```
In [1]:
import numpy as np
import matplotlib.pyplot as plt
```

Excercice 1

1- Lecture des données

```
In [2]:
from sklearn import datasets
iris = datasets.load_iris()
```

```
In [3]:
# Ce n'est pas un DataFrame, mais une sorte de dictionnaire
print(type(iris))
```

```
<class 'sklearn.utils.Bunch'>
```

```
In [4]:
print(dir(iris))
```

```
['DESCR', 'data', 'data_module', 'feature_names', 'filename', 'frame', 'target', 'target_names']
```

```
In [5]:
# Les noms des varaibles descriptives des différentes observations
iris.feature_names
```

```
Out[5]:
['sepal length (cm)',
 'sepal width (cm)',
 'petal length (cm)',
 'petal width (cm)']
```

```
In [6]:
iris.target_names
```

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

```
In [7]:
# Les labels associés à chaque observations
# [0] étant sétosa, [1] étant versicolor et
# [2] virginica

target = iris.target
target
```

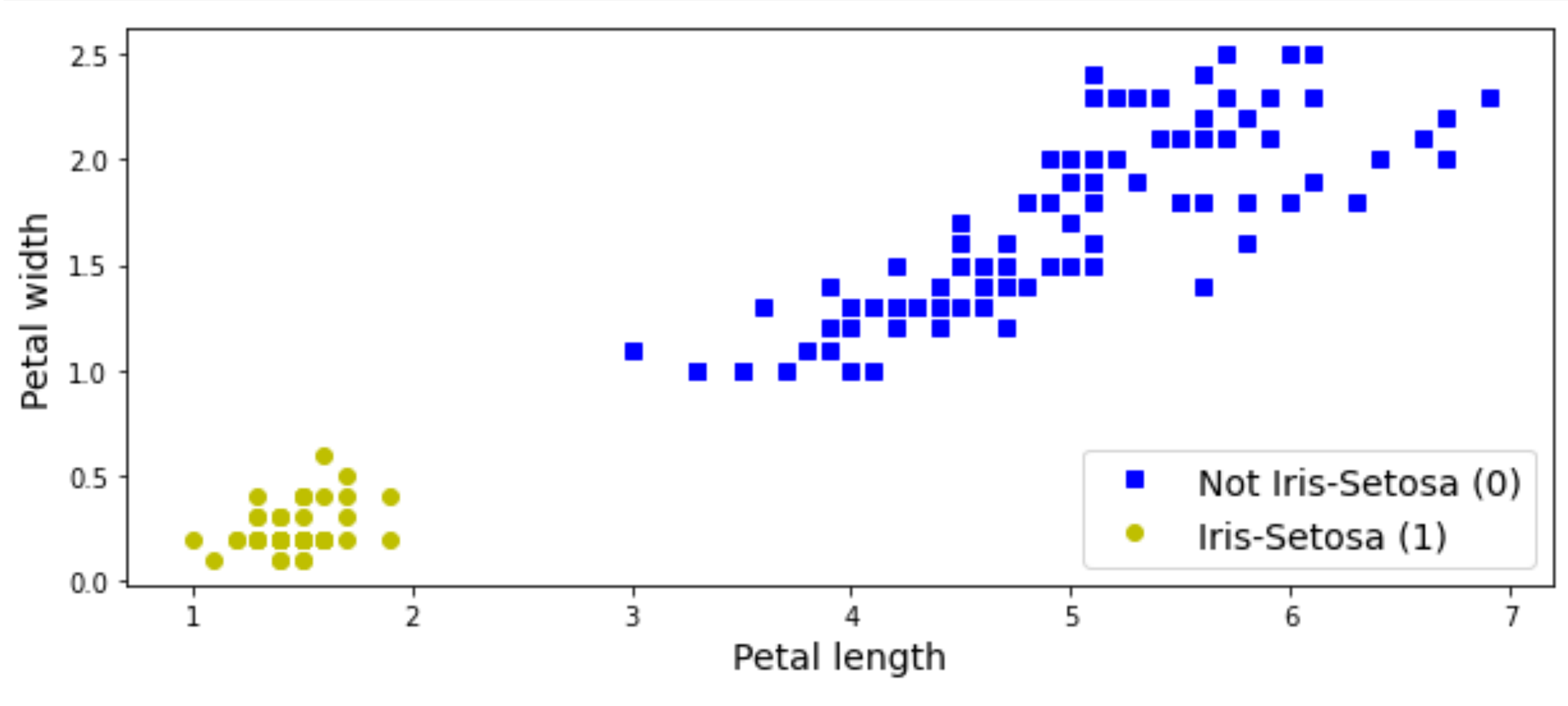
```
Out[7]:
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 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])
```

```
In [8]:
# On limite l'analyse aux deux variables petal length, petal width
# On affecte y=1 pour la clsse Setosa = 1 et 0 ailleurs
X = iris.data[:, (2, 3)]
y = (iris.target == 0).astype(np.int)
```

```
/var/folders/lg/y19wscx13n7g_vwjxll3gd3m0000gn/T/ipykernel_8761/536534188.py:4: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  y = (iris.target == 0).astype(np.int)
```

2- Dispersion des données

```
In [9]:
plt.figure(figsize=(10, 4))
plt.plot(X[y==0, 0], X[y==0, 1], "bs", label="Not Iris-Setosa (0)")
plt.plot(X[y==1, 0], X[y==1, 1], "yo", label="Iris-Setosa (1)")
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.legend(loc="lower right", fontsize=14)
plt.show()
```



3- Entrainement du perceptron

```
In [10]:
# Entrainement du perceptron
# Les hyperparamètres étant le nombre max d'itérations et
# la valeur de la tolérence (critère d'arret) tol=1e-3

from sklearn.linear_model import Perceptron

per_clf = Perceptron(max_iter=100, tol=1e-3, random_state=42)
per_clf.fit(X, y)
```

```
Out[10]:
Perceptron(max_iter=100, random_state=42)
```

4- Paramètres du modèle

```
In [11]:
# Valeur de la constante
per_clf.intercept_
```

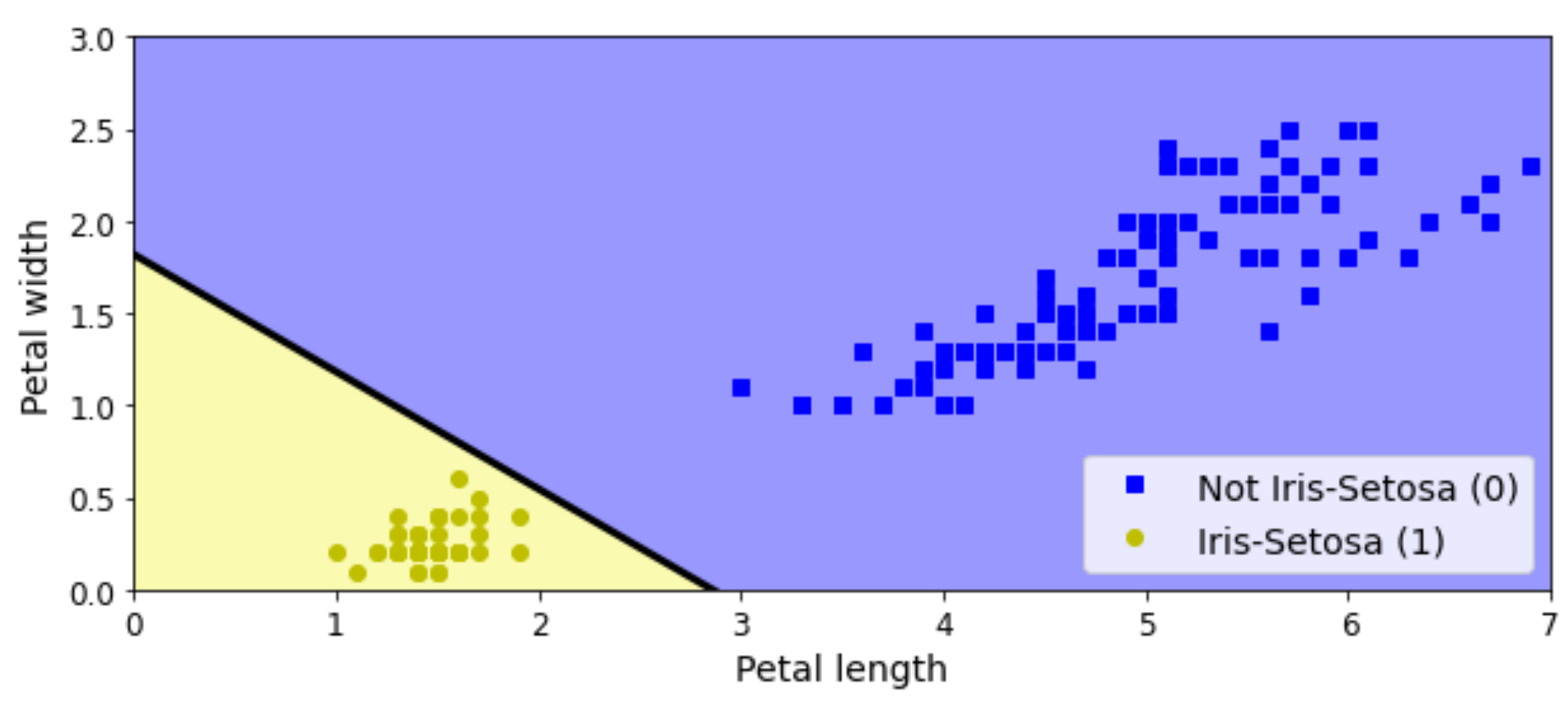
```
Out[11]:
array([4.])
```

```
In [12]:
# Les coefficients de pondérations des caractéristiques
# respectivement petal length, petal width
per_clf.coef_
```

```
Out[12]:
array([[-1.4, -2.2]])
```

4 - Représentation des frontières de décision

```
In [13]:
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rc('axes', labelsize=14)
mpl.rc('xtick', labelsize=12)
mpl.rc('ytick', labelsize=12)

a = -per_clf.coef_[0][0] / per_clf.coef_[0][1]
b = -per_clf.intercept_ / per_clf.coef_[0][1]

axes = [0, 7, 0, 3]

x0, x1 = np.meshgrid(
        np.linspace(axes[0], axes[1], 500).reshape(-1, 1),
        np.linspace(axes[2], axes[3], 200).reshape(-1, 1),
    )
X_new = np.c_[x0.ravel(), x1.ravel()]
y_predict = per_clf.predict(X_new)
zz = y_predict.reshape(x0.shape)

plt.figure(figsize=(10, 4))
plt.plot(X[y==0, 0], X[y==0, 1], "bs", label="Not Iris-Setosa (0)")
plt.plot(X[y==1, 0], X[y==1, 1], "yo", label="Iris-Setosa (1)")

plt.plot([axes[0], axes[1]], [a * axes[0] + b, a * axes[1] + b], "k-", linewidth=3)
from matplotlib.colors import ListedColormap
custom_cmap = ListedColormap(['#9898ff', '#fafab0'])

plt.contourf(x0, x1, zz, cmap=custom_cmap)
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
plt.legend(loc="lower right", fontsize=14)
plt.axis(axes)

plt.show()
```



Excercice 2 : Perceptron multi-couche

```
In [14]:
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

from sklearn.model_selection import train_test_split
```

1- Lecture des données

```
In [15]:
np.random.seed(10) # initialiser le générateur pseudo-aléatoire pour pouvoir répéter l'expérience
```

```
In [16]:
from sklearn import datasets
iris = datasets.load_iris()
```

```
In [17]:
# On limite l'analyse aux deux variables petal length, petal width
# On affecte y=1 pour la clsse Setosa = 1 et 0 ailleurs
X = iris.data[:, :]
y = (iris.target == 0).astype(np.int)
```

```
/var/folders/lg/y19wscx13n7g_vwjxll3gd3m0000gn/T/ipykernel_8761/863911844.py:4: DeprecationWarning: `np.int` is a deprecated alias for the builtin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, check the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  y = (iris.target == 0).astype(np.int)
```

```
In [18]:
# Les labels associés à chaque observations
# [0] étant sétosa, [1] étant versicolor et
# [2] virginica

target = iris.target
target
```

```
Out[18]:
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])
```

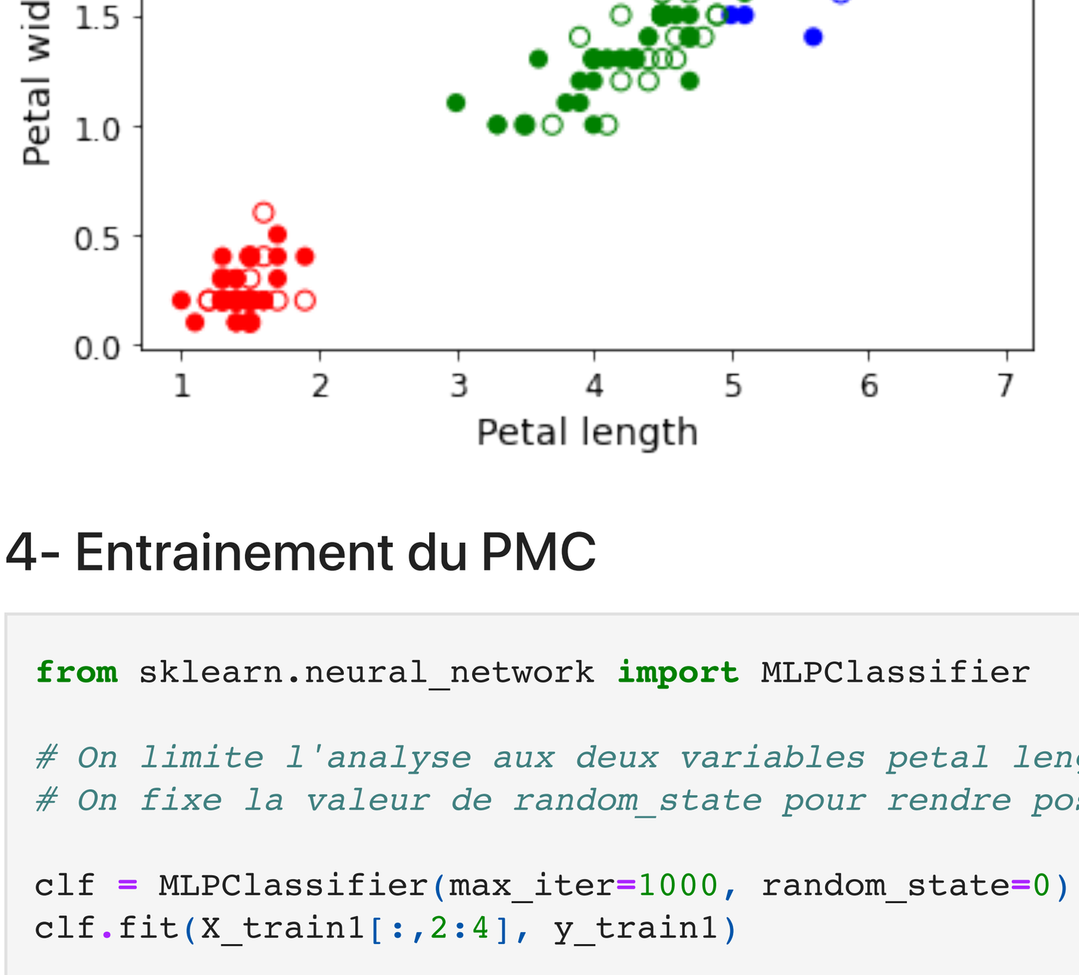
2- Répartion des données en ensembles d'entrainement et de test

```
In [19]:
# X : var. explicatives, y : var. expliquée, train : apprentissage, test : test
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, target, test_size=0.4)
```

3- Dipersion des trois classes

```
In [20]:
# afficher les nuages des données d'apprentissage (pleins) et de test (creux)
# ['r','g','b'] respectivement pour ['setosa', 'versicolor', 'virginica']

cmp = np.array(['r','g','b'])
plt.figure()
plt.scatter(X_train1[:,2],X_train1[:,3],c=cmp[y_train1],s=50,edgecolors='none')
plt.scatter(X_test1[:,2],X_test1[:,3],c='none',s=50,edgecolors=cmp[y_test1])
plt.xlabel("Petal length", fontsize=14)
plt.ylabel("Petal width", fontsize=14)
```

```
Out[20]:
Text(0, 0.5, 'Petal width')
```



4- Entrainement du PMC

```
In [21]:
from sklearn.neural_network import MLPClassifier

# On limite l'analyse aux deux variables petal length, petal width
# On fixe la valeur de random_state pour rendre possible la répétabilité

clf = MLPClassifier(max_iter=1000, random_state=0)
clf.fit(X_train1[:,2:4], y_train1)
```

```
Out[21]:
MLPClassifier(max_iter=1000, random_state=0)
```

5- Performence du modèle

Taux de classification

```
In [22]:
train_score = clf.score(X_train1[:,2:4], y_train1)
print("Le score en train est {}".format(train_score))
test_score = clf.score(X_test1[:,2:4], y_test1)
print("Le score en test est {}".format(test_score))
```

```
Le score en train est 0.9444444444444444
Le score en test est 0.9666666666666667
```

Matrice de confusion

```
In [23]:
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
disp = ConfusionMatrixDisplay.from_estimator(
        clf,
        X_test1[:, 2:4],
        y_test1,
        display_labels=iris.target_names,
        cmap=plt.cm.Blues,
    )

print(disp.confusion_matrix)

plt.show()
```

```
[[18  0  0]
 [ 0 22  2]
 [ 0  0 18]]
```