제2유형_연습하기_와인종류분류

✅ 데이터 분석 순서

1. 라이브러리 및 데이터 확인

2. 데이터 탐색(EDA)

3. 데이터 전처리 및 분리

4. 모델링 및 성능평가

5. 예측값 제출

✅ 1. 라이브러리 및 데이터 확인

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
###############  실기환경 복사 영역  ###############
import pandas as pd
import numpy as np
# 실기 시험 데이터셋으로 셋팅하기 (수정금지)

from sklearn.datasets import load_wine
# 와인 데이터셋을 로드합니다.
wine = load_wine()
x = pd.DataFrame(wine.data, columns=wine.feature_names)
y = pd.DataFrame(wine.target)

# 실기 시험 데이터셋으로 셋팅하기 (수정금지)
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
                                                    stratify=y,
                                                    random_state=2023)

x_test = pd.DataFrame(x_test)
x_train = pd.DataFrame(x_train)
y_train = pd.DataFrame(y_train)

x_test.reset_index()
y_train.columns = ['target']
###############  실기환경 복사 영역  ###############


### 참고사항 ###
# y_test 는 실기 문제상에 주어지지 않음

# ★Tip : X를 대문자로 쓰지말고 소문자 x로 쓰세요. 시험에서 실수하기 쉽습니다.(문제풀기 전에 소문자로 변경!)
# (참고 : 보통 X는 2차원 배열(행렬)이기 때문에 대문자로 쓰고, y는 1차원 배열(벡터)이기 때문에 소문자로 씀)

# (~23년 10월말) 실기시험 데이터 형식 (실제 시험장에서는 다를 수 있으니 반드시 체크)
# X_test = pd.read_csv("data/X_test.csv")
# X_train = pd.read_csv("data/X_train.csv")
# y_train = pd.read_csv("data/y_train.csv")

# ★(23년 10월말~) 기준으로 체험환경에서 제공되는 데이터셋이 조금 변경되었습니다.
# train = pd.read_csv("data/customer_train.csv")
# test = pd.read_csv("data/customer_test.csv")
# x_train과 y_train, x_test를 별도로 할당해주셔야 합니다.
```

빅데이터 자격증 / AI 강의는 메타코드M

와인의 종류를 분류해보자

- 데이터의 결측치, 이상치에 대해 처리하고

- 분류모델을 사용하여 정확도, F1 score, AUC 값을 산출하시오.

- 제출은 result 변수에 담아 양식에 맞게 제출하시오

빅데이터 자격증 / AI 강의는 메타코드M

In [3]:
```
# 데이터 설명
print(wine.DESCR)
```

```
.. _wine_dataset:
```

```
Wine recognition dataset
------------------------

**Data Set Characteristics:**

    :Number of Instances: 178
    :Number of Attributes: 13 numeric, predictive attributes and the class
    :Attribute Information:
            - Alcohol
            - Malic acid
            - Ash
            - Alcalinity of ash
            - Magnesium
            - Total phenols
            - Flavanoids
            - Nonflavanoid phenols
            - Proanthocyanins
            - Color intensity
            - Hue
            - OD280/OD315 of diluted wines
            - Proline

    - class:
        - class_0
        - class_1
        - class_2

    :Summary Statistics:

    ============================= ==== ===== ======= =====
                                   Min   Max   Mean     SD
    ============================= ==== ===== ======= =====
    Alcohol:                      11.0  14.8    13.0   0.8
    Malic Acid:                   0.74  5.80    2.34  1.12
    Ash:                          1.36  3.23    2.36  0.27
    Alcalinity of Ash:            10.6  30.0    19.5   3.3
    Magnesium:                    70.0 162.0    99.7  14.3
    Total Phenols:                0.98  3.88    2.29  0.63
    Flavanoids:                   0.34  5.08    2.03  1.00
    Nonflavanoid Phenols:         0.13  0.66    0.36  0.12
    Proanthocyanins:              0.41  3.58    1.59  0.57
    Colour Intensity:              1.3  13.0     5.1   2.3
    Hue:                          0.48  1.71    0.96  0.23
    OD280/OD315 of diluted wines: 1.27  4.00    2.61  0.71
    Proline:                       278  1680     746   315
    ============================= ==== ===== ======= =====

    :Missing Attribute Values: None
    :Class Distribution: class_0 (59), class_1 (71), class_2 (48)
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

This is a copy of UCI ML Wine recognition datasets.
https://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data

The data is the results of a chemical analysis of wines grown in the same
region in Italy by three different cultivators. There are thirteen different
measurements taken for different constituents found in the three types of
wine.

Original Owners:

Forina, M. et al, PARVUS -
An Extendible Package for Data Exploration, Classification and Correlation.
Institute of Pharmaceutical and Food Analysis and Technologies,
Via Brigata Salerno, 16147 Genoa, Italy.

Citation:

Lichman, M. (2013). UCI Machine Learning Repository
[https://archive.ics.uci.edu/ml]. Irvine, CA: University of California,
School of Information and Computer Science.

.. topic:: References

  (1) S. Aeberhard, D. Coomans and O. de Vel,
  Comparison of Classifiers in High Dimensional Settings,
  Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of
  Mathematics and Statistics, James Cook University of North Queensland.
  (Also submitted to Technometrics).

  The data was used with many others for comparing various
  classifiers. The classes are separable, though only RDA
  has achieved 100% correct classification.
  (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-transformed data))
  (All results using the leave-one-out technique)

  (2) S. Aeberhard, D. Coomans and O. de Vel,
```

```
"THE CLASSIFICATION PERFORMANCE OF RDA"
Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of
Mathematics and Statistics, James Cook University of North Queensland.
(Also submitted to Journal of Chemometrics).
```

✅ 2. 데이터 탐색(EDA)

In [4]:
```python
# 데이터의 행/열 확인

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
```

```
(142, 13)
(36, 13)
(142, 1)
```

In [5]:
```python
# 초기 데이터 확인

print(x_train.head(3))
print(x_test.head(3))
print(y_train.head(3))
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
52     13.82        1.75  2.42               14.0      111.0           3.88
146    13.88        5.04  2.23               20.0       80.0           0.98
44     13.05        1.77  2.10               17.0      107.0           3.00

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
52         3.74                  0.32             1.87             7.05  1.01
146        0.34                  0.40             0.68             4.90  0.58
44         3.00                  0.28             2.03             5.04  0.88

     od280/od315_of_diluted_wines  proline
52                           3.26   1190.0
146                          1.33    415.0
44                           3.35    885.0
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
168    13.58        2.58  2.69               24.5      105.0           1.55
144    12.25        3.88  2.20               18.5      112.0           1.38
151    12.79        2.67  2.48               22.0      112.0           1.48

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
168        0.84                  0.39             1.54             8.66  0.74
144        0.78                  0.29             1.14             8.21  0.65
151        1.36                  0.24             1.26            10.80  0.48

     od280/od315_of_diluted_wines  proline
168                          1.80    750.0
144                          2.00    855.0
151                          1.47    480.0
     target
52        0
146       2
44        0
```


In [6]:
```python
# 변수명과 데이터 타입이 매칭이 되는지, 결측치가 있는지 확인해보세요

print(x_train.info())
print(x_test.info())
print(y_train.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 142 entries, 52 to 115
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       142 non-null    float64
 1   malic_acid                    142 non-null    float64
 2   ash                           142 non-null    float64
 3   alcalinity_of_ash             142 non-null    float64
 4   magnesium                     142 non-null    float64
 5   total_phenols                 142 non-null    float64
 6   flavanoids                    142 non-null    float64
 7   nonflavanoid_phenols          142 non-null    float64
 8   proanthocyanins               142 non-null    float64
 9   color_intensity               142 non-null    float64
 10  hue                           142 non-null    float64
 11  od280/od315_of_diluted_wines  142 non-null    float64
 12  proline                       142 non-null    float64
dtypes: float64(13)
memory usage: 15.5 KB
None
<class 'pandas.core.frame.DataFrame'>
Int64Index: 36 entries, 168 to 55
Data columns (total 13 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       36 non-null     float64
 1   malic_acid                    36 non-null     float64
 2   ash                           36 non-null     float64
 3   alcalinity_of_ash             36 non-null     float64
 4   magnesium                     36 non-null     float64
 5   total_phenols                 36 non-null     float64
 6   flavanoids                    36 non-null     float64
 7   nonflavanoid_phenols          36 non-null     float64
 8   proanthocyanins               36 non-null     float64
 9   color_intensity               36 non-null     float64
 10  hue                           36 non-null     float64
 11  od280/od315_of_diluted_wines  36 non-null     float64
 12  proline                       36 non-null     float64
dtypes: float64(13)
memory usage: 3.9 KB
None
<class 'pandas.core.frame.DataFrame'>
Int64Index: 142 entries, 52 to 115
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   target  142 non-null    int32
dtypes: int32(1)
memory usage: 1.7 KB
None
```

In [7]:

```
# x_train 과 x_test 데이터의 기초통계량을 잘 비교해보세요.

print(x_train.describe())  # x_train.describe().T 둘중에 편한거 사용하세요
print(x_test.describe())
print(y_train.describe())
```

```
          alcohol  malic_acid         ash  alcalinity_of_ash   magnesium  \
count  142.000000  142.000000  142.000000         142.000000  142.000000   
mean    13.025915    2.354296    2.340211          19.354225   98.732394   
std      0.812423    1.142722    0.279910           3.476825   13.581859   
min     11.030000    0.740000    1.360000          10.600000   70.000000   
25%     12.370000    1.610000    2.190000          16.800000   88.000000   
50%     13.050000    1.820000    2.320000          19.300000   97.000000   
75%     13.685000    3.115000    2.510000          21.500000  106.750000   
max     14.830000    5.800000    3.230000          30.000000  151.000000   

       total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
count     142.000000  142.000000            142.000000       142.000000   
mean        2.303592    2.043592              0.361479         1.575070   
std         0.633955    1.033597              0.124627         0.576798   
min         0.980000    0.340000              0.140000         0.410000   
25%         1.757500    1.227500              0.270000         1.242500   
50%         2.335000    2.100000              0.325000         1.555000   
75%         2.800000    2.917500              0.437500         1.950000   
max         3.880000    5.080000              0.630000         3.580000   

       color_intensity         hue  od280/od315_of_diluted_wines      proline  
count       142.000000  142.000000                    142.000000   142.000000  
mean          5.005070    0.950394                      2.603592   742.112676  
std           2.150186    0.220736                      0.709751   317.057395  
min           1.280000    0.540000                      1.270000   290.000000  
25%           3.300000    0.782500                      1.922500   496.250000  
50%           4.850000    0.960000                      2.780000   660.000000  
75%           6.122500    1.097500                      3.170000   981.250000  
max          13.000000    1.710000                      3.920000  1680.000000  
         alcohol  malic_acid        ash  alcalinity_of_ash   magnesium  \
count  36.000000   36.000000  36.000000           36.00000   36.000000   
mean   12.900833    2.265556   2.470278           20.05000  103.722222   
std     0.813112    1.021943   0.226066            2.70275   16.371772   
min    11.640000    0.890000   2.000000           14.60000   84.000000   
25%    12.230000    1.592500   2.300000           18.00000   91.500000   
50%    12.835000    1.885000   2.470000           19.50000  101.000000   
75%    13.635000    2.792500   2.605000           21.70000  112.000000   
max    14.390000    4.950000   3.220000           26.50000  162.000000   

       total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
count      36.000000   36.000000             36.000000        36.000000   
mean        2.261667    1.972778              0.363333         1.653333   
std         0.600259    0.858882              0.125516         0.558012   
min         1.350000    0.660000              0.130000         0.840000   
25%         1.715000    1.175000              0.267500         1.320000   
50%         2.420000    2.175000              0.395000         1.550000   
75%         2.602500    2.682500              0.435000         1.972500   
max         3.850000    3.490000              0.660000         3.280000   

       color_intensity        hue  od280/od315_of_diluted_wines     proline  
count        36.000000  36.000000                     36.000000    36.00000  
mean          5.267222   0.985278                      2.643611   765.75000  
std           2.915076   0.258694                      0.720100   309.94851  
min           2.080000   0.480000                      1.290000   278.00000  
25%           2.875000   0.787500                      2.037500   542.50000  
50%           4.325000   0.985000                      2.790000   682.50000  
75%           6.900000   1.167500                      3.192500   996.25000  
max          11.750000   1.450000                      4.000000  1480.00000  
           target
count  142.000000
mean     0.936620
std      0.773816
min      0.000000
25%      0.000000
50%      1.000000
75%      2.000000
max      2.000000
```

In [8]:
```python
# y데이터도 구체적으로 살펴보세요.
print(y_train.head())
```

```
     target
52        0
146       2
44        0
67        1
43        0
```

In [9]:
```python
# y데이터도 구체적으로 살펴보세요.
print(y_train.value_counts())
```

```
target
1         57
0         47
2         38
dtype: int64
```

✅ 3. 데이터 전처리 및 분리

1) 결측치, 2) 이상치, 3) 변수 처리하기

In [10]:
```
# 결측치 확인
print(x_train.isnull().sum())
print(x_test.isnull().sum())
print(y_train.isnull().sum())
```



```
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
dtype: int64
alcohol                         0
malic_acid                      0
ash                             0
alcalinity_of_ash               0
magnesium                       0
total_phenols                   0
flavanoids                      0
nonflavanoid_phenols            0
proanthocyanins                 0
color_intensity                 0
hue                             0
od280/od315_of_diluted_wines    0
proline                         0
dtype: int64
target    0
dtype: int64
```

In [11]:
```
# 결측치 제거
# df = df.dropna()
# print(df)

# 참고사항
# print(df.dropna().shape)  # 행 기준으로 삭제

# ★주의사항
# x_train의 행을 제거해야 하는 경우, 그에 해당하는 y_train 행도 제거해야 합니다.
# 해결방법 : train = pd.concat([x_train, y_train], axis=1)
# 위와 같이 데이터를 결합한 후에 행을 제거하고 다시 데이터 분리를 수행하면 됩니다.
# (만약 원데이터가 x_train/y_train이 결합된 형태로 주어진다면 전처리를 모두 수행한 후에 분리하셔도 됩니다)
```


In [12]:
```
# 결측치 대체(평균값, 중앙값, 최빈값)
# ** 주의사항 : train 데이터의 중앙값/평균값/최빈값 등으로 test 데이터의 결측치도 변경해줘야 함 **

# 연속형 변수 : 중앙값, 평균값
#  - df['변수명'].median()
#  - df['변수명'].mean()
# 범주형 변수 : 최빈값

# df['변수명'] = df['변수명'].fillna(대체할 값)
```

In [13]:
```
# 이상치 대체(예시)
# df['변수명'] = np.where( df['변수명'] >= 5, 대체할 값, df['변수명'] )
```

In [14]:
```
# 변수처리

# 불필요한 변수 제거
# df = df.drop(columns = ['변수1','변수2'])
# df = df.drop(['변수1','변수2'], axis=1)

# 필요시 변수 추가(파생변수 생성)
# df['파생변수명'] = df['A'] * df['B'] (파생변수 생성 수식)

# 원핫인코딩(가변수 처리)
# x_train = pd.get_dummies(x_train)
# x_test = pd.get_dummies(x_test)
# print(x_train.info())
# print(x_test.info())
```


데이터 분리

In [15]:
```
# 데이터를 훈련 세트와 검증용 세트로 분할 (80% 훈련, 20% 검증용)
from sklearn.model_selection import train_test_split
x_train, x_val, y_train, y_val = train_test_split(x_train,
```

```
                                                    y_train['target'],
                                                    test_size=0.2,
                                                    stratify = y_train['target'],
                                                    random_state=2023)

print(x_train.shape)
print(x_val.shape)
print(y_train.shape)
print(y_val.shape)
```

```
(113, 13)
(29, 13)
(113,)
(29,)
```


✅ 4. 모델링 및 성능평가

In [16]:
```
# 랜덤포레스트 모델 사용  (참고 : 회귀모델은 RandomForestRegressor)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier()
model.fit(x_train, y_train)
```

Out[16]:
```
▾ RandomForestClassifier
RandomForestClassifier()
```

In [17]:
```
# 모델을 사용하여 테스트 데이터 예측
y_pred = model.predict(x_val)
```

In [18]:
```
# 모델 성능 평가 (정확도, F1 score, 민감도, 특이도 등)
from sklearn.metrics import accuracy_score, f1_score, recall_score, precision_score
acc = accuracy_score(y_val, y_pred)              # (실제값, 예측값)
f1 = f1_score(y_val, y_pred, average = 'macro') # (실제값, 예측값)
# auc = roc_auc_score(y_val, y_pred)            # (실제값, 예측값) * AUC는 이진분류
```

In [19]:
```
# 정확도(Accuracy)
print(acc)
```

```
0.9655172413793104
```

In [20]:
```
# macro f1 score
print(f1)
```



```
0.9670588235294119
```

✅ 5. 예측값 제출

(주의) test 셋을 모델에 넣어 나온 예측값을 제출해야함

In [21]:
```
# (실기시험 안내사항)
# 아래 코드 예측변수와 수험번호를 개인별로 변경하여 활용
# pd.DataFrame({ 'result': y_result }).to_csv('수험번호.csv', index=False)

# 모델을 사용하여 테스트 데이터 예측

# 1. 특정 클래스로 분류할 경우 (predict)
y_result = model.predict(x_test)
print(y_result[:5])

# 2. 특정 클래스로 분류될 확률을 구할 경우 (predict_proba)
y_result_prob = model.predict_proba(x_test)
print(y_result_prob[:5])

# 이해해보기
result_prob = pd.DataFrame({
    'result': y_result,
    'prob_0': y_result_prob[:,0],
    'prob_1': y_result_prob[:,1],
    'prob_2': y_result_prob[:,2]
})
# Class 0일 확률 : y_result_prob[:,0]
# Class 1일 확률 : y_result_prob[:,1]
# Class 2일 확률 : y_result_prob[:,2]

print(result_prob[:5])
```

```
[2 2 2 0 1]
[[0.04 0.01 0.95]
 [0.11 0.11 0.78]
 [0.01 0.1  0.89]
 [0.99 0.01 0.  ]
 [0.05 0.91 0.04]]
   result  prob_0  prob_1  prob_2
0       2    0.04    0.01    0.95
1       2    0.11    0.11    0.78
2       2    0.01    0.10    0.89
3       0    0.99    0.01    0.00
4       1    0.05    0.91    0.04
```


In [22]:
```
# ★tip : 데이터를 저장한다음 불러와서 제대로 제출했는지 확인해보자
# pd.DataFrame({'result': y_result}).to_csv('수험번호.csv', index=False)
# df2 = pd.read_csv("수험번호.csv")
# print(df2.head())
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js