분류(classification) 문제 실습 - 항구의 기뢰 찾기

학습 목표

- 데이터을 전처리 후, 이에 대한 머신러닝 모델을 만들어봅니다.
- 모델 비교에 대해 이해해 봅니다.
- 모델 평가에 대해 기본 이해해 봅니다.

학습 내용

- 주어진 데이터를 적절한 비율로 나누어보기
- knn 모델 만들어보기
- 의사결정 트리 모델을 만들어보기
- 모델 평가해 보기

- 데이터 셋 : UC Irvine Data Repository
  - https://archive.ics.uci.edu/ml/datasets.php
  - Connectionist Bench (Sonar, Mines vs. Rocks) Data Set
    - https://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+%28Sonar%2C+Mine
    - sonar.all-data

데이터 설명

- 데이터 셋 특성 : 다변수
- 행의 수 : 208개
- 열의 수 : 60개
- 데이터 설명
  - 다양한 각도와 다양한 조건에서 금속 실린더에서 수중 음파 탐지기 신호를 통해 얻은 다양한 패턴이 포함되어 있음.
  - 군사작전의 결과로 항구에 남아 있는 폭파되지 않은 기뢰를 찾기 위해 소나(Sonar, 수중 음파탐지기)를 이용할 수 있는지 확인하기 위한 어떤 실험으로 만들어짐.
  - 반 정도의 표본은 바위, 나마저 반은 기뢰 모양의 금속 원통을 나타냄.

01. 데이터 준비

In [1]:
```
from urllib.request import urlopen
import sys
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [8]:
```
target_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/undoc
dat = pd.read_csv(target_url, header=None, prefix='V')
dat.head()
```

Out[8]:

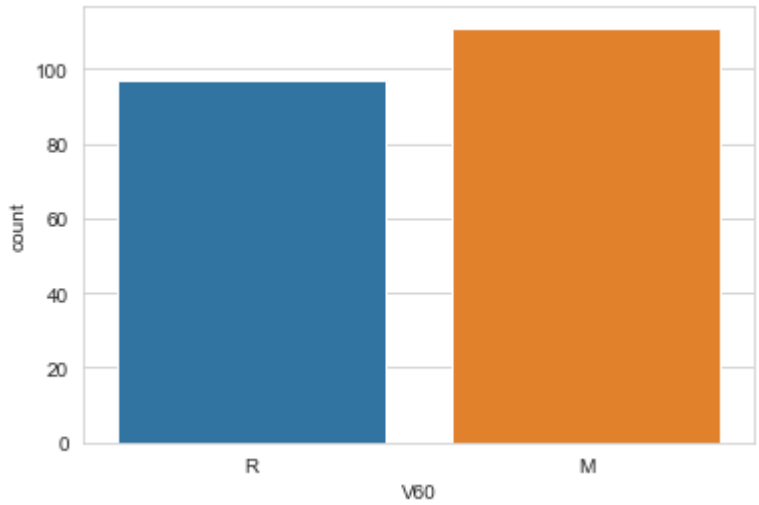
	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V5
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.002
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.008

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V5
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.023
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.012
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.003

5 rows × 61 columns

In [9]:
```
sns.set_style('whitegrid')
sns.countplot(x='V60', data=dat)
```

Out[9]:
```
<AxesSubplot:xlabel='V60', ylabel='count'>
```



In [11]:
```
dat['target']=dat['V60']
dat.columns
```

Out[11]:
```
Index(['V0', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'V29', 'V30',
       'V31', 'V32', 'V33', 'V34', 'V35', 'V36', 'V37', 'V38', 'V39', 'V40',
       'V41', 'V42', 'V43', 'V44', 'V45', 'V46', 'V47', 'V48', 'V49', 'V50',
       'V51', 'V52', 'V53', 'V54', 'V55', 'V56', 'V57', 'V58', 'V59', 'V60',
       'label', 'target'],
      dtype='object')
```

In [12]:
```
### 인코딩 (R:0, M:1)
dat['target'] = dat['V60'].map( {'R':0, 'M':1 }  ).astype(int)
dat['target'].value_counts()
```

Out[12]:
```
1    111
0     97
Name: target, dtype: int64
```

In [13]:
```
dat.head()
```

Out[13]:

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V5
0	0.0200	0.0371	0.0428	0.0207	0.0954	0.0986	0.1539	0.1601	0.3109	0.2111	...	0.015
1	0.0453	0.0523	0.0843	0.0689	0.1183	0.2583	0.2156	0.3481	0.3337	0.2872	...	0.004
2	0.0262	0.0582	0.1099	0.1083	0.0974	0.2280	0.2431	0.3771	0.5598	0.6194	...	0.009
3	0.0100	0.0171	0.0623	0.0205	0.0205	0.0368	0.1098	0.1276	0.0598	0.1264	...	0.015

	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V5
4	0.0762	0.0666	0.0481	0.0394	0.0590	0.0649	0.1209	0.2467	0.3564	0.4459	...	0.010

5 rows × 63 columns

데이터 최종 전처리

- V60 열 삭제
- 입력 열 지정
- 출력(target) 열 지정
- 데이터 나누기(train_test_split)

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [19]:
```
new_df = dat.drop("V60", axis='columns')
X_all = new_df.loc[:,'V0':'V59']  # V0~V59 열 선택
y = new_df['target']              # 'target' 열 선택

# test:10%, train:90%로 데이터 분리
X_train, X_test, y_train, y_test = train_test_split(X_all, y,
                                                    test_size=0.1,
                                                    random_state=0)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[19]:
```
((187, 60), (21, 60), (187,), (21,))
```

In [20]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [21]:
```
### 모델 선택 및 학습
model1 = DecisionTreeClassifier()
model1.fit(X_train, y_train)
model1.score(X_test, y_test)
```

Out[21]:
```
0.7142857142857143
```

In [22]:
```
### 모델 선택 및 학습
model2 = KNeighborsClassifier()
model2.fit(X_train, y_train)
model2.score(X_test, y_test)
```

Out[22]:
```
0.8095238095238095
```

In [23]:
```
### 최종 모델 선택 및 예측
last_model = KNeighborsClassifier()
last_model.fit(X_train, y_train)
pred = last_model.predict(X_test)
pred
```

Out[23]:
```
array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1])
```

In [25]:
```
### 최종 모델 평가
( sum(pred==y_test) / len(pred) ) * 100
```

Out[25]: 80.95238095238095

- 최종 모델은 knn을 선택했고,
- 최종 모델 예측하여 평가 결과 : 81% 의 정확도를 갖는 모델을 만들었다.

실습 과제

- knn 모델의 k의 개수를 변경해 보며, 더 최적의 모델을 선택해 보자.
- V0~V59의 피처를 선택했는데, 이에 대한 일부만을 선택해 보며, 모델의 성능을 측정해 보자.