TP1

Introduction au module scikit-learn

A. Importation des libraires

1 -

from sklearn import \*

2 -

import numpy as np

import matplotlib.pyplot as plt

Manipulation d’un jeu de données

1-

from sklearn import datasets

iris = datasets.load\_iris()

2-

print(iris.data)

print(iris.target\_names)

print(iris.target)

3-

a = []

for b,c in zip(iris.data, iris.target\_names[iris.target]):

a.append(np.append(b,c))

a = np.array(a)

print(a)

4-

print(iris.data.mean(0))

print(iris.data.std(0))

print(iris.data.min(0))

print(iris.data.max(0))

5-

print(iris.data.size)

print(iris.data.shape[1])

print(iris.target\_names.size)

Téléchargement et importation de données

1-

mnist = datasets.fetch\_mldata('MNIST original')

2-

print(mnist.data)

print(mnist.data.size)

print(mnist.data.shape[0])

print(mnist.data.mean(0))

print(mnist.data.std(0))

print(mnist.data.min(0))

print(mnist.data.max(0))

print(np.unique(mnist.target))

Génération de données et affichage

1-

print(help(datasets.make\_blobs))

2-

data, label = datasets.make\_blobs(n\_samples=500, n\_features=2, centers=4)

3-

plt.figure()

plt.xlim(-15, 15)

plt.ylim(-15, 15)

plt.scatter(x=data[:,0], y=data[:,1],c=label)

plt.title("1000 donnees de deux variables\nreparties en 4 groupes")

plt.xlabel("valeur de x")

plt.ylabel("valeur de y")

plt.legend()

plt.show()

4-

a, b = datasets.make\_blobs(n\_samples=50, n\_features=2, centers=2)

c, d = datasets.make\_blobs(n\_samples=250, n\_features=2, centers=3)

plt.figure()

plt.xlim(-15, 15)

plt.ylim(-15, 15)

plt.scatter(x=a[:,0], y=a[:,1], c=b)

plt.title("100 donnees de deux variables\nreparties en 2 groupes")

plt.xlabel("valeur de x")

plt.ylabel("valeur de y")

plt.legend()

plt.show()

plt.figure()

plt.xlim(-15, 15)

plt.ylim(-15, 15)

plt.scatter(x=c[:,0], y=c[:,1], c=d)

plt.title("500 donnees de deux variables\nreparties en 3 groupes")

plt.xlabel("valeur de x")

plt.ylabel("valeur de y")

plt.legend()

plt.show()

lastData, lastLabel = np.vstack((a, c)), np.hstack((b, d))

print a.shape

plt.figure()

plt.xlim(-15, 15)

plt.ylim(-15, 15)

plt.scatter(x=lastData[:,0], y=lastData[:,1], c=lastLabel)

plt.title("Concatenation des donnees")

plt.xlabel("valeur de x")

plt.ylabel("valeur de y")

plt.legend()

plt.show()

TP2

Prétraitement et visualisation de données

A. Normalisation de données

1-

2-

3-

4-

B. Normalisation MinMax

1-

2-

3-

C. Visualisation de données

1-

iris = datasets.load\_iris()

2-

m = 0

plt.figure(1)

for ligne in range(0, len(iris.feature\_names)):

for colonne in range(0, len(iris.feature\_names)):

if ligne != colonne and ligne <= colonne:

m = m + 1

plt.subplot(2, 3, m)

plt.scatter(x=iris.data[:,ligne], y=iris.data[:,colonne], c=iris.target, alpha=0.8)

plt.title("Figure : Ligne "+ str(ligne) + " Colonne " + str(colonne))

plt.show()

D. Réduction de dimensions et visualisation de données

1-

from sklearn.decomposition import PCA

from sklearn.lda import LDA

2-

irisPCA = PCA(n\_components=2).fit(iris.data).transform(iris.data)

irisLDA = LDA().fit(iris.data, iris.target).transform(iris.data)

3-

plt.figure("Figure PCA/LDA de iris data")

plt.subplot(1, 3, 1)

plt.title("Iris PCA")

plt.scatter(irisPCA[:,0], irisPCA[:,1], c=iris.target, alpha=0.8)

plt.legend()

plt.subplot(1, 3, 2)

plt.scatter(irisLDA[:,0], irisLDA[:,1], c=iris.target, alpha=0.8)

plt.subplot(1, 3, 3)

plt.title("Iris LDA")

plt.scatter(irisLDA[:,0], irisLDA[:,1], c=LDA().fit(iris.data, iris.target).predict(iris.data), alpha=0.8)

plt.legend()

plt.show()

print("Le taux de réussite est de " + str(sum(LDA().fit(iris.data, iris.target).predict(iris.data) == iris.target) / (iris.data.shape[0] + 0.0)))