Load the Wine Dataset

I load the built-in **Wine dataset** from scikit-learn.
This dataset contains chemical analysis of wines grown in the same region in Italy but derived from three different cultivars (classes 0, 1, 2).
The goal is to classify wines into their correct cultivar based on 13 numerical features such as alcohol, malic acid, and color intensity.

In [ ]:
```
from sklearn.datasets import load_wine
wine =load_wine()
```

Dataset Details

Print the Wine dataset object to see its description and structure.

In [ ]:
```
print(wine)
```

{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]], shape=(178, 13)), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2]), 'frame': None, 'target_names': array(['class_0', 'class_1', 'class_
2'], dtype='<U7'), 'DESCR': '.. _wine_dataset:\n\nWine recognition dataset\n--------
----------------\n\n**Data Set Characteristics:**\n\n:Number of Instances: 178\n:Num
ber of Attributes: 13 numeric, predictive attributes and the class\n:Attribute Infor
mation:\n   - Alcohol\n   - Malic acid\n   - Ash\n   - Alcalinity of ash\n   -
Magnesium\n   - Total phenols\n   - Flavanoids\n   - Nonflavanoid phenols\n   -
Proanthocyanins\n   - Color intensity\n   - Hue\n   - OD280/OD315 of diluted wine
s\n   - Proline\n   - class:\n   - class_0\n   - class_1\n   - clas
s_2\n\n:Summary Statistics:\n\n============================== ==== ===== ======= ====
=\n   Min   Max   Mean   SD\n========================
===== ==== ===== ======= =====\nAlcohol: 11.0 14.8   13.0
0.8\nMalic Acid: 0.74 5.80   2.34 1.12\nAsh:
1.36 3.23   2.36 0.27\nAlcalinity of Ash: 10.6 30.0   19.5   3.3\nMa
gnesium: 70.0 162.0   99.7 14.3\nTotal Phenols:
0.98 3.88   2.29 0.63\nFlavanoids: 0.34 5.08   2.03 1.00\nNo
nflavanoid Phenols: 0.13 0.66   0.36 0.12\nProanthocyanins:
0.41 3.58   1.59 0.57\nColour Intensity: 1.3 13.0   5.1   2.3\nHu
e: 0.48 1.71   0.96 0.23\nOD280/OD315 of diluted wines:
1.27 4.00   2.61 0.71\nProline: 278 1680   746   315\n==
============================ ==== ===== ======= =====\n\n:Missing Attribute Values: N
one\n:Class Distribution: class_0 (59), class_1 (71), class_2 (48)\n:Creator: R.A. F
isher\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n:Date: July, 1988\n
\nThis is a copy of UCI ML Wine recognition datasets.\nhttps://archive.ics.uci.edu/m
l/machine-learning-databases/wine/wine.data\n\nThe data is the results of a chemical
analysis of wines grown in the same\nregion in Italy by three different cultivators.
There are thirteen different\nmeasurements taken for different constituents found in
the three types of\nwine.\n\nOriginal Owners:\n\nForina, M. et al, PARVUS -\nAn Exte
ndible Package for Data Exploration, Classification and Correlation.\nInstitute of P
harmaceutical and Food Analysis and Technologies,\nVia Brigata Salerno, 16147 Genoa,
Italy.\n\nCitation:\n\nLichman, M. (2013). UCI Machine Learning Repository\n[http
s://archive.ics.uci.edu/ml]. Irvine, CA: University of California,\nSchool of Inform
ation and Computer Science.\n\n.. dropdown:: References\n\n   (1) S. Aeberhard, D.
Coomans and O. de Vel,\n   Comparison of Classifiers in High Dimensional Setting
s,\n   Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of\n   Ma
thematics and Statistics, James Cook University of North Queensland.\n   (Also subm

```
itted to Technometrics).\n\n    The data was used with many others for comparing var
ious\n    classifiers. The classes are separable, though only RDA\n    has achieved
100% correct classification.\n    (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-tr
ansformed data))\n    (All results using the leave-one-out technique)\n\n    (2) S.
Aeberhard, D. Coomans and O. de Vel,\n    "THE CLASSIFICATION PERFORMANCE OF RDA"\n
Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of\n    Mathematic
s and Statistics, James Cook University of North Queensland.\n    (Also submitted to
Journal of Chemometrics).\n', 'feature_names': ['alcohol', 'malic_acid', 'ash', 'alc
alinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols',
'proanthocyanins', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'prolin
e']}
```

Data Shape

Check the number of rows (samples) and columns (features) in the dataset.

In [ ]:
```
print(wine.data.shape)
```

```
(178, 13)
```

Target Shape

Check how many target labels (wine classes) are available in the dataset.

In [ ]:
```
print(wine.target.shape)
```

```
(178,)
```

Feature Names

Display the names of all input features in the Wine dataset.

In [ ]:
```
print(wine.feature_names)
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols',
'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue',
'od280/od315_of_diluted_wines', 'proline']
```

Target Class Names

Show the names of the wine classes (cultivars) present in the dataset.

In [ ]:
```
print(wine.target_names)
```

```
['class_0' 'class_1' 'class_2']
```

Preview Data

Display the first 5 rows of feature values and their corresponding target labels.

In [ ]:
```
print(wine.data[:5])
print(wine.target[:5])
```

```
[[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e+00
  2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 1.120e+01 1.000e+02 2.650e+00 2.760e+00
  2.600e-01 1.280e+00 4.380e+00 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 1.860e+01 1.010e+02 2.800e+00 3.240e+00
  3.000e-01 2.810e+00 5.680e+00 1.030e+00 3.170e+00 1.185e+03]
 [1.437e+01 1.950e+00 2.500e+00 1.680e+01 1.130e+02 3.850e+00 3.490e+00
  2.400e-01 2.180e+00 7.800e+00 8.600e-01 3.450e+00 1.480e+03]
 [1.324e+01 2.590e+00 2.870e+00 2.100e+01 1.180e+02 2.800e+00 2.690e+00
  3.900e-01 1.820e+00 4.320e+00 1.040e+00 2.930e+00 7.350e+02]]
[0 0 0 0 0]
```

Create DataFrame

Convert the dataset into a pandas DataFrame, add the target column, and view the first 5 rows.

In [ ]:
```
import pandas as pd
df = pd.DataFrame(wine.data,columns=wine.feature_names)
df['target']=wine.target
df.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
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	

Summary Statistics

View statistical summary (mean, std, min, max, quartiles) of all features.

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

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flava
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.0
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.9
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.3
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.2
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.1
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.8
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.0

Alcohol Feature Summary

Show statistical details (mean, std, min, max, quartiles) for the alcohol column only.

In [ ]:
```
from sklearn.datasets import load_wine
wine =load_wine()
```

Raw Dataset Object

Print the raw Wine dataset object to check its description and metadata.

In [ ]:
```
print(wine)
```

{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]], shape=(178, 13)), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2]), 'frame': None, 'target_names': array(['class_0', 'class_1', 'class_
2'], dtype='<U7'), 'DESCR': '.. _wine_dataset:\n\nWine recognition dataset\n--------
----------------\n\n**Data Set Characteristics:**\n\n:Number of Instances: 178\n:Num
ber of Attributes: 13 numeric, predictive attributes and the class\n:Attribute Infor
mation:\n   - Alcohol\n   - Malic acid\n   - Ash\n   - Alcalinity of ash\n   -
Magnesium\n   - Total phenols\n   - Flavanoids\n   - Nonflavanoid phenols\n   -
Proanthocyanins\n   - Color intensity\n   - Hue\n   - OD280/OD315 of diluted wine
s\n   - Proline\n   - class:\n   - class_0\n   - class_1\n   - clas
s_2\n\n:Summary Statistics:\n\n============================== ==== ===== ======= ====
=\n   Min   Max   Mean   SD\n========================
===== ==== ===== ======= =====\nAlcohol: 11.0 14.8   13.0
0.8\nMalic Acid: 0.74 5.80   2.34 1.12\nAsh:
1.36 3.23   2.36 0.27\nAlcalinity of Ash: 10.6 30.0   19.5   3.3\nMa
gnesium: 70.0 162.0   99.7 14.3\nTotal Phenols:
0.98 3.88   2.29 0.63\nFlavanoids: 0.34 5.08   2.03 1.00\nNo
nflavanoid Phenols: 0.13 0.66   0.36 0.12\nProanthocyanins:
0.41 3.58   1.59 0.57\nColour Intensity: 1.3 13.0   5.1   2.3\nHu
e: 0.48 1.71   0.96 0.23\nOD280/OD315 of diluted wines:
1.27 4.00   2.61 0.71\nProline: 278 1680   746   315\n==
============================ ==== ===== ======= =====\n\n:Missing Attribute Values: N
one\n:Class Distribution: class_0 (59), class_1 (71), class_2 (48)\n:Creator: R.A. F
isher\n:Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)\n:Date: July, 1988\n
\nThis is a copy of UCI ML Wine recognition datasets.\nhttps://archive.ics.uci.edu/m
l/machine-learning-databases/wine/wine.data\n\nThe data is the results of a chemical
analysis of wines grown in the same\nregion in Italy by three different cultivators.
There are thirteen different\nmeasurements taken for different constituents found in
the three types of\nwine.\n\nOriginal Owners:\n\nForina, M. et al, PARVUS -\nAn Exte
ndible Package for Data Exploration, Classification and Correlation.\nInstitute of P
harmaceutical and Food Analysis and Technologies,\nVia Brigata Salerno, 16147 Genoa,
Italy.\n\nCitation:\n\nLichman, M. (2013). UCI Machine Learning Repository\n[http
s://archive.ics.uci.edu/ml]. Irvine, CA: University of California,\nSchool of Inform
ation and Computer Science.\n\n.. dropdown:: References\n\n   (1) S. Aeberhard, D.
Coomans and O. de Vel,\n   Comparison of Classifiers in High Dimensional Setting
s,\n   Tech. Rep. no. 92-02, (1992), Dept. of Computer Science and Dept. of\n   Ma
thematics and Statistics, James Cook University of North Queensland.\n   (Also subm

```
itted to Technometrics).\n\n    The data was used with many others for comparing var
ious\n    classifiers. The classes are separable, though only RDA\n    has achieved
100% correct classification.\n    (RDA : 100%, QDA 99.4%, LDA 98.9%, 1NN 96.1% (z-tr
ansformed data))\n    (All results using the leave-one-out technique)\n\n    (2) S.
Aeberhard, D. Coomans and O. de Vel,\n    "THE CLASSIFICATION PERFORMANCE OF RDA"\n
Tech. Rep. no. 92-01, (1992), Dept. of Computer Science and Dept. of\n    Mathematic
s and Statistics, James Cook University of North Queensland.\n    (Also submitted to
Journal of Chemometrics).\n', 'feature_names': ['alcohol', 'malic_acid', 'ash', 'alc
alinity_of_ash', 'magnesium', 'total_phenols', 'flavanoids', 'nonflavanoid_phenols',
'proanthocyanins', 'color_intensity', 'hue', 'od280/od315_of_diluted_wines', 'prolin
e']}
```

Feature Data Shape

Check the dimensions of the feature data (rows × columns).

In [ ]:
```
print(wine.data.shape)
```

```
(178, 13)
```

Target Data Shape

Check how many target values (labels) are present in the dataset.

In [ ]:
```
print(wine.target.shape)
```

```
(178,)
```

Feature Names

List all the feature names used in the Wine dataset.

In [ ]:
```
print(wine.feature_names)
```

```
['alcohol', 'malic_acid', 'ash', 'alcalinity_of_ash', 'magnesium', 'total_phenols',
'flavanoids', 'nonflavanoid_phenols', 'proanthocyanins', 'color_intensity', 'hue',
'od280/od315_of_diluted_wines', 'proline']
```

Target Class Labels

Display the names of the wine classes (0 = class_0, 1 = class_1, 2 = class_2).

In [ ]:
```
print(wine.target_names)
```

```
['class_0' 'class_1' 'class_2']
```

Preview of Data and Targets

Show the first 5 rows of feature values along with their corresponding target labels.

In [ ]:
```
print(wine.data[:5])
print(wine.target[:5])
```

```
[[1.423e+01 1.710e+00 2.430e+00 1.560e+01 1.270e+02 2.800e+00 3.060e+00
  2.800e-01 2.290e+00 5.640e+00 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 1.120e+01 1.000e+02 2.650e+00 2.760e+00
  2.600e-01 1.280e+00 4.380e+00 1.050e+00 3.400e+00 1.050e+03]
 [1.316e+01 2.360e+00 2.670e+00 1.860e+01 1.010e+02 2.800e+00 3.240e+00
  3.000e-01 2.810e+00 5.680e+00 1.030e+00 3.170e+00 1.185e+03]
 [1.437e+01 1.950e+00 2.500e+00 1.680e+01 1.130e+02 3.850e+00 3.490e+00
  2.400e-01 2.180e+00 7.800e+00 8.600e-01 3.450e+00 1.480e+03]
 [1.324e+01 2.590e+00 2.870e+00 2.100e+01 1.180e+02 2.800e+00 2.690e+00
  3.900e-01 1.820e+00 4.320e+00 1.040e+00 2.930e+00 7.350e+02]]
[0 0 0 0 0]
```

Create DataFrame

Convert the Wine dataset into a pandas DataFrame, add the target column, and display the first 5 rows.

In [ ]:
```
import pandas as pd
df = pd.DataFrame(wine.data,columns=wine.feature_names)
df['target']=wine.target
df.head()
```

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflav
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	

Dataset Statistics

Generate summary statistics (count, mean, std, min, max, quartiles) for all features in the dataset.

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

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flava
count	178.000000	178.000000	178.000000	178.000000	178.000000	178.000000	178.0
mean	13.000618	2.336348	2.366517	19.494944	99.741573	2.295112	2.0
std	0.811827	1.117146	0.274344	3.339564	14.282484	0.625851	0.9
min	11.030000	0.740000	1.360000	10.600000	70.000000	0.980000	0.3
25%	12.362500	1.602500	2.210000	17.200000	88.000000	1.742500	1.2
50%	13.050000	1.865000	2.360000	19.500000	98.000000	2.355000	2.1
75%	13.677500	3.082500	2.557500	21.500000	107.000000	2.800000	2.8
max	14.830000	5.800000	3.230000	30.000000	162.000000	3.880000	5.0

Alcohol Column Statistics

Show detailed statistics (count, mean, std, min, max, quartiles) for the alcohol feature only.

In [ ]:
```
df['alcohol'].describe()
```

```
count    178.000000
mean      13.000618
std        0.811827
min       11.030000
25%       12.362500
50%       13.050000
75%       13.677500
max       14.830000
Name: alcohol, dtype: float64
```

Dataset Information

Display info about the DataFrame including column names, data types, and non-null counts.

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

```
<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
```

Target Class Distribution

Count the number of samples in each wine class (0, 1, 2).

In [ ]:
```
df['target'].value_counts()
```

```
target
1    71
0    59
2    48
Name: count, dtype: int64
```

Dataset Shape

Check the total number of rows (samples) and columns (features + target) in the dataset.

In [ ]:
```
df.shape
```

```
(178, 14)
```

Check for Missing Values

Display a boolean DataFrame showing True/False for missing values in each cell.

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

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonf
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
173	False	False	False	False	False	False	False	
174	False	False	False	False	False	False	False	
175	False	False	False	False	False	False	False	
176	False	False	False	False	False	False	False	
177	False	False	False	False	False	False	False	

178 rows × 14 columns

Missing Values Count

Show the number of missing values in each column of the dataset.

In [ ]:
```
df.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
target                          0
dtype: int64
```

Total Missing Values

Calculate the total number of missing values across the entire dataset.

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

```
np.int64(0)
```

Any Missing Values?

Check if the dataset contains any missing values (True/False).

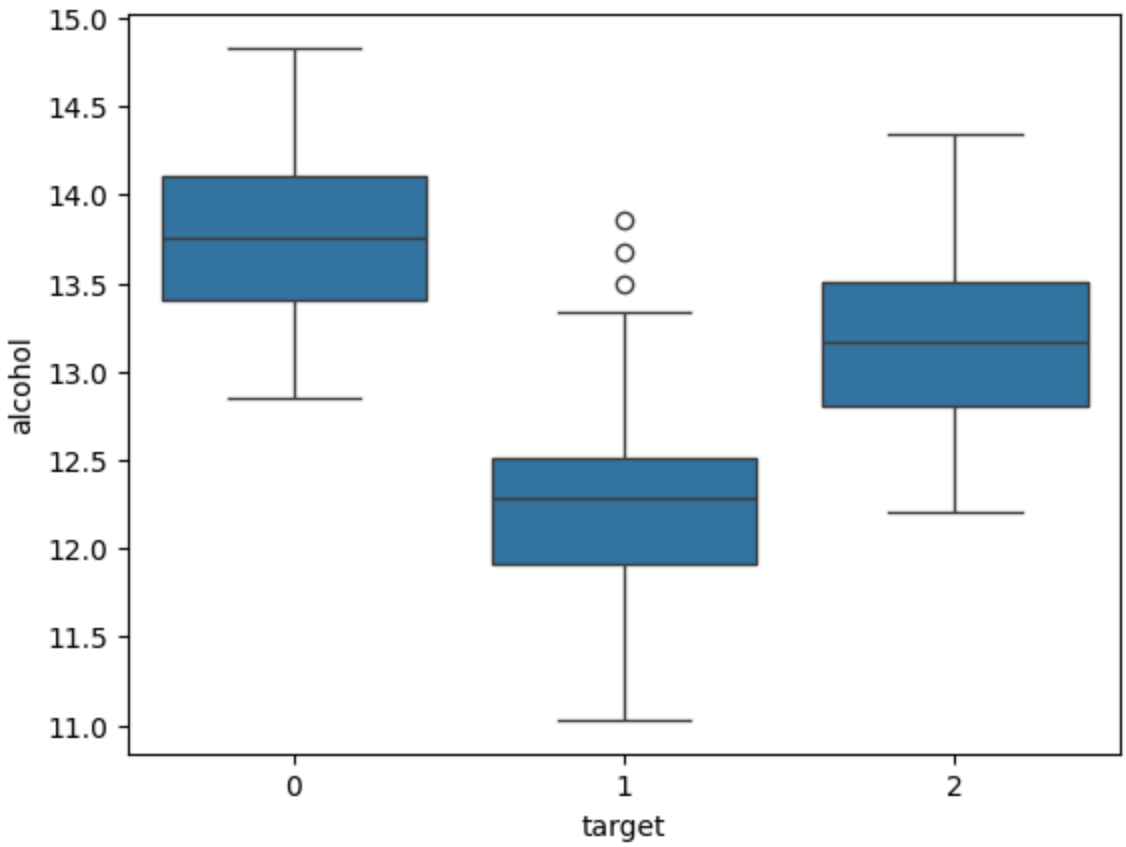
In [ ]:
```
df.isnull().values.any()
```

```
np.False_
```

Boxplot of Alcohol by Target

Visualize the distribution of alcohol content across different wine classes using a boxplot.

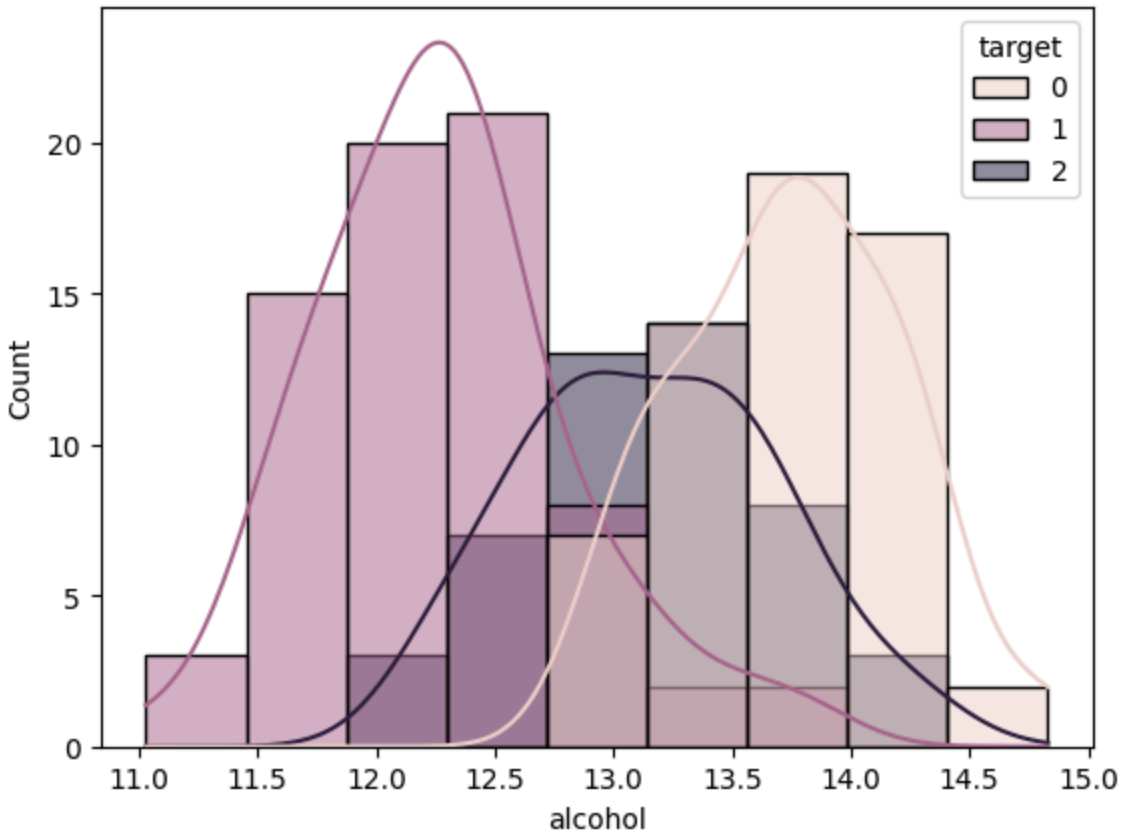
In [ ]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot(x='target',y='alcohol',data=df)
plt.show()
```



Alcohol Distribution by Class

Plot the distribution of alcohol values for each wine class using a histogram with KDE.

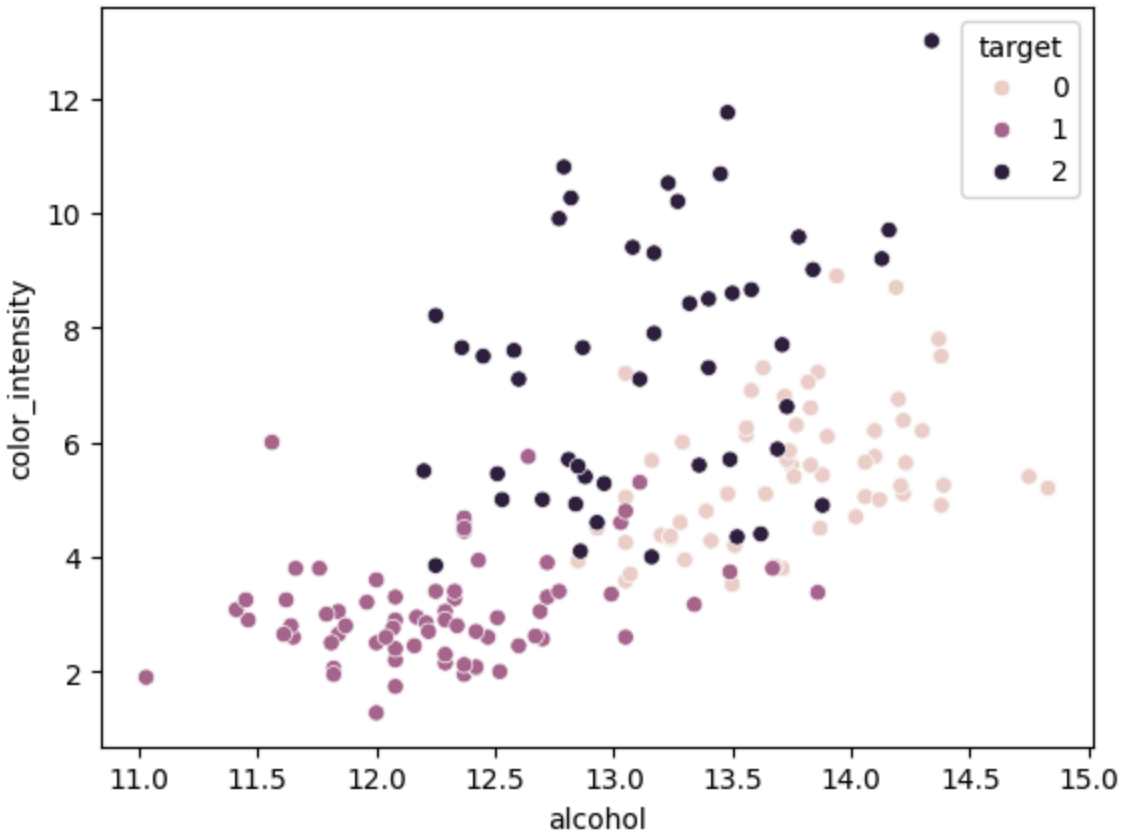
In [ ]:
```
sns.histplot(x='alcohol',hue='target',kde=True,data=df)
plt.show()
```



Scatter Plot (Alcohol vs Color Intensity)

Visualize the relationship between alcohol and color intensity, colored by wine class.

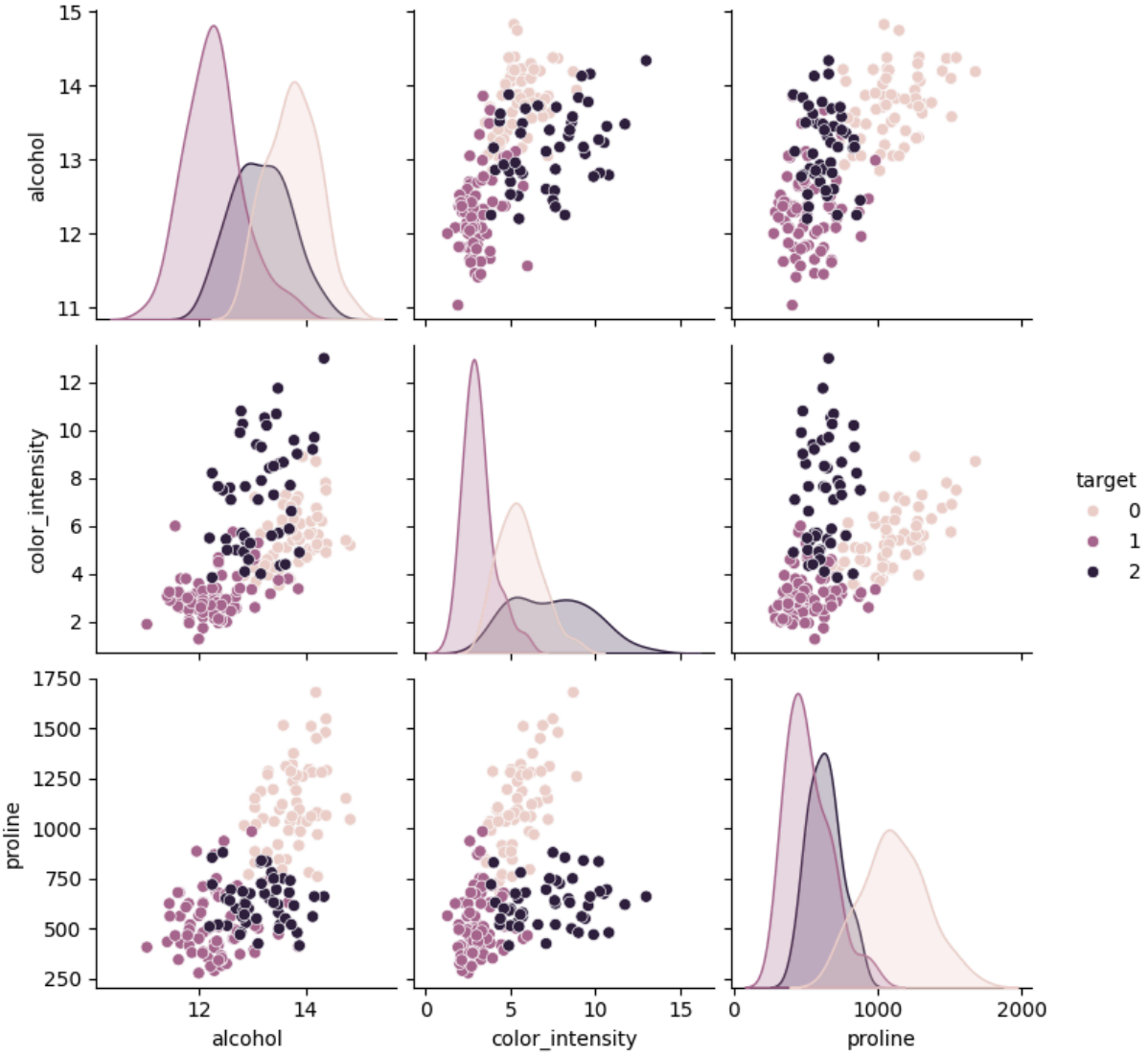
In [ ]:
```
sns.scatterplot(x='alcohol',y='color_intensity',hue='target',data=df)
plt.show()
```



Pairplot of Selected Features

Plot pairwise relationships between alcohol, color intensity, and proline, separated by wine class.

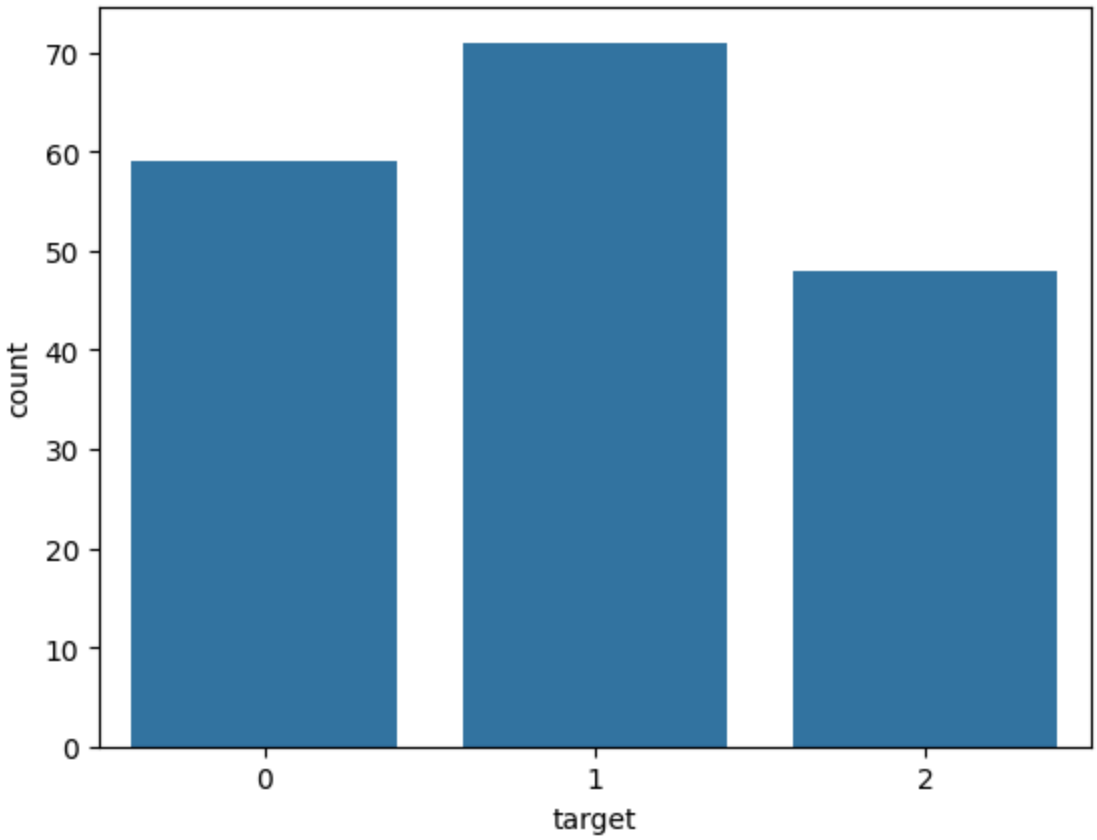
In [ ]:
```
sns.pairplot(df,vars=["alcohol","color_intensity","proline"],hue='target')
plt.show()
```



Target Class Counts

Show the number of samples available in each wine class using a count plot.

In [ ]:
```
sns.countplot(x='target',data=df)
plt.show()
```



Train-Test Split

Split the dataset into training (80%) and testing (20%) sets to evaluate model performance.

In [ ]:
```
from sklearn.model_selection import train_test_split
x=df.drop("target",axis=1)
y=df['target']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
plt.show()
```

Preview Train and Test Sets

Display the feature values of training and testing sets along with their target labels.

In [ ]:
```
print(x_train)
print(x_test)
print(y_train)
print(y_test)
```

```
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
158    14.34        1.68  2.70               25.0       98.0           2.80
137    12.53        5.51  2.64               25.0       96.0           1.79
98     12.37        1.07  2.10               18.5       88.0           3.52
159    13.48        1.67  2.64               22.5       89.0           2.60
38     13.07        1.50  2.10               15.5       98.0           2.40
..       ...         ...   ...                ...        ...            ...
71     13.86        1.51  2.67               25.0       86.0           2.95
106    12.25        1.73  2.12               19.0       80.0           1.65
14     14.38        1.87  2.38               12.0      102.0           3.30
92     12.69        1.53  2.26               20.7       80.0           1.38
102    12.34        2.45  2.46               21.0       98.0           2.56

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
158        1.31                  0.53             2.70            13.00  0.57
137        0.60                  0.63             1.10             5.00  0.82
98         3.75                  0.24             1.95             4.50  1.04
159        1.10                  0.52             2.29            11.75  0.57
38         2.64                  0.28             1.37             3.70  1.18
..          ...                   ...              ...              ...   ...
71         2.86                  0.21             1.87             3.38  1.36
106        2.03                  0.37             1.63             3.40  1.00
14         3.64                  0.29             2.96             7.50  1.20
92         1.46                  0.58             1.62             3.05  0.96
102        2.11                  0.34             1.31             2.80  0.80

     od280/od315_of_diluted_wines  proline
158                          1.96    660.0
137                          1.69    515.0
98                           2.77    660.0
159                          1.78    620.0
38                           2.69   1020.0
..                            ...      ...
71                           3.16    410.0
106                          3.17    510.0
14                           3.00   1547.0
92                           2.06    495.0
102                          3.38    438.0

[142 rows x 13 columns]
     alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols  \
19     13.64        3.10  2.56               15.2      116.0           2.70
45     14.21        4.04  2.44               18.9      111.0           2.85
140    12.93        2.81  2.70               21.0       96.0           1.54
30     13.73        1.50  2.70               22.5      101.0           3.00
67     12.37        1.17  1.92               19.6       78.0           2.11
16     14.30        1.92  2.72               20.0      120.0           2.80
119    12.00        3.43  2.00               19.0       87.0           2.00
174    13.40        3.91  2.48               23.0      102.0           1.80
109    11.61        1.35  2.70               20.0       94.0           2.74
141    13.36        2.56  2.35               20.0       89.0           1.40
24     13.50        1.81  2.61               20.0       96.0           2.53
150    13.50        3.12  2.62               24.0      123.0           1.40
41     13.41        3.84  2.12               18.8       90.0           2.45
118    12.77        3.43  1.98               16.0       80.0           1.63
15     13.63        1.81  2.70               17.2      112.0           2.85
```

111	12.52	2.43	2.17	21.0	88.0	2.55
113	11.41	0.74	2.50	21.0	88.0	2.48
82	12.08	1.13	2.51	24.0	78.0	2.00
9	13.86	1.35	2.27	16.0	98.0	2.98
114	12.08	1.39	2.50	22.5	84.0	2.56
18	14.19	1.59	2.48	16.5	108.0	3.30
66	13.11	1.01	1.70	15.0	78.0	2.98
60	12.33	1.10	2.28	16.0	101.0	2.05
169	13.40	4.60	2.86	25.0	112.0	1.98
171	12.77	2.39	2.28	19.5	86.0	1.39
164	13.78	2.76	2.30	22.0	90.0	1.35
117	12.42	1.61	2.19	22.5	108.0	2.00
65	12.37	1.21	2.56	18.1	98.0	2.42
90	12.08	1.83	2.32	18.5	81.0	1.60
55	13.56	1.73	2.46	20.5	116.0	2.96
29	14.02	1.68	2.21	16.0	96.0	2.65
128	12.37	1.63	2.30	24.5	88.0	2.22
145	13.16	3.57	2.15	21.0	102.0	1.50
31	13.58	1.66	2.36	19.1	106.0	2.86
12	13.75	1.73	2.41	16.0	89.0	2.60
42	13.88	1.89	2.59	15.0	101.0	3.25

	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	\
19	3.03	0.17	1.66	5.100000	0.96	
45	2.65	0.30	1.25	5.240000	0.87	
140	0.50	0.53	0.75	4.600000	0.77	
30	3.25	0.29	2.38	5.700000	1.19	
67	2.00	0.27	1.04	4.680000	1.12	
16	3.14	0.33	1.97	6.200000	1.07	
119	1.64	0.37	1.87	1.280000	0.93	
174	0.75	0.43	1.41	7.300000	0.70	
109	2.92	0.29	2.49	2.650000	0.96	
141	0.50	0.37	0.64	5.600000	0.70	
24	2.61	0.28	1.66	3.520000	1.12	
150	1.57	0.22	1.25	8.600000	0.59	
41	2.68	0.27	1.48	4.280000	0.91	
118	1.25	0.43	0.83	3.400000	0.70	
15	2.91	0.30	1.46	7.300000	1.28	
111	2.27	0.26	1.22	2.000000	0.90	
113	2.01	0.42	1.44	3.080000	1.10	
82	1.58	0.40	1.40	2.200000	1.31	
9	3.15	0.22	1.85	7.220000	1.01	
114	2.29	0.43	1.04	2.900000	0.93	
18	3.93	0.32	1.86	8.700000	1.23	
66	3.18	0.26	2.28	5.300000	1.12	
60	1.09	0.63	0.41	3.270000	1.25	
169	0.96	0.27	1.11	8.500000	0.67	
171	0.51	0.48	0.64	9.899999	0.57	
164	0.68	0.41	1.03	9.580000	0.70	
117	2.09	0.34	1.61	2.060000	1.06	
65	2.65	0.37	2.08	4.600000	1.19	
90	1.50	0.52	1.64	2.400000	1.08	
55	2.78	0.20	2.45	6.250000	0.98	
29	2.33	0.26	1.98	4.700000	1.04	
128	2.45	0.40	1.90	2.120000	0.89	
145	0.55	0.43	1.30	4.000000	0.60	

```
31         3.19                  0.22            1.95         6.900000  1.09
12         2.76                  0.29            1.81         5.600000  1.15
42         3.56                  0.17            1.70         5.430000  0.88

     od280/od315_of_diluted_wines  proline
19                           3.36    845.0
45                           3.33   1080.0
140                          2.31    600.0
30                           2.71   1285.0
67                           3.48    510.0
16                           2.65   1280.0
119                          3.05    564.0
174                          1.56    750.0
109                          3.26    680.0
141                          2.47    780.0
24                           3.82    845.0
150                          1.30    500.0
41                           3.00   1035.0
118                          2.12    372.0
15                           2.88   1310.0
111                          2.78    325.0
113                          2.31    434.0
82                           2.72    630.0
9                            3.55   1045.0
114                          3.19    385.0
18                           2.82   1680.0
66                           3.18    502.0
60                           1.67    680.0
169                          1.92    630.0
171                          1.63    470.0
164                          1.68    615.0
117                          2.96    345.0
65                           2.30    678.0
90                           2.27    480.0
55                           3.03   1120.0
29                           3.59   1035.0
128                          2.78    342.0
145                          1.68    830.0
31                           2.88   1515.0
12                           2.90   1320.0
42                           3.56   1095.0
158    2
137    2
98     1
159    2
38     0
      ..
71     1
106    1
14     0
92     1
102    1
Name: target, Length: 142, dtype: int64
19     0
45     0
140    2
```

```
30     0
67     1
16     0
119    1
174    2
109    1
141    2
24     0
150    2
41     0
118    1
15     0
111    1
113    1
82     1
9      0
114    1
18     0
66     1
60     1
169    2
171    2
164    2
117    1
65     1
90     1
55     0
29     0
128    1
145    2
31     0
12     0
42     0
Name: target, dtype: int64
```

Train-Test Shapes

Check the dimensions of training and testing sets for both features and target labels.

In [ ]:
```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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

Preview Training Data

Show the first few rows of training features and their corresponding target labels.

In [ ]:
```
print("X_train rows:",x_train.head())
print("Y_train rows:",y_train.head())
```

```
X_train rows:      alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_ph
enols  \
158    14.34        1.68  2.70               25.0       98.0           2.80
137    12.53        5.51  2.64               25.0       96.0           1.79
98     12.37        1.07  2.10               18.5       88.0           3.52
159    13.48        1.67  2.64               22.5       89.0           2.60
38     13.07        1.50  2.10               15.5       98.0           2.40

     flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hue  \
158        1.31                  0.53             2.70            13.00  0.57
137        0.60                  0.63             1.10             5.00  0.82
98         3.75                  0.24             1.95             4.50  1.04
159        1.10                  0.52             2.29            11.75  0.57
38         2.64                  0.28             1.37             3.70  1.18

     od280/od315_of_diluted_wines  proline
158                          1.96    660.0
137                          1.69    515.0
98                           2.77    660.0
159                          1.78    620.0
38                           2.69   1020.0
Y_train rows: 158    2
137    2
98     1
159    2
38     0
Name: target, dtype: int64
```

Feature Scaling

Standardize the training and testing feature data so that all features have mean 0 and standard deviation 1.

In [ ]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Verify Scaled Data

Display the first 5 rows of scaled training data and check that the mean is ~0 and standard deviation is ~1.

In [ ]:
```
print("First 5 rows of scaled training data:\n",x_train_scaled[:5])
print("\n Mean of scaled training data(approx 0):\n",x_train_scaled.mean(axis=0))
print("\n Std of scaled training data(approx 1):\n",x_train_scaled.std(axis=0))
```

```
First 5 rows of scaled training data:
 [[ 1.66529275 -0.60840587  1.21896194  1.60540017 -0.16738426  0.80400157
  -0.6916784   1.26722552  1.8775398   3.41947305 -1.65632857 -0.87940904
  -0.24860607]
 [-0.54952506  2.7515415   1.00331502  1.60540017 -0.30437887 -0.78538376
  -1.40123291  2.04959953 -0.87350523 -0.0248012  -0.58463272 -1.25462095
  -0.72992237]
 [-0.74531007 -1.14354109 -0.93750727 -0.28270426 -0.8523573   1.93702874
   1.7467906  -1.00165913  0.58798744 -0.24006834  0.35845962  0.2462267
  -0.24860607]
 [ 0.61294837 -0.61717858  1.00331502  0.87920616 -0.78385999  0.4892718
  -0.90154664  1.18898812  1.17258451  2.8813052  -1.65632857 -1.12955031
  -0.38138298]
 [ 0.11124931 -0.76631462 -0.93750727 -1.15413707 -0.16738426  0.17454204
   0.63748708 -0.68870952 -0.40926638 -0.58449577  0.95860929  0.1350528
   0.94638614]]

Mean of scaled training data(approx 0):
[ 8.30321727e-15 -5.73387191e-16  4.72196617e-15  1.22359087e-16
-3.69813726e-16  1.83343169e-16  9.90795865e-16 -1.01268759e-15
 3.63558948e-16  3.19775505e-16  1.24626444e-15  1.76697467e-16
-1.56369440e-18]

Std of scaled training data(approx 1):
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

Random Forest Classifier

Train a Random Forest model with 100 trees, make predictions on the test set, and evaluate performance using accuracy, confusion matrix, and classification report.

In [ ]:
```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

rf=RandomForestClassifier(n_estimators=100,random_state=42)
rf.fit(x_train,y_train)

#predictions
y_pred=rf.predict(x_test)

# Comparing prediction and actual answers
print("Random Forest Accuracy:",accuracy_score(y_test,y_pred))
print("\nConfusion_matrix:\n",confusion_matrix(y_test,y_pred))
print("\nClassification_report:\n",classification_report(y_test,y_pred))
```

```
Random Forest Accuracy: 1.0

Confusion_matrix:
 [[14  0  0]
 [ 0 14  0]
 [ 0  0  8]]

Classification_report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        14
           1       1.00      1.00      1.00        14
           2       1.00      1.00      1.00         8

    accuracy                           1.00        36
   macro avg       1.00      1.00      1.00        36
weighted avg       1.00      1.00      1.00        36
```

Logistic Regression

Train a Logistic Regression model on the training set, predict on the test set, and evaluate using accuracy, confusion matrix, and classification report.

In [ ]:
```
from sklearn.linear_model import LogisticRegression

log_reg = LogisticRegression(max_iter=500,random_state=42)
log_reg.fit(x_train,y_train)

y_pred_lr = log_reg.predict(x_test)

print("Logistic Regression accuracy score:",accuracy_score(y_test,y_pred_lr))
print("\nLogistic Regression confusion_matrix:\n",confusion_matrix(y_test,y_pred_lr
print("\nLogistic Regression classification_report:\n",classification_report(y_test
```

```
Logistic Regression accuracy score: 0.9722222222222222

Logistic Regression confusion_matrix:
 [[13  1  0]
 [ 0 14  0]
 [ 0  0  8]]

Logistic Regression classification_report:
              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
```

```
c:\Users\rohan\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\linear_model\_logistic.py:473: ConvergenceWarning: lbfgs failed to converge after 500 iteration(s) (status=1):
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT

Increase the number of iterations to improve the convergence (max_iter=500).
You might also want to 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-regression
  n_iter_i = _check_optimize_result(
```

Model Accuracy Comparison

Print and compare the accuracy scores of Random Forest and Logistic Regression models.

In [ ]:
```
print("Random Forest Accuracy:",accuracy_score(y_test,y_pred))
print("Logistic Regression accuracy score:",accuracy_score(y_test,y_pred_lr))
```

```
Random Forest Accuracy: 1.0
Logistic Regression accuracy score: 0.9722222222222222
```

Short Observations

The Random Forest Classifier achieved an accuracy of 100%, while Logistic Regression achieved an accuracy of 97.2%. Random Forest performed slightly better and captured the dataset patterns perfectly, making it the stronger model for this task. Logistic Regression, though a bit less accurate, still gave very good results and can be used as a simple, fast baseline model.

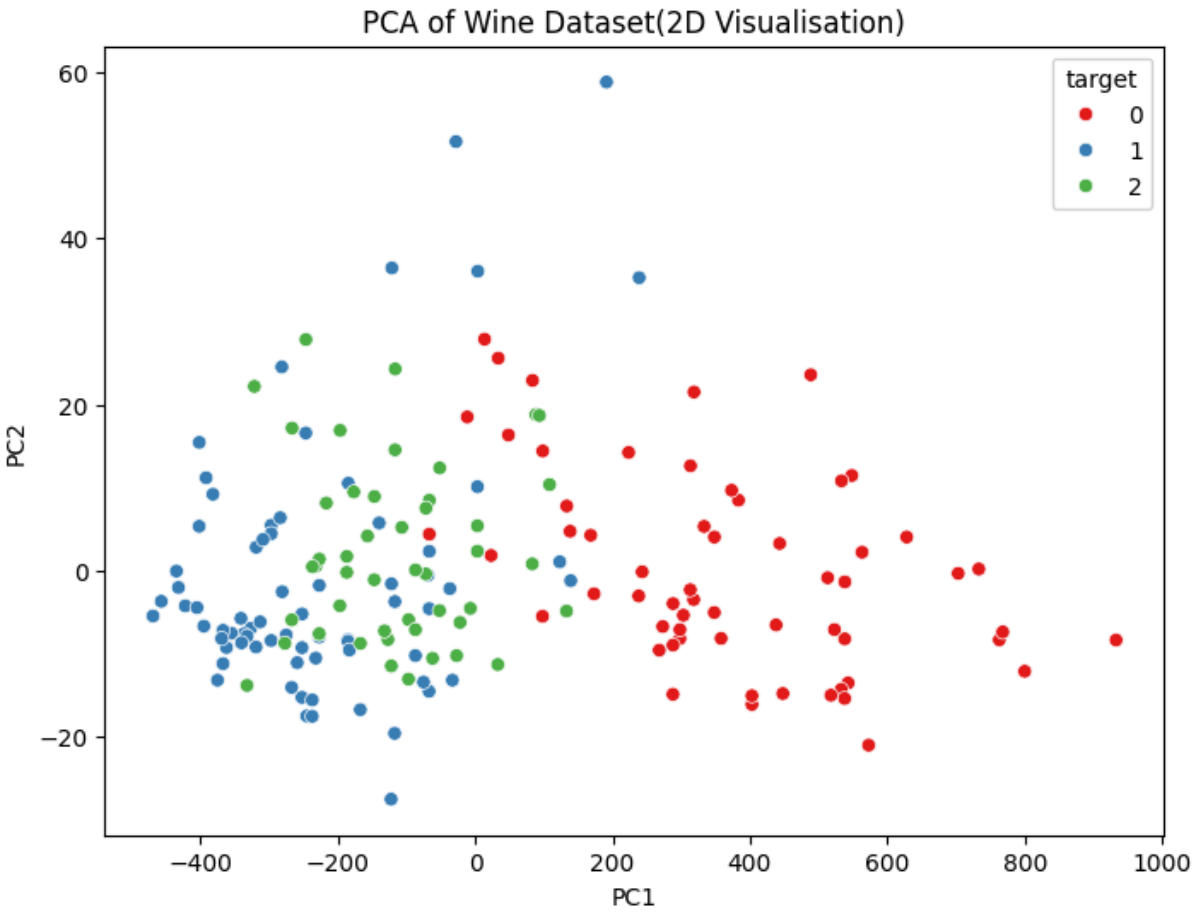
PCA 2D Visualization

Apply Principal Component Analysis (PCA) to reduce the features to 2 components and plot them to visualize class separation in 2D.

In [ ]:
```
from sklearn.decomposition import PCA


wine = load_wine()
df =pd.DataFrame(wine.data,columns=wine.feature_names)
df["target"]=wine.target
X=df.drop("target",axis=1)
y=df['target']
pca =PCA(n_components=2)
x_pca=pca.fit_transform(X)
pca_df=pd.DataFrame(data=x_pca,columns=['PC1','PC2'])
pca_df['target']=y
plt.figure(figsize=(8,6))
sns.scatterplot(x="PC1",y="PC2",hue='target',palette='Set1',data=pca_df)
```

```
plt.title("PCA of Wine Dataset(2D Visualisation)")
plt.show()
```



Hyperparameter Tuning with GridSearchCV

Use GridSearchCV to find the best Random Forest parameters (number of trees, maximum depth, minimum samples split) and display the best parameters with cross-validation accuracy.

In [ ]:
```
from sklearn.model_selection import GridSearchCV

rf=RandomForestClassifier(random_state=42)
rf.fit(x_train,y_train)
param_grid={
    'n_estimators':[50,100,200],
    'max_depth':[None,5,10],
    'min_samples_split':[2,5,10]
}
grid_search = GridSearchCV(estimator=rf,param_grid=param_grid,cv=5,scoring='accurac
grid_search.fit(x_train,y_train)
print("Best parameters:",grid_search.best_params_)
print("Best Cross Validation Accuracy:",grid_search.best_score_)
```

```
Best parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 100}
Best Cross Validation Accuracy: 0.9785714285714286
```

Feature Importance Table

Extract feature importances from the Random Forest model and display them in descending order.

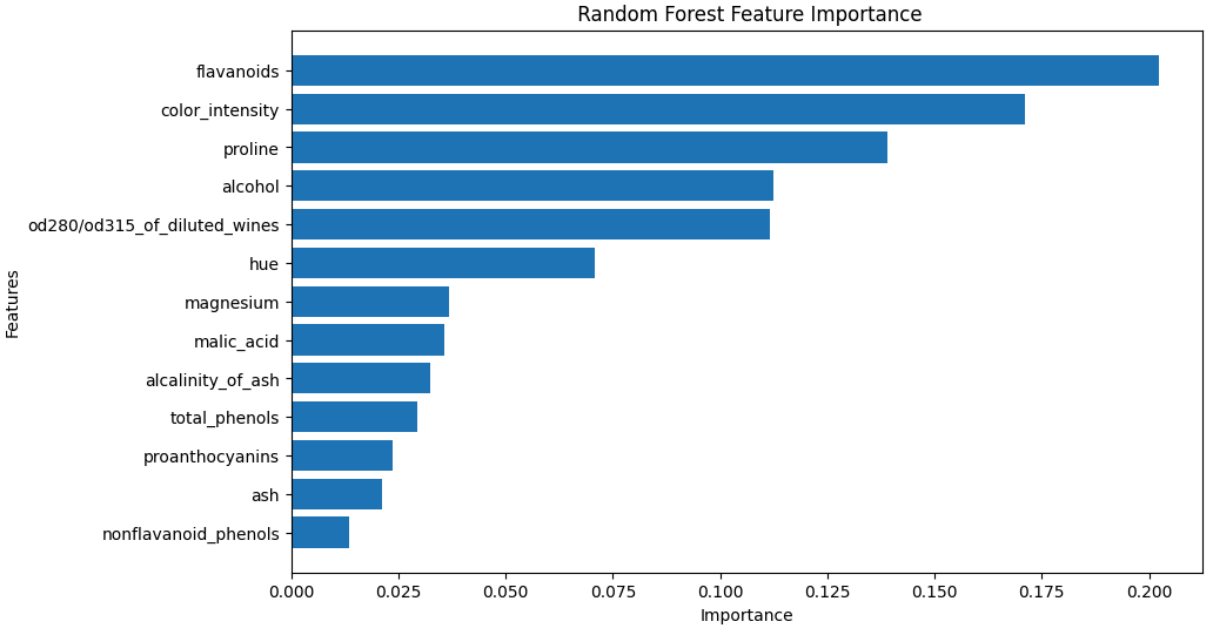
In [ ]:
```
importances = rf.feature_importances_
feature_names = x_train.columns
feat_imp = pd.DataFrame({'Features':feature_names,'Importance':importances })
feat_imp = feat_imp.sort_values(by="Importance",ascending=False)
print(feat_imp)
```

```
                       Features  Importance
6                     flavanoids    0.202293
9                color_intensity    0.171202
12                       proline    0.139046
0                        alcohol    0.112398
11  od280/od315_of_diluted_wines    0.111564
10                           hue    0.070891
4                      magnesium    0.036841
1                    malic_acid    0.035703
3              alcalinity_of_ash    0.032425
5                  total_phenols    0.029279
8                proanthocyanins    0.023561
2                            ash    0.021282
7          nonflavanoid_phenols    0.013515
```

Feature Importance Plot

Visualize the Random Forest feature importances as a horizontal bar chart to see which features contribute most to classification.

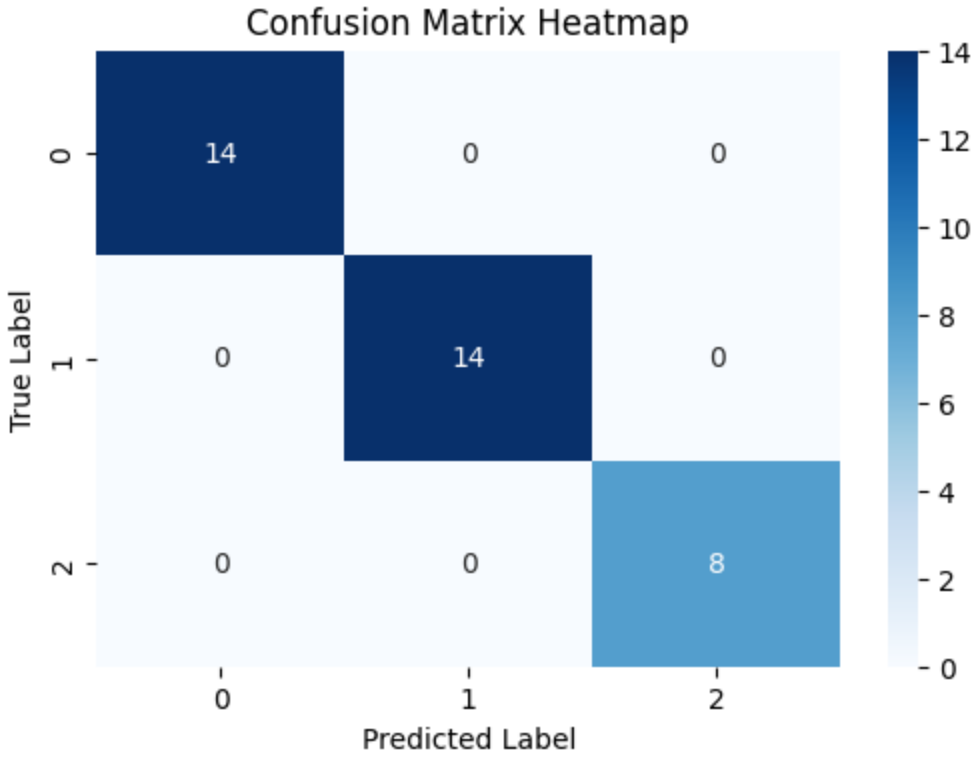
In [ ]:
```
plt.figure(figsize=(10,6))
plt.barh(feat_imp["Features"],feat_imp["Importance"])
plt.title(" Random Forest Feature Importance")
plt.gca().invert_yaxis()
plt.xlabel("Importance")
plt.ylabel("Features")
plt.show()
```



Confusion Matrix Heatmap

Plot the confusion matrix as a heatmap to clearly visualize correct and incorrect predictions across the three wine classes.

In [ ]:
```
cm=confusion_matrix(y_test,y_pred)
plt.figure(figsize=(6,4))
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues',xticklabels=[0,1,2],yticklabels=[0,1
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix Heatmap')
plt.show()
```


Confusion Matrix Heatmap
True Label
Predicted Label
0
1
2
14
0
0
0
14
0
0
0
8
14
12
10
8
6
4
2
0