TD Informatique TSI2

Chap 5. Machine_Learning_Part2

EXERCICE 01

Parmi les assertions suivantes, lesquelles sont vraies ?

A. Il y a 2^n partitions possibles d'un ensemble à n éléments

B. Dans un apprentissage supervisé, on dispose d'un ensemble de données classées.

C. L'ensemble d'apprentissage utilisé pour calculer la matrice de confusion est un ensemble de données classées.

D. Pour calculer la matrice de confusion, on utilise un ensemble test de données non classées.

E. Les coefficients `m[i][i]` d'une matrice de confusion `m` donnent les seules prédictions correctes.

F. Un faux positif est un positif qui a été prédit négatif.

EXERCICE 02

1. L'idée ici est de faire un dessin qui ressemble à la figure créé quand on a parlé de matrice de confusion. On avait eu un nuage de points avec deux niveaux de gris. On désire plus précisément dans cet exercice avoir un nuage de points avec trois niveaux de gris.

   Taper `from random import randint`

   Tapez ensuite la procédure `creer_points` d'arguments `nb`, `dim` et `coul` qui renvoie maintenant une liste de triplets (x, y, c) avec `c = randint(0,2)` et donc `c` peut prendre trois valeurs au lieu de deux.

   Appliquer enfin :

   `couleurs = ["0.01", "0.5", "0.8"]` et `creer_points(50,40,couleurs)`

2. On crée un point `P` comme dans le cours et on veut tracer le nuage de points où apparaîtra `P`.

   On tapera donc :

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

In [2]: points = creer_points(50,40,couleurs)
In [3]: P = (randint(0,40), randint(0,40),"k")
In [4]: x = [p[0] for p in points]
In [5]: y = [p[1] for p in points]
In [6]: c = [p[2] for p in points]

In [7]: plt.scatter(x,y, linestyle='None', color=c, marker="o");
plt.plot(P[0],P[1], P[2] + "X");
plt.text(P[0]+0.4,P[1],"P"); plt.show()
```

   Le `"X"` permet de visualiser une croix au niveau de `"P"`

3. On va utiliser l'algorithme des k plus proches voisins afin d'affecter à `P` la couleur dominante parmi ses $k = 4$ voisins les plus proches. On cherche donc à prédire le niveau de gris de `P`.

   (a) Construire une procédure `distance(p1,p2)` qui renvoie le carré de la distance des points `(x1,y1,c1) = p1` et de `(x2,y2,c2)= p2`

   (b) Construire une procédure `tri(E,P,d)` et `knn(E,p,d,k)` comme dans le cours.
   On pose `k = 4` et `vs = knn(points,P,distance,k)`. Taper ces codes et vérifier que `vs` donne une liste cohérente.

T.S.V.P →

(c) Taper ensuite la fonction `couleur_maj` d'arguments `vs` et `coul` qui détermine la couleur majoritaire parmi les couleurs des voisins ou plutôt l'indice de cette couleur majoritaire

```
In [1]: def couleur_maj(vs,coul):
            cpt = [0] * len(coul)
            for v in vs :
                for i in range(len(coul)):
                    if v[2] == coul[i] :
                        cpt[i] += 1
            ind_maxi = 0
            for i in range(len(cpt)):
                if cpt[i] > cpt[ind_maxi]:
                    ind_maxi = i
            return ind_maxi
```

Tapez alors `couleurs[couleur_maj(vs,couleurs)]`