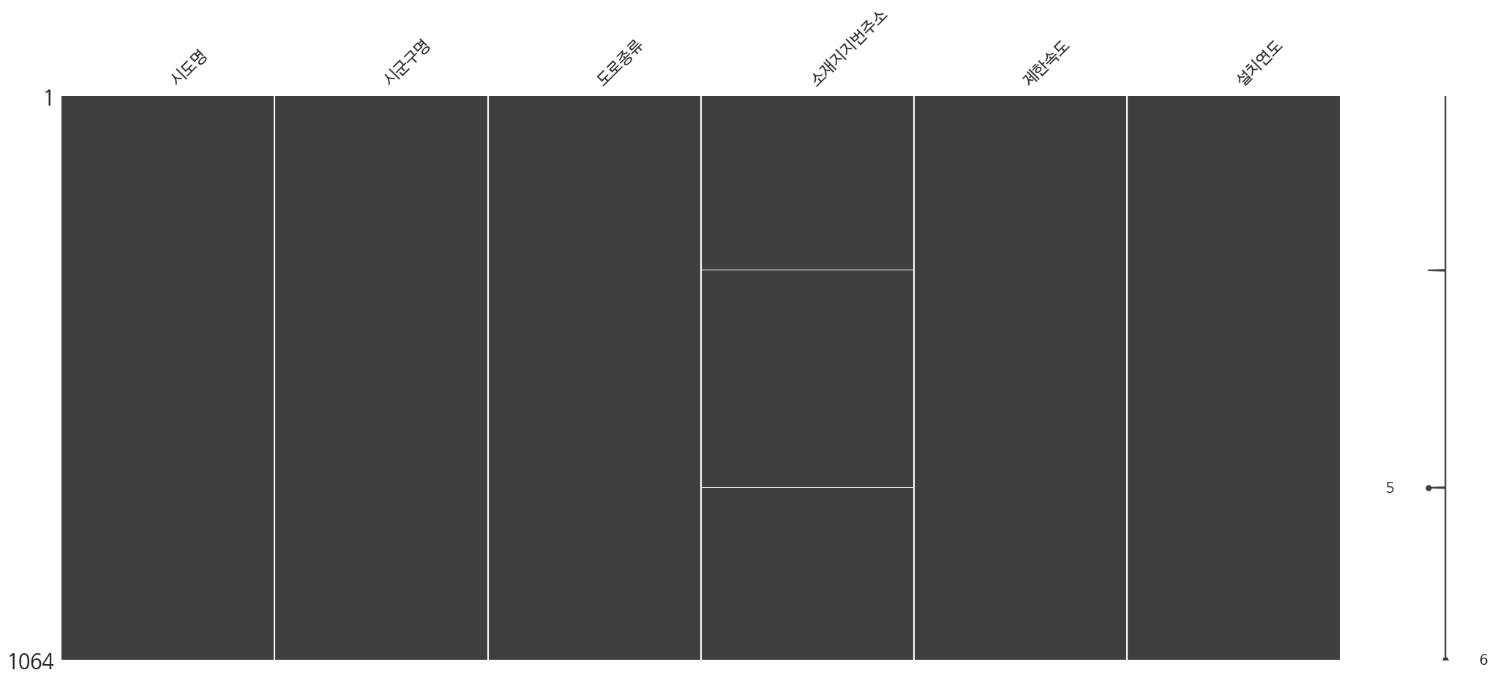
가장 높은 빈도를 보이고 있는 단일로에 비해서 , 교차로가 어떤 식으로 변화를 하는지 알기 위해서 교차로의 비율을 계산할 경우 , 월~수 , 토에 교차로에서의 사고의 비율이 조금 올라가는 것을 볼 수 있다.

**# Summary**

* 이상치들은 주로 금요일~일요일 사이에 많이 발생하는 패턴을 보이고 있음.
* 교차로의 사고는 차와 사람의 사고가 많이 있고 , 그 비율은 월~수 , 토에 많이 올라가지만 , 여전히 단일로에서의 사고가 많기 때문에 모델 학습에 영향을 미칠 수 있는 데이터는 차와 차의 사고가 비중이 높다고 판단이 가능하다.
* 요일에서 원천데이터 분석에서 본 분석과 상이한 부분이 있었지만 , 빈도분석의 결과 대부분이 원천데이터에서의 빈도분석의 결과를 따라가는 형태를 보이고 있기 때문에 , 이상치의 처리는 최빈값으로 대체하는 것으로 채택.

**■ 대구 CCTV 데이터**

**■ 결측값 확인**

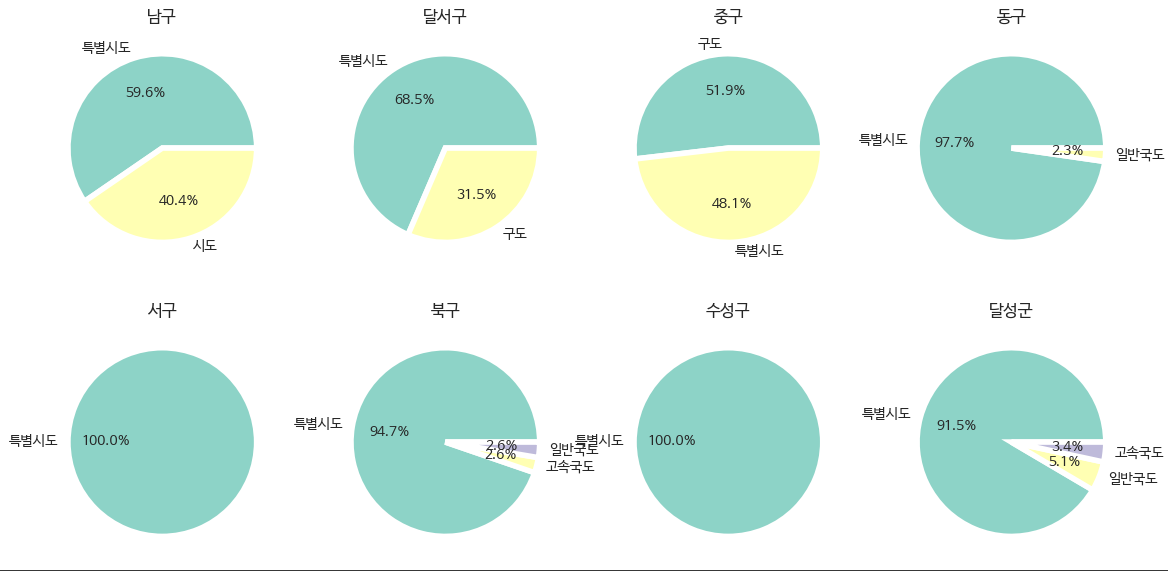


결측값을 시각화 하였다.

성찰

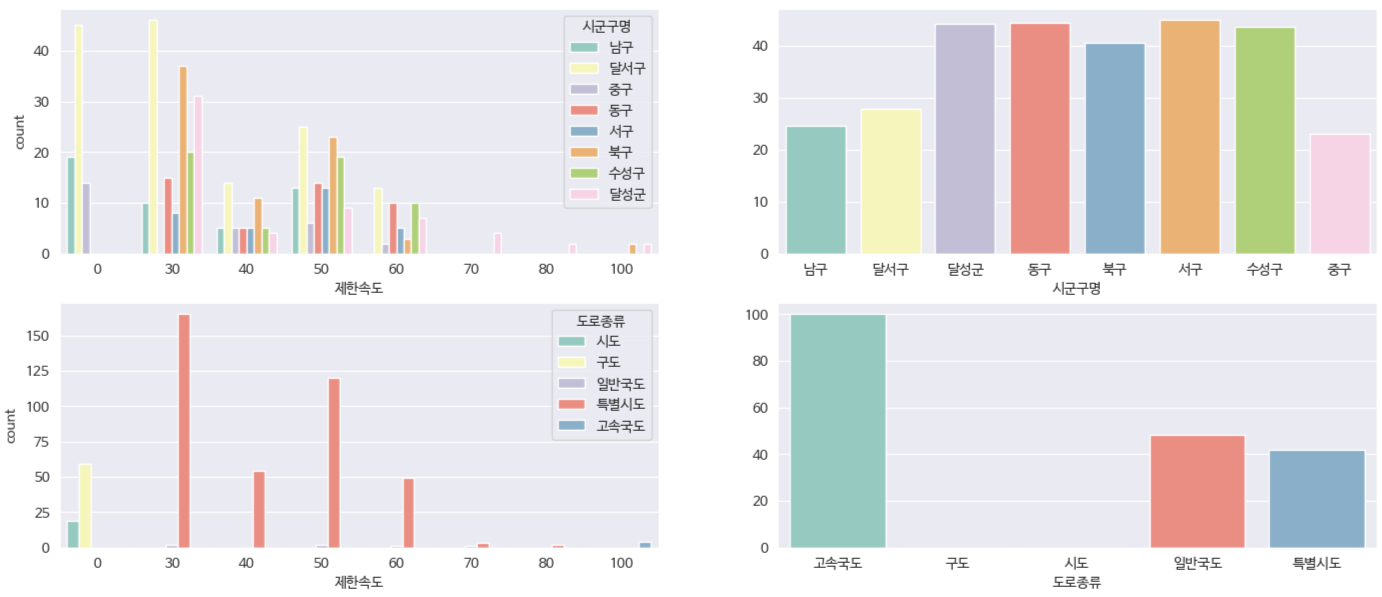
- 위의 그래프에서 , 결측값이 가로로 그어진 선인데 , 전체 데이터의 수(1064)에 비해서 두개밖에 없기 때문에 , 0.19%정도이다. 그렇기에 무시할 수 있을 정도의 크기이기 때문에 dropna()을 통해서 결측값 제거를 채택.

**■ 데이터 분석**



대구 CCTV 데이터의 경우 , 장소를 중심하고 데이터분석을 진행

도로가 관리를 받는 주체에 따른 지역별 비율을 보면 , 대구특별시에서 관리하는 특별시도가 대부분이고 , 그 외에 구도 , 시도 , 일반국도 , 고속국도 등이 분포하는 것을 볼 수 있다.



제한속도의 데이터를 분석을 할 경우 , 위처럼 볼 수 있다. 제한속도가 매우 높은 지역은 달성군 , 북구 등이 있는데 둘다 다른 지역에 비해 고속국도가 자리잡고 있는 곳이기도 하였다.

성찰

* 제한속도의 경우 , 해당 도로 종류와 엮어서 데이터를 구성할 경우 , 해당 도로의 사고 위험율과 직접 연관지을 수 있을 것이다.
* 뉴스와 같은 정보에서 입수한 바로는 , 고속국도가 더 사고가 많이 나지 않는다는 정보를 획득했기에 , 그러한 사고와 연관되어있는 내용을 데이터에 반영할 수 있을 것이다.

**2 데이터 전처리**

**■ 결측값 전처리**

1) CCTV 데이터 처리

##############################################################

cctv = cctv.dropna()

##############################################################

데이터의 결측값이 많지 않았기 때문에 , dropna() 함수를 사용해서 , 처리한다.

2) 병합 후의 데이터 처리

데이터 병합 후에 생긴 NA값에 대해서 처리를 한다.

##############################################################

# 데이터 merge

merged\_data = pd.merge(train , cctv , on = '시군구' , how = 'left')

# NA값 비율

for col in merged\_data.columns:

print(f"{col} NA Percentage : {merged\_data[col].isna().sum() / len(merged\_data) \* 100}%")

merged\_data.dropna()

# 병합 후 생긴 중복 값 처리

duplicated\_rows = merged\_data[merged\_data.duplicated()]

merged\_data = merged\_data.drop\_duplicates()

###############################################################

위의 과정에서 확인한 NA의 비율은

< NA 비율 >

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 : 1.7542639970337413%

시군구명 NA Percentage : 1.7542639970337413%

도로종류 NA Percentage : 1.7542639970337413%

소재지지번주소 NA Percentage : 1.7542639970337413%

제한속도 NA Percentage : 1.7542639970337413%

설치연도 NA Percentage : 1.7542639970337413%

포함된 비율이 크지 않기 때문에 , dropna() 사용

**■ 이상치 처리**

###############################################################

# 이상치 처리

# 이상치 처리

# 최빈값 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)

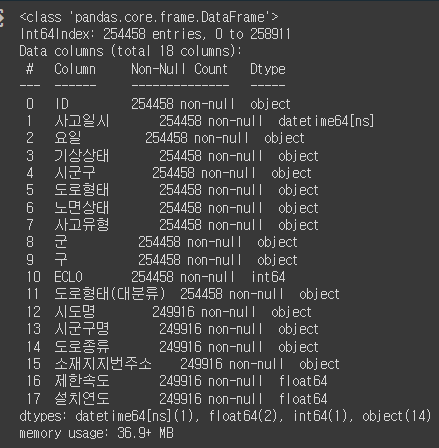
train['ECLO'] = replace\_outlier(train , 'ECLO') ###############################################################

|  |  |  |
| --- | --- | --- |
| **Ⅳ** |  | **모델 구축 및 검증** |

|  |  |
| --- | --- |
| 1 | 모델 구축 |

1) 변수 선택

먼저 모델을 만들기 전에 필요없는 컬럼을 제거한다.



위처럼 모든 컬럼을 사용할 수는 없기 때문에 , 변수 선택을 지금까지 분석한 내용에 근거하여 해야 한다.

< 제거 대상 >

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.

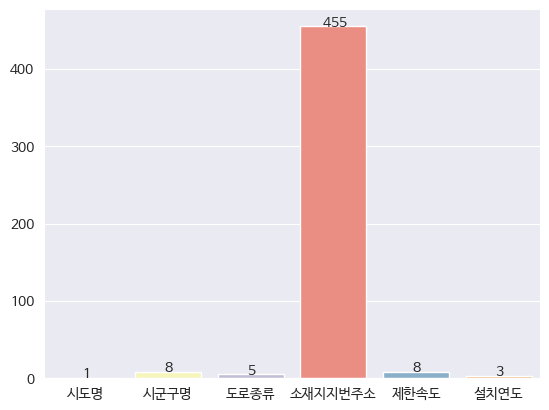
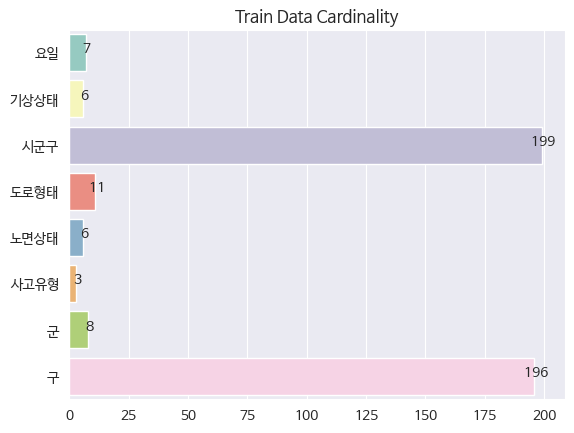
소재지지번주소 -> 시군구와 마찬가지이다.

2) 모델 구축

2-1) 인코딩

우리의 최종목표는 인명피해 심각도에 따라서 지역별로 군집분석을 하는 것이기 때문에 , 데이터에 인명피해 심각도에 대한 데이터가 반영이 되어있어야 한다.

데이터의 거의 대부분이 , 범주형 변수이기 때문에 수치형으로 변환을 하지 않으면 모델 학습에 사용하지 못한다.



범주형 데이터에서 우리가 사용하는 컬럼만 봐도 , 카디널리티가 적은 것이 아니기 때문에 , 적절한 인코딩 방법을 사용하여야 한다.

채택 방법 : TargetEncoding

타겟 인코딩은 한 범주에 대한 예측하려는 타켓(종속변수)의 평균값으로 해당 범주를 인코딩 하는 방식이다.

이럴 경우 , 종속변수에 대한 정보도 데이터에 반영이 되기 때문에 , 군집분석을 할 때 좋다고 생각하여 채택.

#############################################################################

!pip install category\_encoders

from category\_encoders.target\_encoder import TargetEncoder

categorical\_features = ['요일' , '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)

#############################################################################

2-2) 군집분석

#############################################################################

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score

def elbow(df):

sse = []

for i in range(1 , len(df)):

km = KMeans(n\_clusters = i)

km.fit(df)

sse.append(km.inertia\_)

sns.lineplot(

x = range(1 , len(df)) ,

y = sse ,

marker = 'o' ,

color = 'g'

)

plt.title("Elbow Method")

plt.xlabel("number of cluster")

plt.ylabel("sum of square error")

def silhouette(df):

silhouette = []

for i in range(2 , len(df)):

km = KMeans(n\_clusters = i)

km.fit(df)

score = silhouette\_score(df , km.labels\_)

silhouette.append(score)

sns.lineplot(

x = range(2 , len(df)) ,

y = silhouette ,

marker = 'o' ,

color = 'g'

)

plt.title("Silhouette Method")

plt.xlabel("number of cluster")

plt.ylabel("Score")

#############################################################################

군집 분석을 할 때 지역을 판별해야 하기 때문에 , 시군구가 인덱스로 오게 데이터를 가공한다.

#############################################################################

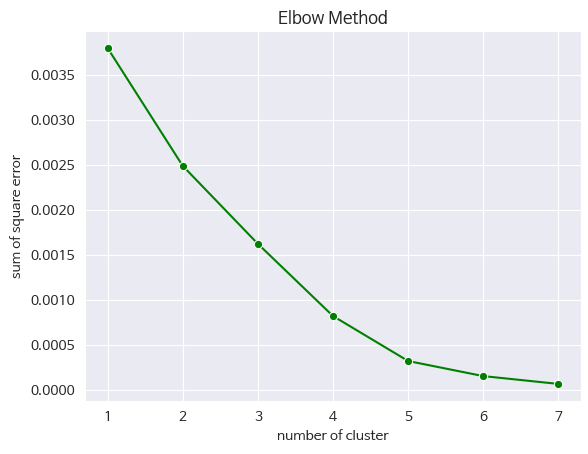
merged\_data.drop(['시군구'] , axis = 1 , inplace = True)

merged\_data\_group = merged\_data.groupby(['구'])['요일' , '도로형태' , '노면상태' ,

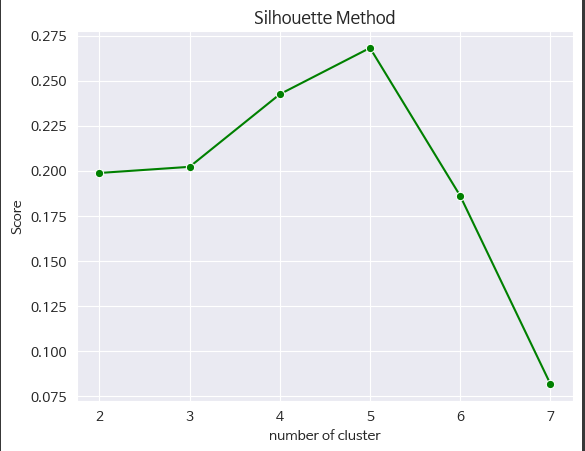
'사고유형' , '군' , '도로종류' , '제한속도'].mean()

#############################################################################

해당 코드를 통해서 , ECLO의 평균값으로 인코딩 된 각 지역 데이터들을 군집분석을 하기 위해서 적절 한 군집수를 찾기 위해 , 위에서 선언한 두개의 함수를 사용한다.



엘보우 방법의 경우 , 군집이 되는 중앙점 , 즉 포인트에서 어느정도로 데이터들이 떨어져 있는지를 보는 오차를 표시해서 , 아래로 내려갈수록 오차가 적지만 군집의 수는 늘어나기 때문에 군집분석의 의미가 없어지므로 가장 많이 떨어지는 부분 , 즉 해당 그래프에서는 3~4개가 적절하다.



실루엣 분석의 경우 , 군집의 밀집도를 나타낸다. 즉 이 그래프에서는 올라가는 지점이 가장 좋은데 , 5개가 가장 높지만 5개의 군집으로 만들 경우 너무 많을 수도 있기 때문에 3~4개가 적절하다고 판단이 가능하다.

#############################################################################

km = KMeans(n\_clusters = 4)

km.fit(merged\_data\_group)

merged\_data\_group['Cluster'] = km.predict(merged\_data\_group)

#############################################################################

Kmeans 알고리즘을 사용하여 , 군집분석을 시행

#############################################################################

from sklearn.decomposition import PCA

plt.figure(figsize = (8 , 8))

pca = PCA(n\_components = 2)

data\_pca = pca.fit\_transform(merged\_data\_group.drop(['Cluster'] , axis = 1))

data\_pca = pd.DataFrame(data\_pca , columns = ['PC1' , 'PC2'])

data\_pca['Cluster'] = list(merged\_data\_group['Cluster'])

sns.scatterplot(x = 'PC1' ,

y = 'PC2' ,

hue = 'Cluster' ,

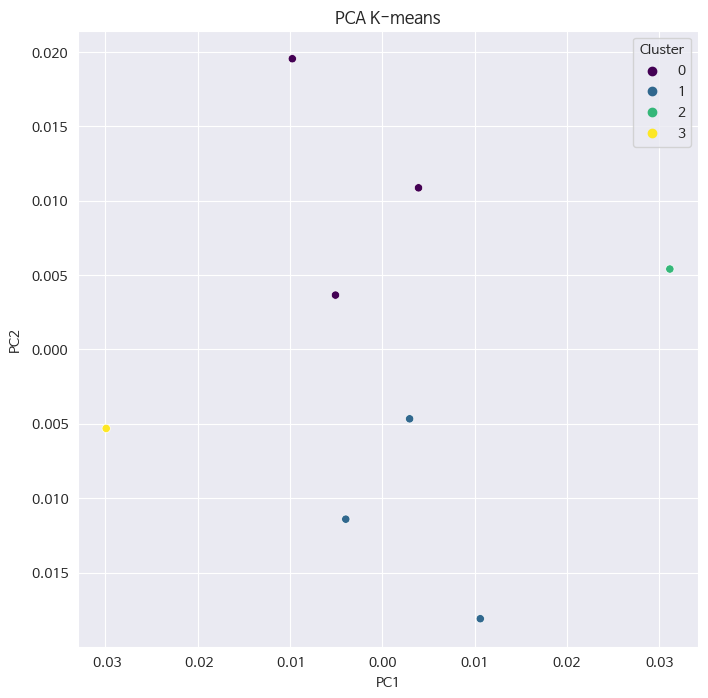
data = data\_pca ,

palette = 'viridis')

plt.title("PCA K-means")

#############################################################################

군집 그래프를 그릴 경우 , 군집분석에 사용한 데이터는 8개정도의 컬럼을 가지고 있어서 다차원 그래프를 그릴 수는 없으므로 , 주성분 분석(PCA)를 통해서 가장 설명력이 있는 두개의 데이터로 차원을 줄인 후에 , 그래프를 출력한다.



2-3) 모델 구성

여기에서 얻은 군집을 레이블로 하여 , 새로운 데이터를 재구성 한 뒤에 , 모델을 학습 시킨다.

사용 모델 : RandomForestClassifier

|  |  |
| --- | --- |
| 2 | 모델 검증 |

1). 모델 검증 방법

모델 검증 방법은 , 분류 문제이기 때문에 , f1 score , precision , recall등을 한번에 볼 수 있는 classification\_report를 사용하여 모델 검증을 시행한다.

2) 모델 평가 과정

##################################################################

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

trainX , validX , trainY , validY = train\_test\_split(smote\_df , smote\_target , test\_size = 0.2 , random\_state = 42)

rf\_model = RandomForestClassifier()

rf\_model.fit(trainX , trainY)

from sklearn import metrics

prediction = rf\_model.predict(validX)

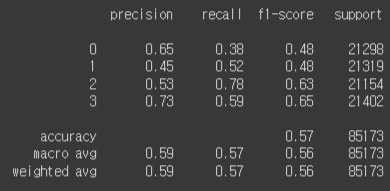
print(metrics.classification\_report(validY , prediction))

#################################################################

|  |  |  |
| --- | --- | --- |
| **Ⅴ** |  | **결론** |

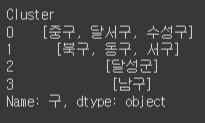
|  |  |
| --- | --- |
| 1 | 분석 결과 |

* + 1. 모델 성능 평가



Classification\_report() 함수를 사용해서 , 성능을 평가하였고 , 모델의 성능 자체는 좋지 못하였다. 모델의 성능을 올리기 위해서 , 모델의 하이퍼패러미터 튜닝 작업이나 , 수치형으로 바꾼 후의 데이터에 대한 스케일링 , 범주형의 binning등을 하지 않았기 때문에 아직 개선의 여지는 있다고 생각한다.

* + 1. 해석



군집 분석의 결과를 보면

0번 군집 : 중구 , 달서구 , 수성구

1번 군집 : 북구 , 동구 , 서구

2번 군집 : 달성군

3번 군집 : 남구

이렇게 되었다. 지금까지 분석한 내용들을 토대로 보면

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.

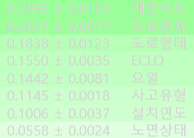
1. 2번 군집

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.

* + 1. 중요 변수



모델 기반 변수 선택을 위해 , PermutationImportance를 사용해서 모델의 중요도를 시각화 함.

지금까지 분석에서 얻은 성찰대로 , 제한속도 , 도로종류 , 도로형태등이 중요하고 그리고 위험도의 직접적인 척도가 되는 ECLO의 경우에도 중요하게 나왔다 , 사고유형이 생각보다 낮게 나왔지만 , 종속변수와는 독립으로 판단이 된 노면상태의 경우에도 가장 낮게나왔다.

그러나 어느정도 모델의 성능은 올라가며 , 내려가는 수치는 보이지 않기 때문에 모든 컬럼이 유의미한 정보를 갖는다고 판단이 된다.

|  |  |
| --- | --- |
| 2 | 시사점 및 개선점 |

1. 시사점   
 1) 교통사고의 발생 시기는 주말이 많다.

-> 따라서 주말의 교통안전 강화가 필요

2) 교통사고의 주요 발생 지역은 도로교통법상 과속 단속 구역

-> 따라서 운전자의 교통안전 의식 제고와 과속 단속 강화가 필요

3) 교통사고의 위험이 높은 도로의 종류가 존재

-> 우선순위를 결정하여 , 사고 대처 필요

4) 교통사고의 위험도는 차대차 사고가 가장 높다

-> 자동차에 대한 기술적인 부분 뿐만 아니라 , 정기적으로 운전자들의 운전 능력 파악 필요.

2. 개선점

1) 교통사고 데이터의 품질 향상

-> 프로젝트에서 활용된 교통사고 데이터는 대구시에서 수집한 데이터를 기반으로 하였으나, 일부 데이터의 부정확성 및 누락으로 인해 분석 결과에 제한이 있었습니다. 따라서 교통사고 데이터의 품질을 향상시키기 위한 노력이 필요.

2) 교통사고 분석의 심층성 강화

-> 향후 교통사고 분석의 심층성을 강화하여, 교통사고의 발생 원인을 보다 정확하게 파악하고, 이를 바탕으로 보다 효과적인 예방 및 감소 방안을 마련할 필요가 있음.

3) 교통사고 예방 및 감소 방안의 실효성 제고

-> 프로젝트에서 도출된 교통사고 예방 및 감소 방안은 정책 수립의 기초자료로 활용될 수 있으나, 실제 정책으로 실행되기 위해서는 실효성을 제고하기 위한 노력이 필요. 이를 위해서는 방안의 타당성 검토 및 실효성 평가 등을 통해 방안의 개선을 지속적으로 추진해야함.

4) 기계학습 모델의 성능 향상 필요

-> 데이터를 도로에 국한하는 것이 아닌 여러 방면에서 현재 상황을 분석하여 , 현재 문제 상황을 잘 반영할 수 있는 데이터셋을 만드는 것이 중요. 그 외에도 , 모델 튜닝 , 모델 선택 등 기술적인 부분에서도 기계학습 모델에 따라서 다루는 데이터의 특징이 다르기 때문에 모델을 잘 선택하여 , 문제에 맞게 모델의 패러미터를 적절히 설정 하는 것이 중요함.

|  |  |  |
| --- | --- | --- |
| **Ⅵ** |  | **산출물 목록** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **목록 코드** | **목록 명** | **작성일자** | **작성자** | **비고** |
| CNR-001 | 데이터 유효성 검사 함수 파이썬 코드 | 2023.11.30 | 김대양 |  |
| CNR-002 | 그래프 작성 함수 파이썬 코드 | 2023.12.02 | 김대양 ,  서동윤 |  |
| CNR-003 | Cramer’s V 상관계수 계산 함수 파이썬 코드 | 2023.12.05 | 김대양 |  |
| CNR-004 | t-test 함수 파이썬 코드 | 2023.12.05 | 김대양 |  |
| CNR-005 | Tukey 이상치 추출 함수 파이썬 코드 | 2023.12.10 | 서동윤 |  |
| CNR-006 | 이상치 상한 대체 함수 파이썬 코드 | 2023.12.12 | 서동윤 |  |
| CNR-007 | 엘보우 메소드 파이썬 코드 | 2023.12.15 | 김대양 |  |
| CNR-008 | 실루엣 검사 함수 파이썬 코드 | 2023.12.15 | 김대양 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |