



Master Informatique EID2

Traitement numérique des données

TP3 Introduction à la classification

Mohamed Ben Saad Elias ABDELLI

```
1 import numpy as np
 2 import matplotlib.pyplot as plt
 3 from sklearn import *
 4 from sklearn.preprocessing import *
 5 from sklearn.neighbors import KNeighborsClassifier
 6 from sklearn.metrics.pairwise import euclidean_distances
 1 iris = datasets.load_iris()
 2 X = iris.data
 3 Y = iris.target
 2 # Plus Proche Voisin #
 6 ## 1. Créez une fonction PPV(X,Y) ##
 7 #-----#
 8
9 def PPV(X,Y):
10 w = metrics.pairwise.euclidean_distances(X, X)
14 return pred
 2 ## 2- Modifiez la fonction pour calculer et renvoyer l'erreur de prédiction ##
 3 #-----
 5 def erreur_PPV(Y1,Y2):
 6 dif = Y1 == Y2
   pct = 0.0
 7
   n = len(dif)
for i in range(n):
 8
 9
    if dif[i] == 0 :
10
        pct = pct + 1
11
12 pct = pct/n*100.0
13 return pct
1 #-----#
2 ## 3. Testez sur les données Iris ##
3 #-----#
5 P = PPV(X,Y)
6 print('le plus proche voisin :\n', PPV(X,Y))
7 print('Erreur de :',erreur_PPV(Y,P))
le plus proche voisin :
2 2]
Erreur de : 4.0
2 ## 4. Testez la fonction des K Plus Proches Voisins de sklearn (avec ici K = 1) ##
5 def kPPV(X,Y,k):
6 neigh = KNeighborsClassifier(n_neighbors=k)
 neigh.fit(X, Y)
7
8 return neigh.predict(X)
```

```
2 # ===== Les résultats sont-ils différents ? ======= #
3 # En effet on a 0% d'érreur avec la KPPV de sklearn #
5 print(kPPV(X,Y,1))
6 print(erreur_PPV(Y,kPPV(X,Y,1)))
2 2]
0.0
2 # ==== Testez avec d'autres valeurs de K ====#
3 #-----#
4
5 print("k=6\n",kPPV(X,Y,6))
6 print(erreur_PPV(kPPV(X,Y,6),Y))
8 print("k=10\n", kPPV(X,Y,10))
9 print(erreur_PPV(kPPV(X,Y,10),Y))
10
11 print("k=25\n", kPPV(X,Y,25))
12 print(erreur_PPV(kPPV(X,Y,25),Y))
13
14 print("k=50\n",kPPV(X,Y,50))
15 print(erreur PPV(kPPV(X,Y,50),Y))
k=6
2 2]
2.666666666666667
k=10
2 2]
2 0
k=25
2 2]
2.0
k=50
2 2]
6.0
```

```
2 ## 5. BONUS ##
3 #----#
5 #======#
6 # Fonction pour trouver le plus fréquent #
7 #=======#
9 def most_frequent(List):
10
  dict = {}
  count, itm = 0, ''
11
12
   for item in reversed(List):
13
    dict[item] = dict.get(item, 0) + 1
14
     if dict[item] >= count :
15
        count, itm = dict[item], item
   return(itm)
16
17
18 def PPVK(X,Y,k) :
19 w = metrics.pairwise.euclidean_distances(X, X)
20
   t = np.argsort(w)
   pred = np.zeros((X.shape[0]),int)
21
22
   cmin = np.array(t[:,1:k+1])
23
   p = Y[cmin]
  for i in range(X.shape[0]):
24
25
    pred[i] = int(most_frequent(p[i]))
26
   return pred
27
28 print(PPVK(X,Y,6))
29 print('Erreur de :',erreur_PPV(PPVK(X,Y,6),Y))
2 2]
Erreur de : 4.0
```

```
2 # Classifieur Bayesien Naïf #
 6 ## 1. Créez une fonction CBN(X,Y) ##
9 def prob(X, m, s):
10 a = (X - m)**2
   b = 2 * s**2
11
12 ex = np.exp(-(a / b))
13
    res = np.log(ex / (np.sqrt(2 * np.pi)*s))
14
    return res
15
16 def pred(X,md):
     p = [[sum(prob(i,*j) for j, i in zip(sums,x)) for sums in md] for x in X]
17
18
     return p
19
20 def CBN(X,Y):
y = np.unique(Y)
     sep = [[x for x, t in zip(X, Y) if t == c] for c in y]
23
     md = [np.c_[np.mean(i, axis=0), np.std(i, axis=0)] for i in sep]
    mdl = np.array(md)
24
cbn = np.argmax(pred(X,mdl), axis=1)
26 return cbn
```

```
1 #-----#
2 ## 2. Testez sur les données Iris et Fonction erreur ##
5 \text{ irisCB} = CBN(X,Y)
6 print(irisCB)
7 print('Erreur de :',erreur_PPV(irisCB,Y))
2 2]
Erreur de : 4.0
2 ## 3. Testez la fonction du Classifieur Bayesien Naïf inclut dans sklearn ##
4
5 irisCBN = naive_bayes.GaussianNB().fit(X,Y).predict(X)
6 print(irisCBN)
7 print('Erreur de :',erreur_PPV(irisCBN,Y))
2 2]
```

Erreur de : 4.0