IDL TP Final `scikit-learn` Wine dataset testing

Un Premier Exemple

Les données

In [1]:
```
from sklearn import datasets
wine = datasets.load_wine()
type(wine)
```

Out[1]:
```
sklearn.utils._bunch.Bunch
```

In [2]:
```
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.

|details-start|
**References**
|details-split|

(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).

|details-end|
```

In [3]:
```
wine.feature_names
```

Out[3]:
```
['alcohol',
 'malic_acid',
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']
```

In [4]:
```
wine.target_names
```

Out[4]:
```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

In [5]:
```
import pandas as pd
```

In [6]:
```
wine_data_structure = {
        "Type": type(wine),
        "Keys": wine.keys(),
        "Feature names": wine.feature_names,
        "Target names": wine.target_names,
        "Number of samples": wine.data.shape[0],
        "Number of features": wine.data.shape[1],
        "Sample data point": wine.data[0],
        "Sample target": wine.target[0]
}
```

In [7]:
```
df_wine = pd.DataFrame(data=wine.data, columns=wine.feature_names)
df_wine['target'] = wine.target
```

In [8]:
```
wine_data_structure, df_wine.head()
```

Out[8]:
```
({'Type': sklearn.utils._bunch.Bunch,
  'Keys': dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'f
eature_names']),
  '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 names': array(['class_0', 'class_1', 'class_2'], dtype='<U7'),
  'Number of samples': 178,
  'Number of features': 13,
  'Sample data point': array([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]),
  'Sample target': 0},
    alcohol  malic_acid   ash  alcalinity_of_ash  magnesium  total_phenols
\
 0    14.23        1.71  2.43               15.6      127.0           2.80
 1    13.20        1.78  2.14               11.2      100.0           2.65
 2    13.16        2.36  2.67               18.6      101.0           2.80
 3    14.37        1.95  2.50               16.8      113.0           3.85
 4    13.24        2.59  2.87               21.0      118.0           2.80

    flavanoids  nonflavanoid_phenols  proanthocyanins  color_intensity   hu
e  \
 0        3.06                  0.28             2.29             5.64  1.0
4
 1        2.76                  0.26             1.28             4.38  1.0
5
 2        3.24                  0.30             2.81             5.68  1.0
3
 3        3.49                  0.24             2.18             7.80  0.8
6
 4        2.69                  0.39             1.82             4.32  1.0
4

    od280/od315_of_diluted_wines  proline  target
 0                          3.92   1065.0       0
 1                          3.40   1050.0       0
 2                          3.17   1185.0       0
 3                          3.45   1480.0       0
 4                          2.93    735.0       0  )
```

In [9]:
```python
import polars as pl
```

In [10]:
```python
df = pl.DataFrame(
        data=wine.data, schema=wine.feature_names).with_columns(
        target=pl.Series(wine.target)
)
df.head()
```

Out[10]: shape: (5, 14)

alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavano
f64	f64	f64	f64	f64	f64	
14.23	1.71	2.43	15.6	127.0	2.8	3
13.2	1.78	2.14	11.2	100.0	2.65	2
13.16	2.36	2.67	18.6	101.0	2.8	3
14.37	1.95	2.5	16.8	113.0	3.85	3
13.24	2.59	2.87	21.0	118.0	2.8	2

In [11]:
```python
X_wine, y_wine = wine.data, wine.target
```

In [12]:
```python
X_wine.shape
```

Out[12]:
```
(178, 13)
```

In [13]:
```python
y_wine
```

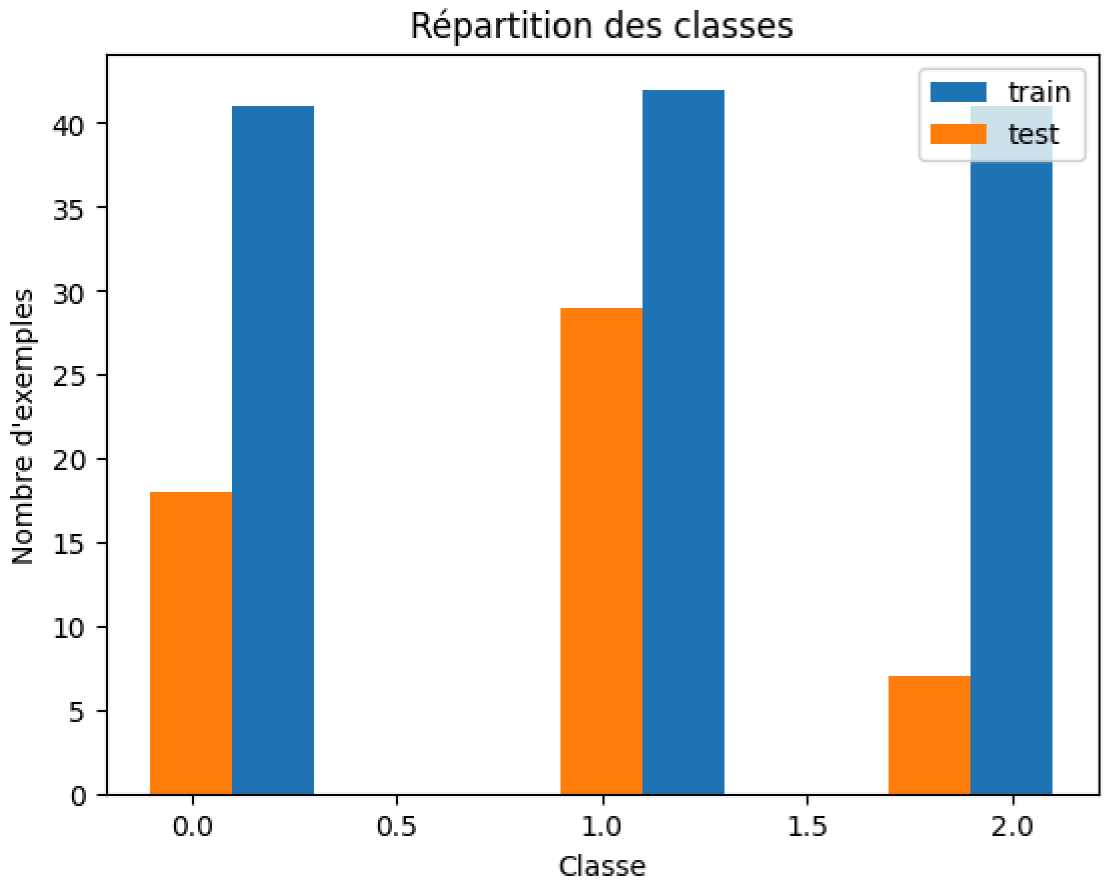
Out[13]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])
```

In [14]:
```python
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_wine, y_wine, test_siz
y_train
```

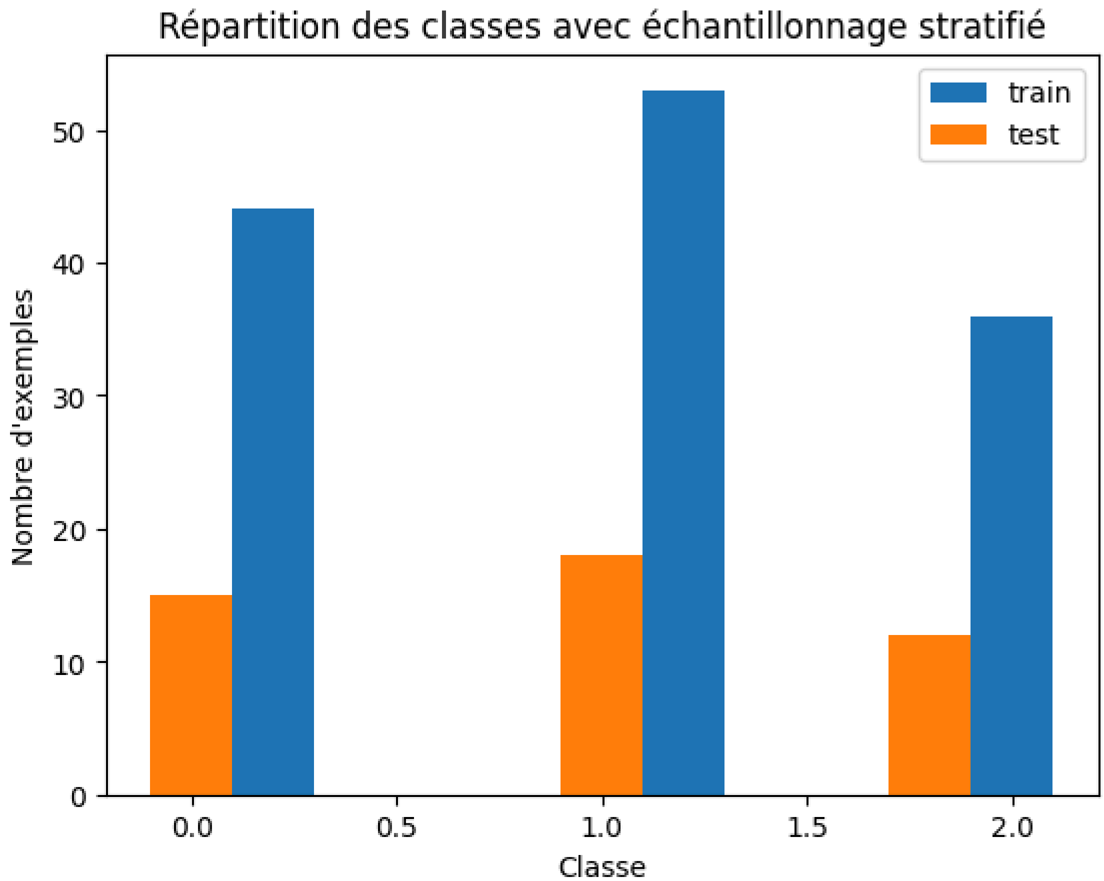
Out[14]:
```
array([1, 1, 2, 2, 0, 1, 2, 2, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 0, 1, 1, 2, 0, 1, 2, 1, 0, 0, 1, 2, 2, 2, 2, 1, 2, 1, 1, 0, 2,
       2, 1, 0, 2, 2, 2, 2, 2, 1, 0, 2, 1, 0, 1, 0, 0, 2, 1, 1, 1, 0, 2,
       2, 0, 0, 2, 0, 2, 0, 2, 2, 1, 0, 0, 2, 1, 2, 0, 1, 1, 0, 0, 1, 2,
       1, 2, 2, 0, 1, 1, 1, 2, 2, 1, 2, 0, 1, 2, 0, 1, 1, 2, 0, 2, 0, 2,
       2, 0, 0, 0, 0, 1, 0, 1, 0, 2, 2, 0, 0, 1])
```

In [15]:
```python
import matplotlib.pyplot as plt
%matplotlib inline

plt.hist(y_train, align="right", label="train")
plt.hist(y_test, align="left", label="test")
plt.legend()
```

```
plt.xlabel("Classe")
plt.ylabel("Nombre d'exemples")
plt.title("Répartition des classes")
plt.show()
```



In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X_wine, y_wine, test_siz
plt.hist(y_train, align="right", label="train")
plt.hist(y_test, align="left", label="test")
plt.legend()
plt.xlabel("Classe")
plt.ylabel("Nombre d'exemples")
plt.title("Répartition des classes avec échantillonnage stratifié")
plt.show()
```



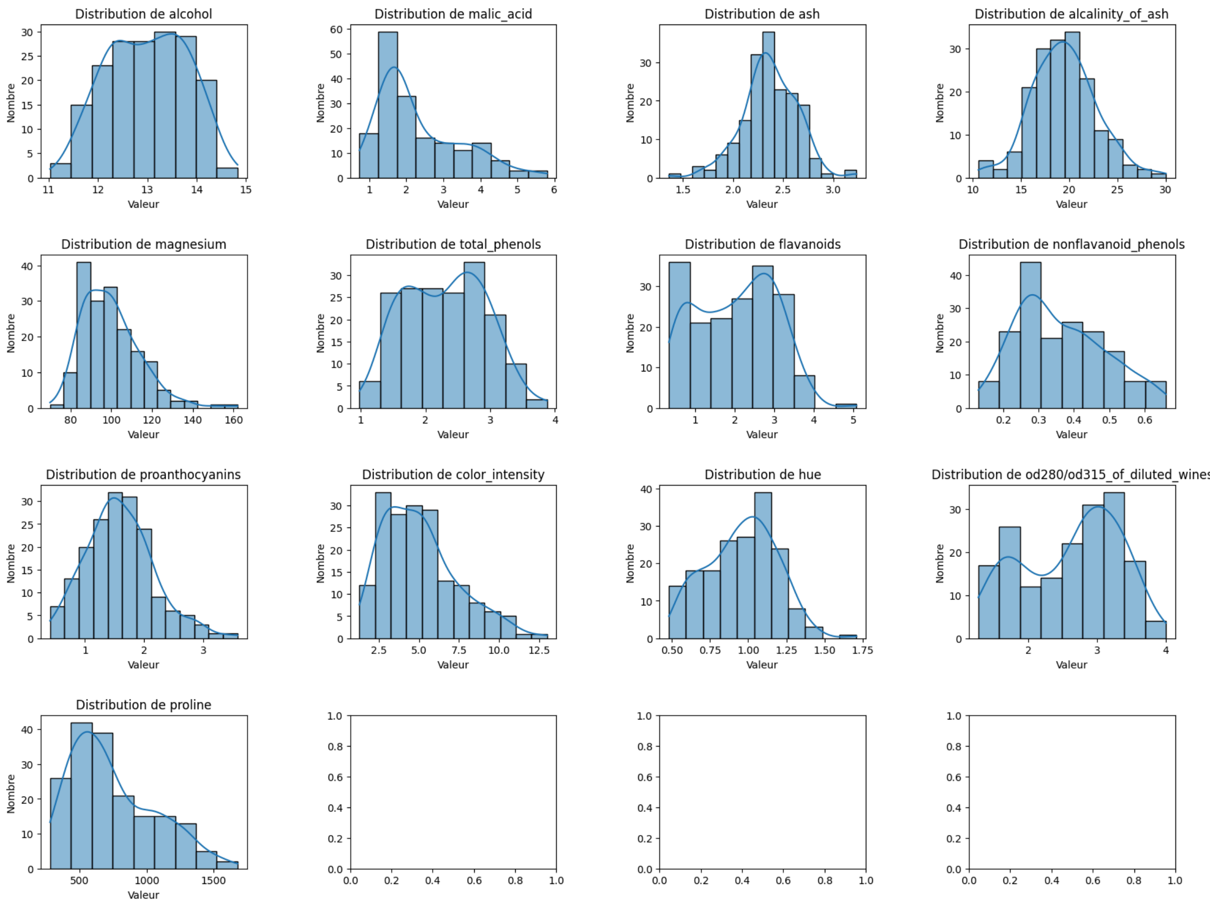
Visualisation de données

In [17]:
```python
import seaborn as sns
print("Description statistique de base des caractéristiques :")
print(df_wine.describe())
```

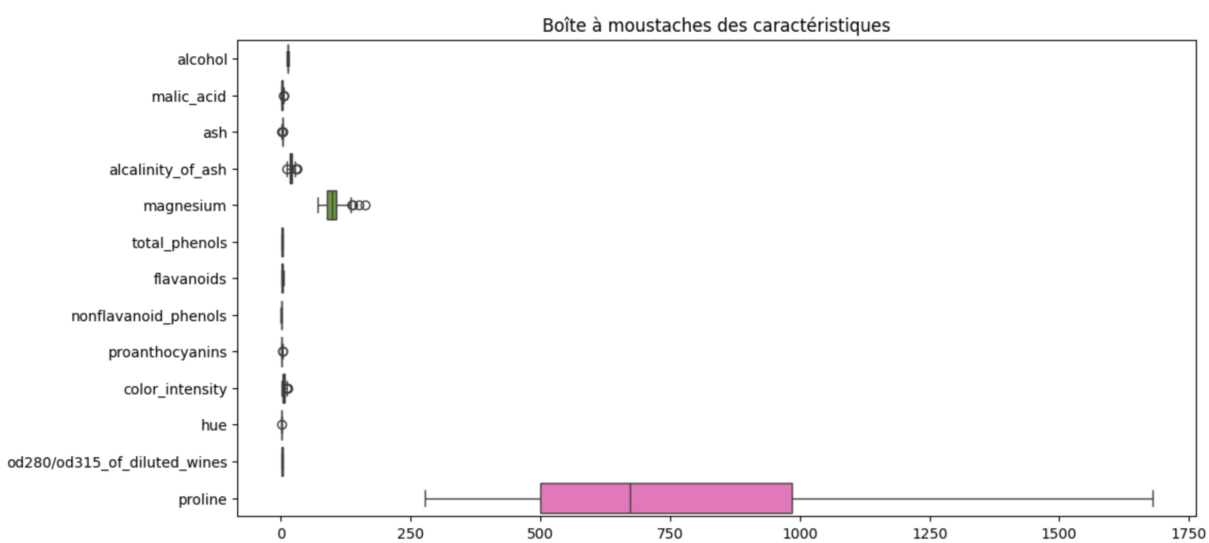
```
Description statistique de base des caractéristiques :
          alcohol  malic_acid         ash  alcalinity_of_ash   magnesium  \
count  178.000000  178.000000  178.000000         178.000000  178.000000   
mean    13.000618    2.336348    2.366517          19.494944   99.741573   
std      0.811827    1.117146    0.274344           3.339564   14.282484   
min     11.030000    0.740000    1.360000          10.600000   70.000000   
25%     12.362500    1.602500    2.210000          17.200000   88.000000   
50%     13.050000    1.865000    2.360000          19.500000   98.000000   
75%     13.677500    3.082500    2.557500          21.500000  107.000000   
max     14.830000    5.800000    3.230000          30.000000  162.000000   

       total_phenols  flavanoids  nonflavanoid_phenols  proanthocyanins  \
count     178.000000  178.000000            178.000000       178.000000   
mean        2.295112    2.029270              0.361854         1.590899   
std         0.625851    0.998859              0.124453         0.572359   
min         0.980000    0.340000              0.130000         0.410000   
25%         1.742500    1.205000              0.270000         1.250000   
50%         2.355000    2.135000              0.340000         1.555000   
75%         2.800000    2.875000              0.437500         1.950000   
max         3.880000    5.080000              0.660000         3.580000   

       color_intensity         hue  od280/od315_of_diluted_wines      prolin
e  \
count       178.000000  178.000000                    178.000000   178.00000
0   
mean          5.058090    0.957449                      2.611685   746.89325
8   
std           2.318286    0.228572                      0.709990   314.90747
4   
min           1.280000    0.480000                      1.270000   278.00000
0   
25%           3.220000    0.782500                      1.937500   500.50000
0   
50%           4.690000    0.965000                      2.780000   673.50000
0   
75%           6.200000    1.120000                      3.170000   985.00000
0   
max          13.000000    1.710000                      4.000000  1680.00000
0   

           target  
count  178.000000  
mean     0.938202  
std      0.775035  
min      0.000000  
25%      0.000000  
50%      1.000000  
75%      2.000000  
max      2.000000  
```

In [18]:
```python
fig, axs = plt.subplots(nrows=4, ncols=4, figsize=(20, 15))
fig.subplots_adjust(hspace=0.5, wspace=0.5)
axs = axs.flatten()

for i, col in enumerate(df_wine.columns[:-1]):
        sns.histplot(df_wine[col], kde=True, ax=axs[i])
```

```python
        axs[i].set_title(f'Distribution de {col}')
        axs[i].set_ylabel('Nombre')
        axs[i].set_xlabel('Valeur')
```



In [19]:
```python
plt.figure(figsize=(12, 6))
sns.boxplot(data=df_wine.drop(columns=['target']), orient='h')
plt.title('Boîte à moustaches des caractéristiques')
plt.show()
```



Données Standardisées

In [20]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

In [21]:
```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_wine.drop(columns=['target']))
```

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, df_wine['targe
```

In [23]:
```
print(f"Taille de l'ensemble d'entraînement : {X_train.shape}, Taille de l'e
```

```
Taille de l'ensemble d'entraînement : (124, 13), Taille de l'ensemble de tes
ts : (54, 13)
```

Entraînement

In [24]:
```
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report, confusion_matrix, accurac
```

In [25]:
```
clf = LinearSVC(dual=True)
clf.fit(X_train, y_train)
```

Out[25]:

```
▾ LinearSVC
LinearSVC(dual=True)
```

Évaluation

In [26]:
```
y_pred = clf.predict(X_test)
```

In [27]:
```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)
```

In [28]:
```
print(f"Taux d'exactitude du modèle : {accuracy}")
print("Matrice de confusion :")
print(conf_matrix)
print("Rapport de classification :")
print(class_report)
```

```
Taux d'exactitude du modèle : 0.9814814814814815
Matrice de confusion :
[[18  0  0]
 [ 1 20  0]
 [ 0  0 15]]
Rapport de classification :
              precision    recall  f1-score   support

           0       0.95      1.00      0.97        18
           1       1.00      0.95      0.98        21
           2       1.00      1.00      1.00        15

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

In [29]:
```python
clf.score(X_test, y_test)
```

Out[29]:
```
0.9814814814814815
```

Optimisation des hyperparamètres

In [30]:
```python
from sklearn.model_selection import GridSearchCV
```

In [31]:
```python
model = LinearSVC(dual=False, random_state=42)  # dual=False when n_samples
param_grid = {
        'C': [0.1, 1, 10, 100],
        'loss': ['hinge', 'squared_hinge']
}
```

In [32]:
```python
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy', verb
```

In [33]:
```python
def load_data() -> pd.DataFrame:
        """Load wine dataset and return data as a pandas DataFrame."""
        wine = datasets.load_wine()
        df_wine = pd.DataFrame(data=wine.data, columns=wine.feature_names)
        df_wine['target'] = wine.target.astype(str)
        return df_wine

def standardize_data(df: pd.DataFrame) -> tuple:
        """Standardize the data using the StandardScaler."""
        X = df.drop(columns=['target']).values
        y = df['target'].values
        scaler = StandardScaler()
        X_scaled = scaler.fit_transform(X)
        return X_scaled, y

def train_test_split_data(X: pd.DataFrame, y: pd.DataFrame, test_size=0.3, r
        """Split the data into training and test sets."""
        X_train, X_test, y_train, y_test = train_test_split(
                X, y, test_size=test_size, random_state=random_state, strati
        return X_train, X_test, y_train, y_test
```

```
df = load_data()
X, y = standardize_data(df)
X_train, X_test, y_train, y_test = train_test_split_data(X, y)
```

In [34]:
```
grid_search.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 8 candidates, totalling 40 fits
/home/viet/.local/lib/python3.10/site-packages/sklearn/model_selection/_validation.py:542: FitFailedWarning:
20 fits failed out of a total of 40.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
20 fits failed with the following error:
Traceback (most recent call last):
  File "/home/viet/.local/lib/python3.10/site-packages/sklearn/model_selection/_validation.py", line 890, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/home/viet/.local/lib/python3.10/site-packages/sklearn/base.py", line 1351, in wrapper
    return fit_method(estimator, *args, **kwargs)
  File "/home/viet/.local/lib/python3.10/site-packages/sklearn/svm/_classes.py", line 325, in fit
    self.coef_, self.intercept_, n_iter_ = _fit_liblinear(
  File "/home/viet/.local/lib/python3.10/site-packages/sklearn/svm/_base.py", line 1216, in _fit_liblinear
    solver_type = _get_liblinear_solver_type(multi_class, penalty, loss, dual)
  File "/home/viet/.local/lib/python3.10/site-packages/sklearn/svm/_base.py", line 1047, in _get_liblinear_solver_type
    raise ValueError(
ValueError: Unsupported set of arguments: The combination of penalty='l2' and loss='hinge' are not supported when dual=False, Parameters: penalty='l2', loss='hinge', dual=False

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/home/viet/.local/lib/python3.10/site-packages/sklearn/model_selection/_search.py:1051: UserWarning: One or more of the test scores are non-finite: [
nan 0.98366667        nan 0.97566667        nan 0.97566667
        nan 0.97566667]
  warnings.warn(
```

Out[34]:

▸ **GridSearchCV**

▸ **estimator: LinearSVC**

▸ LinearSVC

In [35]:
```
print("Best parameters:", grid_search.best_params_)
print("Best cross-validation score: {:.2f}".format(grid_search.best_score_))
```

```
Best parameters: {'C': 0.1, 'loss': 'squared_hinge'}
Best cross-validation score: 0.98
```

In [36]:
```
def evaluate_model(clf: LinearSVC, X_test: pd.DataFrame, y_test: pd.DataFram
        """Evaluate the trained model on the test data."""
        y_pred = clf.predict(X_test)
        accuracy = accuracy_score(y_test, y_pred)
        conf_matrix = confusion_matrix(y_test, y_pred)
        class_report = classification_report(y_test, y_pred)
        print(f"Taux d'exactitude du modèle : {accuracy}")
        print("Matrice de confusion :")
        print(conf_matrix)
        print("Rapport de classification :")
        print(class_report)
        return accuracy, conf_matrix, class_report

best_model = grid_search.best_estimator_
accuracy, conf_matrix, class_report = evaluate_model(best_model, X_test, y_t
```

```
Taux d'exactitude du modèle : 0.9814814814814815
Matrice de confusion :
[[18  0  0]
 [ 1 20  0]
 [ 0  0 15]]
Rapport de classification :
              precision    recall  f1-score   support

           0       0.95      1.00      0.97        18
           1       1.00      0.95      0.98        21
           2       1.00      1.00      1.00        15

    accuracy                           0.98        54
   macro avg       0.98      0.98      0.98        54
weighted avg       0.98      0.98      0.98        54
```

Validation croisée

In [37]:
```
from sklearn.model_selection import cross_validate, cross_val_score
```

In [38]:
```
def cross_validate_model(model, X, y, cv=5):
        """Perform cross-validation on a given model."""
        scores = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
        print(f"Cross-validation scores: {scores}")
        print(f"Average cross-validation score: {scores.mean():.3f}")
        return scores
```

In [39]:
```
cross_validate_model(best_model, X, y, cv=10)
```

```
Cross-validation scores: [1.         0.94444444 1.         0.94444444 1.
0.94444444
 1.         1.         1.         1.        ]
Average cross-validation score: 0.983
```

Out[39]:
```
array([1.        , 0.94444444, 1.        , 0.94444444, 1.        ,
       0.94444444, 1.        , 1.        , 1.        , 1.        ])
```

Classification de textes

In [40]:
```python
from sklearn.datasets import fetch_20newsgroups

categories = [
        "sci.crypt",
        "sci.electronics",
        "sci.med",
        "sci.space",
]

data_train = fetch_20newsgroups(
        subset="train",
        categories=categories,
        shuffle=True,
)

data_test = fetch_20newsgroups(
        subset="test",
        categories=categories,
        shuffle=True,
)
```

In [41]:
```python
print(len(data_train.data))
print(len(data_test.data))
```

```
2373
1579
```

In [42]:
```python
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer(stop_words="english")
X_train = vectorizer.fit_transform(data_train.data) # données de train vecto
y_train = data_train.target
X_train.shape
```

Out[42]:
```
(2373, 38377)
```

In [43]:
```python
print(X_train[0, :])
```

```
  (0, 6241)	5
  (0, 15189)	3
  (0, 10848)	2
  (0, 13811)	2
  (0, 33463)	1
  (0, 34197)	1
  (0, 10718)	1
  (0, 17565)	1
  (0, 29885)	1
  (0, 26255)	1
  (0, 11642)	1
  (0, 25772)	1
  (0, 36427)	1
  (0, 36264)	1
  (0, 13639)	1
  (0, 24869)	1
  (0, 22325)	1
  (0, 1445)	1
  (0, 6521)	1
  (0, 25423)	1
  (0, 14415)	1
  (0, 6636)	1
  (0, 24306)	1
  (0, 37875)	1
  (0, 38132)	1
  :	:
  (0, 21381)	1
  (0, 35071)	1
  (0, 37833)	1
  (0, 31496)	1
  (0, 27109)	1
  (0, 37263)	1
  (0, 29319)	1
  (0, 20482)	2
  (0, 29303)	1
  (0, 36278)	1
  (0, 21030)	1
  (0, 7811)	1
  (0, 19718)	1
  (0, 36275)	1
  (0, 28309)	1
  (0, 14829)	1
  (0, 23764)	1
  (0, 36101)	1
  (0, 28961)	1
  (0, 20458)	1
  (0, 34472)	1
  (0, 10579)	1
  (0, 25040)	1
  (0, 34126)	1
  (0, 20151)	1
```

In [44]:
```
X_test = vectorizer.transform(data_test.data)
y_test = data_test.target
```

In [45]:
```python
clf = LinearSVC(C=0.5)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.93      0.95       396
           1       0.85      0.93      0.89       393
           2       0.92      0.90      0.91       396
           3       0.98      0.93      0.96       394

    accuracy                           0.92      1579
   macro avg       0.93      0.92      0.92      1579
weighted avg       0.93      0.92      0.92      1579
```

```
/home/viet/.local/lib/python3.10/site-packages/sklearn/svm/_classes.py:31: F
utureWarning: The default value of `dual` will change from `True` to `'aut
o'` in 1.5. Set the value of `dual` explicitly to suppress the warning.
  warnings.warn(
```

In [46]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer(
        sublinear_tf=True,
        max_df=0.5,
        stop_words='english'
)
X_train = vectorizer.fit_transform(data_train.data) # données de train vecto
y_train = data_train.target
X_train.shape

X_test = vectorizer.transform(data_test.data)
y_test = data_test.target

clf = LinearSVC(C=0.5)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96       396
           1       0.88      0.96      0.92       393
           2       0.95      0.93      0.94       396
           3       0.98      0.95      0.97       394

    accuracy                           0.95      1579
   macro avg       0.95      0.95      0.95      1579
weighted avg       0.95      0.95      0.95      1579
```

```
/home/viet/.local/lib/python3.10/site-packages/sklearn/svm/_classes.py:31: F
utureWarning: The default value of `dual` will change from `True` to `'aut
o'` in 1.5. Set the value of `dual` explicitly to suppress the warning.
  warnings.warn(
```

In [ ]:

```

```