Classification des iris par kNN

3 variétés d'Iris

Dans ce TP, l'objectif est se familiariser avec les objets/algorithmes décrits dans le cours sur l'apprentissage automatique, plus précisément ceux liés à l'apprentissage supervisé via l'**algorithme des k plus proches voisins**.
Vous pouvez ouvrir le fichier nommé `tp_iris_kNN_trame.py`, qui sera complété ensuite.

1 Iris : Pétales

Question 1

Valider la partie 1 du script et décrire ce que contient `iris`.

Question 2

Valider la partie 2 : à quoi correspondent `X,y` et `A` ?

Script 3

La fonction `montrer_iris` permet d'illustrer graphiquement notre jeu de données.
A vous de la comprendre et de la tester.

Script 4

La fonction `rechercher_plus_proches_voisins` permet de déterminer les indices des k plus proches voisins d'une entrée nommée `nouvelle`.
On ne demande pas de comprendre en détail son code, mais juste de la tester différentes valeurs de `k` (et la variable `nouvelle` imposée par l'énoncé, qui correspond à une nouvelle entrée non étiquetée).

Script 5

La fonction `classifier_plus_proches_voisins` permet de déterminer le numéro de la classe majoritaire parmi les k plus proches voisins de `nouvelle`.
Là aussi, on ne demande pas de comprendre le code mais plutôt d'utiliser cette fonction pour prédire l'étiquette de `nouvelle`.
On fera varier `k`.

Script 6

La fonction `partager_apprentissage_test` permet de séparer/partitionner aléatoirement un jeu de données en un jeu d'apprentissage et un jeu de test. La variable `ratio_test` vaut par défaut 0.4 : la taille du jeu de test correspond à 40% de celle du jeu de données initial (et donc celle du jeu d'apprentissage est de 60% de celle du jeu initial).
Cette fonction conserve la répartition entre classe (1/3 du jeu test est formé de *setosa*, 1/3 de *versicolor* et 1/3 de *virginica*, comme dans le jeu initial)

Valider cette fonction ainsi que son test.
Vérifier que le `ratio_test` est respecté puis illustrer graphiquement chacun des deux jeux.

Script 7

Écrire une fonction `tout_classifier_plus_proches_voisins(X_test, X_app, y_app, k_voisins=3)` qui, à partir de du jeu `(X_app,y_app)`, classifie par plus proches voisins toutes les données du jeu `X_test` et retourne la classe majoritaire de chaque élément de `X_test` sous forme de tableau à une dimension (i.e. d'une liste) nommée `y_pred`.
Illustrer la prédiction obtenue.

Script 8

Proposer plusieurs lignes de codes permettant de construire la matrice de confusion et afficher celle-ci.
Conclusion ?

Question 9

Afin d'évaluer notre algorithme, on souhaite mettre au point un taux de précision.
Comment feriez-vous pour le définir ?

Script 10

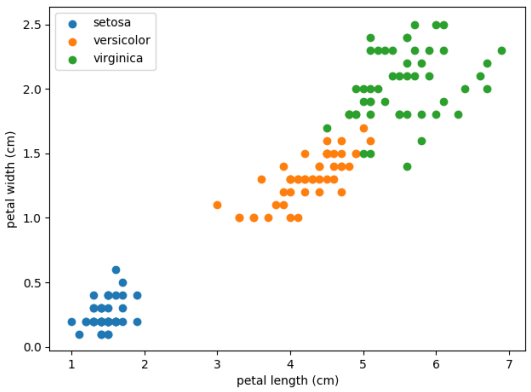
Calculer ce taux de précision et le faire varier en fonction de k.

2 Iris : Sépales

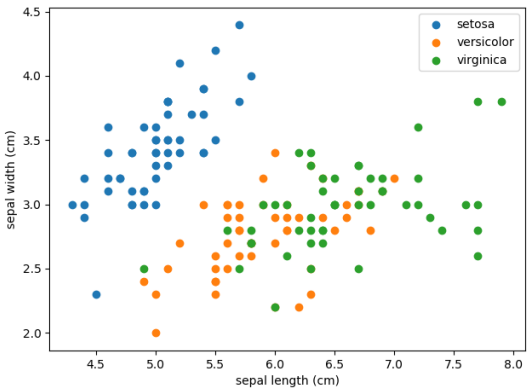
On souhaite désormais se focaliser sur de nouveaux descripteurs : la longueur et la largeur des sépales.

Script 11

Reprendre les Questions 1,2 et 3 en modifiant ce qui est nécessaire.
Reprendre maintenant le Script 10 : Conclusion ?



descripteurs : pétales



descripteurs : sépales