**from** sklearn.datasets **import** load\_iris

**import** pandas **as** pd

**import** numpy **as** np

**import** seaborn **as** sns

iris **=** load\_iris()

In [2]:

iris**.**keys()

Out[2]:

dict\_keys(['data', 'target', 'frame', 'target\_names', 'DESCR', 'feature\_names', 'filename', 'data\_module'])

In [3]:

data **=** pd**.**DataFrame(iris**.**data, columns **=** iris**.**feature\_names)

data**.**head()

Out[3]:

|  | **sepal length (cm)** | **sepal width (cm)** | **petal length (cm)** | **petal width (cm)** |
| --- | --- | --- | --- | --- |
| **0** | 5.1 | 3.5 | 1.4 | 0.2 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 |

In [4]:

data['label'] **=** iris**.**target

In [5]:

set(iris**.**target), iris**.**target\_names

Out[5]:

({0, 1, 2}, array(['setosa', 'versicolor', 'virginica'], dtype='<U10'))

In [6]:

data**.**sample(5)

Out[6]:

|  | **sepal length (cm)** | **sepal width (cm)** | **petal length (cm)** | **petal width (cm)** | **label** |
| --- | --- | --- | --- | --- | --- |
| **122** | 7.7 | 2.8 | 6.7 | 2.0 | 2 |
| **46** | 5.1 | 3.8 | 1.6 | 0.2 | 0 |
| **35** | 5.0 | 3.2 | 1.2 | 0.2 | 0 |
| **58** | 6.6 | 2.9 | 4.6 | 1.3 | 1 |
| **132** | 6.4 | 2.8 | 5.6 | 2.2 | 2 |

In [7]:

data**.**info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 150 entries, 0 to 149

Data columns (total 5 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 sepal length (cm) 150 non-null float64

1 sepal width (cm) 150 non-null float64

2 petal length (cm) 150 non-null float64

3 petal width (cm) 150 non-null float64

4 label 150 non-null int32

dtypes: float64(4), int32(1)

memory usage: 5.4 KB

In [8]:

setosa **=** data[data['label']**==** 0]

In [9]:

setosa**.**sample(5)

Out[9]:

|  | **sepal length (cm)** | **sepal width (cm)** | **petal length (cm)** | **petal width (cm)** | **label** |
| --- | --- | --- | --- | --- | --- |
| **38** | 4.4 | 3.0 | 1.3 | 0.2 | 0 |
| **31** | 5.4 | 3.4 | 1.5 | 0.4 | 0 |
| **17** | 5.1 | 3.5 | 1.4 | 0.3 | 0 |
| **40** | 5.0 | 3.5 | 1.3 | 0.3 | 0 |
| **35** | 5.0 | 3.2 | 1.2 | 0.2 | 0 |

In [10]:

plot\_setosa **=** setosa**.**drop('label', axis**=**1)

**import** matplotlib.pyplot **as** plt

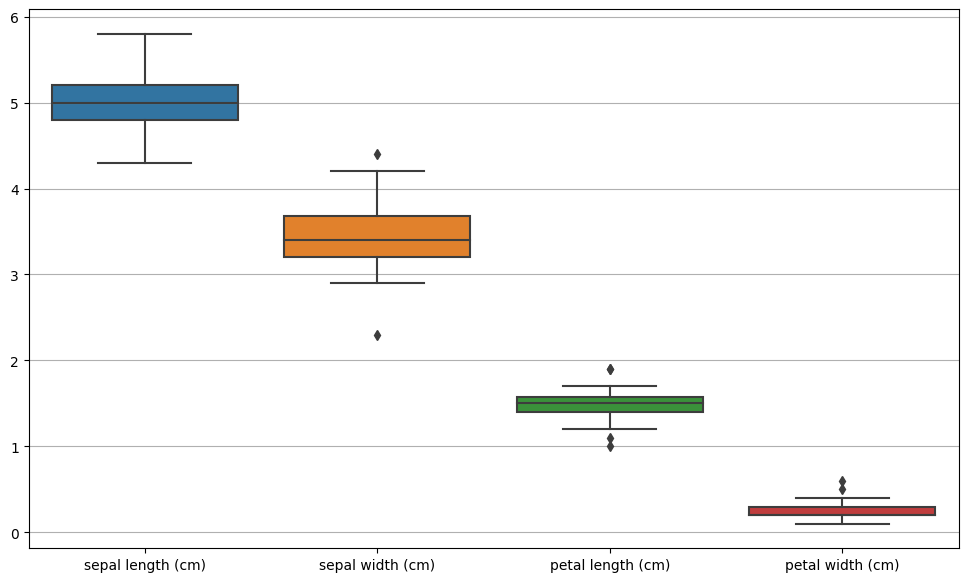
plt**.**figure(figsize **=** (12,7))

plt**.**grid()

sns**.**boxplot(data **=** plot\_setosa)

Out[10]:

<AxesSubplot:>

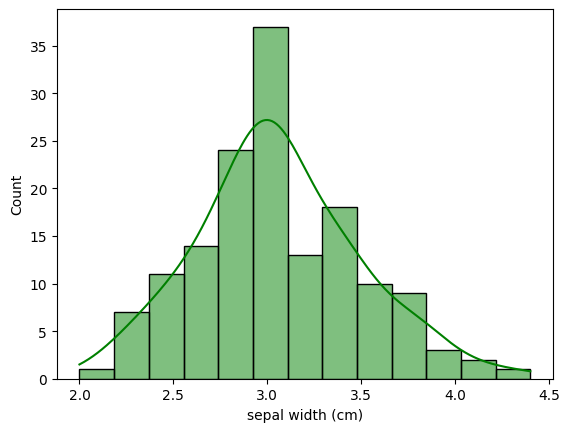


In [11]:

**from** matplotlib.cbook **import** boxplot\_stats

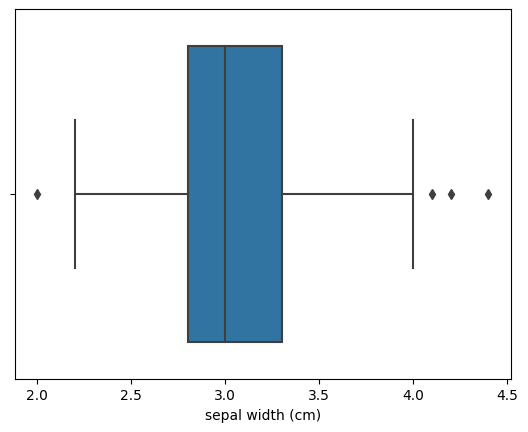
In [12]:

sns**.**histplot(data["sepal width (cm)"], kde**=** **True**, color**=** 'green');



In [13]:

sns**.**boxplot(data**=**data, x**=**'sepal width (cm)');



In [14]:

stats **=** boxplot\_stats(data['sepal width (cm)'])

In [15]:

stats

Out[15]:

[{'mean': 3.0573333333333337,

'iqr': 0.5,

'cilo': 2.9359050183971735,

'cihi': 3.0640949816028265,

'whishi': 4.0,

'whislo': 2.2,

'fliers': array([2. , 4.4, 4.1, 4.2]),

'q1': 2.8,

'med': 3.0,

'q3': 3.3}]

In [16]:

**def** detect\_outliers\_iqr(data):

Q1 **=** data**.**quantile(0.25)

Q3 **=** data**.**quantile(0.75)

IQR **=** Q3 **-** Q1

print(Q1,Q3,IQR)

lower\_bound **=** Q1 **-** 1.5 **\*** IQR

upper\_bound **=** Q3 **+** 1.5 **\*** IQR

outliers **=** data[(data **<** lower\_bound) **|** (data **>** upper\_bound)]

**return** outliers

In [17]:

outliers **=** detect\_outliers\_iqr(data['sepal width (cm)'])

2.8 3.3 0.5

In [18]:

data\_wo\_outliers **=** data[**~**data['sepal width (cm)']**.**isin(outliers)]

In [19]:

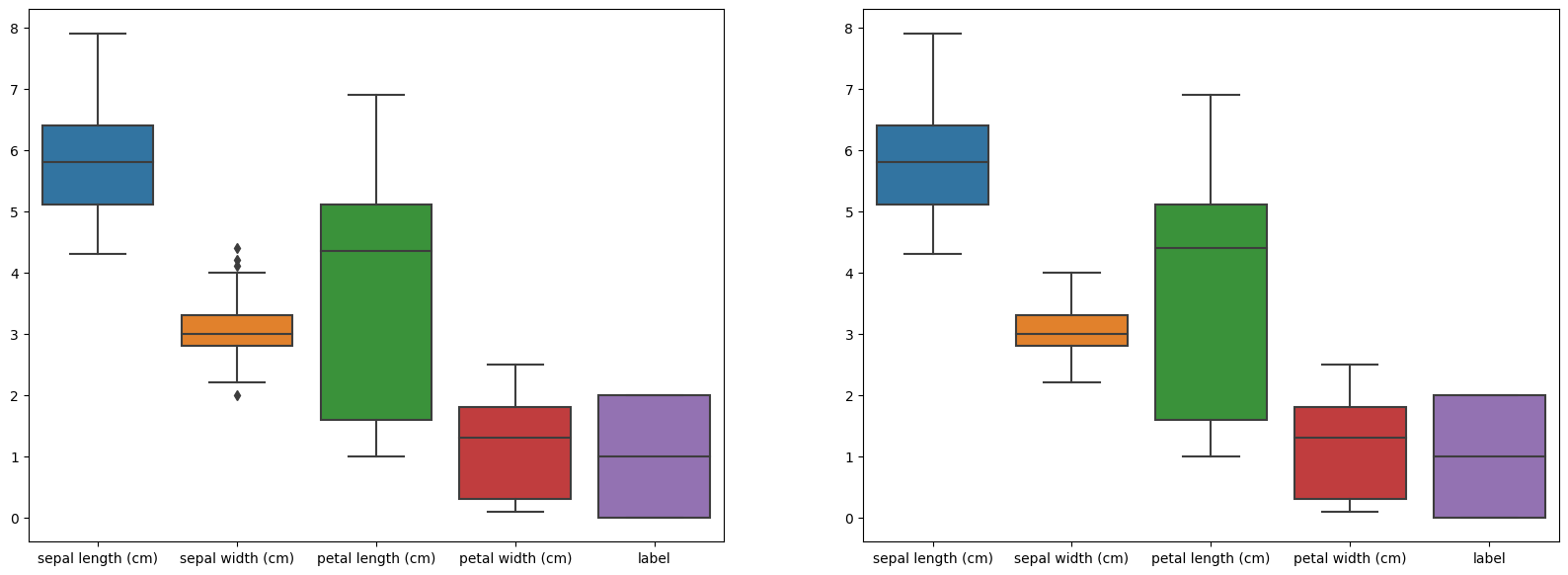
fig, axs **=** plt**.**subplots(1, 2, figsize**=**(20, 7))

sns**.**boxplot(ax**=**axs[0], data**=**data)

sns**.**boxplot(ax**=**axs[1], data**=**data\_wo\_outliers)

Out[19]:

<AxesSubplot:>



All statistical methods custom functions

In [24]:

**import** numpy **as** np

**def** calculate\_mean(dataframe):

means **=** {}

**for** column **in** dataframe**.**columns:

column\_values **=** dataframe[column]**.**values

means[column] **=** sum(column\_values) **/** len(column\_values)

**return** means

**def** calculate\_median(dataframe):

medians **=** {}

**for** column **in** dataframe**.**columns:

column\_values **=** dataframe[column]**.**values

sorted\_values **=** sorted(column\_values)

length **=** len(sorted\_values)

**if** length **%** 2 **==** 0:

median **=** (sorted\_values[length **//** 2 **-** 1] **+** sorted\_values[length **//** 2]) **/** 2

**else**:

median **=** sorted\_values[length **//** 2]

medians[column] **=** median

**return** medians

**def** calculate\_mode(dataframe):

modes **=** {}

**for** column **in** dataframe**.**columns:

column\_values **=** dataframe[column]**.**values

value\_counts **=** dataframe[column]**.**value\_counts()

max\_count **=** value\_counts**.**max()

mode **=** value\_counts[value\_counts **==** max\_count]**.**index**.**tolist()

modes[column] **=** mode

**return** modes

**def** calculate\_standard\_deviation(dataframe):

standard\_deviations **=** {}

**for** column **in** dataframe**.**columns:

column\_values **=** dataframe[column]**.**values

mean **=** sum(column\_values) **/** len(column\_values)

squared\_diff\_sum **=** sum((x **-** mean) **\*\*** 2 **for** x **in** column\_values)

variance **=** squared\_diff\_sum **/** len(column\_values)

standard\_deviations[column] **=** np**.**sqrt(variance)

**return** standard\_deviations

**def** calculate\_variance(dataframe):

variances **=** {}

**for** column **in** dataframe**.**columns:

column\_values **=** dataframe[column]**.**values

mean **=** sum(column\_values) **/** len(column\_values)

squared\_diff\_sum **=** sum((x **-** mean) **\*\*** 2 **for** x **in** column\_values)

variances[column] **=** squared\_diff\_sum **/** len(column\_values)

**return** variances

Mean

using python method

In [25]:

setosa**.**mean()

Out[25]:

sepal length (cm) 5.006

sepal width (cm) 3.428

petal length (cm) 1.462

petal width (cm) 0.246

label 0.000

dtype: float64

Using custom functions

In [26]:

calculate\_mean(setosa)

Out[26]:

{'sepal length (cm)': 5.005999999999999,

'sepal width (cm)': 3.428000000000001,

'petal length (cm)': 1.4620000000000002,

'petal width (cm)': 0.2459999999999999,

'label': 0.0}

Median

using python methods

In [28]:

setosa**.**median()

Out[28]:

sepal length (cm) 5.0

sepal width (cm) 3.4

petal length (cm) 1.5

petal width (cm) 0.2

label 0.0

dtype: float64

In [29]:

calculate\_median(setosa)

Out[29]:

{'sepal length (cm)': 5.0,

'sepal width (cm)': 3.4,

'petal length (cm)': 1.5,

'petal width (cm)': 0.2,

'label': 0.0}

Mode

Using python methods

In [52]:

setosa**.**drop('label', axis**=**1)**.**mode()

Out[52]:

|  | **sepal length (cm)** | **sepal width (cm)** | **petal length (cm)** | **petal width (cm)** |
| --- | --- | --- | --- | --- |
| **0** | 5.0 | 3.4 | 1.4 | 0.2 |
| **1** | 5.1 | NaN | 1.5 | NaN |

Using custom function

In [53]:

calculate\_mode(setosa)

Out[53]:

{'sepal length (cm)': 5.1,

'sepal width (cm)': 3.4,

'petal length (cm)': 1.5,

'petal width (cm)': 0.2,

'label': 0}

In [56]:

*#check*

setosa['sepal length (cm)']**.**value\_counts()

Out[56]:

5.1 8

5.0 8

5.4 5

4.8 5

4.9 4

4.6 4

4.4 3

5.2 3

4.7 2

5.7 2

5.5 2

4.3 1

5.8 1

4.5 1

5.3 1

Name: sepal length (cm), dtype: int64