피할 수 없다면 예측해보자!

목차

"서울학교 안전사고, 11월 두번째 많아…위험개선 필요"

등록 2022.11.02 10:17:02 | 수정 2022.11.02 10:47:43

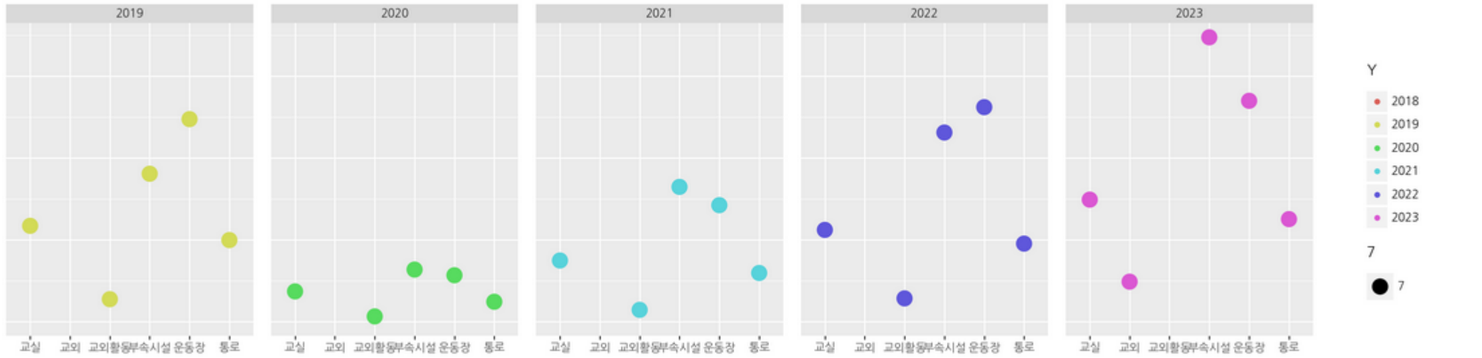
가 가

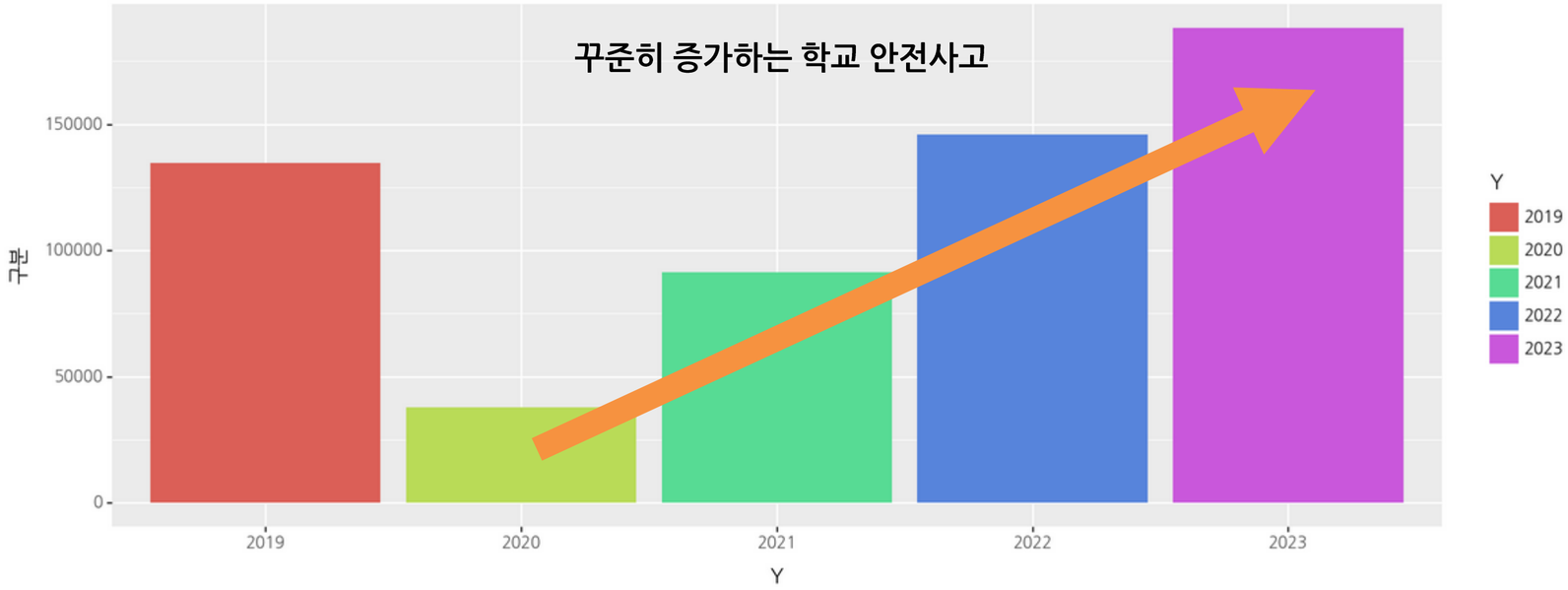
"응급처치만 알았다면"...부족한 학교 안전교육

학교서 안전교육받은 88% '기억 못해'
교사 "시청각 자료로 형식적 수업"
코로나 이후 33시간으로 줄이기도
전문가 "안전교육, 실습 중심으로"

등록 2022-11-01 오후 2:44:5(
수정 2022-11-01 오후 9:38:4(

스마트폰이 대중화된 사회에서 사고 부위를 알게 된다면 이동하면서 처치 방법을 확인 가능





학교 안전 중앙 공재회에서 배포해준 학교 안전사고 데이터를 사용

```
_2019 = pd.read_excel('asd/★2019~2023 학교안전사고 데이터_수정.xlsx',sheet_name='2019')
_2020 = pd.read_excel('asd/★2019~2023 학교안전사고 데이터_수정.xlsx',sheet_name='2020')
_2021 =pd.read_excel('asd/★2019~2023 학교안전사고 데이터_수정.xlsx',sheet_name='2021')
_2022 =pd.read_excel('asd/★2019~2023 학교안전사고 데이터_수정.xlsx',sheet_name='2022')
_2023 =pd.read_excel('asd/★2019~2023 학교안전사고 데이터_수정.xlsx',sheet_name='2023')
```

```
_2019 = _2019.query('사고발생일 >="2018-12-01" & 사고발생일 <="2019-12-31"')
_2020 = _2020.query('사고발생일 >="2020-01-01" & 사고발생일 <="2020-12-31"')
_2021 = _2021.query('사고발생일 >="2021-01-01" & 사고발생일 <="2021-12-31"')
_2022 = _2022.query('사고발생일 >="2022-01-01" & 사고발생일 <="2022-12-31"')
_2023 = _2023.query('사고발생일 >="2023-01-01" & 사고발생일 <="2023-12-31"')
```

사고 발생시각을 시간과 분으로 구분하는 작업과 통합 및 결측치 확인

```
[ ]  place['사고발생시각'] = pd.to_datetime(place['사고발생시각'], format='%H:%M').dt.strftime('%H:%M')
```

```
[ ]  place['Hour'] = place['사고발생시각'].str[:2]
```

```
[ ]  place['minute'] = place['사고발생시각'].str[3:5]
```

```
]:  place = pd.concat([_2019,_2020,_2021,_2022,_2023])
```

```
place.isnull().sum()
```

```
구분              0
학교급             0
지역              0
교육청             0
설립유형            0
사고자구분           1
사고자성별           0
사고자학년        1500
사고발생일           0
사고발생요일          0
사고발생시각          0
사고시간            0
사고장소            0
사고부위            0
사고형태            0
사고당시활동          0
사고매개물      188284
YM              0
Hour            0
매개물        411535
dtype: int64
```

사고부위에서 전혀 예측하기 어려운 기타 제거 및 사고 발생일 분리

```
[ ]  filter_place = place[place['사고부위'] !='기타']
```

```
[ ]  filter_place['사고부위'].unique()
     place = filter_place
```

```
[ ]  place = place.reset_index().drop('index',axis=1)
```

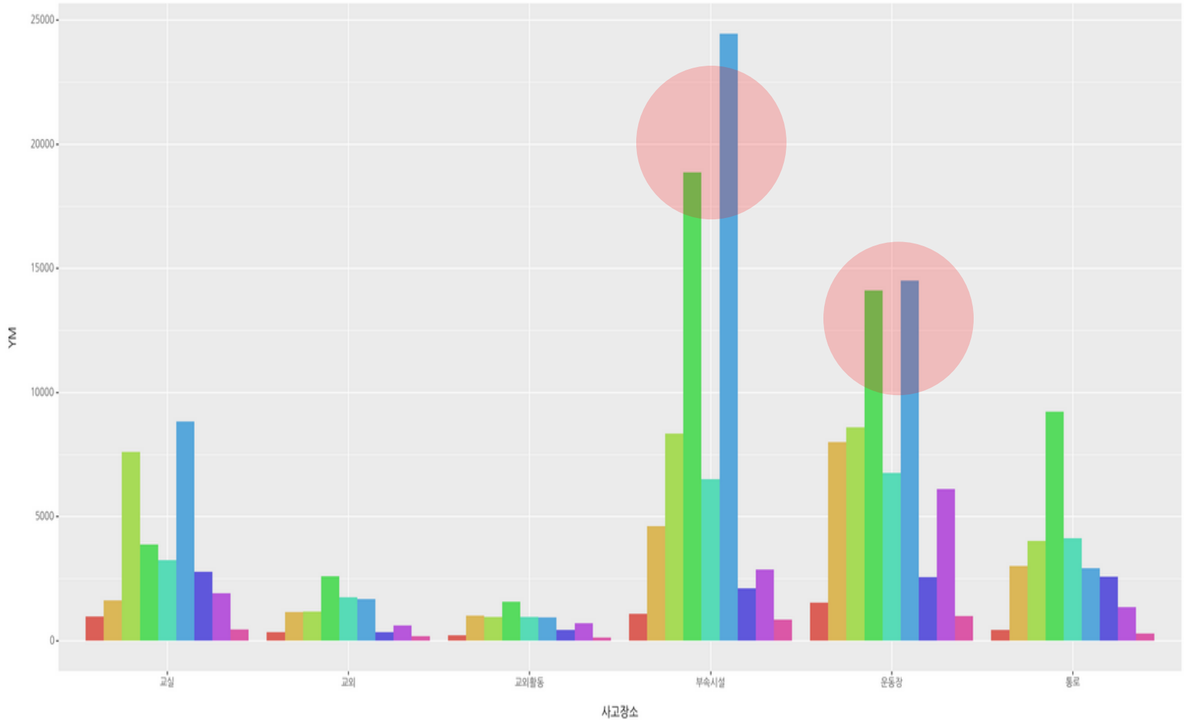
```
[ ]  place['Month'] = place['사고발생일'].str[5:7]
```

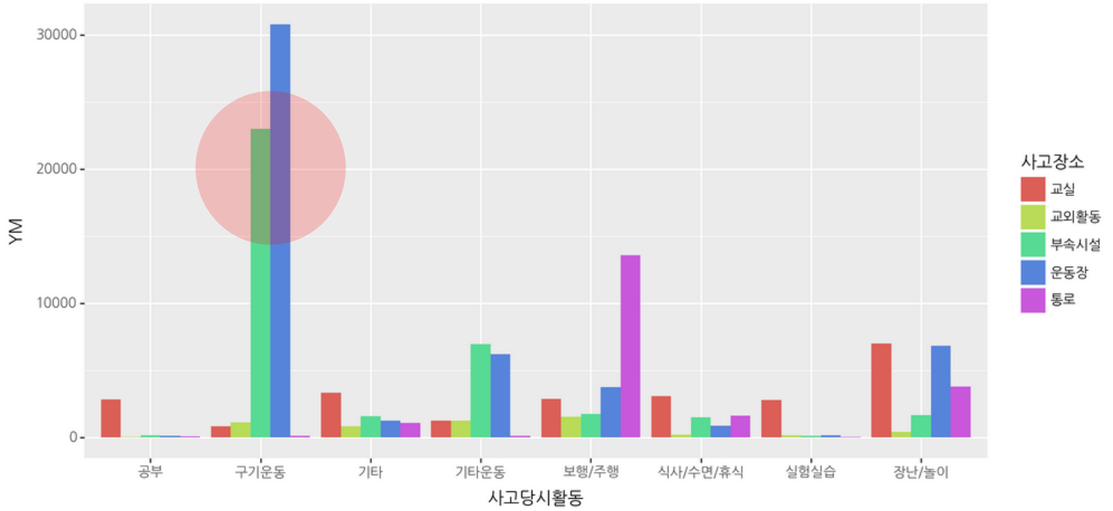
범주형 자료를 숫자형 범주형 변수로 변환

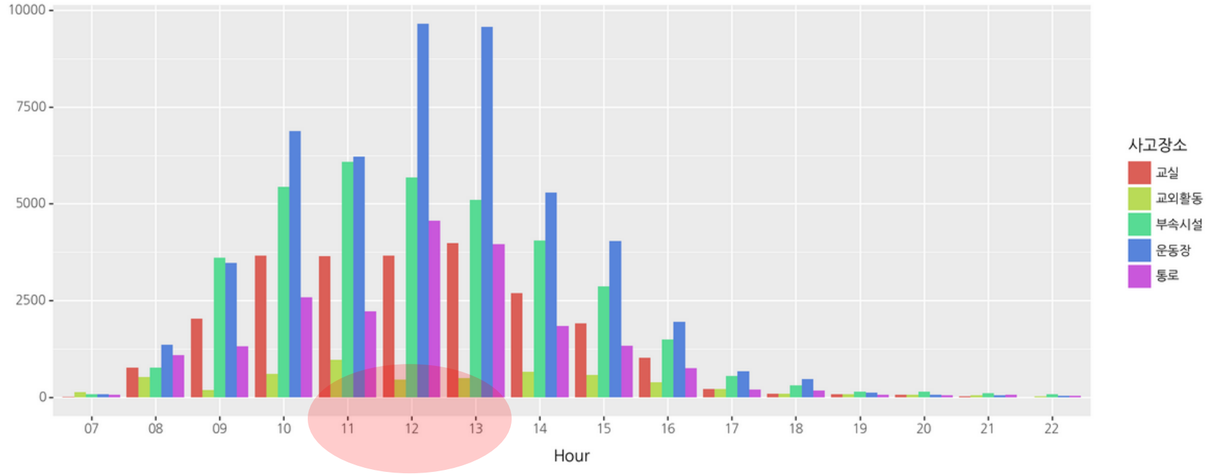
```
label_encoder = LabelEncoder()
place['사고장소'] = label_encoder.fit_transform(place['사고장소'])
place['사고부위'] = label_encoder.fit_transform(place['사고부위'])
place['사고당시활동'] = label_encoder.fit_transform(place['사고당시활동'])
place['사고자학년'] = label_encoder.fit_transform(place['사고자학년'])
place['사고시간'] = label_encoder.fit_transform(place['사고시간'])
place['설립유형'] = label_encoder.fit_transform(place['설립유형'])
place['사고자구분'] = label_encoder.fit_transform(place['사고자구분'])
place['사고자성별'] = label_encoder.fit_transform(place['사고자성별'])
place['사고형태'] = label_encoder.fit_transform(place['사고형태'])
#place['Month'] = label_encoder.fit_transform(place['Month'])
#place['minute'] = label_encoder.fit_transform(place['minute'])
```

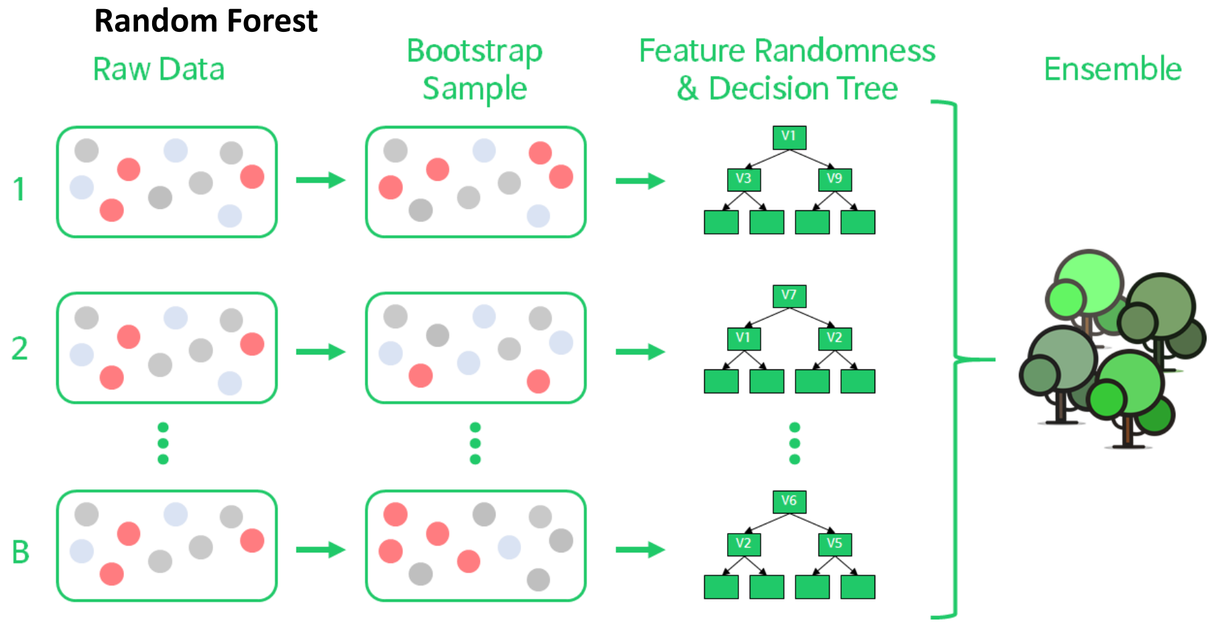
완성된 데이터 프레임

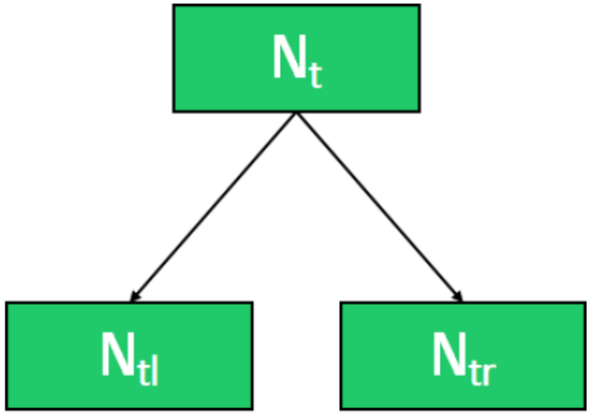
	구분	학교급	지역	교육청	설립유형	사고자구분	사고자성별	사고자학년	사고발생일	사고발생요일	...	사고부위	사고형태	사고당시활동	사고매개물	YM	매개물	Hour	minute	Y	Month
0	A0000001	기타학교	경기	경기도교육청	사립	일반학생	여	1	2018-12-28	금	...	4	물리적힘 노출	6	날카로운 물건(칼/가위/송곳 등)	2018-12	NaN	13.0	10.0	2018	12.0
1	A0000002	초등학교	경기	광주하남교육지원청	공립	일반학생	여	1	2018-12-27	목	...	0	낙상-미끄러짐	4	건물(문/창문/바닥/벽 등)	2018-12	NaN	11.0	35.0	2018	12.0
2	A0000003	초등학교	경기	용인교육지원청	공립	일반학생	남	5	2018-12-28	금	...	1	물리적힘 노출	4	건물(문/창문/바닥/벽 등)	2018-12	NaN	12.0	40.0	2018	12.0
3	A0000005	중학교	경기	광주하남교육지원청	공립	일반학생	여	2	2018-12-24	월	...	0	낙상-미끄러짐	4	건물(문/창문/바닥/벽 등)	2018-12	NaN	14.0	0.0	2018	12.0
4	A0000006	초등학교	경기	구리남양주교육지원청	공립	일반학생	남	2	2018-12-27	목	...	5	물리적힘 노출	4	건물(문/창문/바닥/벽 등)	2018-12	NaN	14.0	0.0	2018	12.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
586604	E0193173	중학교	제주	서귀포시교육지원청	공립	일반학생	여	1	2023-12-11	월	...	6	물리적힘 노출	7	NaN	2023-12	건물(문/창문/바닥/벽 등)	14.0	10.0	2023	12.0
586605	E0193174	중학교	제주	제주시교육지원청	공립	일반학생	남	0	2023-12-08	금	...	0	사람과의 충돌	5	NaN	2023-12	자연(사람/동물/식물 등)	13.0	20.0	2023	12.0
586606	E0193175	중학교	제주	제주시교육지원청	공립	일반학생	여	2	2023-12-04	월	...	4	물리적힘 노출	1	NaN	2023-12	운동(놀이)용 장비/기구(공/운동기구/운동장 기구 등)	9.0	20.0	2023	12.0
586607	E0193176	중학교	제주	제주시교육지원청	공립	일반학생	남	0	2023-12-13	수	...	2	낙상	7	NaN	2023-12	자연(사람/동물/식물 등)	12.0	25.0	2023	12.0
586608	E0193177	중학교	제주	제주시교육지원청	공립	일반학생	남	0	2023-11-30	목	...	3	물리적힘 노출	1	NaN	2023-11	운동(놀이)용 장비/기구(공/운동기구/운동장 기구 등)	15.0	50.0	2023	11.0


25000
20000
15000
10000
5000
0
YM
교실
교외
교외활동
부속시설
운동장
통로
사고장소
사고부위
기타
다리
머리(두부)
발
복합부위
손
치아(구강)
팔
흉복부


30000
20000
10000
0
YM
공부
구기운동
기타
기타운동
보행/주행
식사/수면/휴식
실험실습
장난/놀이
사고당시활동
사고장소
교실
교외활동
부속시설
운동장
통로


10000
7500
5000
2500
0
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
Hour
사고장소
교실
교외활동
부속시설
운동장
통로





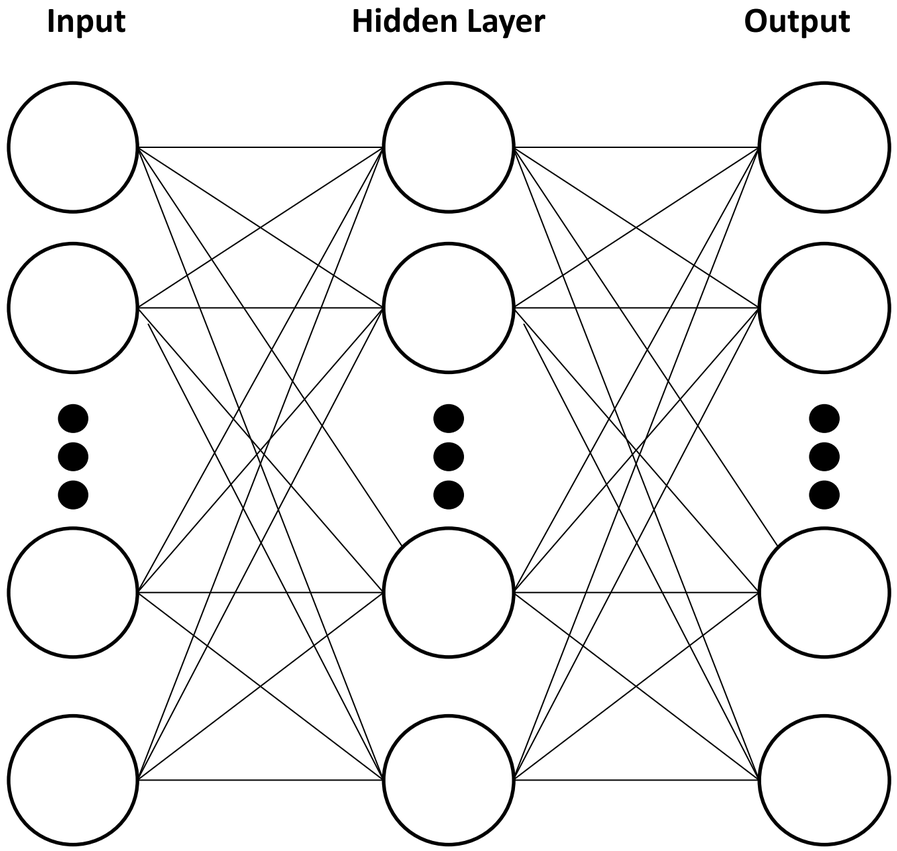
$$\Delta i(t) = i(t) - \frac{N_{tl}}{N_t} i(t_l) - \frac{N_{tr}}{N_t} i(t_r)$$

아래와 같은 칼럼을 후보군으로 넣어 각 변수의 중요도를 파악

```
'사고장소','사고당시활동','사고자학년','사고시간','설립유형','사고자구분''
사고자성별','Month','minute','사고형태','Hour'
```

Random Forest Result:

	Feature	Importance
9	사고형태	0.930070
1	사고당시활동	0.017263
2	사고자학년	0.011351
0	사고장소	0.010957
3	사고시간	0.006917
8	minute	0.006303
7	Month	0.006186
10	Hour	0.005498
6	사고자성별	0.002926
4	설립유형	0.001479
5	사고자구분	0.001050



딥러닝 모델의 예시로 깊은 신경망을 사용.
장소,당시활동,사고자학년,Month,minute,사고시간,Hour,사고형태 7개의 변수를 사용하여 사고 부위 8개를 분류하는 예측 모형을 사용.

Optimizr : Adam

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$\theta = \theta - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}}\hat{m}_t$$

$$\omega_{t+1} = \omega_t - m_t \frac{\eta}{\sqrt{v_t + \epsilon}}$$

Cross Entrophy Loss()

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -w_{y_n} \log \frac{\exp(x_{n,y_n})}{\sum_{c=1}^{C} \exp(x_{n,c})} \cdot 1\{y_n \neq \text{ignore\_index}\}$$

$$\ell(x, y) = \begin{cases} \sum_{n=1}^{N} \frac{1}{\sum_{n=1}^{N} w_{y_n} \cdot 1\{y_n \neq \text{ignore\_index}\}} l_n, & \text{if reduction} = \text{‘mean’;} \\ \sum_{n=1}^{N} l_n, & \text{if reduction} = \text{‘sum’.} \end{cases}$$

사고 장소, 사고당시 활동, 사고 시간 … 등
8개의 범주형 및 숫자형 변수

```
# 모델 정의
net = nn.Sequential(
    torch.nn.Linear(in_features=9, out_features=512),
    torch.nn.ReLU(),
    torch.nn.Linear(512, 256),
    torch.nn.ReLU(),
    torch.nn.Linear(256, 128),
    torch.nn.ReLU(),
    torch.nn.Linear(128,64),
    torch.nn.ReLU(),
    torch.nn.Dropout(0.5),
    torch.nn.Linear(64, 9)
)

# GPU 설정 확인
device = torch.device("cuda" if torch.cuda.is_available() else 'cpu')
print(f'Using device: {device}')

# 모델을 GPU로 이동
net.to(device)

# 손실 함수와 옵티마이저 정의
criterion = torch.nn.CrossEntropyLoss()
optimizer = optim.Adam(net.parameters(), lr=0.001)

# 데이터를 GPU로 이동 (예시로 X_train과 y_train, X_test와 y_test가 텐서라고 가정)
X_train = X_train.to(device)
y_train = y_train.to(device)
X_test = X_test.to(device)
y_test = y_test.to(device)

# 학습
num_epochs = 100
for epoch in range(num_epochs):
    net.train()
    optimizer.zero_grad()
    outputs = net(X_train)
    loss = criterion(outputs, y_train)
    loss.backward()
    optimizer.step()
```

Deep Learning Model

사고 부위 8개를 숫자형으로 범주형화 시킨 것으로 표현 0~7 총 8 부위

```
Using device: cuda
Epoch [10/100], Loss: 1.8323
Epoch [20/100], Loss: 1.4896
Epoch [30/100], Loss: 1.2093
Epoch [40/100], Loss: 0.9925
Epoch [50/100], Loss: 0.8218
Epoch [60/100], Loss: 0.6789
Epoch [70/100], Loss: 0.5611
Epoch [80/100], Loss: 0.4753
Epoch [90/100], Loss: 0.4089
Epoch [100/100], Loss: 0.3570
Test Accuracy: 0.91
Probabilities for the first 5 test samples:
tensor([[1.7243e-02, 9.7886e-01, 3.8919e-03, 5.0882e-07, 2.1544e-10, 2.3938e-13,
         4.8258e-11, 1.3635e-12, 4.5015e-13],
        [3.0174e-06, 2.9247e-03, 9.8503e-01, 1.2028e-02, 1.6621e-05, 8.8106e-09,
         3.1306e-08, 1.5508e-08, 5.4419e-10],
        [1.2161e-08, 5.6054e-05, 6.7966e-02, 8.8043e-01, 5.1439e-02, 9.0191e-05,
         1.3914e-05, 4.9450e-06, 3.4345e-09],
        [1.7019e-06, 4.0511e-03, 9.9419e-01, 1.7518e-03, 5.3881e-07, 8.2842e-11,
         8.3538e-10, 2.7676e-10, 1.0211e-11],
        [1.0843e-05, 6.3703e-03, 9.7667e-01, 1.6903e-02, 4.0929e-05, 5.4925e-08,
         1.7289e-07, 7.0254e-08, 3.1090e-09]], device='cuda:0')
```

이는 전에 말한 7개의 변수를 통하여 사고부위를 예측하는 모델로 테스트 데이터에 직접 실험해보니 정확도 0.91로 매우 높은 결과를 나타냅니다.

실제로 테스트 데이터가 아닌 임의로 7개의 변수를 넣는다면 예측해줍니다.
또한 여기서 임의의 변수만 바꿔준다면 사고 부위를 통해서 사고 형태를 예측이 가능하며 정확도도 높았습니다.

	precision	recall	f1-score	support
0	1.00	1.00	1.00	10968
1	1.00	0.99	1.00	19127
2	0.99	0.98	0.98	30640
3	0.91	0.78	0.84	10438
4	0.88	0.96	0.92	31653
5	0.64	0.26	0.36	5190
6	0.64	0.95	0.76	7521
7	0.00	0.00	0.00	1785
accuracy			0.91	117322
macro avg	0.76	0.74	0.73	117322
weighted avg	0.90	0.91	0.90	117322

각 사고부위에 대한 정확도로 꽤나 높은 것을 파악할 수 있습니다.

데이터의 한계점으로는 안전사고형태는 매우 다양하다는 것입니다. 제공된 데이터와 같이 매우 관리가 잘된 데이터에 의거하여 안전사고가 발생한다면 사고부위나 사고형태의 예측을 정확하게 할 수 있으나 정말 알 수 없는 상황이나 새로운 사례가 발생한다면 예측이 어려울 것입니다. 이는 위의 모델이 특정 변수에 매우 취약하다는 것을 알 수 있으며 따라서 이걸 해결하기 위해서는 계속 데이터를 갱신 시키며 변수를 추가시켜서 모델을 업데이트 해줘야 할 것입니다.

안전사고가 발생한다면 최초의 제보자가 사고자의 정보를 정확하게 가져와야 한다는 점입니다. 위의 모델은 정확한 정보를 통해서 구현되었기 때문에 만약 헷갈리거나 잘못된 정보를 제공할 경우 처치하는 사람이 당혹감을 느껴 처치에 어려움을 느낄 수 있을 것입니다.

사고 형태와 사고 부위는 매우 영향이 큰 데이터라는 점입니다. 실제로 사고부위를 예측할 때 사고 형태 변수를 넣지 않게 되면 예측력은 매우 떨어져 거의 40퍼 대로 내려가게 됩니다. 이는 사고 형태를 통한 사고 부위의 예측 사고 부위를 통한 사고 형태를 예측하는 것에는 유용하나 두개의 데이터가 부재라면 예측이 매우 어렵게 될 것입니다.

또한 사고 발생은 포아송분포처럼 주기마다 발생하는 것이 아닌 갑작스럽게 발생하는 예측이 어려운 자료라고 생각하기에 이런 모델에 의존이 아닌 모델은 참고하되 담당자의 신속한 처치가 필요할 것입니다.