TP1 : Introduction au module scikit-learn

A. Importation des librairies :

```
#importation de scikit-learn
from sklearn import *

# Importation des librairies numpy (calcul scientifique) et matplotlib.pyplot (figures).
import numpy as np
import matplotlib.pyplot as plt
```

B. Manipulation d'un jeu de données

1- Chargez les données Iris

```
iris = datasets.load_iris()
```

2- Affichez les données, les noms des variables et le nom des classes

```
features = iris.data
targets = iris.target
print('feature_names :', iris.feature_names)
print('target_names :', iris.target_names)
print('Les 5 premières features :\n', iris.features[:5])
print('targets :\n', targets)

feature_names = ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
target_names = ['setosa' 'versicolor' 'virginica']
Les 5 premières features :
 [[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]]
targets :
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 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]
```

3- Affichez le nom des classes pour chaque donnée :

```
print('Le nom des classes pour chaque donnée :')

for features, targets in zip(iris.data, iris.target):
    print('>> les données : ', features, ' > la classe : ', iris.target_names[targets])

Le nom des classes pour chaque donnée :
>> les données :  [5.1 3.5 1.4 0.2]  > la classe :  setosa
>> les données :  [4.9 3.  1.4 0.2]  > la classe :  setosa
>> les données :  [4.7 3.2 1.3 0.2]  > la classe :  setosa
>> les données :  [4.6 3.1 1.5 0.2]  > la classe :  setosa
>> les données :  [5.  3.6 1.4 0.2]  > la classe :  setosa
>> les données :  [5.4 3.9 1.7 0.4]  > la classe :  setosa
>> les données :  [4.6 3.4 1.4 0.3]  > la classe :  setosa
>> les données :  [5.  3.4 1.5 0.2]  > la classe :  setosa
>> les données :  [4.4 2.9 1.4 0.2]  > la classe :  setosa
>> les données :  [4.9 3.1 1.5 0.1]  > la classe :  setosa
>> les données :  [5.4 3.7 1.5 0.2]  > la classe :  setosa
>> les données :  [4.8 3.4 1.6 0.2]  > la classe :  setosa
>> les données :  [4.8 3.  1.4 0.1]  > la classe :  setosa
>> les données :  [4.3 3.  1.1 0.1]  > la classe :  setosa
>> les données :  [5.8 4.  1.2 0.2]  > la classe :  setosa
>> les données :  [5.7 4.4 1.5 0.4]  > la classe :  setosa
>> les données :  [5.4 3.9 1.3 0.4]  > la classe :  setosa
>> les données :  [5.1 3.5 1.4 0.3]  > la classe :  setosa
>> les données :  [5.7 3.8 1.7 0.3]  > la classe :  setosa
>> les données :  [5.1 3.8 1.5 0.3]  > la classe :  setosa
>> les données :  [5.4 3.4 1.7 0.2]  > la classe :  setosa
>> les données :  [5.1 3.7 1.5 0.4]  > la classe :  setosa
>> les données :  [4.6 3.6 1.  0.2]  > la classe :  setosa
>> les données :  [5.1 3.3 1.7 0.5]  > la classe :  setosa
>> les données :  [4.8 3.4 1.9 0.2]  > la classe :  setosa
>> les données :  [5.  3.  1.6 0.2]  > la classe :  setosa
>> les données :  [5.  3.4 1.6 0.4]  > la classe :  setosa
>> les données :  [5.2 3.5 1.5 0.2]  > la classe :  setosa
>> les données :  [5.2 3.4 1.4 0.2]  > la classe :  setosa
>> les données :  [4.7 3.2 1.6 0.2]  > la classe :  setosa
>> les données :  [4.8 3.1 1.6 0.2]  > la classe :  setosa
>> les données :  [5.4 3.4 1.5 0.4]  > la classe :  setosa
>> les données :  [5.2 4.1 1.5 0.1]  > la classe :  setosa
>> les données :  [5.5 4.2 1.4 0.2]  > la classe :  setosa
>> les données :  [4.9 3.1 1.5 0.2]  > la classe :  setosa
>> les données :  [5.  3.2 1.2 0.2]  > la classe :  setosa
>> les données :  [5.5 3.5 1.3 0.2]  > la classe :  setosa
>> les données :  [4.9 3.6 1.4 0.1]  > la classe :  setosa
>> les données :  [4.4 3.  1.3 0.2]  > la classe :  setosa
>> les données :  [5.1 3.4 1.5 0.2]  > la classe :  setosa
>> les données :  [5.  3.5 1.3 0.3]  > la classe :  setosa
>> les données :  [4.5 2.3 1.3 0.3]  > la classe :  setosa
>> les données :  [4.4 3.2 1.3 0.2]  > la classe :  setosa
>> les données :  [5.  3.5 1.6 0.6]  > la classe :  setosa
>> les données :  [5.1 3.8 1.9 0.4]  > la classe :  setosa
>> les données :  [4.8 3.  1.4 0.3]  > la classe :  setosa
>> les données :  [5.1 3.8 1.6 0.2]  > la classe :  setosa
>> les données :  [4.6 3.2 1.4 0.2]  > la classe :  setosa
>> les données :  [5.3 3.7 1.5 0.2]  > la classe :  setosa
>> les données :  [5.  3.3 1.4 0.2]  > la classe :  setosa
>> les données :  [7.  3.2 4.7 1.4]  > la classe :  versicolor
>> les données :  [6.4 3.2 4.5 1.5]  > la classe :  versicolor
>> les données :  [6.9 3.1 4.9 1.5]  > la classe :  versicolor
>> les données :  [5.5 2.3 4.  1.3]  > la classe :  versicolor
>> les données :  [6.5 2.8 4.6 1.5]  > la classe :  versicolor
>> les données :  [5.7 2.8 4.5 1.3]  > la classe :  versicolor
>> les données :  [6.3 3.3 4.7 1.6]  > la classe :  versicolor
>> les données :  [4.9 2.4 3.3 1.  ]  > la classe :  versicolor
>> les données :  [6.6 2.9 4.6 1.3]  > la classe :  versicolor
>> les données :  [5.2 2.7 3.9 1.4]  > la classe :  versicolor
>> les données :  [5.  2.  3.5 1.  ]  > la classe :  versicolor
>> les données :  [5.9 3.  4.2 1.5]  > la classe :  versicolor
>> les données :  [6.  2.2 4.  1.  ]  > la classe :  versicolor
>> les données :  [6.1 2.9 4.7 1.4]  > la classe :  versicolor
>> les données :  [5.6 2.9 3.6 1.3]  > la classe :  versicolor
>> les données :  [6.7 3.1 4.4 1.4]  > la classe :  versicolor
>> les données :  [5.6 3.  4.5 1.5]  > la classe :  versicolor
>> les données :  [5.8 2.7 4.1 1.  ]  > la classe :  versicolor
>> les données :  [6.2 2.2 4.5 1.5]  > la classe :  versicolor
>> les données :  [5.6 2.5 3.9 1.1]  > la classe :  versicolor
>> les données :  [5.9 3.2 4.8 1.8]  > la classe :  versicolor
>> les données :  [6.1 2.8 4.  1.3]  > la classe :  versicolor
>> les données :  [6.3 2.5 4.9 1.5]  > la classe :  versicolor
>> les données :  [6.1 2.8 4.7 1.2]  > la classe :  versicolor
>> les données :  [6.4 2.9 4.3 1.3]  > la classe :  versicolor
>> les données :  [6.6 3.  4.4 1.4]  > la classe :  versicolor
>> les données :  [6.8 2.8 4.8 1.4]  > la classe :  versicolor
>> les données :  [6.7 3.  5.  1.7]  > la classe :  versicolor
>> les données :  [6.  2.9 4.5 1.5]  > la classe :  versicolor
>> les données :  [5.7 2.6 3.5 1.  ]  > la classe :  versicolor
>> les données :  [5.5 2.4 3.8 1.1]  > la classe :  versicolor
>> les données :  [5.5 2.4 3.7 1.  ]  > la classe :  versicolor
>> les données :  [5.8 2.7 3.9 1.2]  > la classe :  versicolor
>> les données :  [6.  2.7 5.1 1.6]  > la classe :  versicolor
>> les données :  [5.4 3.  4.5 1.5]  > la classe :  versicolor
>> les données :  [6.  3.4 4.5 1.6]  > la classe :  versicolor
>> les données :  [6.7 3.1 4.7 1.5]  > la classe :  versicolor
>> les données :  [6.3 2.3 4.4 1.3]  > la classe :  versicolor
>> les données :  [5.6 3.  4.1 1.3]  > la classe :  versicolor
>> les données :  [5.5 2.5 4.  1.3]  > la classe :  versicolor
>> les données :  [5.5 2.6 4.4 1.2]  > la classe :  versicolor
>> les données :  [6.1 3.  4.6 1.4]  > la classe :  versicolor
>> les données :  [5.8 2.6 4.  1.2]  > la classe :  versicolor
>> les données :  [5.  2.3 3.3 1.  ]  > la classe :  versicolor
>> les données :  [5.6 2.7 4.2 1.3]  > la classe :  versicolor
>> les données :  [5.7 3.  4.2 1.2]  > la classe :  versicolor
>> les données :  [5.7 2.9 4.2 1.3]  > la classe :  versicolor
>> les données :  [6.2 2.9 4.3 1.3]  > la classe :  versicolor
>> les données :  [5.1 2.5 3.  1.1]  > la classe :  versicolor
>> les données :  [5.7 2.8 4.1 1.3]  > la classe :  versicolor
>> les données :  [6.3 3.3 6.  2.5]  > la classe :  virginica
>> les données :  [5.8 2.7 5.1 1.9]  > la classe :  virginica
>> les données :  [7.1 3.  5.9 2.1]  > la classe :  virginica
>> les données :  [6.3 2.9 5.6 1.8]  > la classe :  virginica
>> les données :  [6.5 3.  5.8 2.2]  > la classe :  virginica
>> les données :  [7.6 3.  6.6 2.1]  > la classe :  virginica
>> les données :  [4.9 2.5 4.5 1.7]  > la classe :  virginica
>> les données :  [7.3 2.9 6.3 1.8]  > la classe :  virginica
>> les données :  [6.7 2.5 5.8 1.8]  > la classe :  virginica
>> les données :  [7.2 3.6 6.1 2.5]  > la classe :  virginica
>> les données :  [6.5 3.2 5.1 2.  ]  > la classe :  virginica
>> les données :  [6.4 2.7 5.3 1.9]  > la classe :  virginica
>> les données :  [6.8 3.  5.5 2.1]  > la classe :  virginica
>> les données :  [5.7 2.5 5.  2.  ]  > la classe :  virginica
>> les données :  [5.8 2.8 5.1 2.4]  > la classe :  virginica
>> les données :  [6.4 3.2 5.3 2.3]  > la classe :  virginica
>> les données :  [6.5 3.  5.5 1.8]  > la classe :  virginica
>> les données :  [7.7 3.8 6.7 2.2]  > la classe :  virginica
>> les données :  [7.7 2.6 6.9 2.3]  > la classe :  virginica
>> les données :  [6.  2.2 5.  1.5]  > la classe :  virginica
>> les données :  [6.9 3.2 5.7 2.3]  > la classe :  virginica
>> les données :  [5.6 2.8 4.9 2.  ]  > la classe :  virginica
>> les données :  [7.7 2.8 6.7 2.  ]  > la classe :  virginica
>> les données :  [6.3 2.7 4.9 1.8]  > la classe :  virginica
>> les données :  [6.7 3.3 5.7 2.1]  > la classe :  virginica
>> les données :  [7.2 3.2 6.  1.8]  > la classe :  virginica
>> les données :  [6.2 2.8 4.8 1.8]  > la classe :  virginica
>> les données :  [6.1 3.  4.9 1.8]  > la classe :  virginica
>> les données :  [6.4 2.8 5.6 2.1]  > la classe :  virginica
>> les données :  [7.2 3.  5.8 1.6]  > la classe :  virginica
>> les données :  [7.4 2.8 6.1 1.9]  > la classe :  virginica
>> les données :  [7.9 3.8 6.4 2.  ]  > la classe :  virginica
>> les données :  [6.4 2.8 5.6 2.2]  > la classe :  virginica
>> les données :  [6.3 2.8 5.1 1.5]  > la classe :  virginica
>> les données :  [6.1 2.6 5.6 1.4]  > la classe :  virginica
>> les données :  [7.7 3.  6.1 2.3]  > la classe :  virginica
>> les données :  [6.3 3.4 5.6 2.4]  > la classe :  virginica
>> les données :  [6.4 3.1 5.5 1.8]  > la classe :  virginica
>> les données :  [6.  3.  4.8 1.8]  > la classe :  virginica
>> les données :  [6.9 3.1 5.4 2.1]  > la classe :  virginica
>> les données :  [6.7 3.1 5.6 2.4]  > la classe :  virginica
>> les données :  [6.9 3.1 5.1 2.3]  > la classe :  virginica
>> les données :  [5.8 2.7 5.1 1.9]  > la classe :  virginica
>> les données :  [6.8 3.2 5.9 2.3]  > la classe :  virginica
>> les données :  [6.7 3.3 5.7 2.5]  > la classe :  virginica
>> les données :  [6.7 3.  5.2 2.3]  > la classe :  virginica
>> les données :  [6.3 2.5 5.  1.9]  > la classe :  virginica
>> les données :  [6.5 3.  5.2 2.  ]  > la classe :  virginica
>> les données :  [6.2 3.4 5.4 2.3]  > la classe :  virginica
>> les données :  [5.9 3.  5.1 1.8]  > la classe :  virginica
```

4- Affichez la moyenne (mean), l'ecart-type (std), le min et le max pour chaque variable.

```
print(" la moyenne (la moyenne de chaque variable) : ", iris.data.mean(0))
print(" l'ecart-type (std) : ", iris.data.std(0))
print(" min : ", iris.data.min(0))
print(" max : ", iris.data.max(0))

 la moyenne (la moyenne de chaque variable) :  [5.84333333 3.05733333 3.758    1.19933333]
 l'ecart-type (std) :  [0.82530129 0.43441097 1.75940407 0.75969263]
 min :  [4.3 2.  1.  0.1]
 max :  [7.9 4.4 6.9 2.5]
```

5- En utilisant les attributs size et shape, affichez le nombre de données, le nombre de variables et le nombre de classes.

```
print('le nombre de données : ', iris.data.shape[0])
print('le nombre de variables : ', iris.data.shape[1])
print('le nombre de classes : ', iris.target_names.size)

le nombre de données :  150
le nombre de variables :  4
le nombre de classes :  3
```

C. Téléchargement et importation de données

1- Importez les données 'MNIST original'.

```
# mnist = datasets.fetch_mldata('MNIST original')

mnist = datasets.fetch_openml('mnist_784', version=1, cache=True)
print('fin de téléchargement')

fin de téléchargement
```

2- Affichez la matrice des données, le nombre de données et de variables, les numéros de classes pour chaque donnée, ainsi que la moyenne, l'écart type, les valeurs min et max pour chaque variable ; enfin affichez le nombre d'images avec la fonction unique.

```
print('le nombre de données : ', mnist.data.shape[0])
print('le nombre de variables : ', mnist.data.shape[1])
print('la matrice des données :\n', mnist.data)

le nombre de données :  70000
le nombre de variables :  784
la matrice des données :
 [[0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

```
print('Il le nombre de 5 premiers classes pour chaque donnée :')
for i, target in enumerate(mnist.target[:5]):
    print('> la donnée à l\'index', i, '> sa classe est : ', target)

# la nombre de 5 premiers classes pour chaque donnée :
> la donnée à l'index 0 > sa classe est :  5
> la donnée à l'index 1 > sa classe est :  0
> la donnée à l'index 2 > sa classe est :  4
> la donnée à l'index 3 > sa classe est :  1
> la donnée à l'index 4 > sa classe est :  9
```

l'affichage de moyenne (mean), l'ecart-type (std), le min et le max pour chaque variable.

```
features_mnist = mnist.data
targets_mnist = mnist.target

mean_mnist = features_mnist.mean(0)
std_mnist = features_mnist.std(0)
min_mnist = features_mnist.min(0)
max_mnist = features_mnist.max(0)
print(' la moyenne pour chaque variable \t: ',mean_mnist[400:465])
print(' l\'ecart-type pour chaque variable \t: ',std_mnist[400:465])
print(' le min pour chaque variable \t\t: ',min_mnist[400:465])
print(' le max pour chaque variable \t\t: ',max_mnist[400:465])

 la moyenne pour chaque variable  :  [ 16.16153439 45.4515     75.59607143 159.12380952 166.38431633
  194.37278571  81.99058052  67.13854592  32.44126013  39.4835667 ]
 l'ecart-type pour chaque variable  :  [ 31.  0.  0.  0.  0.
  255. 255. 255. 255. 255.]
 le min pour chaque variable  :  [0. 0. 0. 0. 0.
 le max pour chaque variable  :  [255. 255. 255. 255. 255.]
```

3- Le nombre de classe avec la fonction unique.

```
classes_num = np.unique(targets_mnist)
print('les classes des données mnist : ',classes_num)
print('le nombre de classes est : ',classes_num.size)

les classes des données mnist :  ['0' '1' '2' '3' '4' '5' '6' '7' '8' '9']
le nombre de classes est :  10
```

D. Génération de données et affichage

1- Utiliser l'aide (help) pour voir comment utiliser la fonction datasets.make_blobs.

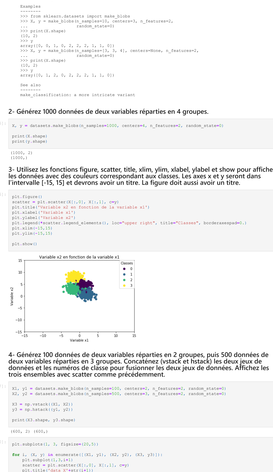
```
help(datasets.make_blobs)

Help on function make_blobs in module sklearn.datasets._samples_generator:

make_blobs(n_samples=100, n_features=2, *, centers=None, cluster_std=1.0, center_box=(-10.0, 10.0), shuffle=True, random_state=None, return_centers=False)
    Generate isotropic Gaussian blobs for clustering.

    Read more in the :ref:`User Guide <sample_generators>`.

    Parameters
    ----------
    n_samples : int or array-like, optional (default=100)
        If int, it is the total number of points equally divided among
        clusters.
        If array-like, each element of the sequence indicates
        the number of samples per cluster.

        .. versionchanged:: v0.20
            one can now pass an array-like to the ``n_samples`` parameter

    n_features : int, optional (default=2)
        The number of features for each sample.

    centers : int or array of shape [n_centers, n_features], optional
            (default=None)
        The number of centers to generate, or the fixed center locations.
        If n_samples is an int and centers is None, 3 centers are generated.
        If n_samples is array-like, centers must be
        either None or an array of length equal to the length of n_samples.

    cluster_std : float or sequence of floats, optional (default=1.0)
        The standard deviation of the clusters.

    center_box : pair of floats (min, max), optional (default=(-10.0, 10.0))
        The bounding box for each cluster center when centers are
        generated at random.

    shuffle : boolean, optional (default=True)
        Shuffle the samples.

    random_state : int, RandomState instance, default=None
        Determines random number generation for dataset creation. Pass an int
        for reproducible output across multiple function calls.
        See :term:`Glossary <random_state>`.

    return_centers : bool, optional (default=False)
        If True, then return the centers of each cluster

        .. versionadded:: 0.23

    Returns
    -------
    X : array of shape [n_samples, n_features]
        The generated samples.

    y : array of shape [n_samples]
        The integer labels for cluster membership of each sample.

    centers : array, shape [n_centers, n_features]
        The centers of each cluster. Only returned if
        ``return_centers=True``.

    Examples
    --------
    >>> from sklearn.datasets import make_blobs
    >>> X, y = make_blobs(n_samples=10, centers=3, n_features=2,
    ...                   random_state=0)
    >>> print(X.shape)
    (10, 2)
    >>> y
    array([0, 0, 1, 0, 2, 2, 2, 1, 1, 0])
    >>> X, y = make_blobs(n_samples=3, 5, 6), centers=None, n_features=2,
    ...                   random_state=0)
    >>> print(X.shape)
    (14, 2)
    >>> y
    array([5, 1, 5, 0, 2, 2, 2, 1, 1, 0])

    See also
    --------
    make_classification: a more intricate variant
```

2- Générez 1000 données de deux variables réparties en 4 groupes.

```
X, y = datasets.make_blobs(n_samples=1000, centers=4, n_features=2, random_state=0)

print(X.shape)
print(y.shape)

(1000, 2)
(1000,)
```

3- Utilisez les fonctions figure, scatter, title, xlim, ylim, xlabel, ylabel et show pour afficher les données avec des couleurs correspondant aux classes. Les axes x et y seront dans l'intervalle [-15, 15] et certaines avoir un titre. La figure doit aussi avoir un titre.

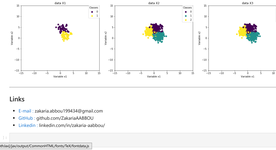
```
plt.figure()
scatter = plt.scatter(X[:,0], X[:,1], c=y)
plt.title("Variable x2 en fonction de la variable x1")
plt.xlabel("Variable x1")
plt.ylabel("Variable x2")
plt.legend(*scatter.legend_elements(), loc="upper right", title="Classes", borderaxespad=1.)
plt.xlim(-15,15)
plt.ylim(-15,15)

plt.show()
```



4- Générez 100 données de deux variables réparties en 2 groupes, puis 500 données de deux variables réparties en 3 groupes. Concaténez (vstack et hstack) les deux jeux de données et les matrices de classe pour fusionner les deux jeux de données. Affichez les trois ensembles avec scatter comme précédemment.

```
X1, y1 = datasets.make_blobs(n_samples=50, centers=2, n_features=2, random_state=0)
X2, y2 = datasets.make_blobs(n_samples=150, centers=3, n_features=2, random_state=0)

X3 = np.vstack((X1, X2))
y3 = np.hstack((y1, y2))

print(X3.shape, y3.shape)

(650, 2) (650,)
```

```
plt.subplots(1, 3, figsize=(20,5))

for i, (x, y) in enumerate([(X1, y1), (X2, y2), (X3, y3)]):
    plt.subplot(1,3,i+1)
    scatter = plt.scatter(X[:,0], X[:,1], c=y)
    plt.title("Data "+str(i+1))
    plt.xlabel("Variable x1")
    plt.ylabel("Variable x2")
    plt.legend(*scatter.legend_elements(), loc="upper right", title="Classes", borderaxespad=1.)
    plt.xlim(-15,15)
    plt.ylim(-15,15)

plt.show()
```



Links

- E-mail : zakaria.abbou1994@gmail.com
- GitHub : github.com/ZakariaABBOU
- LinkedIn : linkedin.com/in/zakaria-aabbou/