importation des modules basiques

(aprés leurs installation avec cmd) sinon erreur: ModuleNotFoundError: No module named '----'

oub1 anaconda va faciliter la tâche

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import os
import matplotlib.pyplot as plt
```

téléchargement du dataset

In [2]:

```
# methode 1 : on peut charger directement dataset du biblio sns puisqu'elle existe déjà
data=sns.load_dataset("iris")
data
```

Out[2]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica

	sepal_length	sepal_width	petal_length	petal_width	species
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

In [3]:

```
#methode2: à partir du fichier télechargé
df=pd.read_table('C:/iris.csv')
```

In [4]:

```
df
```

Out[4]:

	sepal.length,"sepal.width","petal.length","petal.width","variety"
0	5.1,3.5,1.4,.2,"Setosa"
1	4.9,3,1.4,.2,"Setosa"
2	4.7,3.2,1.3,.2,"Setosa"
3	4.6,3.1,1.5,.2,"Setosa"
4	5,3.6,1.4,.2,"Setosa"
...	...
145	6.7,3,5.2,2.3,"Virginica"
146	6.3,2.5,5,1.9,"Virginica"
147	6.5,3,5.2,2,"Virginica"
148	6.2,3.4,5.4,2.3,"Virginica"
149	5.9,3,5.1,1.8,"Virginica"

150 rows × 1 columns

In [5]:

```
df=pd.read_table('C:/iris.csv',sep=',')
```

In [6]:
```
df
```

Out[6]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Virginica
146	6.3	2.5	5.0	1.9	Virginica
147	6.5	3.0	5.2	2.0	Virginica
148	6.2	3.4	5.4	2.3	Virginica
149	5.9	3.0	5.1	1.8	Virginica

150 rows × 5 columns

In [7]:
```
# les statistiques
df.describe()
```

Out[7]:

	sepal.length	sepal.width	petal.length	petal.width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000

	sepal.length	sepal.width	petal.length	petal.width
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [8]:

```
# info sur les types des données
df.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  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   variety       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

prétraitement

In [9]:

```
# le nb des exemplaires de chaque classe

   # notre dataset contient la colone "classe"
   #  =>  data est dite "supervisée"

df['variety'].value_counts()
```

Out[9]:

```
Virginica     50
Versicolor    50
Setosa        50
Name: variety, dtype: int64
```

In [10]:

```
# on remarque: pas besion de normaliser la data  puisqu'elle  est déjà normalisée ( f=N est la m pr chaue classe)
```

In [11]:
```
# des valeur nulles ?
df. isnull().sum()
 # on peut raisonner sur tt les attributs  (cad sans utiliser .sum ())
```

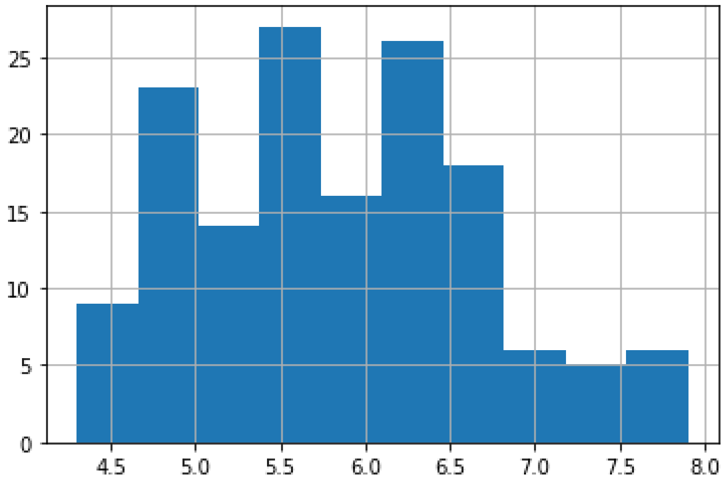
Out[11]:
```
sepal.length    0
sepal.width     0
petal.length    0
petal.width     0
variety         0
dtype: int64
```

visualisation

In [12]:
```
#sous forme d'histogramme
```

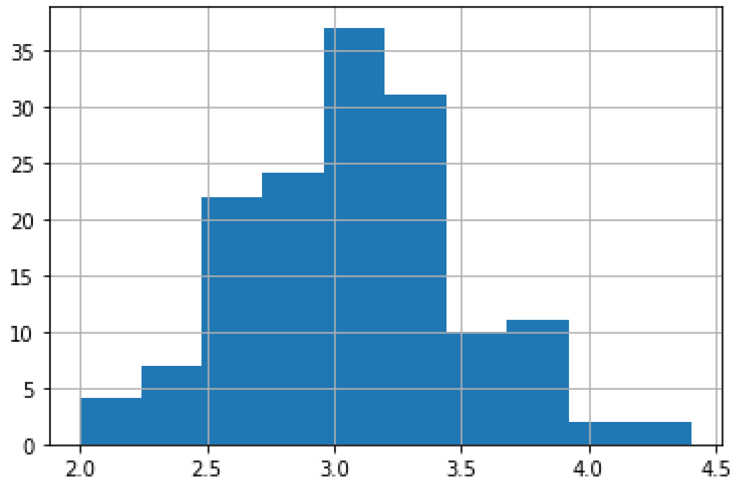
In [13]:
```
df['sepal.length'].hist()
```

Out[13]: <AxesSubplot:>



In [14]:
```
df['sepal.width'].hist()
```

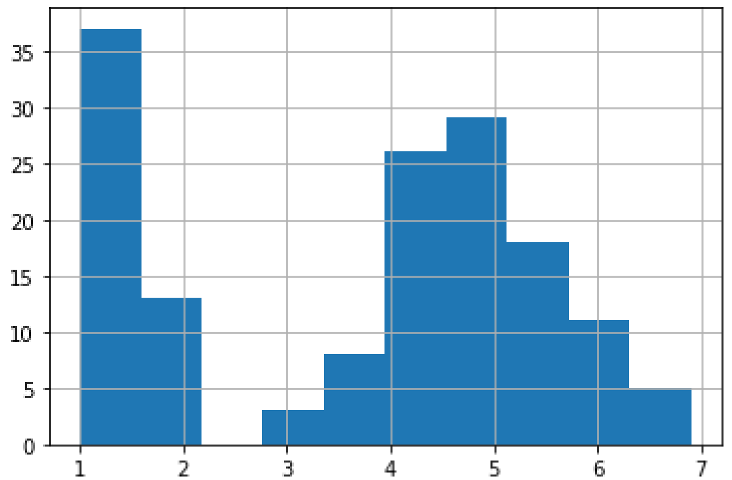
Out[14]: <AxesSubplot:>



In [15]:

```
df['petal.length'].hist()
```

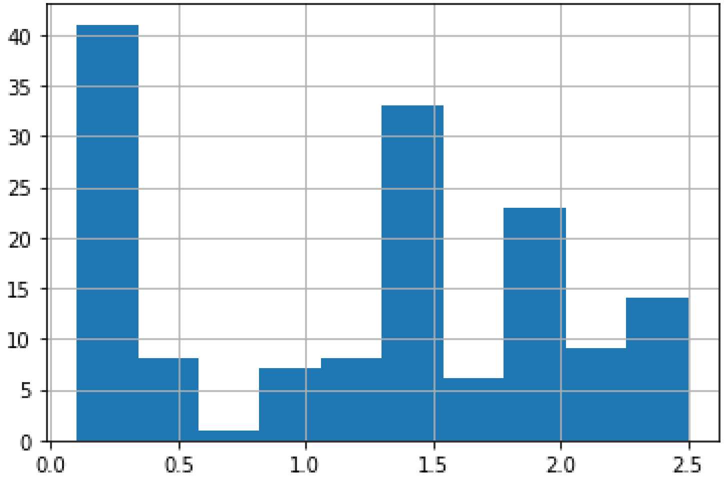
Out[15]: <AxesSubplot:>



In [16]:

```
df['petal.width'].hist()
```

Out[16]: <AxesSubplot:>



In [17]:

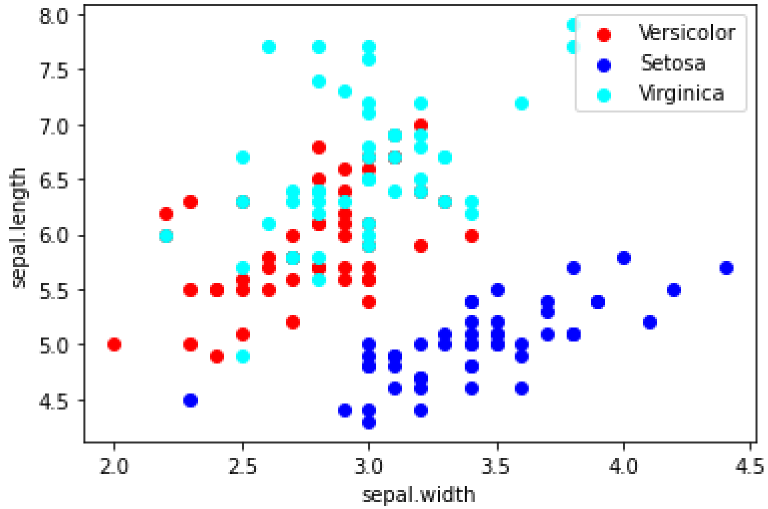
```
#sous forme de : nuage de points
```

In [18]:

```
colors=['red','blue','aqua']
variety=['Versicolor','Setosa','Virginica' ]
```

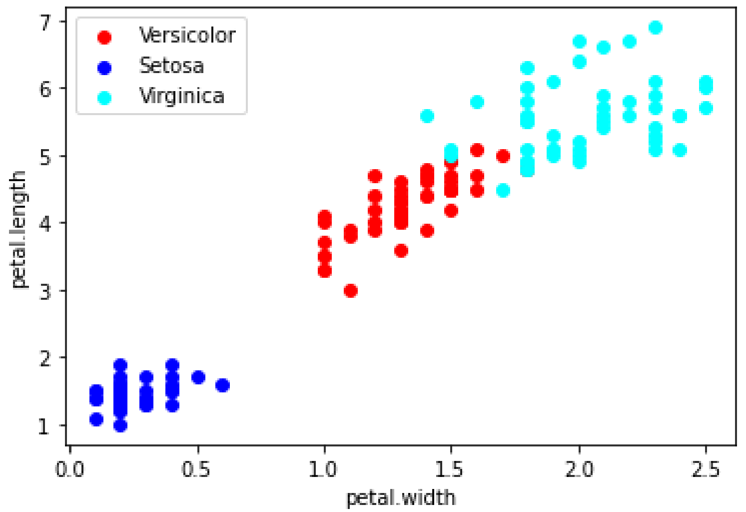
In [19]:

```
for i in range(3):
    x=df[df['variety']==variety[i]]
    plt.scatter(x['sepal.width'],x['sepal.length'],c=colors[i],label=variety[i])
    plt.xlabel("sepal.width")
    plt.ylabel("sepal.length")
    plt.legend()
```



In [20]:

```
for i in range(3):
    x=df[df['variety']==variety[i]]
    plt.scatter(x['petal.width'],x['petal.length'],c=colors[i],label=variety[i])
    plt.xlabel("petal.width")
    plt.ylabel("petal.length")
    plt.legend()
```



corrélation

In [21]:

```
# matrice de corrélation :
#un tab qui montre -1<=coeff de corrélation<=1

# chaq c du tab montre une corrélation entre 2 var
   #si jamais 2 var on une corrélation élvée
   #=>on n'églige l'1 des 2 var
df.corr()
```

Out[21]:

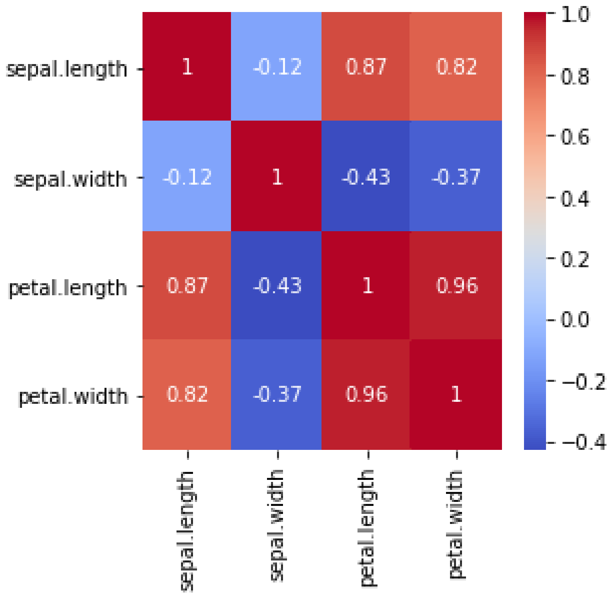
	sepal.length	sepal.width	petal.length	petal.width
sepal.length	1.000000	-0.117570	0.871754	0.817941
sepal.width	-0.117570	1.000000	-0.428440	-0.366126
petal.length	0.871754	-0.428440	1.000000	0.962865
petal.width	0.817941	-0.366126	0.962865	1.000000

In [22]:

```
#heat map (puisque les couleurs sont mieux visibles que les nombres)
corr=df.corr()
fig,ax=plt.subplots(figsize=(4,4)) # pr ajuster la taille
sns.heatmap(corr,annot=True,ax=ax,cmap="coolwarm") #annot=True pr afficher les val du matrice #on ajoute l'attribut c
```

Out[22]:

```
<AxesSubplot:>
```



In [23]:
```
# on a seulement 4  paramètres c'est pas la peines de minimiser le nbre
```

codage string-->int

cette etape va me faciliter la tâche avec les modèles

In [24]:
```
#Encode target labels with value between 0 and n_classes-1

# Setosa------->0
# Versicolor--->1
# Virginica---->2
```

In [25]:
```
from sklearn.preprocessing import LabelEncoder
le= LabelEncoder()
```

In [26]:
```
df['variety']= le.fit_transform(df['variety'])
```

```
df.head() # pr la visualisation
```

Out[26]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [27]:

```
df
```

Out[27]:

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	2
146	6.3	2.5	5.0	1.9	2
147	6.5	3.0	5.2	2.0	2
148	6.2	3.4	5.4	2.3	2
149	5.9	3.0	5.1	1.8	2

150 rows × 5 columns

les modèles

In [28]:

```
# division du dataset en 2 :je vais choisir  test_size =0.3 donc 70% 4 training 30% 4 testing
from sklearn.model_selection import train_test_split

x= df.drop(columns=['variety'])
y= df['variety']

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)

print("la base traitement est de la forme : ",x_train.shape)
print("la base test est de la forme : ",x_test.shape)
```

```
la base traitement est de la forme :  (105, 4)
la base test est de la forme :  (45, 4)
```

In [29]:

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
```

In [30]:

```
model.fit(x_train,y_train)
```

Out[30]:

```
LogisticRegression()
```

In [31]:

```
predictions=model.predict(x_test)
print(predictions)
print(y_test)
```

```
[0 2 1 2 2 0 2 1 1 2 0 1 0 2 1 2 0 2 2 0 1 1 1 1 1 0 0 2 0 2 0 0 0 2 0 2 1
 1 2 2 0 1 1 2 1]
49     0
144    2
61     1
137    2
133    2
17     0
116    2
54     1
58     1
101    2
```

```
30     0
65     1
29     0
107    2
84     1
145    2
37     0
118    2
120    2
4      0
63     1
96     1
94     1
66     1
81     1
48     0
36     0
105    2
21     0
138    2
25     0
32     0
13     0
122    2
14     0
103    2
73     1
59     1
142    2
100    2
24     0
75     1
67     1
83     1
71     1
Name: variety, dtype: int32
```

In [32]:

```
#from sklearn.metric import classification_report,accurancy_score
#print(classification_report(y_test,predictions))
#print(accurancy_score(y_test,predictions))
```

In [33]:

```
print("l'occurance du model LogisticRegressionest est de : ",model.score(x_test,y_test)*100)
```

```
l'occurance du model LogisticRegressionest est de :  97.77777777777777
```

In [38]:

```python
#from sklearn.neighbors import KNeighbhorsClassifier
#model=KNeighbhorsClassifier()


# ImportError: cannot import name 'KNeighbhorsClassifier' from 'sklearn.neighbors' (c:\python\python3.9.1\lib\site-pa
```

In [39]:

```python
from sklearn import tree

model = tree.DecisionTreeClassifier()
```

In [40]:

```python
model.fit(x_train,y_train)
```

Out[40]:

```
DecisionTreeClassifier()
```

In [41]:

```python
print("l'occurance du model DecisionTree est de : ",model.score(x_test,y_test)*100)
```

```
l'occurance du model DecisionTree est de :  95.55555555555556
```

In [ ]:

In [ ]:

In [1]:
```python
from sklearn import datasets

iris = datasets.load_iris()
iris
```

Out[1]:
```
{'data': array([[5.1, 3.5, 1.4, 0.2],
        [4.9, 3. , 1.4, 0.2],
        [4.7, 3.2, 1.3, 0.2],
        [4.6, 3.1, 1.5, 0.2],
        [5. , 3.6, 1.4, 0.2],
        [5.4, 3.9, 1.7, 0.4],
        [4.6, 3.4, 1.4, 0.3],
        [5. , 3.4, 1.5, 0.2],
        [4.4, 2.9, 1.4, 0.2],
        [4.9, 3.1, 1.5, 0.1],
        [5.4, 3.7, 1.5, 0.2],
        [4.8, 3.4, 1.6, 0.2],
        [4.8, 3. , 1.4, 0.1],
        [4.3, 3. , 1.1, 0.1],
        [5.8, 4. , 1.2, 0.2],
        [5.7, 4.4, 1.5, 0.4],
        [5.4, 3.9, 1.3, 0.4],
        [5.1, 3.5, 1.4, 0.3],
        [5.7, 3.8, 1.7, 0.3],
        [5.1, 3.8, 1.5, 0.3],
        [5.4, 3.4, 1.7, 0.2],
        [5.1, 3.7, 1.5, 0.4],
        [4.6, 3.6, 1. , 0.2],
        [5.1, 3.3, 1.7, 0.5],
        [4.8, 3.4, 1.9, 0.2],
        [5. , 3. , 1.6, 0.2],
        [5. , 3.4, 1.6, 0.4],
        [5.2, 3.5, 1.5, 0.2],
        [5.2, 3.4, 1.4, 0.2],
        [4.7, 3.2, 1.6, 0.2],
        [4.8, 3.1, 1.6, 0.2],
```

```
[5.4, 3.4, 1.5, 0.4],
[5.2, 4.1, 1.5, 0.1],
[5.5, 4.2, 1.4, 0.2],
[4.9, 3.1, 1.5, 0.2],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.6, 1.4, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
```

```
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
```

```
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
[6.7, 3.3, 5.7, 2.1],
[7.2, 3.2, 6. , 1.8],
[6.2, 2.8, 4.8, 1.8],
[6.1, 3. , 4.9, 1.8],
[6.4, 2.8, 5.6, 2.1],
[7.2, 3. , 5.8, 1.6],
[7.4, 2.8, 6.1, 1.9],
[7.9, 3.8, 6.4, 2. ],
[6.4, 2.8, 5.6, 2.2],
[6.3, 2.8, 5.1, 1.5],
[6.1, 2.6, 5.6, 1.4],
[7.7, 3. , 6.1, 2.3],
[6.3, 3.4, 5.6, 2.4],
[6.4, 3.1, 5.5, 1.8],
[6. , 3. , 4.8, 1.8],
[6.9, 3.1, 5.4, 2.1],
[6.7, 3.1, 5.6, 2.4],
[6.9, 3.1, 5.1, 2.3],
[5.8, 2.7, 5.1, 1.9],
[6.8, 3.2, 5.9, 2.3],
[6.7, 3.3, 5.7, 2.5],
[6.7, 3. , 5.2, 2.3],
[6.3, 2.5, 5. , 1.9],
[6.5, 3. , 5.2, 2. ],
```

```
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]),
 '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, 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]),
 'target_names': array(['setosa', 'versicolor', 'virginica'], dtype='<U
10'),
 'DESCR': '.. _iris_dataset:\n\nIris plants dataset\n------------------
--\n\n**Data Set Characteristics:**\n\n    :Number of Instances: 150 (5
0 in each of three classes)\n    :Number of Attributes: 4 numeric, pred
ictive attributes and the class\n    :Attribute Information:\n        -
sepal length in cm\n        - sepal width in cm\n        - petal length
in cm\n        - petal width in cm\n        - class:\n                -
Iris-Setosa\n                - Iris-Versicolour\n                - Iris
-Virginica\n                \n    :Summary Statistics:\n\n    =========
===== ==== ==== ======= ===== ====================\n                
Min  Max   Mean    SD   Class Correlation\n    ============== ==== ====
======= ===== ====================\n    sepal length:   4.3  7.9   5.84
   0.83    0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194
\n    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n    pe
tal width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n    ==========
==== ==== ==== ======= ===== ====================\n\n    :Missing Attri
bute Values: None\n    :Class Distribution: 33.3% for each of 3 classe
s.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%P
LU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris database,
first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s pap
er. Note that it\'s the same as in R, but not as in the UCI\nMachine Le
arning Repository, which has two wrong data points.\n\nThis is perhaps
the best known database to be found in the\npattern recognition literat
```

```
ure.  Fisher\'s paper is a classic in the field and\nis referenced freq
uently to this day.  (See Duda & Hart, for example.)  The\ndata set con
tains 3 classes of 50 instances each, where each class refers to a\ntyp
e of iris plant.  One class is linearly separable from the other 2; the
\nlatter are NOT linearly separable from each other.\n\n.. topic:: Refe
rences\n\n   - Fisher, R.A. "The use of multiple measurements in taxono
mic problems"\n     Annual Eugenics, 7, Part II, 179-188 (1936); also i
n "Contributions to\n     Mathematical Statistics" (John Wiley, NY, 195
0).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Sc
ene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.
See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the Neighborh
ood: A New System\n     Structure and Classification Rule for Recogniti
on in Partially Exposed\n     Environments".  IEEE Transactions on Patt
ern Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-7
1.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE
Transactions\n     on Information Theory, May 1972, 431-433.\n   - See
also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n
  conceptual clustering system finds 3 classes in the data.\n   - Many,
many more ...',
 'feature_names': ['sepal length (cm)',
  'sepal width (cm)',
  'petal length (cm)',
  'petal width (cm)'],
 'filename': 'C:\\ProgramData\\Anaconda3\\lib\\site-packages\\sklearn
\\datasets\\data\\iris.csv'}
```

Lire LE DataSet IRIS

In [2]:
```
data = iris.data
data
```

Out[2]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
```

```
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
       [5.1, 3.8, 1.5, 0.3],
       [5.4, 3.4, 1.7, 0.2],
       [5.1, 3.7, 1.5, 0.4],
       [4.6, 3.6, 1. , 0.2],
       [5.1, 3.3, 1.7, 0.5],
       [4.8, 3.4, 1.9, 0.2],
       [5. , 3. , 1.6, 0.2],
       [5. , 3.4, 1.6, 0.4],
       [5.2, 3.5, 1.5, 0.2],
       [5.2, 3.4, 1.4, 0.2],
       [4.7, 3.2, 1.6, 0.2],
       [4.8, 3.1, 1.6, 0.2],
       [5.4, 3.4, 1.5, 0.4],
       [5.2, 4.1, 1.5, 0.1],
       [5.5, 4.2, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.2],
       [5. , 3.2, 1.2, 0.2],
       [5.5, 3.5, 1.3, 0.2],
       [4.9, 3.6, 1.4, 0.1],
       [4.4, 3. , 1.3, 0.2],
       [5.1, 3.4, 1.5, 0.2],
       [5. , 3.5, 1.3, 0.3],
       [4.5, 2.3, 1.3, 0.3],
       [4.4, 3.2, 1.3, 0.2],
       [5. , 3.5, 1.6, 0.6],
       [5.1, 3.8, 1.9, 0.4],
       [4.8, 3. , 1.4, 0.3],
```

```
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
```

```
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
[5.7, 3. , 4.2, 1.2],
[5.7, 2.9, 4.2, 1.3],
[6.2, 2.9, 4.3, 1.3],
[5.1, 2.5, 3. , 1.1],
[5.7, 2.8, 4.1, 1.3],
[6.3, 3.3, 6. , 2.5],
[5.8, 2.7, 5.1, 1.9],
[7.1, 3. , 5.9, 2.1],
[6.3, 2.9, 5.6, 1.8],
[6.5, 3. , 5.8, 2.2],
[7.6, 3. , 6.6, 2.1],
[4.9, 2.5, 4.5, 1.7],
[7.3, 2.9, 6.3, 1.8],
[6.7, 2.5, 5.8, 1.8],
[7.2, 3.6, 6.1, 2.5],
[6.5, 3.2, 5.1, 2. ],
[6.4, 2.7, 5.3, 1.9],
[6.8, 3. , 5.5, 2.1],
[5.7, 2.5, 5. , 2. ],
[5.8, 2.8, 5.1, 2.4],
[6.4, 3.2, 5.3, 2.3],
[6.5, 3. , 5.5, 1.8],
[7.7, 3.8, 6.7, 2.2],
[7.7, 2.6, 6.9, 2.3],
[6. , 2.2, 5. , 1.5],
[6.9, 3.2, 5.7, 2.3],
[5.6, 2.8, 4.9, 2. ],
[7.7, 2.8, 6.7, 2. ],
[6.3, 2.7, 4.9, 1.8],
```

```
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
       [6.1, 2.6, 5.6, 1.4],
       [7.7, 3. , 6.1, 2.3],
       [6.3, 3.4, 5.6, 2.4],
       [6.4, 3.1, 5.5, 1.8],
       [6. , 3. , 4.8, 1.8],
       [6.9, 3.1, 5.4, 2.1],
       [6.7, 3.1, 5.6, 2.4],
       [6.9, 3.1, 5.1, 2.3],
       [5.8, 2.7, 5.1, 1.9],
       [6.8, 3.2, 5.9, 2.3],
       [6.7, 3.3, 5.7, 2.5],
       [6.7, 3. , 5.2, 2.3],
       [6.3, 2.5, 5. , 1.9],
       [6.5, 3. , 5.2, 2. ],
       [6.2, 3.4, 5.4, 2.3],
       [5.9, 3. , 5.1, 1.8]])
```

In [3]:
```
import numpy as np
data.shape
```

Out[3]:
```
(150, 4)
```

lire les noms des colonnes de iris

In [4]:
```
iris.feature_names
['sepal length (cm)',
 'sepal width (cm)',
```

```
 'petal length (cm)',
 'petal width (cm)']
```

Out[4]:
```
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

lire les targets de iris

In [5]:
```
target = iris.target
target
```

Out[5]:
```
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, 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])
```

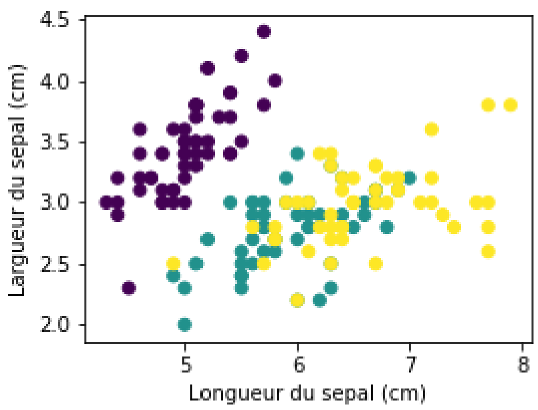
In [6]:
```
from array import array
iris.target_names
```

Out[6]:
```
array(['setosa', 'versicolor', 'virginica'], dtype='<U10')
```

shema permet de présenter les classes de iris en fonction de longeur et largeur de sepal en utilisant matplotlib

In [7]:
```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(4,3))
plt.scatter(data[:, 0], data[:, 1], c=target)
plt.xlabel('Longueur du sepal (cm)')
plt.ylabel('Largueur du sepal (cm)')
```

Out[7]:
```
Text(0, 0.5, 'Largueur du sepal (cm)')
```



In [8]:
```
from sklearn import neighbors
clf = neighbors.KNeighborsClassifier()
```

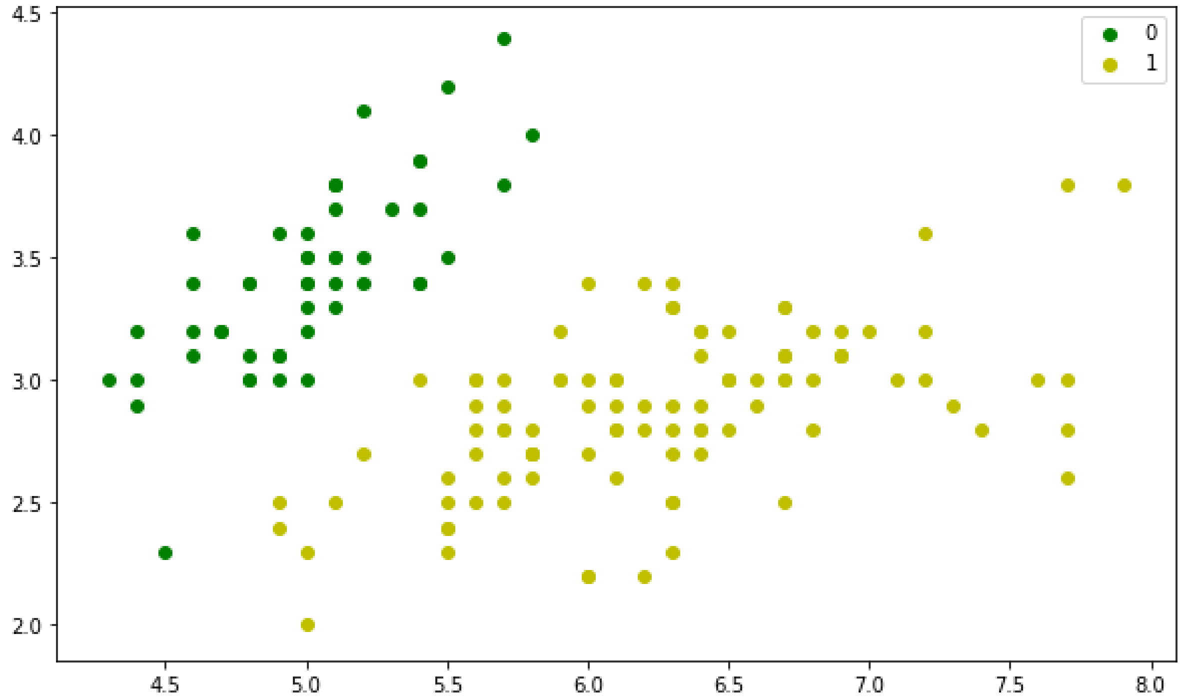
In [9]:
```
clf.fit(data, target)
```

Out[9]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=
2,
                     weights='uniform')
```

In [10]:
```
clf.predict(data[::10])
```

Out[10]:
```
array([0, 0, 0, 0, 0, 1, 1, 2, 1, 1, 2, 2, 2, 2, 2])
```

In [11]:
```
target[::10]
```
Out[11]:
```
array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2])
```
In [12]:
```
data_train = data[::2]
data_test = data[1::2]
target_train = target[::2]
target_test = target[1::2]
clf.fit(data_train, target_train)
```
Out[12]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowsk
i',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=
2,
                     weights='uniform')
```
In [13]:
```
np.sum(clf.predict(data_test) - target_test)
```
Out[13]:
```
1
```
In [14]:
```
X = iris.data[:, :2] # Utiliser les deux premiers colonnes afin d'avoir
 un problème de classification binaire. 
y = (iris.target != 0) * 1 # re-étiquetage des fleurs
```
In [15]:
```
#visualisation des données
plt.figure(figsize=(10, 6))
plt.scatter(X[y == 0][:, 0], X[y == 0][:, 1], color='g', label='0')
plt.scatter(X[y == 1][:, 0], X[y == 1][:, 1], color='y', label='1')
plt.legend();
```



In [16]:
```
from sklearn.linear_model import LogisticRegression # import de la clas
se

model = LogisticRegression(C=1e20) # construction d'un objet de Régress
ion logistique
model.fit(X, y) # Entrainement du modèle
```

Out[16]:
```
LogisticRegression(C=1e+20, class_weight=None, dual=False, fit_intercep
t=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbo
se=0,
                   warm_start=False)
```

In [17]:
```
Iries_To_Predict = [
    [5.5, 2.5],
```

```
    [7, 3],
    [3,2],
    [5,3]
]
```

In [18]:
```
model.predict(Iries_To_Predict)
```

Out[18]:
```
array([1, 1, 0, 0])
```

In [19]:
```
import pandas as pd
import numpy as np
import sklearn.metrics as sm
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
```

In [20]:
```
print(iris)
print(iris.data)
print(iris.feature_names)
print(iris.target)
print(iris.target_names)
```

```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
```

```
[5.1, 3.5, 1.4, 0.3],
[5.7, 3.8, 1.7, 0.3],
[5.1, 3.8, 1.5, 0.3],
[5.4, 3.4, 1.7, 0.2],
[5.1, 3.7, 1.5, 0.4],
[4.6, 3.6, 1. , 0.2],
[5.1, 3.3, 1.7, 0.5],
[4.8, 3.4, 1.9, 0.2],
[5. , 3. , 1.6, 0.2],
[5. , 3.4, 1.6, 0.4],
[5.2, 3.5, 1.5, 0.2],
[5.2, 3.4, 1.4, 0.2],
[4.7, 3.2, 1.6, 0.2],
[4.8, 3.1, 1.6, 0.2],
[5.4, 3.4, 1.5, 0.4],
[5.2, 4.1, 1.5, 0.1],
[5.5, 4.2, 1.4, 0.2],
[4.9, 3.1, 1.5, 0.2],
[5. , 3.2, 1.2, 0.2],
[5.5, 3.5, 1.3, 0.2],
[4.9, 3.6, 1.4, 0.1],
[4.4, 3. , 1.3, 0.2],
[5.1, 3.4, 1.5, 0.2],
[5. , 3.5, 1.3, 0.3],
[4.5, 2.3, 1.3, 0.3],
[4.4, 3.2, 1.3, 0.2],
[5. , 3.5, 1.6, 0.6],
[5.1, 3.8, 1.9, 0.4],
[4.8, 3. , 1.4, 0.3],
[5.1, 3.8, 1.6, 0.2],
[4.6, 3.2, 1.4, 0.2],
[5.3, 3.7, 1.5, 0.2],
[5. , 3.3, 1.4, 0.2],
[7. , 3.2, 4.7, 1.4],
[6.4, 3.2, 4.5, 1.5],
[6.9, 3.1, 4.9, 1.5],
[5.5, 2.3, 4. , 1.3],
[6.5, 2.8, 4.6, 1.5],
[5.7, 2.8, 4.5, 1.3],
```

```
[6.3, 3.3, 4.7, 1.6],
[4.9, 2.4, 3.3, 1. ],
[6.6, 2.9, 4.6, 1.3],
[5.2, 2.7, 3.9, 1.4],
[5. , 2. , 3.5, 1. ],
[5.9, 3. , 4.2, 1.5],
[6. , 2.2, 4. , 1. ],
[6.1, 2.9, 4.7, 1.4],
[5.6, 2.9, 3.6, 1.3],
[6.7, 3.1, 4.4, 1.4],
[5.6, 3. , 4.5, 1.5],
[5.8, 2.7, 4.1, 1. ],
[6.2, 2.2, 4.5, 1.5],
[5.6, 2.5, 3.9, 1.1],
[5.9, 3.2, 4.8, 1.8],
[6.1, 2.8, 4. , 1.3],
[6.3, 2.5, 4.9, 1.5],
[6.1, 2.8, 4.7, 1.2],
[6.4, 2.9, 4.3, 1.3],
[6.6, 3. , 4.4, 1.4],
[6.8, 2.8, 4.8, 1.4],
[6.7, 3. , 5. , 1.7],
[6. , 2.9, 4.5, 1.5],
[5.7, 2.6, 3.5, 1. ],
[5.5, 2.4, 3.8, 1.1],
[5.5, 2.4, 3.7, 1. ],
[5.8, 2.7, 3.9, 1.2],
[6. , 2.7, 5.1, 1.6],
[5.4, 3. , 4.5, 1.5],
[6. , 3.4, 4.5, 1.6],
[6.7, 3.1, 4.7, 1.5],
[6.3, 2.3, 4.4, 1.3],
[5.6, 3. , 4.1, 1.3],
[5.5, 2.5, 4. , 1.3],
[5.5, 2.6, 4.4, 1.2],
[6.1, 3. , 4.6, 1.4],
[5.8, 2.6, 4. , 1.2],
[5. , 2.3, 3.3, 1. ],
[5.6, 2.7, 4.2, 1.3],
```

```
       [5.7, 3. , 4.2, 1.2],
       [5.7, 2.9, 4.2, 1.3],
       [6.2, 2.9, 4.3, 1.3],
       [5.1, 2.5, 3. , 1.1],
       [5.7, 2.8, 4.1, 1.3],
       [6.3, 3.3, 6. , 2.5],
       [5.8, 2.7, 5.1, 1.9],
       [7.1, 3. , 5.9, 2.1],
       [6.3, 2.9, 5.6, 1.8],
       [6.5, 3. , 5.8, 2.2],
       [7.6, 3. , 6.6, 2.1],
       [4.9, 2.5, 4.5, 1.7],
       [7.3, 2.9, 6.3, 1.8],
       [6.7, 2.5, 5.8, 1.8],
       [7.2, 3.6, 6.1, 2.5],
       [6.5, 3.2, 5.1, 2. ],
       [6.4, 2.7, 5.3, 1.9],
       [6.8, 3. , 5.5, 2.1],
       [5.7, 2.5, 5. , 2. ],
       [5.8, 2.8, 5.1, 2.4],
       [6.4, 3.2, 5.3, 2.3],
       [6.5, 3. , 5.5, 1.8],
       [7.7, 3.8, 6.7, 2.2],
       [7.7, 2.6, 6.9, 2.3],
       [6. , 2.2, 5. , 1.5],
       [6.9, 3.2, 5.7, 2.3],
       [5.6, 2.8, 4.9, 2. ],
       [7.7, 2.8, 6.7, 2. ],
       [6.3, 2.7, 4.9, 1.8],
       [6.7, 3.3, 5.7, 2.1],
       [7.2, 3.2, 6. , 1.8],
       [6.2, 2.8, 4.8, 1.8],
       [6.1, 3. , 4.9, 1.8],
       [6.4, 2.8, 5.6, 2.1],
       [7.2, 3. , 5.8, 1.6],
       [7.4, 2.8, 6.1, 1.9],
       [7.9, 3.8, 6.4, 2. ],
       [6.4, 2.8, 5.6, 2.2],
       [6.3, 2.8, 5.1, 1.5],
```

```
        [6.1, 2.6, 5.6, 1.4],
        [7.7, 3. , 6.1, 2.3],
        [6.3, 3.4, 5.6, 2.4],
        [6.4, 3.1, 5.5, 1.8],
        [6. , 3. , 4.8, 1.8],
        [6.9, 3.1, 5.4, 2.1],
        [6.7, 3.1, 5.6, 2.4],
        [6.9, 3.1, 5.1, 2.3],
        [5.8, 2.7, 5.1, 1.9],
        [6.8, 3.2, 5.9, 2.3],
        [6.7, 3.3, 5.7, 2.5],
        [6.7, 3. , 5.2, 2.3],
        [6.3, 2.5, 5. , 1.9],
        [6.5, 3. , 5.2, 2. ],
        [6.2, 3.4, 5.4, 2.3],
        [5.9, 3. , 5.1, 1.8]]), '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, 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]), 'target_
names': array(['setosa', 'versicolor', 'virginica'], dtype='<U10'), 'DE
SCR': '.. _iris_dataset:\n\nIris plants dataset\n--------------------\n
\n**Data Set Characteristics:**\n\n    :Number of Instances: 150 (50 in
each of three classes)\n    :Number of Attributes: 4 numeric, predictiv
e attributes and the class\n    :Attribute Information:\n        - sepa
l length in cm\n        - sepal width in cm\n        - petal length in
cm\n        - petal width in cm\n        - class:\n                - Ir
is-Setosa\n                - Iris-Versicolour\n                - Iris-V
irginica\n                \n    :Summary Statistics:\n\n    ===========
=== ==== ==== ======= ===== ====================\n                    M
in  Max   Mean    SD   Class Correlation\n    ============== ==== ====
```

```
======= ===== ====================\n    sepal length:   4.3  7.9   5.84
   0.83    0.7826\n    sepal width:    2.0  4.4   3.05   0.43   -0.4194
\n    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)\n    pe
tal width:    0.1  2.5   1.20   0.76    0.9565  (high!)\n    ==========
==== ==== ==== ======= ===== ====================\n\n    :Missing Attri
bute Values: None\n    :Class Distribution: 33.3% for each of 3 classe
s.\n    :Creator: R.A. Fisher\n    :Donor: Michael Marshall (MARSHALL%P
LU@io.arc.nasa.gov)\n    :Date: July, 1988\n\nThe famous Iris database,
first used by Sir R.A. Fisher. The dataset is taken\nfrom Fisher\'s pap
er. Note that it\'s the same as in R, but not as in the UCI\nMachine Le
arning Repository, which has two wrong data points.\n\nThis is perhaps
the best known database to be found in the\npattern recognition literat
ure.  Fisher\'s paper is a classic in the field and\nis referenced freq
uently to this day.  (See Duda & Hart, for example.)  The\ndata set con
tains 3 classes of 50 instances each, where each class refers to a\ntyp
e of iris plant.  One class is linearly separable from the other 2; the
\nlatter are NOT linearly separable from each other.\n\n.. topic:: Refe
rences\n\n   - Fisher, R.A. "The use of multiple measurements in taxono
mic problems"\n     Annual Eugenics, 7, Part II, 179-188 (1936); also i
n "Contributions to\n     Mathematical Statistics" (John Wiley, NY, 195
0).\n   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Sc
ene Analysis.\n     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.
See page 218.\n   - Dasarathy, B.V. (1980) "Nosing Around the Neighborh
ood: A New System\n     Structure and Classification Rule for Recogniti
on in Partially Exposed\n     Environments".  IEEE Transactions on Patt
ern Analysis and Machine\n     Intelligence, Vol. PAMI-2, No. 1, 67-7
1.\n   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE
Transactions\n     on Information Theory, May 1972, 431-433.\n   - See
also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II\n
  conceptual clustering system finds 3 classes in the data.\n   - Many,
many more ...', 'feature_names': ['sepal length (cm)', 'sepal width (c
m)', 'petal length (cm)', 'petal width (cm)'], 'filename': 'C:\\Program
Data\\Anaconda3\\lib\\site-packages\\sklearn\\datasets\\data\\iris.cs
v'}
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
```

```
[5.4 3.9 1.7 0.4]
[4.6 3.4 1.4 0.3]
[5.  3.4 1.5 0.2]
[4.4 2.9 1.4 0.2]
[4.9 3.1 1.5 0.1]
[5.4 3.7 1.5 0.2]
[4.8 3.4 1.6 0.2]
[4.8 3.  1.4 0.1]
[4.3 3.  1.1 0.1]
[5.8 4.  1.2 0.2]
[5.7 4.4 1.5 0.4]
[5.4 3.9 1.3 0.4]
[5.1 3.5 1.4 0.3]
[5.7 3.8 1.7 0.3]
[5.1 3.8 1.5 0.3]
[5.4 3.4 1.7 0.2]
[5.1 3.7 1.5 0.4]
[4.6 3.6 1.  0.2]
[5.1 3.3 1.7 0.5]
[4.8 3.4 1.9 0.2]
[5.  3.  1.6 0.2]
[5.  3.4 1.6 0.4]
[5.2 3.5 1.5 0.2]
[5.2 3.4 1.4 0.2]
[4.7 3.2 1.6 0.2]
[4.8 3.1 1.6 0.2]
[5.4 3.4 1.5 0.4]
[5.2 4.1 1.5 0.1]
[5.5 4.2 1.4 0.2]
[4.9 3.1 1.5 0.2]
[5.  3.2 1.2 0.2]
[5.5 3.5 1.3 0.2]
[4.9 3.6 1.4 0.1]
[4.4 3.  1.3 0.2]
[5.1 3.4 1.5 0.2]
[5.  3.5 1.3 0.3]
[4.5 2.3 1.3 0.3]
[4.4 3.2 1.3 0.2]
[5.  3.5 1.6 0.6]
```

```
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]
 [7.  3.2 4.7 1.4]
 [6.4 3.2 4.5 1.5]
 [6.9 3.1 4.9 1.5]
 [5.5 2.3 4.  1.3]
 [6.5 2.8 4.6 1.5]
 [5.7 2.8 4.5 1.3]
 [6.3 3.3 4.7 1.6]
 [4.9 2.4 3.3 1. ]
 [6.6 2.9 4.6 1.3]
 [5.2 2.7 3.9 1.4]
 [5.  2.  3.5 1. ]
 [5.9 3.  4.2 1.5]
 [6.  2.2 4.  1. ]
 [6.1 2.9 4.7 1.4]
 [5.6 2.9 3.6 1.3]
 [6.7 3.1 4.4 1.4]
 [5.6 3.  4.5 1.5]
 [5.8 2.7 4.1 1. ]
 [6.2 2.2 4.5 1.5]
 [5.6 2.5 3.9 1.1]
 [5.9 3.2 4.8 1.8]
 [6.1 2.8 4.  1.3]
 [6.3 2.5 4.9 1.5]
 [6.1 2.8 4.7 1.2]
 [6.4 2.9 4.3 1.3]
 [6.6 3.  4.4 1.4]
 [6.8 2.8 4.8 1.4]
 [6.7 3.  5.  1.7]
 [6.  2.9 4.5 1.5]
 [5.7 2.6 3.5 1. ]
 [5.5 2.4 3.8 1.1]
 [5.5 2.4 3.7 1. ]
 [5.8 2.7 3.9 1.2]
```

```
[6.  2.7 5.1 1.6]
[5.4 3.  4.5 1.5]
[6.  3.4 4.5 1.6]
[6.7 3.1 4.7 1.5]
[6.3 2.3 4.4 1.3]
[5.6 3.  4.1 1.3]
[5.5 2.5 4.  1.3]
[5.5 2.6 4.4 1.2]
[6.1 3.  4.6 1.4]
[5.8 2.6 4.  1.2]
[5.  2.3 3.3 1. ]
[5.6 2.7 4.2 1.3]
[5.7 3.  4.2 1.2]
[5.7 2.9 4.2 1.3]
[6.2 2.9 4.3 1.3]
[5.1 2.5 3.  1.1]
[5.7 2.8 4.1 1.3]
[6.3 3.3 6.  2.5]
[5.8 2.7 5.1 1.9]
[7.1 3.  5.9 2.1]
[6.3 2.9 5.6 1.8]
[6.5 3.  5.8 2.2]
[7.6 3.  6.6 2.1]
[4.9 2.5 4.5 1.7]
[7.3 2.9 6.3 1.8]
[6.7 2.5 5.8 1.8]
[7.2 3.6 6.1 2.5]
[6.5 3.2 5.1 2. ]
[6.4 2.7 5.3 1.9]
[6.8 3.  5.5 2.1]
[5.7 2.5 5.  2. ]
[5.8 2.8 5.1 2.4]
[6.4 3.2 5.3 2.3]
[6.5 3.  5.5 1.8]
[7.7 3.8 6.7 2.2]
[7.7 2.6 6.9 2.3]
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
```

```
 [7.7 2.8 6.7 2. ]
 [6.3 2.7 4.9 1.8]
 [6.7 3.3 5.7 2.1]
 [7.2 3.2 6.  1.8]
 [6.2 2.8 4.8 1.8]
 [6.1 3.  4.9 1.8]
 [6.4 2.8 5.6 2.1]
 [7.2 3.  5.8 1.6]
 [7.4 2.8 6.1 1.9]
 [7.9 3.8 6.4 2. ]
 [6.4 2.8 5.6 2.2]
 [6.3 2.8 5.1 1.5]
 [6.1 2.6 5.6 1.4]
 [7.7 3.  6.1 2.3]
 [6.3 3.4 5.6 2.4]
 [6.4 3.1 5.5 1.8]
 [6.  3.  4.8 1.8]
 [6.9 3.1 5.4 2.1]
 [6.7 3.1 5.6 2.4]
 [6.9 3.1 5.1 2.3]
 [5.8 2.7 5.1 1.9]
 [6.8 3.2 5.9 2.3]
 [6.7 3.3 5.7 2.5]
 [6.7 3.  5.2 2.3]
 [6.3 2.5 5.  1.9]
 [6.5 3.  5.2 2. ]
 [6.2 3.4 5.4 2.3]
 [5.9 3.  5.1 1.8]]
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal w
idth (cm)']
[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 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]
['setosa' 'versicolor' 'virginica']
```

In [84]:
```
#Stocker les données en tant que DataFrame Pandas
x=pd.DataFrame(iris.data)
# définir les noms de colonnes
x.columns=['Sepal_Length','Sepal_width','Petal_Length','Petal_width']
y=pd.DataFrame(iris.target)
y.columns=['Targets']
```

In [85]:
```
#Cluster K-means
model=KMeans(n_clusters=3)
#adapter le modèle de données
model.fit(df)
```

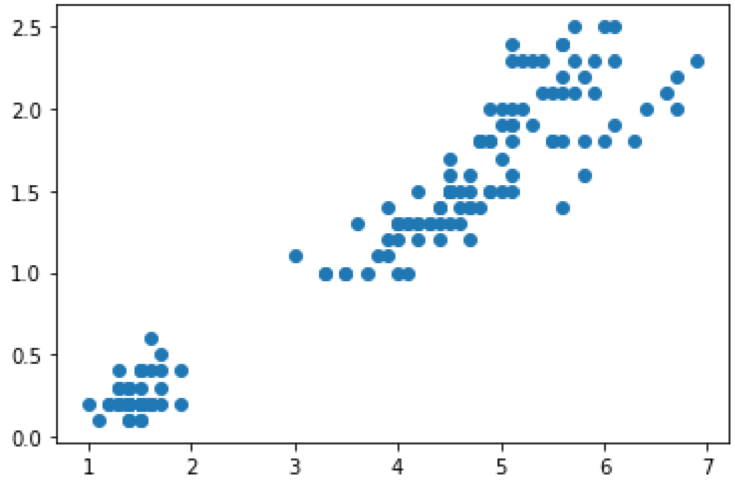
Out[85]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='aut
o',
       random_state=None, tol=0.0001, verbose=0)
```

In [86]:
```
print(model.labels_)
```

```
[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 2 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 2 2 2 1 2 2
2 2
 2 2 1 1 2 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 2 2 1 2 2 2 2 1 2 2 2 1 2 2 2
1 2
 2 1]
```

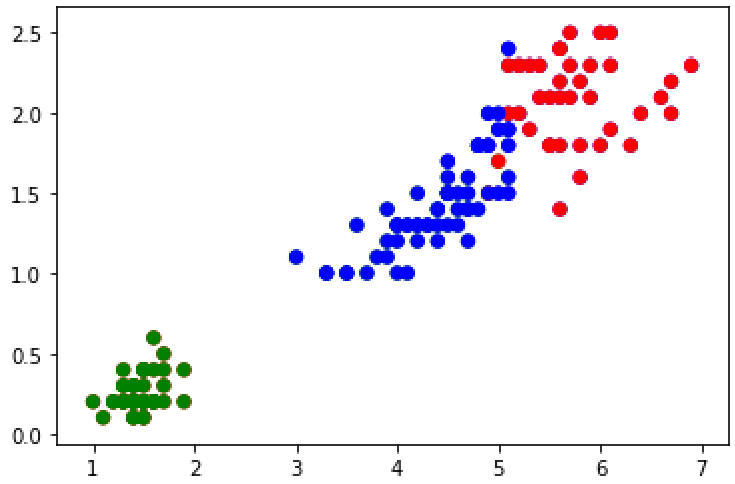
In [88]:
```
plt.scatter(x.Petal_Length, x.Petal_width)
```

Out[88]:
```
<matplotlib.collections.PathCollection at 0x2a6788e3c88>
```



In [25]:
```
colormap=np.array(['Red','green','blue'])
plt.scatter(df.Petal_Length, df.Petal_width,c=colormap[y.Targets],s=40)
plt.scatter(df.Petal_Length, df.Petal_width,c=colormap[model.labels_],s
=40)
```

Out[25]: <matplotlib.collections.PathCollection at 0x2a67495d848>



In [26]:
```
from sklearn.naive_bayes import GaussianNB
```

```
tr=GaussianNB()
tr = tr.fit(data,target)
tr
```

Out[26]: GaussianNB(priors=None, var_smoothing=1e-09)

In [72]:
```
x=pd.DataFrame(iris.data)
# définir les noms de colonnes
x.columns=['Sepal_Length','Sepal_width','Petal_Length','Petal_width']
y=pd.DataFrame(iris.target)
y.columns=['Targets']
```

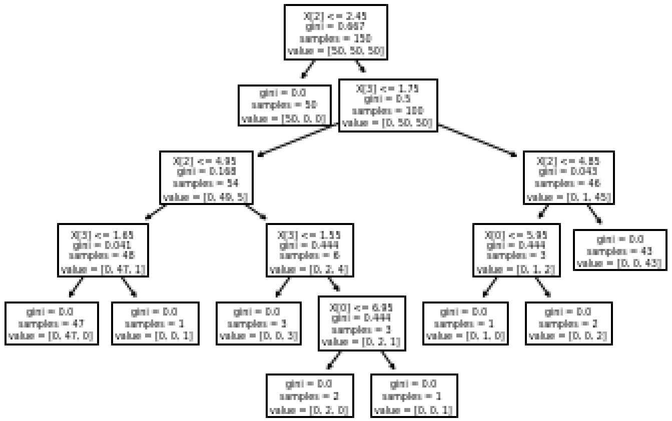
In [79]:
```
from sklearn import tree

clf = tree.DecisionTreeClassifier()
clf = clf.fit(df,y)
clf
```

Out[79]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gin
i',
                       max_depth=None, max_features=None, max_leaf_node
s=None,
                       min_impurity_decrease=0.0, min_impurity_split=No
ne,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=None, splitter='best')
```

In [80]:
```
tree.plot_tree(clf)
```

Out[80]:
```
[Text(167.4, 199.32, 'X[2] <= 2.45\ngini = 0.667\nsamples = 150\nvalu
e = [50, 50, 50]'),
 Text(141.64615384615385, 163.07999999999998, 'gini = 0.0\nsamples =
50\nvalue = [50, 0, 0]'),
 Text(193.15384615384616, 163.07999999999998, 'X[3] <= 1.75\ngini =
0.5\nsamples = 100\nvalue = [0, 50, 50]'),
 Text(103.01538461538462, 126.83999999999999, 'X[2] <= 4.95\ngini =
0.168\nsamples = 54\nvalue = [0, 49, 5]'),
```

```
 Text(51.50769230769231, 90.6, 'X[3] <= 1.65\ngini = 0.041\nsamples =
48\nvalue = [0, 47, 1]'),
 Text(25.753846153846155, 54.359999999999985, 'gini = 0.0\nsamples =
47\nvalue = [0, 47, 0]'),
 Text(77.26153846153846, 54.359999999999985, 'gini = 0.0\nsamples = 1
\nvalue = [0, 0, 1]'),
 Text(154.52307692307693, 90.6, 'X[3] <= 1.55\ngini = 0.444\nsamples
= 6\nvalue = [0, 2, 4]'),
 Text(128.76923076923077, 54.359999999999985, 'gini = 0.0\nsamples =
3\nvalue = [0, 0, 3]'),
 Text(180.27692307692308, 54.359999999999985, 'X[0] <= 6.95\ngini =
0.444\nsamples = 3\nvalue = [0, 2, 1]'),
 Text(154.52307692307693, 18.119999999999976, 'gini = 0.0\nsamples =
2\nvalue = [0, 2, 0]'),
 Text(206.03076923076924, 18.119999999999976, 'gini = 0.0\nsamples =
1\nvalue = [0, 0, 1]'),
 Text(283.2923076923077, 126.83999999999999, 'X[2] <= 4.85\ngini = 0.
043\nsamples = 46\nvalue = [0, 1, 45]'),
 Text(257.53846153846155, 90.6, 'X[0] <= 5.95\ngini = 0.444\nsamples
= 3\nvalue = [0, 1, 2]'),
 Text(231.7846153846154, 54.359999999999985, 'gini = 0.0\nsamples = 1
\nvalue = [0, 1, 0]'),
 Text(283.2923076923077, 54.359999999999985, 'gini = 0.0\nsamples = 2
\nvalue = [0, 0, 2]'),
 Text(309.04615384615386, 90.6, 'gini = 0.0\nsamples = 43\nvalue =
[0, 0, 43]')]
```



In [40]:
```
conda install graphviz
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3

  added / updated specs:
    - graphviz


The following packages will be SUPERSEDED by a higher-priority channel:

  graphviz                                        anaconda --> pkgs/ma
in


Preparing transaction: ...working... done
Verifying transaction: ...working... failed

Note: you may need to restart the kernel to use updated packages.
```

```
EnvironmentNotWritableError: The current user does not have write permi
ssions to the target environment.
  environment location: C:\ProgramData\Anaconda3
```

In [41]:
```
conda install python-graphviz
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3
```

```
  added / updated specs:
    - python-graphviz


The following packages will be UPDATED:

  python-graphviz     anaconda::python-graphviz-0.14.2-py_0 --> pkgs/ma
in::python-graphviz-0.15-pyhd3eb1b0_0


Preparing transaction: ...working... done
Verifying transaction: ...working... failed

Note: you may need to restart the kernel to use updated packages.
```

```
EnvironmentNotWritableError: The current user does not have write permi
ssions to the target environment.
  environment location: C:\ProgramData\Anaconda3
```

In [82]:

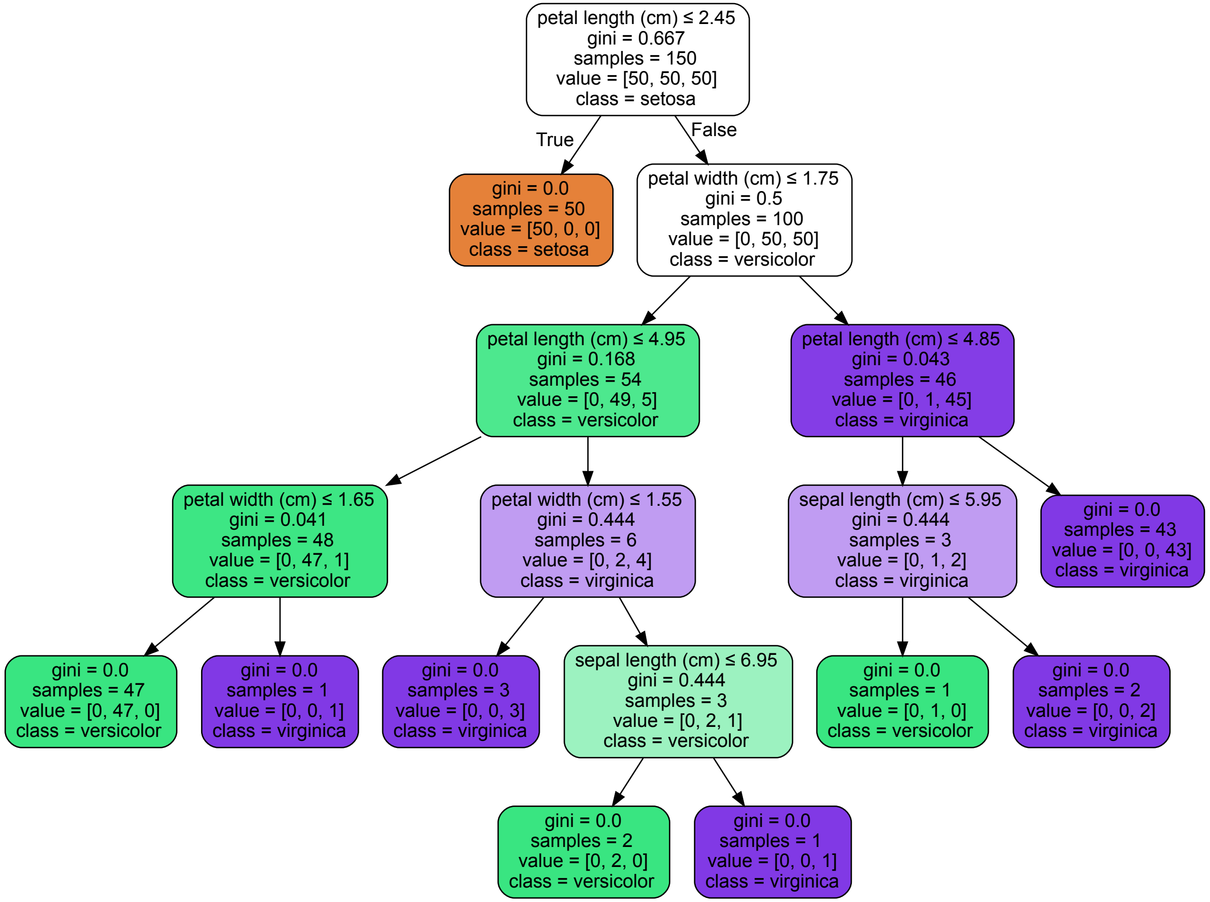
```python
import graphviz
dot_data = tree.export_graphviz(clf, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("iris")
```

Out[82]: 'iris.pdf'

In [83]:

```python
dot_data = tree.export_graphviz(clf, out_file=None,
feature_names=iris.feature_names,
class_names=iris.target_names,
filled=True, rounded=True,
special_characters=True)
graph = graphviz.Source(dot_data)
graph
```

Out[83]:



In [33]:

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_text
iris = load_iris()
decision_tree = DecisionTreeClassifier(random_state=0, max_depth=2)
decision_tree = decision_tree.fit(iris.data, iris.target)
r = export_text(decision_tree, feature_names=iris['feature_names'])
print(r)
```

```
|--- petal width (cm) <= 0.80
|   |--- class: 0
```

```
|--- petal width (cm) >  0.80
|   |--- petal width (cm) <= 1.75
|   |   |--- class: 1
|   |--- petal width (cm) >  1.75
|   |   |--- class: 2
```

In [34]:

```
from sklearn import tree
X = [[0, 0], [2, 2]]
y = [0.5, 2.5]
clf = tree.DecisionTreeRegressor()
clf = clf.fit(X, y)
clf.predict([[1, 1]])
```

Out[34]:

```
array([0.5])
```

In [35]:

```
#from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
X, y = load_iris(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5
, random_state=0)
gnb = GaussianNB()
y_pred = gnb.fit(X_train, y_train).predict(X_test)
print("Number of mislabeled points out of a total %d points : %d"
      % (X_test.shape[0], (y_test != y_pred).sum()))
```

```
Number of mislabeled points out of a total 75 points : 4
```

iris prediction fausse et pourcentage de prediction

utilisation de l'algorithme de random forest

In [36]:

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble  import RandomForestClassifier
from sklearn.metrics import accuracy_score
x=iris.data
y=iris.target
X_train,X_test,y_train,y_test=train_test_split(x,y,test_size = 0.5, ran
dom_state=0)
sc=StandardScaler()
X_train_std=sc.fit_transform(X_train)
X_test_std=sc.fit_transform(X_test)
forest=RandomForestClassifier(criterion='entropy',n_estimators=10,rando
m_state=1,n_jobs=2)
forest.fit(X_train_std,y_train)
y_pred=forest.predict(X_test_std)
print('wrong prediction out of total')
print((y_test !=y_pred).sum(),'/',((y_test== y_pred).sum()+(y_test !=y_
pred).sum()))
print('percentage accuracy',100*accuracy_score(y_test, y_pred))
```

```
wrong prediction out of total
5 / 75
percentage accuracy 93.33333333333333
```

installation keras

In [37]:
```
#pip install keras
```

installation tensorflow

In [38]:
```
#pip install --upgrade tensorflow
```

probleme au niveau de tensorflow qui empeche l'algorithme de s'executer

In [39]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
iris = load_iris()
X=iris['data']
Y=to_categorical(iris['target'])
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.3)
model =Sequential()
model.add(Dense(10,input_dim=4,activation='relu'))
model.add(Dense(3,activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer='adam',metrics=
['accuracy'])
model.fit(X_train,Y_train, validation_data=(X_test,Y_test),epochs=200,b
atch_size=10)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:516: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:517: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:518: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:519: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:520: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\framework
\dtypes.py:525: FutureWarning: Passing (type, 1) or '1type' as a synony
m of type is deprecated; in a future version of numpy, it will be under
stood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorboard\compat\tensorflo
w_stub\dtypes.py:541: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be
understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorboard\compat\tensorflo
w_stub\dtypes.py:542: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be
understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorboard\compat\tensorflo
w_stub\dtypes.py:543: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be
understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorboard\compat\tensorflo
w_stub\dtypes.py:544: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be
understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorboard\compat\tensorflo
w_stub\dtypes.py:545: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be
understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\ProgramData\Anaconda3\lib\site-packages\tensorboard\compat\tensorflo
w_stub\dtypes.py:550: FutureWarning: Passing (type, 1) or '1type' as a
synonym of type is deprecated; in a future version of numpy, it will be
```

```
understood as (type, (1,)) / '(1,)type'.
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
C:\ProgramData\Anaconda3\lib\site-packages\keras\__init__.py in <module>
      2 try:
----> 3     from tensorflow.keras.layers.experimental.preprocessing import RandomRotation
      4 except ImportError:

ModuleNotFoundError: No module named 'tensorflow.keras.layers.experimental.preprocessing'

During handling of the above exception, another exception occurred:

ImportError                               Traceback (most recent call last)
<ipython-input-39-ff498d46e478> in <module>
----> 1 from keras.models import Sequential
      2 from keras.layers import Dense
      3 from keras.utils import to_categorical
      4 from sklearn.datasets import load_iris
      5 from sklearn.model_selection import train_test_split

C:\ProgramData\Anaconda3\lib\site-packages\keras\__init__.py in <module>
      4 except ImportError:
      5     raise ImportError(
----> 6         'Keras requires TensorFlow 2.2 or higher. '
      7         'Install TensorFlow via `pip install tensorflow`')
      8

ImportError: Keras requires TensorFlow 2.2 or higher. Install TensorFlow via `pip install tensorflow`
```

In [43]:
```
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from sklearn import datasets
from sklearn import svm
X, y = datasets.load_iris(return_X_y=True)
X.shape, y.shape
```

Out[43]: ((150, 4), (150,))

In [44]:
```
X_train, X_test, y_train, y_test = train_test_split(
 X, y, test_size=0.4, random_state=0)
X_train.shape, y_train.shape
```

Out[44]: ((90, 4), (90,))

In [45]:
```
X_test.shape, y_test.shape
```

Out[45]: ((60, 4), (60,))

In [46]:
```
clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
clf.score(X_test, y_test)
```

Out[46]: 0.9666666666666667

cross val score

In [47]:
```
from sklearn.model_selection import cross_val_score
clf = svm.SVC(kernel='linear', C=1, random_state=42)
scores = cross_val_score(clf, X, y, cv=5)
scores
```

Out[47]: array([0.96666667, 1.   , 0.96666667, 0.96666667, 1.   ])

In [48]:
```
print("%0.2f accuracy with a standard deviation of %0.2f" % (scores.mea
n(), scores.std()))
```

0.98 accuracy with a standard deviation of 0.02

In [49]:
```python
from sklearn import metrics
scores = cross_val_score(
clf, X, y, cv=5, scoring='f1_macro')
scores
```

Out[49]:
```
array([0.96658312, 1.        , 0.96658312, 0.96658312, 1.        ])
```

In [50]:
```python
from sklearn.model_selection import ShuffleSplit
n_samples = X.shape[0]
cv = ShuffleSplit(n_splits=5, test_size=0.3, random_state=0)
cross_val_score(clf, X, y, cv=cv)
```

Out[50]:
```
array([0.97777778, 0.97777778, 1.        , 0.95555556, 1.        ])
```

In [51]:
```python
def custom_cv_2folds(X):
    n = X.shape[0]
    i = 1
    while i <= 2:
        idx = np.arange(n * (i - 1) / 2, n * i / 2, dtype=int)
        yield idx, idx
        i += 1
custom_cv = custom_cv_2folds(X)
cross_val_score(clf, X, y, cv=custom_cv)
```

Out[51]:
```
array([1.        , 0.97333333])
```

data processing

In [52]:
```python
from sklearn import preprocessing
X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size=0.4, random_state=0)
scaler = preprocessing.StandardScaler().fit(X_train)
X_train_transformed = scaler.transform(X_train)
clf = svm.SVC(C=1).fit(X_train_transformed, y_train)
X_test_transformed = scaler.transform(X_test)
clf.score(X_test_transformed, y_test)
```

Out[52]: 0.9333333333333333

In [53]:
```
from sklearn.pipeline import make_pipeline
clf = make_pipeline(preprocessing.StandardScaler(), svm.SVC(C=1))
cross_val_score(clf, X, y, cv=cv)
```

Out[53]: array([0.97777778, 0.93333333, 0.95555556, 0.93333333, 0.97777778])

In [54]:
```
from sklearn.model_selection import cross_validate
from sklearn.metrics import recall_score
scoring = ['precision_macro', 'recall_macro']
clf = svm.SVC(kernel='linear', C=1, random_state=0)
scores = cross_validate(clf, X, y, scoring=scoring)
sorted(scores.keys())

scores['test_recall_macro']
```

Out[54]: array([0.96666667, 1.   , 0.96666667, 0.96666667, 1.   ])

In [55]:
```
from sklearn.metrics import make_scorer
scoring = {'prec_macro': 'precision_macro',
           'rec_macro': make_scorer(recall_score, average='macro')}
scores = cross_validate(clf, X, y, scoring=scoring,
                        cv=5, return_train_score=True)
sorted(scores.keys())

scores['train_rec_macro']
```

Out[55]: array([0.975   , 0.975   , 0.99166667, 0.98333333, 0.98333333])

score validation score keys

In [56]:
```
scores = cross_validate(clf, X, y,
                      scoring='precision_macro', cv=5,
                      return_estimator=True)
sorted(scores.keys())
```

Out[56]: ['estimator', 'fit_time', 'score_time', 'test_score']

In [ ]:

In [ ]:

In [ ]:

In [ ]: