Exercice 5
Classification et prédiction

Distances

Distance de Manhattan :	$d(A , B) = \sum_{i=1}^{n} \lvert A_i - B_i \rvert$
Distance euclidienne :	$d(A , B) = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$
Distance de Minkowski :	$d(A , B) = (\sum_{i=1}^{n} \lvert A_i - B_i \rvert^p)^{1/p}$
Distance de cosine :	$d(A , B) = 1 - \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} \sqrt{\sum_{i=1}^{n} {B_i}^2}}$
Distance de Hamming :	$d(A , B) = \#\{ i : A_i \neq B_i \}$

Algorithme

Algorithme : k-NN

Entrées : *D* : Dataset ; *k* : nombre de voisins à considérer ; *Inst* : l'instance à classifier ;
Sorties : y: la classe de l'instance à classifier
Var
dist : tableau de [1..N] de paire (instance , distance) ; //avec N la taille de D = |D|
knn : tableau de [1..k] d'instances ;
Début
Pour chaque instance X de D **faire**
dist[X] ←Calculer la distance entre *X* et *Inst* ;
Fait;
dist ←Trier dist dans l'ordre croissant des distances;
knn ←les *k* premières instances de *dist* ;
y ←La classe dominante dans *knn* ;
Retourner y;
Fin.

Questions :

1- Écrire une fonction Python permettant de calculer la distance entre deux instances du dataset (implémenter les distances applicables parmi ceux lister).
2- Écrire une fonction Python permettant de trier les instances du dataset selon la valeur d'une distance calculée "d".
3- Écrire une fonction Python permettant de retourner la classe dominante parmi un ensemble de K classes.
4- Implémenter en Python l'algorithme KNN ayant comme paramètres : k, la métrique de distance à employé.
5- Déduire la classe de l'instance <5.2, 3.5 , 1.41, 0.25> avec K = 3 puis avec K = 5.

BELKADI Widad Hassina (G1-G2): belkadi.wh.usthb@gmail.com
HOUACINE Naila Aziza (G3-G4) : n.a.houacine@gmail.com