# 누아보 해커톤

## -범죄 유형 분류 AI-


◆ 제 출 일 자 : 2024년 2월 21일

◆ 담당 교수님 : 이석원 교수님

◆ 학 과 : 인공지능융합학과

◆ 팀 : 4팀

◆ 학 번/이 름 : 202021931 이제이
202126162 성예담
201820196 김진찬

# 목 차

# Ⅰ. 라이브러리 불러오기

```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
import random
import os
import matplotlib.pyplot as plt
import seaborn as sns

# 한글 폰트
from matplotlib import font_manager, rc
font_path = "./malgun.ttf"
font_name = font_manager.FontProperties(fname=font_path).get_name()
rc('font', family=font_name)

import scipy.linalg as la
from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score

# !pip install catboost
import catboost
from catboost import CatBoostClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier

# !pip install optuna
import optuna
from optuna import Trial, visualization
from optuna.samplers import TPESampler
```

# Ⅱ. 랜덤 시드 고정

```
def seed_everything(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)

seed_everything(42)
```

해당 코드는 랜덤한 결과를 얻는 프로그램에서 결과의 재현성을 보장하기 위해 사용된다.

- random.seed(seed): random 모듈의 난수 생성기의 시드를 설정한다.
- os.environ['PYTHONHASHSEED'] = str(seed): os 모듈의 해싱 시드를 지정된 시드 값으로 설정한다.
- np.random.seed(seed): numpy 모듈의 난수 생성기의 시드를 설정한다.
- seed_everything(42): 시드를 42로 설정함으로써 이 함수를 호출한 이후 랜덤한 프로그램을 실행할 때, 모든 실행에서 동일한 난수 시퀀스를 얻을 수 있도록 한다. 이는 디버깅과 결과의 재현성을 보장하는 데 유용하다.

## Ⅲ. 데이터 파악

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84406 entries, 0 to 84405
Data columns (total 20 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   ID        84406 non-null  object
 1   월         84406 non-null  int64
 2   요일        84406 non-null  object
 3   시간        84406 non-null  int64
 4   소관경찰서     84406 non-null  int64
 5   소관지역      84406 non-null  float64
 6   사건발생거리    84406 non-null  float64
 7   강수량(mm)   84406 non-null  float64
 8   강설량(mm)   84406 non-null  float64
 9   적설량(cm)   84406 non-null  float64
 10  풍향        84406 non-null  float64
 11  안개        84406 non-null  float64
 12  짙은안개      84406 non-null  float64
 13  번개        84406 non-null  float64
 14  진눈깨비      84406 non-null  float64
 15  서리        84406 non-null  float64
 16  연기/연무     84406 non-null  float64
 17  눈날림       84406 non-null  float64
 18  범죄발생지     84406 non-null  object
 19  TARGET    84406 non-null  int64
dtypes: float64(13), int64(4), object(3)
memory usage: 12.9+ MB
```

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17289 entries, 0 to 17288
Data columns (total 19 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   ID        17289 non-null  object
 1   월         17289 non-null  int64
 2   요일        17289 non-null  object
 3   시간        17289 non-null  int64
 4   소관경찰서     17289 non-null  int64
 5   소관지역      17289 non-null  float64
 6   사건발생거리    17289 non-null  float64
 7   강수량(mm)   17289 non-null  float64
 8   강설량(mm)   17289 non-null  float64
 9   적설량(cm)   17289 non-null  float64
 10  풍향        17289 non-null  float64
 11  안개        17289 non-null  float64
 12  짙은안개      17289 non-null  float64
 13  번개        17289 non-null  float64
 14  진눈깨비      17289 non-null  float64
 15  서리        17289 non-null  float64
 16  연기/연무     17289 non-null  float64
 17  눈날림       17289 non-null  float64
 18  범죄발생지     17289 non-null  object
dtypes: float64(13), int64(3), object(3)
memory usage: 2.5+ MB
```

train data와 test data를 info()로 확인해본 결과 누락값은 없었다. ID, 요일, 범죄발생지는 문자형이며, 월, 소관경찰서, train data의 TARGET은 정수형이다. 나머지 컬럼은 모두 실수형인 것을 확인할 수 있었다. 표로 정리한 결과는 다음과 같다.

| Dtype | Column |
|---|---|
| 실수형(float64) | 소관지역, 사건발생거리, 강수량(mm),<br>강설량(mm), 적설량(cm), 풍향, 안개,<br>짙은안개, 번개, 진눈깨비, 서리, 연기/연무, 눈날림 |
| 문자형(object) | ID, 요일, 범죄발생지 |
| 정수형(int64) | 월, 소관경찰서, TARGET(train data) |

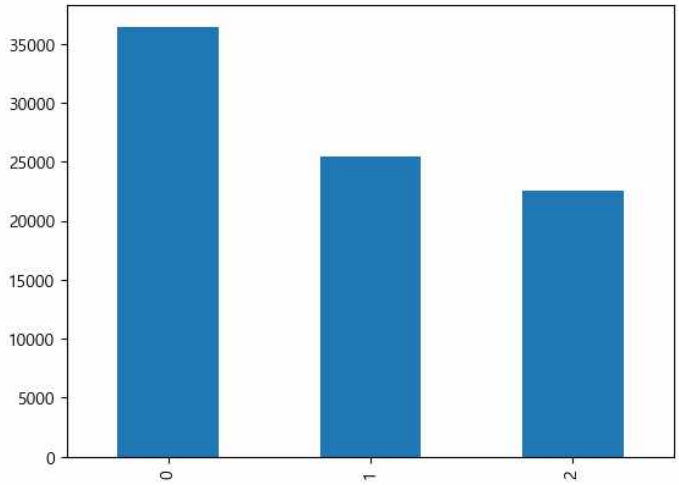
EDA를 하기 전 세운 가설은 다음과 같다.

- **가설 1**: 범죄발생지 유형을 보아 유동인구가 많은 시간대에 범죄가 많이 발생할 것이다.
- **가설 2**: 특정 범죄발생지에서 특정 범죄가 많이 발생할 것이다. (예를 들어 주자장의 경우 빈 차를 노린 절도가 많이 발생했을 것이다.)
- **가설 3**: 비나 눈이 오는 날 교통사고로 인한 상해가 많이 발생했을 것이다.

# Ⅳ. EDA

### 가. 타겟 변수의 분포 확인

타겟 변수가 고르게 분포되어 있는지 확인했다.



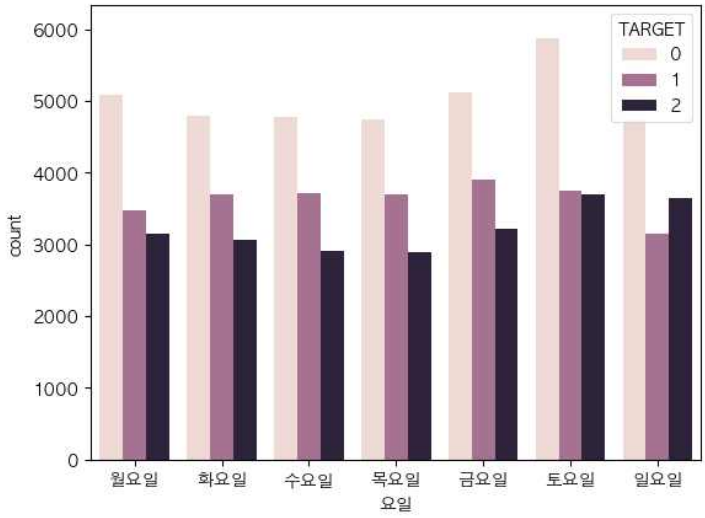

```
범죄 유형 비율 (%)
0    43.187688
1    30.089093
2    26.723219
Name: TARGET, dtype: float64
```

(■ 0 → 강도 / ■ 1 → 절도 / ■ 2 → 상해)

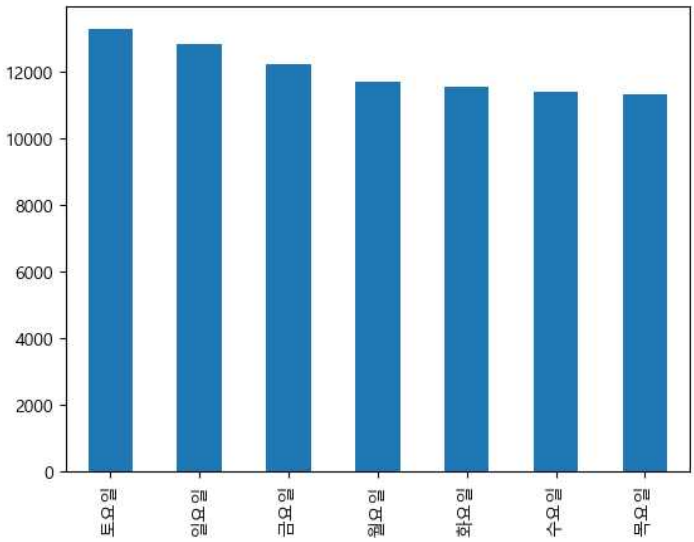
타겟 변수에 불균형이 있는 것을 확인했다. 강도의 빈도가 가장 많았다.

### 나. 요일별 타겟값의 분포

요일별 타겟값이 어떻게 분포되어 있는지 확인해본 결과 다음과 같았다.



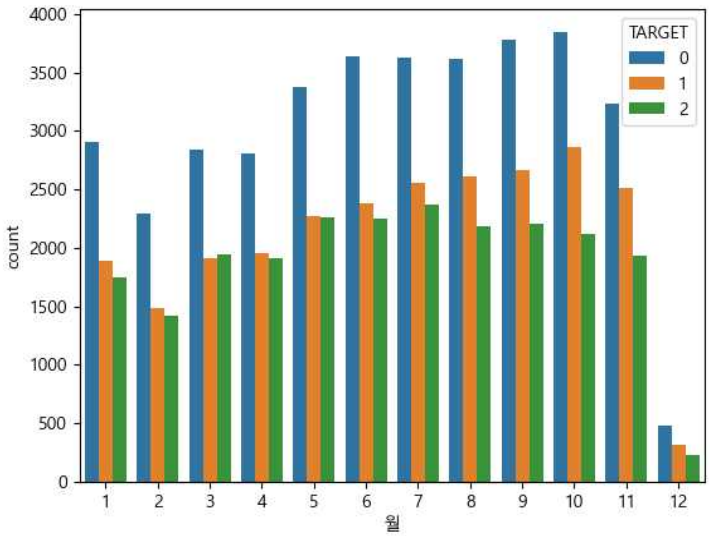

월요일, 화요일, 수요일, 목요일, 금요일은 비슷한 추이로 타겟값이 분포되어 있다. 토요일에 강도의 발생이 가장 많았으며, 일요일은 상해의 빈도수가 많은 특이한 분포를 볼 수 있다.




또한 주말과 금요일에 범죄 발생 수가 가장 많은 것을 볼 수 있다. 이는 유동인구가 많기 때문에 그런 것으로 예상된다.

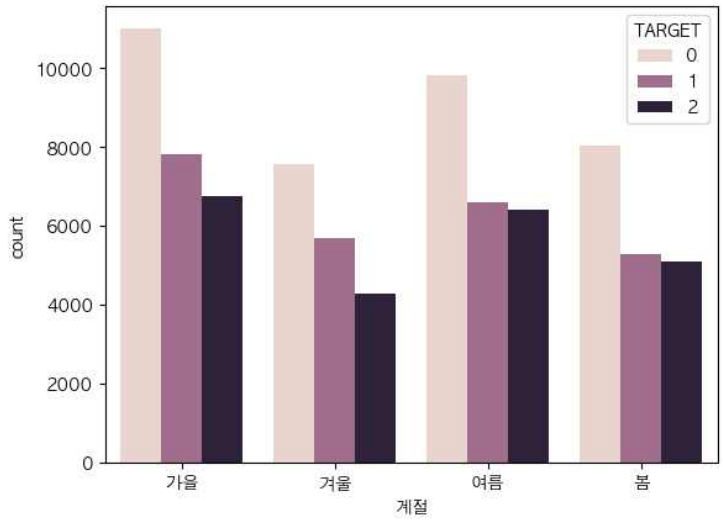
### 다. 월별 타겟값의 분포

월별 타겟값이 어떻게 분포되어 있는지 확인해본 결과 다음과 같았다.



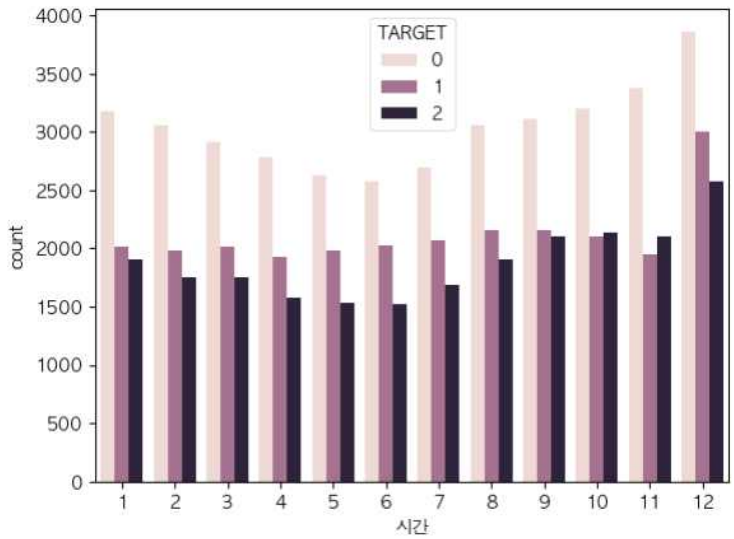

12월 데이터의 수가 확연히 적다는 것을 확인할 수 있다. 만약 데이터가 다음 해의 연초에 조사되었다면, 바로 직전인 12월의 데이터는 제대로 집계되지 않았을 수 있다.

월별 타겟값의 분포를 한눈에 파악하기 어려워 봄, 여름, 가을, 겨울 계절별로 묶어 확인해본 결과 아래 그림과 같았다.

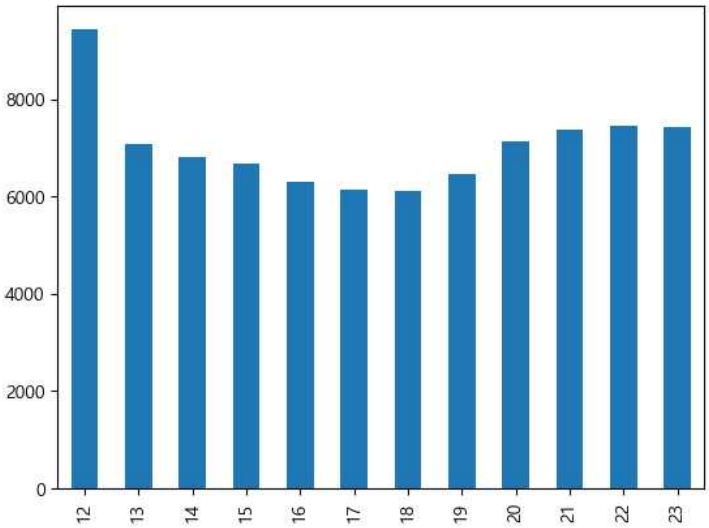



가을에 범죄 발생량이 가장 많았으며 그 중 강도의 수가 가장 많았다. 겨울에는 범죄 발생량이 가장 적었으며, 이는 12월 데이터가 적기 때문인 것으로 파악된다.

**라. 시간별 타겟값의 분포**

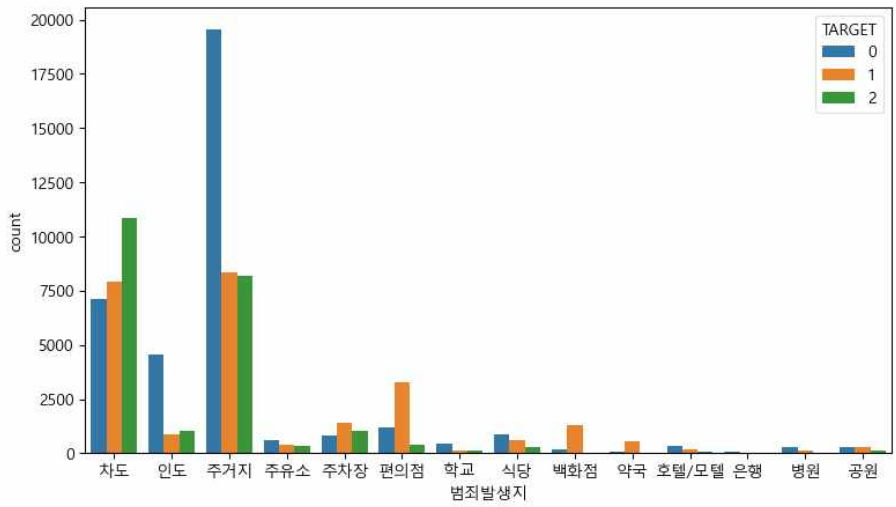



시간별로 타겟값의 분포를 확인해본 결과 12시에 범죄가 가장 많이 일어나는 것으로 파악된다. 10, 11 시간대에는 상해가 절도보다 많이 일어나는 것을 알 수 있다. 백화점 등에 모든 시간대가 골고루 존재하는 것을 파악했기 때문에 시간은 오후 12시부터 오후 23시인 것으로 추정된다.




12시부터 6시까지 범죄 발생 수가 줄어드는 경향이 있다가 그 이후부터 다시 늘어나는 경향이 있다. 12시는 점심시간, 18시 이후는 퇴근시간(저녁시간)과 가까우므로 유동인구가 많기 때문에 범죄 발생 수가 많은 경향이 나타난 것으로 보인다.

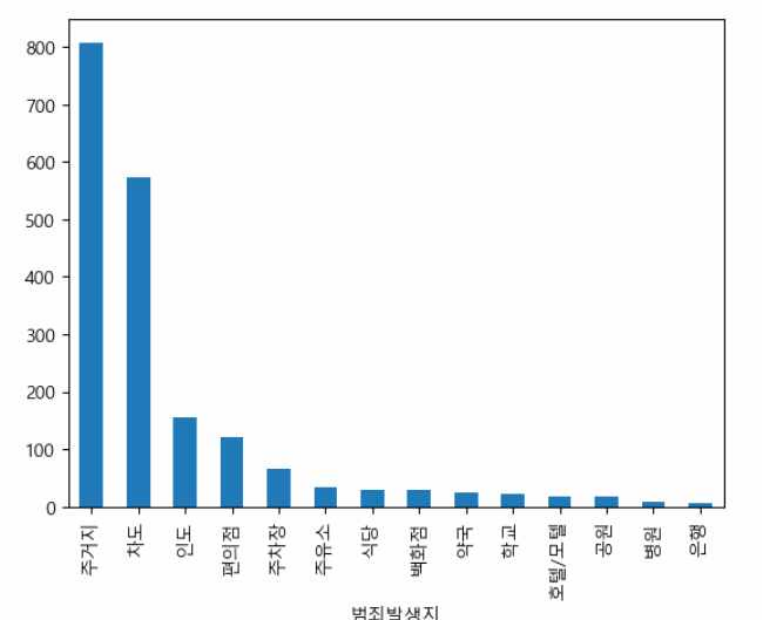
### 마. 장소에 따른 타겟값의 분포



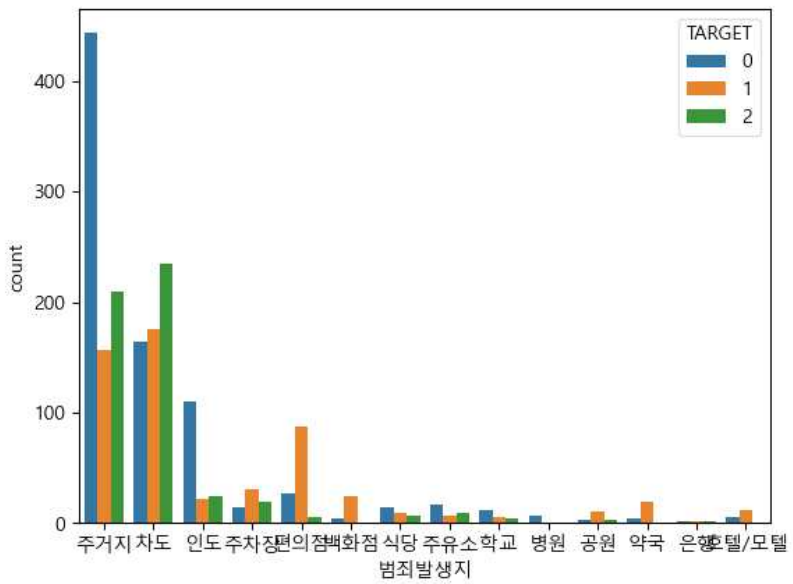

편의점, 백화점, 약국 등 판매업소에서 절도가 특히 많이 일어난다. 주차장에서도 절도가 많이 일어나는데 사람이 없는 빈 차가 많기 때문으로 예상된다. 차도에서 상해가 특히 많이 일어나는데 이는 교통사고로 추정된다. 주유소, 학교, 식당, 호텔/모텔, 은행, 병원, 공원은 데이터의 수가 적었으며 특이사항이 없었다.

### 바. 날씨에 따른 범죄발생지

#### 1) 비가 많이 오는 날

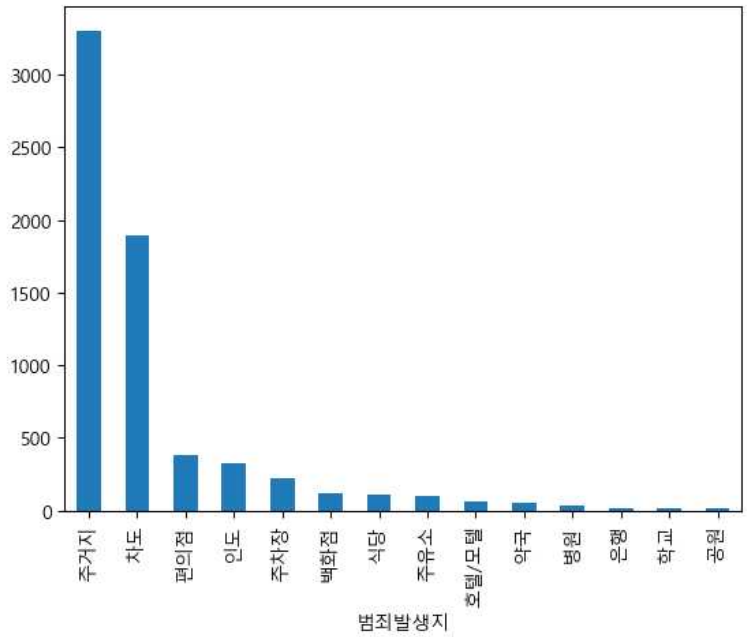



강수량이 200mm 이상인 비가 많이 오는 날 범죄발생지의 분포를 시각화 한 그래프이다. 주거지, 차도, 인도, 편의점 순으로 범죄가 많이 일어나는 것을 알 수 있다.

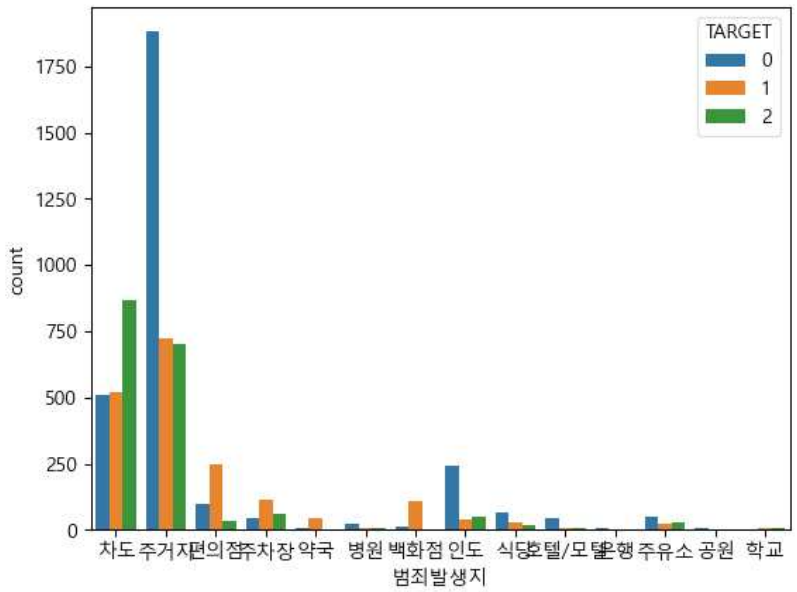



비가 많이 오는 날 주거지의 경우 강도의 발생량이 가장 많았다. 비가 오는 날 집에 침입해 절도를 하려다 몸싸움이 일어나는 등의 사건이 많이 일어나기 때문인 것으로 예상된다. 또한 상해, 절도 순으로 사건이 많았다. 차도의 경우 상해가 가장 많이 일어났는데 이는 비가 오는 날 교통사고 발생 건수가 증가하는 경향이 있기 때문에 그런 것으로 예상된다. 편의점은 절도가, 인도는 강도의 발생 건수가 많았다.

2) 눈이 오는 날

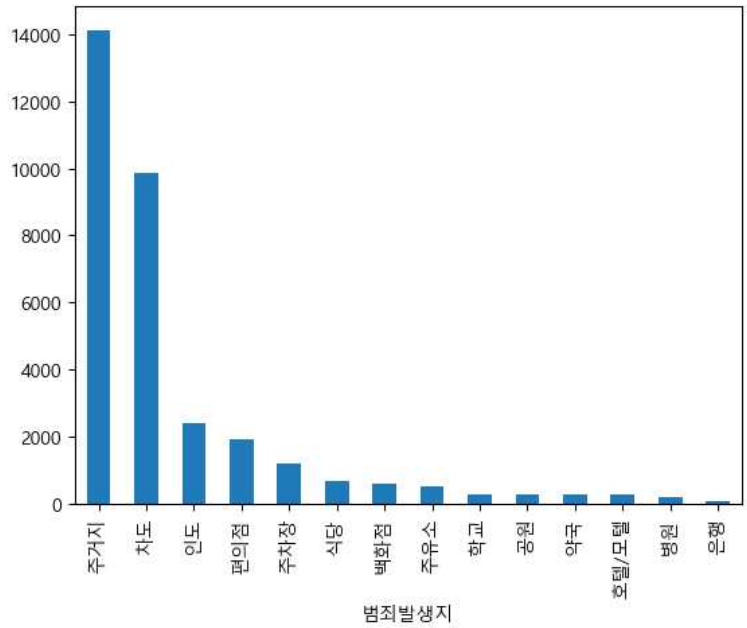



눈이 오는 날의 경우 주거지, 차도, 편의점, 인도, 주차장 순으로 범죄 발생 건수가 많았다.

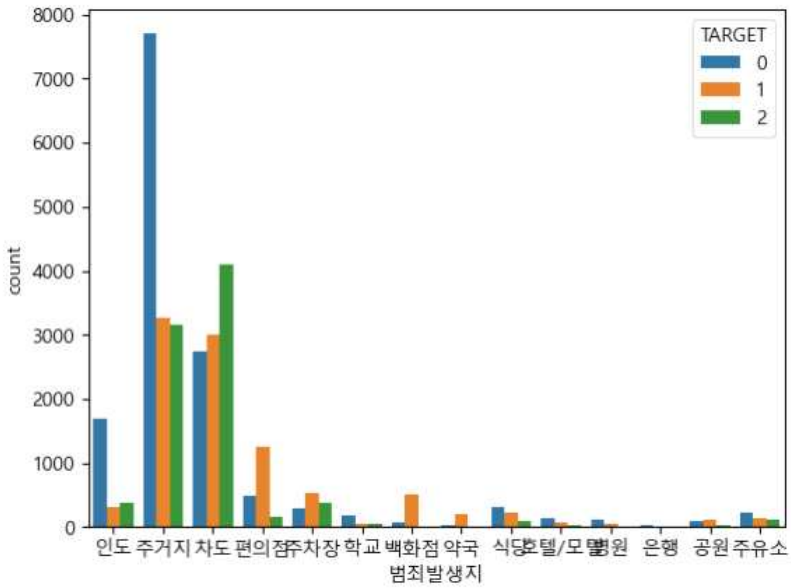



주거지의 경우 강도가 가장 많았으며, 절도와 상해의 건수는 비슷했다, 차도의 경우 상해가 가장 많았다, 편의점은 절도가, 인도는 강도의 발생량이 많았다.

3) 안개가 낀 날

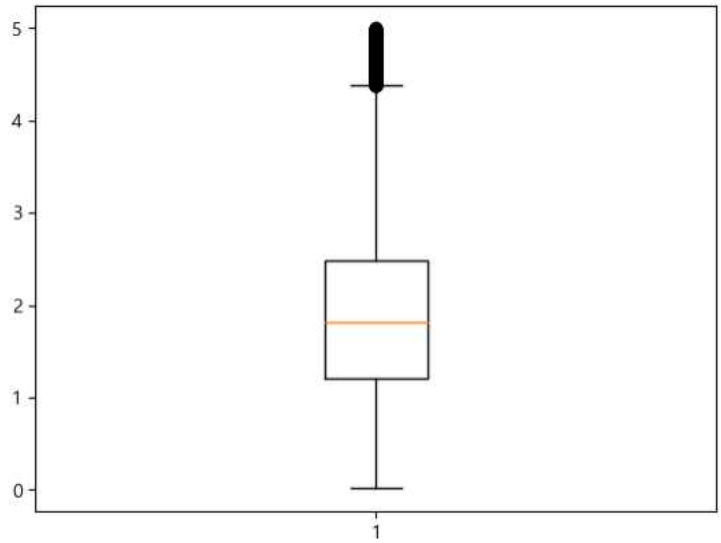



안개가 낀 날의 경우 주거지, 차도, 인도, 편의점 순으로 범죄 발생 건수가 많았다.

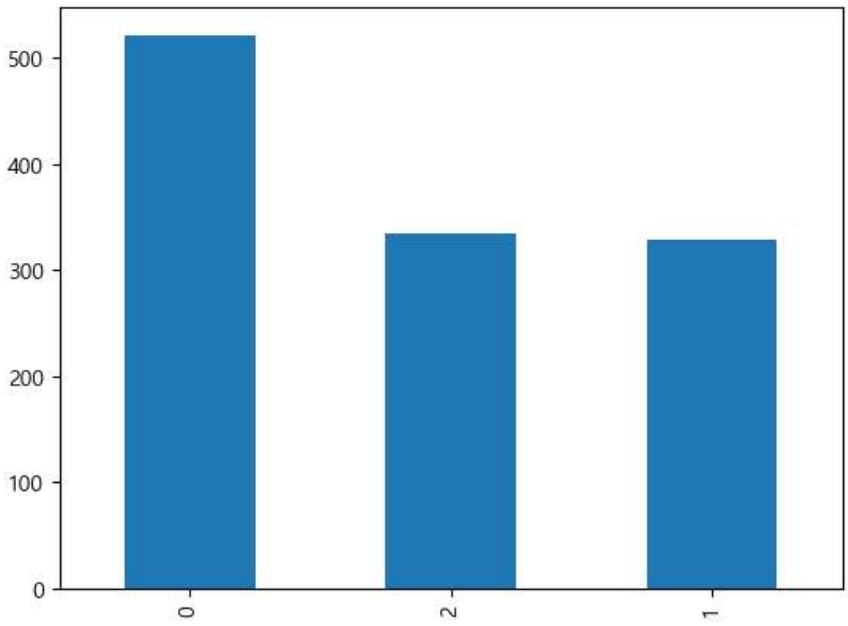



주거지의 경우 강도가 가장 많았으며, 절도와 상해의 건수는 비슷했다, 차도의 경우 상해가 가장 많았다, 편의점은 절도가, 인도는 강도의 발생량이 많았다.

**사. 사건 발생거리 이상치 확인**



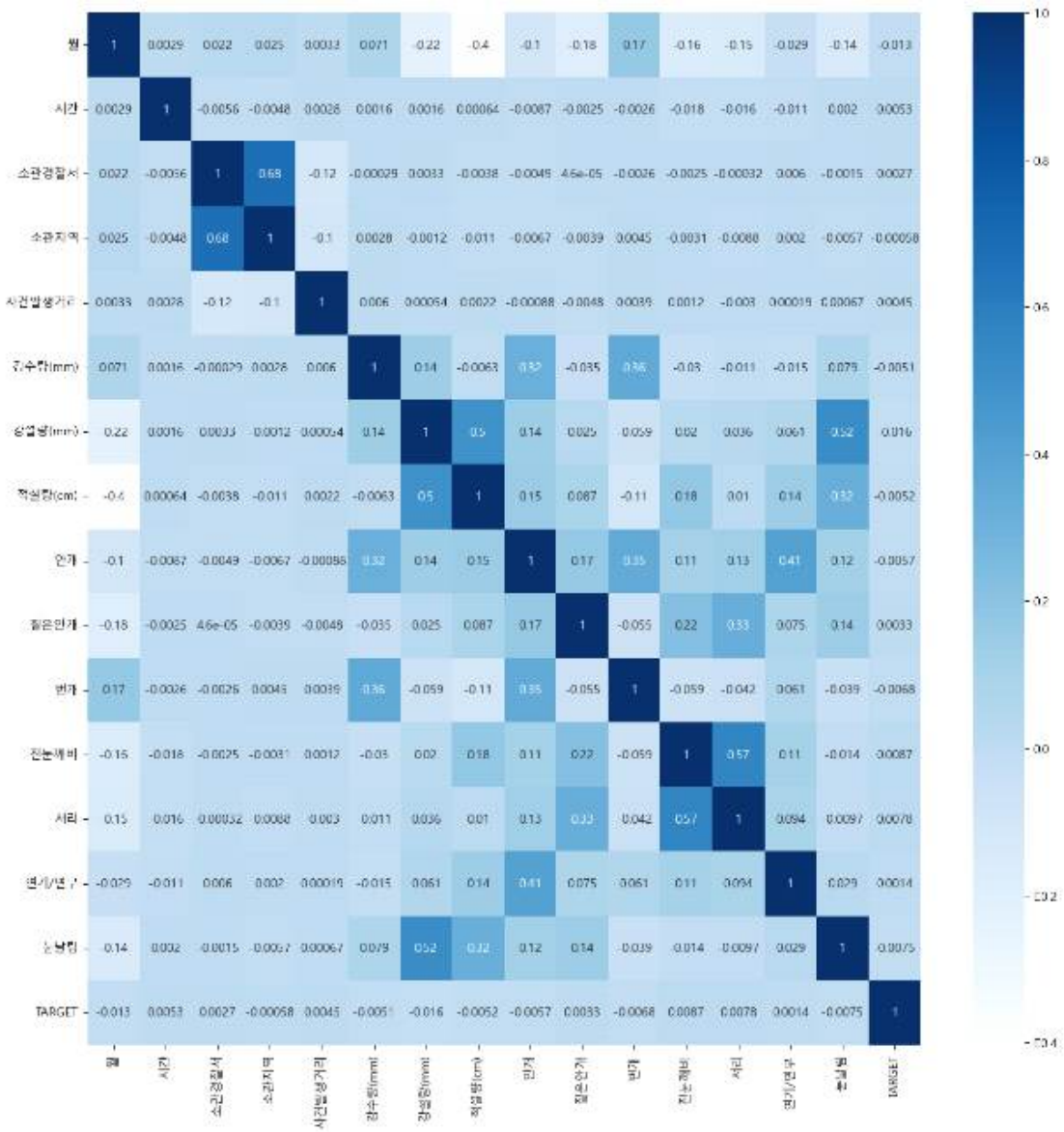

사건 발생 거리를 박스플롯으로 시각화해 본 결과 이상치가 존재한다는 사실을 알 수 있었다.
사건 발생 거리가 멀다고 특정 사건이 더 발생하진 않는다는 사실을 알 수 있었다.



# V. 인사이트 도출

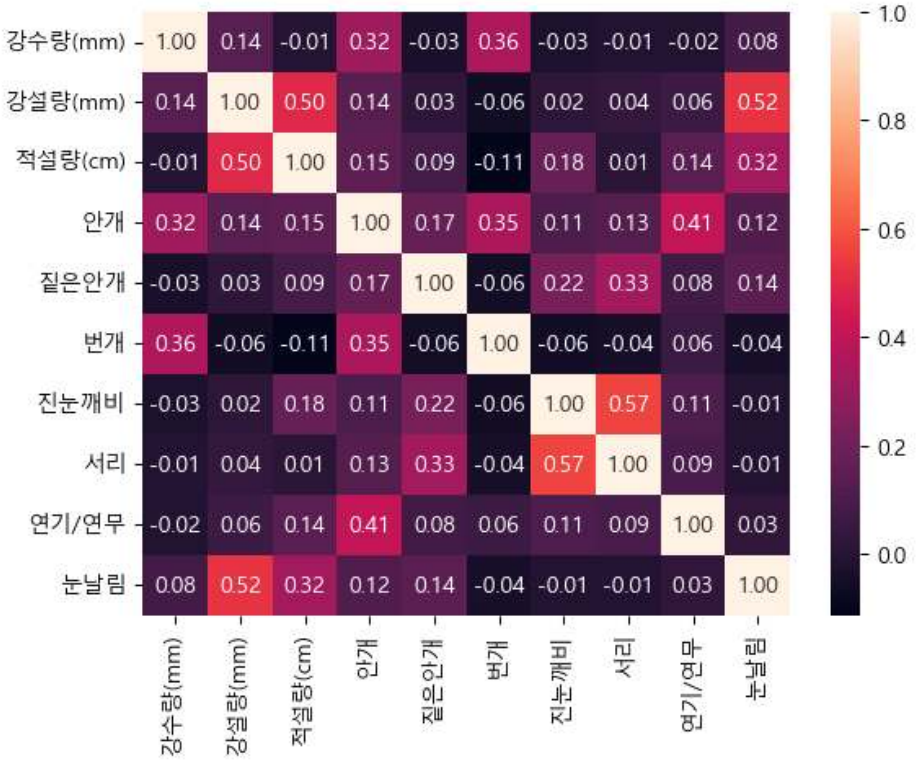
## 가. 상관계수 분석

두 변수 사이의 통계적 관계를 표현하기 위해 상관관계의 정도를 수치적으로 나타낸 계수인 상관계수 분석을 실시했다. 변수들의 상관계수를 시각화해 본 결과 다음과 같다.




<상관계수 값의 해석>

| 절대값 | 의미 |
|---|---|
| 0.9~1.0 | 매우 높은 음/양의 상관관계 |
| 0.7~0.9 | 높은 음/양의 상관관계 |
| 0.5~0.7 | moderate 음/양의 상관관계 |
| 0.0~0.1 | 상관관계가 거의 없음 |




기상 정보 칼럼만을 추출해 상관계수를 파악해 보았다. 서리와 진눈깨비, 강설량과 눈날림, 적설량과 강설량 등의 경우 서로 비슷한 변수들이기 때문에 다중공선성의 문제가 생길 수 있다. 겹치는 변수가 많기 때문에 차원을 축소하거나 일부 변수를 제거하는 방향으로 전처리를 진행해야한다.

### 나. 주성분 분석

겹치는 컬럼이 많아 원 데이터의 분포를 최대한 보존하면서 고차원 공간의 데이터들을 저차원 공간으로 변환할 수 있도록 주성분분석을 실시했다. 여러 변수들 간에 내재하는 상관관계, 연관성을 이용해 소수의 주성분으로 차원을 축소해 다중공선성으로 인해 모형이 잘못 만들어져 문제가 생기는 것을 방지할 수 있다.

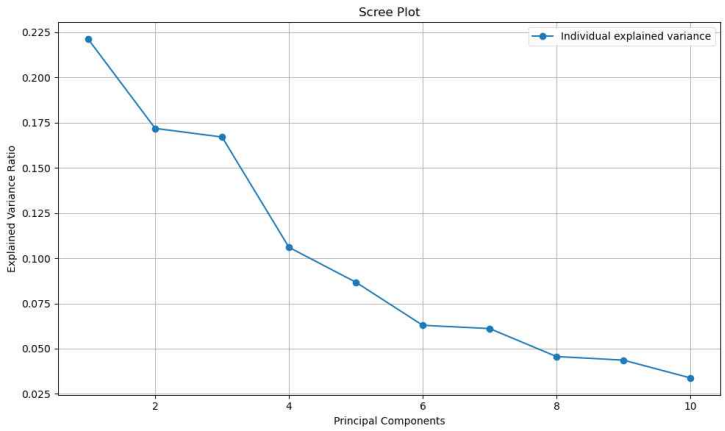
```
train_df_weather = train_df.iloc[:, 6:-2]

scaler = StandardScaler()
x = pd.DataFrame(scaler.fit_transform(train_df_weather), columns=train_df_weather.columns)

result_corr = la.eig(x.corr(method='pearson'))
sorted(result_corr[0], reverse=True) # 고윳값 1 이상: 4개
```

```
[(2.211699323888107+0j),
 (1.7185971071441544+0j),
 (1.670690177389527+0j),
 (1.0604511591233734+0j),
 (0.8675007092644866+0j),
 (0.6291664626700502+0j),
 (0.6109480637372497+0j),
 (0.4563025945032726+0j),
 (0.43640672977953454+0j),
 (0.3382376725002512+0j)]
```

Eigen value(고윳값)을 확인해본 결과 1 이상인 변수는 4개였다.

| | Variance_percent | Cumulative_variance_percent |
|---|---|---|
| **PC1** | 0.221170 | 0.221170 |
| **PC2** | 0.171860 | 0.393030 |
| **PC3** | 0.167069 | 0.560099 |
| **PC4** | 0.106045 | 0.666144 |
| **PC5** | 0.086750 | 0.752894 |
| **PC6** | 0.062917 | 0.815810 |
| **PC7** | 0.061095 | 0.876905 |
| **PC8** | 0.045630 | 0.922536 |
| **PC9** | 0.043641 | 0.966176 |
| **PC10** | 0.033824 | 1.000000 |

누적 기여율은 5개의 주성분을 사용했을 때 약 75%이다. 누적 기여율은 70% ~ 90%가 될 때 충분하다고 판단된다. 누적 기여율을 Scree Plot으로 표현해보면 아래 그림과 같다.




고윳값, 누적 기여율, Scree Plot을 모두 고려해 사용할 주성분의 개수를 5개로 결정했다.

주성분의 로딩 도표와 각 주성분을 라벨링한 결과는 다음과 같다.

| | PC1 | PC2 | PC3 | PC4 | PC5 |
|---|---|---|---|---|---|
| 강수량(mm) | 0.251653 | -0.396335 | -0.509887 | 0.458962 | -0.125921 |
| 강설량(mm) | 0.627183 | -0.405553 | 0.408644 | 0.124002 | -0.124026 |
| 적설량(cm) | 0.611565 | -0.191060 | 0.410006 | -0.146111 | -0.258386 |
| 안개 | 0.577419 | -0.151490 | -0.572917 | -0.229632 | 0.096957 |
| 짙은안개 | 0.417331 | 0.426579 | -0.010591 | 0.131014 | 0.717190 |
| 번개 | 0.100418 | -0.310087 | -0.710748 | 0.193304 | -0.023076 |
| 진눈깨비 | 0.448052 | 0.659107 | -0.040687 | 0.152295 | -0.393500 |
| 서리 | 0.428945 | 0.698675 | -0.113968 | 0.249536 | -0.125784 |
| 연기/연무 | 0.413192 | 0.048507 | -0.284302 | -0.768106 | -0.015619 |
| 눈날림 | 0.554856 | -0.366803 | 0.382921 | 0.174270 | 0.272573 |

- PC1: 강설량(+), 적설량(+), 안개(+), 눈날림(+) → 눈이 오는 정도
- PC2: 진눈깨비(+), 서리(+)
- PC3: 강수량(-), 안개(-), 번개(-) → 비가 오지 않는 정도
- PC4: 연기/연무(-) → 시야 확보 정도
- PC5: 짙은 안개(+) → 시야 방해 정도

# Ⅵ. 데이터 전처리

### 가. '요일' 변수 전처리

```
for i in range(len(train_df)):
    if train_df['요일'].iloc[i] in ['월요일', '화요일', '수요일', '목요일']:
        train_df.loc[i, '요일'] = '평일'
```

■ '요일'에서 월요일, 화요일, 수요일, 목요일은 특이한 경향을 보이지 않으며, 비슷한 경향을 보이므로 '평일'이라는 하나의 범주로 고려한다.

■ 금요일, 토요일, 일요일은 범죄 발생 수가 다소 높거나 상해의 비율이 높은 등 고유의 경향이 있으므로 분리하여 고려한다.

### 나. '시간' 변수 전처리

```
for i in range(len(train_df)):

    if train_df['시간'].iloc[i] in [12]:
        train_df.loc[i, '시간'] = '점심'

    elif train_df['시간'].iloc[i] in [13, 14, 15, 16, 17]:
        train_df.loc[i, '시간'] = '중간'

    elif train_df['시간'].iloc[i] in [18, 19, 20, 21, 22, 23]:
        train_df.loc[i, '시간'] = '저녁'
```

■ 12시의 범죄 발생률이 확연하게 높으며, 그 이후부터 18시까지 범죄 발생 수가 줄어드는 경향을 보인다. 또한 18시 이후부터는 다시 범죄 발생 수가 늘어나는 경향을 보인다. 이는 12시는 점심시간, 18시 이후부터는 퇴근시간(저녁시간)과 가까우므로 유동 인구의 수에 따른 것으로 보인다. 따라서 비슷한 경향을 보이는 시간대끼리 하나의 범주로 묶어 고려한다.

### 다. '범죄발생지' 변수 전처리

```
for i in range(len(train_df)):
    if train_df['범죄발생지'][i] in ['주유소', '학교', '식당', '호텔/모텔', '은행', '병원', '공원']:
        train_df.loc[i, '범죄발생지'] = '기타'
```

■ 주유소, 학교, 식당, 호텔/모텔, 은행, 병원, 공원은 train data에서 데이터의 수가 적으며, 특이한 경향을 보이지 않으므로 '기타'라는 하나의 범주로 고려한다.

### 라. 기상 정보 데이터 전처리

```
train_df_weather = features.iloc[:, 10:]
train_df = train_df.iloc[:, [0, 1, 2, 3, 4, 5, 16, 17]]
train_df = pd.concat([train_df, train_df_weather], axis=1)
```

■ 기상 정보끼리의 상관관계가 존재하기 때문에 강수량부터 눈날림 등의 기상 정보와 관련된 데이터를 Ⅴ에서 실시한 PCA(n=5)를 토대로 차원을 축소한다. 각 주성분은 다음과 같다.

- PC1: 눈이 오는 정도 / PC2: 진눈깨비, 서리와 높은 양의 상관관계를 갖는 변수 / PC3: 비가 오지 않는 정도 / PC4: 시야 확보 정도 / PC5: 시야 방해 정도

### 마. '사건발생거리' 이상치 처리

```
Q1 = train_df['사건발생거리'].quantile(0.25)
Q3 = train_df['사건발생거리'].quantile(0.75)
IQR = Q3 - Q1

outlier_condition = (train_df['사건발생거리'] < (Q1 - 1.5 * IQR)) | (train_df['사건발생거리'] > (Q3 + 1.5 * IQR))
distance_outlier = train_df[outlier_condition]
```

```
train_df.loc[distance_outlier.index, '사건발생거리'] = np.nan
df_no_outliers = train_df.dropna()
train_df = train_df.fillna(df_no_outliers.mean())
```

■ 사건발생거리가 이상치라고 해서 특정 범죄가 더/덜 발생하지 않았다. 따라서 이상치를 평균으로 대체하였다.

### 바. Label encoding

```
label_encoder = LabelEncoder()
columns_to_encode = ['요일', '시간', '범죄발생지']
for column in columns_to_encode:
    train_df[column] = label_encoder.fit_transform(train_df[column])

train_df['소관지역'] = train_df['소관지역'].astype("int64")
```

■ Label encoding은 범주형 데이터를 수치형 데이터로 변환하는 기술 중 하나이다. 머신러닝 알고리즘은 일반적으로 수치형 데이터를 입력으로 받기 때문에 범주형 데이터를 수치형으로 변환해야 한다. '소관지역'의 경우 실수형 데이터로 되어 있지만, 범주형 변수이기 때문에 정수형으로 변환했다.

### 사. 독립변수/종속변수 분리

```
target = train_df['TARGET']
del train_df['TARGET']
```

■ 본 모델의 목적은 종속변수인 범죄유형('TARGET')을 제외한 나머지 독립변수 변수들로 범죄 유형을 알아내는 것이다. 따라서 'TARGET'과 나머지 변수를 분리한다.

## Ⅶ. 모델링

optuna를 이용해 하이퍼파라미터 최적화 진행했으며, CatBoost, XGBoost 등의 모델을 학습시켜 보았다.

■ optuna는 하이퍼파라미터 최적화를 도와주는 파이썬의 프레임워크이다. 파라미터의 범위, 목록을 설정하면 매 Trial마다 파라미터를 변경하면서 최적의 파라미터를 찾아낸다.
■ CatBoost, XGBoost는 boosting 모델의 일종이다. boosting 알고리즘은 여러 개의 알고리즘이 순차적으로 학습-예측을 하면서 이전의 알고리즘이 틀린 데이터를 올바르게 예측할 수 있도록 다음 알고리즘에 가중치를 부여하여 학습과 예측을 진행한다.

```
x_tr, x_val, y_tr, y_val = train_test_split(train_df, target, test_size=0.2, stratify=target, random_state=42)
```

■ 해당 코드는 train data를 훈련용과 검증용으로 나누기 위해 사용된다. 훈련 데이터에 과도하게 적합된 모델은 새로운 데이터에 대한 일반화 능력이 낮을 수 있다. 따라서 검증 데이터를 사용하여 모델이 훈련 데이터뿐만 아니라 새로운 데이터에 대해서도 잘 일반화 되는지 확인해야 한다. 또한 하이퍼파라미터를 조정하는 과정에서도 검증 데이터를 사용한다.

```
weight = dict()

n_classes = 3
cnt = np.bincount(y_tr)
n_samples = cnt.sum()

for i in range(n_classes):
  weight[i] =  n_samples / (n_classes * cnt[i])
```

■ 해당 코드는 클래스 간 불균형을 해결하기 위해 클래스별 가중치를 계산한다.

### 가. CatBoost

```
# 모델 학습 + optuna

def objectiveCAT(trial: Trial, x_tr, y_tr, x_val, y_val, weight):
    param = {
        'iterations': trial.suggest_int('iterations', 100, 1000),
        'depth': trial.suggest_int('depth', 3, 8),
        'learning_rate': trial.suggest_float('learning_rate', 0.001, 0.1),
        'random_state': 42,
        'class_weights': weight,
        'cat_features': cat_cols}

    # 학습 모델 생성
    model = CatBoostClassifier(**param)
    cat_model = model.fit(x_tr, y_tr, verbose=True)

    # 모델 성능 확인
    pred = cat_model.predict(x_val)
    score = f1_score(y_val, pred, average="macro")
    return score
```

```
# 하이퍼 파라미터 튜닝

study = optuna.create_study(direction='maximize', sampler=TPESampler(seed=42))
study.optimize(lambda trial: objectiveCAT(trial, x_tr, y_tr, x_val, y_val, weight), n_trials=30)
```

■ iterations (반복 횟수): 모델이 학습 데이터를 몇 번 반복해서 학습할지를 결정하는 파라미터이다. 100에서 1000까지의 정수값을 시도한다.

■ depth (트리 깊이): 각 트리의 최대 깊이를 나타내며, 트리가 얼마나 복잡한 패턴을 학습할 수 있는지를 결정한다. 3에서 8까지의 정수값을 시도한다.

■ learning_rate (학습률): 각 시도에서 모델의 가중치를 얼마나 조정할지 결정하는 파라미터로, 0.001에서 0.1까지의 실수값을 시도한다.

■ random_state (랜덤 시드): 랜덤 시드를 설정하여 모델 학습의 재현성을 제공한다.

■ class_weights (클래스 가중치): 각 클래스에 대한 손실 함수의 가중치를 설정하는 데 사용된다.

■ cat_features (범주형 특성): CatBoost에서 범주형 변수로 처리해야 하는 열의 인덱스를 나타낸다.

### 나. XGBoost

```
# 모델 학습 + optuna

def objectiveXGB(trial: Trial, x_tr, y_tr, x_val, y_val, weight):
    param = {
        'n_estimators': trial.suggest_int('n_estimators', 50, 200),
        'max_depth': trial.suggest_int('max_depth', 5, 20),
        'learning_rate': trial.suggest_float('learning_rate', 0.001, 0.1),
        'random_state': 42,
        'scale_pos_weight': weight}

    # 학습 모델 생성
    model = XGBClassifier(**param)
    xgb_model = model.fit(x_tr, y_tr, verbose=True)

    # 모델 성능 확인
    pred = xgb_model.predict(x_val)
    score = f1_score(y_val, pred, average="macro")
    return score
```

```
# 하이퍼 파라미터 튜닝

study = optuna.create_study(direction='maximize', sampler=TPESampler(seed=42))
study.optimize(lambda trial: objectiveXGB(trial, x_tr, y_tr, x_val, y_val, weight), n_trials=30)
```

■ n_estimators (트리 개수): 생성할 트리의 개수를 나타내는 파라미터로, 많은 트리를 사용할수록 모델의 성능이 향상될 수 있으나, 과적합의 위험이 있으므로 적절한 값을 찾아야 한다. 50에서 200까지의 정수값을 시도한다.

■ max_depth (트리 최대 깊이): 각 트리의 최대 깊이를 나타내며, 모델이 학습 데이터의 복잡한 패턴을 학습할 수 있는 정도를 결정한다. 5에서 20까지의 정수값을 시도한다.

■ learning_rate (학습률): 각 트리의 예측값에 대한 가중치를 결정하는 파라미터로, 0.001에서 0.1까지의 실수값을 시도한다.

■ random_state (랜덤 시드): 랜덤 시드를 설정하여 모델 학습의 재현성을 제공한다.

■ scale_pos_weight (클래스 가중치): 각 클래스에 대한 가중치를 설정하는 데 사용된다.

## Ⅷ. 평가산식 산출

### 가. Catboost

```
predictions1 = cat.predict(x_val)

f1 = f1_score(y_val, predictions1, average='macro')
print(f'F1 Score: {f1}')
```

```
F1 Score: 0.5336710133993811
```

### 나. XGboost

```
predictions2 = xgb.predict(x_val)

f1 = f1_score(y_val, predictions2, average='macro')
print(f'F1 Score: {f1}')
```

```
F1 Score: 0.5253635966198621
```

■ F1-score: 정밀도(Precision)와 재현율(Recall)의 조화 평균을 나타내는 평가지표이다. 정밀도는 예측을 Positive로 한 대상 중에서 실제값이 positive로 일치한 데이터의 비율이며, 재현율은 실제값이 positive인 대상 중에서 예측을 positive로 한 데이터의 비율이다. F1-score을 계산할 때 macro 평균을 이용했으며, 이는 클래스별 F1-score를 산술평균한 것이다. F1-score은 클래스 간 불균형이 있는 상황에서 주로 사용되기 때문에 해당 평가지표를 선택했다.
■ CatBoost와 XGBoost의 F1-score을 비교해본 결과, CatBoost의 점수가 더 높아 최종적으로 Catboost 모델을 선택했다.

# Ⅸ. 고찰

범죄 유형을 분류하는 것은 여러 측면에서 중요하다. 범죄 유형을 분류하면 범죄의 성격과 특성을 파악할 수 있으며, 범죄의 원인을 이해할 수 있다. 범죄 유형을 분류하면 범죄 데이터를 분석하여 범죄 패턴을 발견할 수 있다. 이는 범죄 조사 및 수사에 도움이 될 뿐만 아니라 범죄 예방에도 유용하다. 예를 들어, 특정 유형의 범죄가 특정 시간대나 장소에서 자주 발생한다면 해당 시간대나 장소에 대한 감시를 강화할 수 있다.

본 연구는 주어진 범죄 발생 기록 데이터를 통해 범죄 유형을 분류를 할 수 있는 인공지능 모델을 개발하여 각각의 범죄 유형별로 높은 영향력을 갖는 변수들을 찾아 범죄 예방 인사이트를 제공하고자 한다. 주어진 데이터는 84406개의 레코드와 종속변수를 포함하여 20개의 피쳐 값을 갖는다. 데이터 자체적으로 범주형 종속변수가 주어졌기 때문에 지도학습-분류 문제로 접근하였다.

본격적인 인공지능 모델을 만들기 전, 데이터를 파악하는 것이 중요하다. 우선, 종속 변수를 파악하는 것이 중요하다. 종속변수를 확인한 결과, 타깃 변수의 불균형이 있음을 확인할 수 있었다. 종속 변수의 분포를 확인한 후, 종속변수와 독립변수와의 관계를 시각화하여 파악해 보려고 하였다. 그리고 유의미한 결과가 보이면 파생 변수를 만들어 모델의 정확도를 높일려고 하였다.

요일과 타깃 값과의 관계를 분석한 결과, 평일에는 비슷한 경향성을 보여 “평일”이라는 파생변수를 만들었다. 월과 타깃 값과의 관계를 눈에 쉽게 파악하기 위해 계절별로 파생변수를 만들어서 분석을 진행하였다. 시간과 타깃 값과의 관계를 분석한 결과 12시에 눈에 띄게 범죄 발생률이 높았고 잠시 줄었다가 다시 높아지는 경향을 보여, ‘점심’, ‘중간’, ‘저녁’으로 하나의 범주로 묶어 고려하였다. 범죄 발생지와 타깃 값의 관계를 파악한 결과, 주유소, 학교, 식당, 호텔/모텔, 은행, 병원, 공원은 특이한 경향을 보이지 않음으로 ‘기타’라는 파생 변수를 만들었다.

데이터를 확인하면 연속형 데이터들도 존재하지만 범주형 데이터들도 존재한다는 것을 알 수 있었다. 흔히 범주형 데이터가 독립변수에 속할 때, 이를 숫자로 인코딩하는 과정이 필요하다. 만일 해당 데이터에 순서가 없을 시, one-hot encoding을 해도 되나, 해당 데이터 같은 경우, 순서가 있었기 때문에 라벨인코딩을 통해 숫자를 인코딩하는 작업을 거쳤다.

데이터를 충분히 파악 하였다면, 모델과 상관성이 높은 변수들을 선별하는 단계를 거쳐야 한다. 이를 흔히 “변수 추출 단계”(Feature Selection)라고도 한다. 가장 보편적으로 활용되는 변수 추출 방법인 상관관계를 통해 불필요한 데이터를 제거하는 과정을 거쳤다.

데이터 전처리가 되었으면 데이터 특성에 맞는 모델을 선정하는 과정이 중요하다. 이를 위해서는 데이터의 특성, 그리고 상황에 맞는 모델을 선정할 필요가 있다. 본 연구에서는 단일 학습기로.분류 모델을 만드는 것보다는 다양한 모델을 결합하여 높은 성능을 얻을 수 있는 앙상블 모델을 채택하였다. 그중에서 약한 학습기를 결합하여 강한학습기를 만들어내는 앙상블 모델인 부스팅 모델을 선택하였다. 부스팅 모델을 선택하게 된 배경은 해당 데이터의 피쳐 들이 충분히 해당 도메인의 일반성을 지닌 데이터 셋이라고 판단하였고 모델의 일반성 보다는 모델이 정확도가 더 중요할 것이라고 판단하였기 때문이다. 또한 데이터 셋이 특출나게 크지도 않다고 판단하였다. 부스팅 모델 같은 경우에도 다양한 모델이 있지만 가장 대표적인 부스팅 모델인 xgboost와 주어진 데이터 특성상 범주형 데이터가 많이 포함이 되기 때문에 CatBoost 모델을 후보로 하였다. 모델을 돌린 후 가장 성능이 좋은 모델을 선정하기로 하였다. 모델을 돌린 결과, CatBoost 모델이 XGBoost보다 좋은 성과를 보이는 것을 알 수 있다. 이는 데이터 특성상 범주형 데이터가 많이 포함해 해당 결과가 나온 것이라고 유추할 수 있다.

부족한 점 및 아쉬운점:

이번 프로젝트를 진행하면서 아쉬운점과 부족한 점들이 많았던 것 같다.
가장 아쉬웠던 점 중 하나는 데이터가 주어져 해당 도메인 특성에 맞는 데이터를 구축하는 과정이 없었다는 점이다. 물론 데이터를 전처리하고 모델을 돌리는 과정을 통해 머신러닝적인 기법들에 대해서 다시 한번 환기할 수 있는 기회를 갖을 수 있다는 점에서 좋았지만 풀고자 하는 문제에 대해 창의적으로 접근해 볼 수 있는 기회는 적었던 것 같다.

부족한 점 같은 경우, 세부적인 데이터 분석 처리가 조금 미흡했던 것 같다. 예를 들어, 종속변수가 균일하게 분포하지 않을 경우에는 통상적으로 SMOTE, TomekLink와 같은 기법을 통해 오버 샘플링, 언더 샘플링을 한다.  해당 과정을 거쳤으면 더 좋은 결과가 있었을 것이라고 예상한다. 또한 특성 추출 과정에서도 PCA를 활용하였는데, 기상 정보의 대다수 데이터들은 binary한 특징을 갖고 있기 때문에 PCA를 통해 차원 축소를 하는 것은 유의미하지 않았었던 것 같다. 파생 변수를 생성하는 과정 또한 시간적인 여유가 됐었다면, 도메인 특성에 맞게 논문을 참고하여 유의미한 변수들을 생성할 수 있었을 것 같다