Importing libraries

In [2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
```

load dataset

In [3]:
```
df=pd.read_csv('wine_classification_dataset.csv')
df.head()
```

Out[3]:

| | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols | flavanoids | nor |
|---|---|---|---|---|---|---|---|---|
| **0** | 14.23 | 1.71 | 2.43 | 15.6 | 127.0 | 2.80 | 3.06 | |
| **1** | 13.20 | 1.78 | 2.14 | 11.2 | 100.0 | 2.65 | 2.76 | |
| **2** | 13.16 | 2.36 | 2.67 | 18.6 | 101.0 | 2.80 | 3.24 | |
| **3** | 14.37 | 1.95 | 2.50 | 16.8 | 113.0 | 3.85 | 3.49 | |
| **4** | 13.24 | 2.59 | 2.87 | 21.0 | 118.0 | 2.80 | 2.69 | |

Columns list in the "wine_classification_dataset" dataset

In [4]:
```
df.columns
```

Out[4]:
```
Index(['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium',
       'total_phenols', 'flavanoids', 'nonflavanoid_phenols',
       'proanthocyanins', 'color_intensity', 'hue',
       'od280/od315_of_diluted_wines', 'proline', 'target'],
      dtype='object')
```

In [5]:
```
df.info()
df.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    int64  
dtypes: float64(13), int64(1)
memory usage: 19.6 KB
```

Out[5]:
```
(178, 14)
```

In [6]:
```
df.describe()
```

Out[6]:

| | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols |
|---|---|---|---|---|---|---|
| **count** | 178.000000 | 178.000000 | 178.000000 | 178.000000 | 178.000000 | 178.000000 |
| **mean** | 13.000618 | 2.336348 | 2.366517 | 19.494944 | 99.741573 | 2.295112 |
| **std** | 0.811827 | 1.117146 | 0.274344 | 3.339564 | 14.282484 | 0.625851 |
| **min** | 11.030000 | 0.740000 | 1.360000 | 10.600000 | 70.000000 | 0.980000 |
| **25%** | 12.362500 | 1.602500 | 2.210000 | 17.200000 | 88.000000 | 1.742500 |
| **50%** | 13.050000 | 1.865000 | 2.360000 | 19.500000 | 98.000000 | 2.355000 |
| **75%** | 13.677500 | 3.082500 | 2.557500 | 21.500000 | 107.000000 | 2.800000 |
| **max** | 14.830000 | 5.800000 | 3.230000 | 30.000000 | 162.000000 | 3.880000 |

Checking null values in each column

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
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
target                          0
dtype: int64
```

Features and Target

In [8]:
```
features = df.drop('target', axis='columns')
target_df = df['target']
features.head()
```

Out[8]:

| | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols | flavanoids | nor |
|---|---|---|---|---|---|---|---|---|
| **0** | 14.23 | 1.71 | 2.43 | 15.6 | 127.0 | 2.80 | 3.06 | |
| **1** | 13.20 | 1.78 | 2.14 | 11.2 | 100.0 | 2.65 | 2.76 | |
| **2** | 13.16 | 2.36 | 2.67 | 18.6 | 101.0 | 2.80 | 3.24 | |
| **3** | 14.37 | 1.95 | 2.50 | 16.8 | 113.0 | 3.85 | 3.49 | |
| **4** | 13.24 | 2.59 | 2.87 | 21.0 | 118.0 | 2.80 | 2.69 | |

In [9]:
```
target_df
```

Out[9]:
```
0      0
1      0
2      0
3      0
4      0
      ..
173    2
174    2
175    2
176    2
177    2
Name: target, Length: 178, dtype: int64
```

train_test_split

In [10]:
```
X_train, X_test, y_train, y_test = train_test_split(features, target_df,
```

In [11]:
```
print(X_train.shape)
X_train.head()
```

```
(142, 13)
```

Out[11]:

| | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols | flavanoids | n |
|---|---|---|---|---|---|---|---|---|
| **158** | 14.34 | 1.68 | 2.70 | 25.0 | 98.0 | 2.80 | 1.31 | |
| **137** | 12.53 | 5.51 | 2.64 | 25.0 | 96.0 | 1.79 | 0.60 | |
| **98** | 12.37 | 1.07 | 2.10 | 18.5 | 88.0 | 3.52 | 3.75 | |
| **159** | 13.48 | 1.67 | 2.64 | 22.5 | 89.0 | 2.60 | 1.10 | |
| **38** | 13.07 | 1.50 | 2.10 | 15.5 | 98.0 | 2.40 | 2.64 | |

In [12]:
```
print(X_test.shape)
X_test.head()
```

```
(36, 13)
```

Out[12]:

| | alcohol | malic_acid | ash | alcalinity_of_ash | magnesium | total_phenols | flavanoids | n |
|---|---|---|---|---|---|---|---|---|
| **19** | 13.64 | 3.10 | 2.56 | 15.2 | 116.0 | 2.70 | 3.03 | |
| **45** | 14.21 | 4.04 | 2.44 | 18.9 | 111.0 | 2.85 | 2.65 | |
| **140** | 12.93 | 2.81 | 2.70 | 21.0 | 96.0 | 1.54 | 0.50 | |
| **30** | 13.73 | 1.50 | 2.70 | 22.5 | 101.0 | 3.00 | 3.25 | |
| **67** | 12.37 | 1.17 | 1.92 | 19.6 | 78.0 | 2.11 | 2.00 | |

In [13]:
```
print(y_train.shape)
y_train.head()
```

```
(142,)
```

Out[13]:
```
158    2
137    2
98     1
159    2
38     0
Name: target, dtype: int64
```

In [14]:
```
print(y_test.shape)
y_test.head()
```

```
(36,)
```

Out[14]:
```
19     0
45     0
140    2
30     0
67     1
Name: target, dtype: int64
```

In [15]:
```
model=LogisticRegression()
model.fit(X_train, y_train)
```

```
C:\Users\sindhu\anaconda3\Lib\site-packages\sklearn\linear_model\_logistic.py
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regres
  n_iter_i = _check_optimize_result(
```

Out[15]:
```
▾ LogisticRegression
LogisticRegression()
```

In [16]:
```
y_pred=model.predict(X_test)
y_pred
```

Out[16]:
```
array([0, 0, 2, 0, 1, 0, 1, 2, 1, 2, 1, 2, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       1, 2, 2, 2, 1, 1, 1, 0, 0, 1, 2, 0, 0, 0], dtype=int64)
```

Accuracy

In [17]:
```
accuracy = accuracy_score(y_test, y_pred)
print(accuracy)
```

```
0.9722222222222222
```

In [18]:
```
classification_report(y_test, y_pred)
```

Out[18]:
```
'              precision    recall  f1-score   support\n\n           0
1.00      0.93      0.96        14\n           1       0.93      1.00
0.97        14\n           2       1.00      1.00      1.00         8\n\n
accuracy                           0.97        36\n   macro avg       0.98
0.98      0.98        36\nweighted avg       0.97      0.97      0.97
```

In [ ]:
```
                precision    recall  f1-score   support

           0       1.00      0.93      0.96        14
           1       0.93      1.00      0.97        14
           2       1.00      1.00      1.00         8

    accuracy                           0.97        36
   macro avg       0.98      0.98      0.98        36
weighted avg       0.97      0.97      0.97        36
```