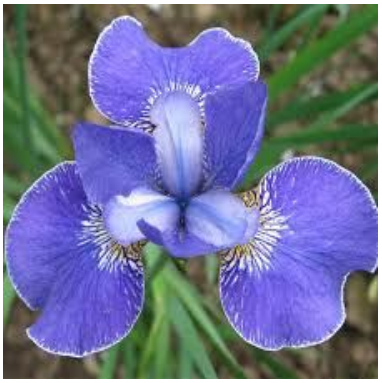
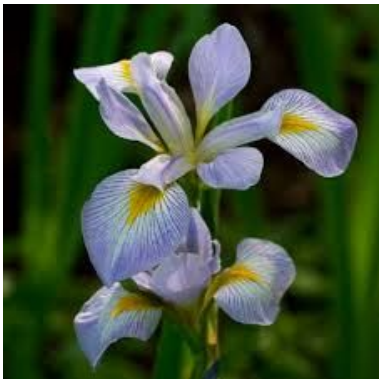
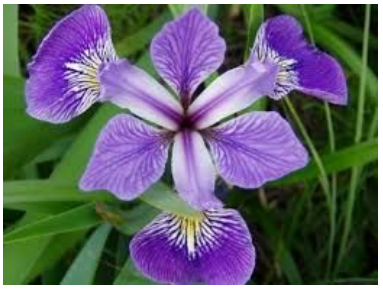
TP19 : Algorithme des “k plus proches voisins”

Nous allons travailler sur un algorithme d'apprentissage automatique (appelé même en français, algorithme de *machine learning*) celui des *k plus proches voisins* ou algorithme KNN.

L'idée est d'utiliser un grand nombre de données afin "d'apprendre à la machine" à résoudre un certain type de problème.

Afin de travailler sur un exemple, nous allons utiliser un jeu de données relativement connu dans le monde du *machine learning* : le jeu de données "iris".

En 1936, *Edgar Anderson,* un botaniste américain, a collecté des données sur 3 espèces d'iris : *iris setosa*, *iris virginica* et *iris versicolor* :

		
iris setosa	iris virginica	iris versicolor

Pour chaque iris mesuré, *Edgar Anderson* a aussi noté l'espèce ("iris setosa", "iris virginica" ou "iris versicolor").

Ouvrez le fichier *iris.csv*. Vous y trouverez 50 valeurs de ces mesures collectés par *Edgar Anderon*.

En résumé, vous trouverez dans ce fichier :

- la longueur des pétales
- la largeur des pétales
- l'espèce de l'iris

Au lieu d'utiliser les noms des espèces, on utilisera des chiffres :

- 0 pour "iris setosa"
- 1 pour "iris virginica"
- 2 pour "iris versicolor"

	A	B	C
1	petal_length	petal_width	species
2	1.4	0.2	0
3	1.4	0.2	0
4	1.3	0.2	0
5	1.5	0.2	0
6	1.4	0.2	0
7	1.7	0.4	0
8	1.4	0.3	0
9	1.5	0.2	0
10	1.4	0.2	0
11	1.5	0.1	0

Avant d'entrer dans le vif du sujet (algorithme knn), nous allons chercher à obtenir une représentation graphique des données contenues dans le fichier iris.csv

Exercice 1 : analyse d'un code

Après avoir placé le fichier `iris.csv` dans le même répertoire que votre fichier Python, analyser et tester le code suivant :

```
1  import pandas
2  import matplotlib.pyplot as plt
3  iris=pandas.read_csv("iris.csv")
4  x=iris.loc[:,"petal_length"]
5  y=iris.loc[:,"petal_width"]
6  lab=iris.loc[:,"species"]
7  plt.scatter(x[lab == 0], y[lab == 0], color='g', label='setosa')
8  plt.scatter(x[lab == 1], y[lab == 1], color='r', label='virginica')
9  plt.scatter(x[lab == 2], y[lab == 2], color='b', label='versicolor')
10 plt.legend()
11 plt.show()
```

1. À quoi sert la librairie `pandas` ligne 1 ?
2. À quoi sert la librairie `matplotlib` à la ligne 2 ?
3. À quoi sert le `as plt` dans l'import de la librairie à la ligne 2 ?
4. Que permet de faire la fonction `scatter` de la librairie `matplotlib` ? Faire une recherche sur Internet. Ne pas faire de copier/coller mais essayer d'être synthétique.
5. Que permet de faire `.loc` aux lignes 4 et 5 ?
6. Que permet de faire `plt.scatter` aux lignes 7, 8 et 9 ?
7. A quoi correspondent `x`, `y` et `lab` ?
8. A quoi servent les lignes 10 et 11 ? Essayer de les supprimer pour comprendre leur rôle.

La bibliothèque *Scikit Learn* de Python propose un grand nombre d'algorithmes liés au machine learning (c'est sans aucun doute la bibliothèque la plus utilisée en machine learning). Parmi tous ces algorithmes, *Scikit Learn* propose l'algorithme des k-plus proches voisins.

Exercice 2

Après avoir placé le fichier **`iris.csv`** dans le même répertoire que votre fichier Python, analyser et tester le code suivant :

```
1   import pandas
2   import matplotlib.pyplot as plt
3   from sklearn.neighbors import KNeighborsClassifier
4
5   #traitement CSV
6   iris=pandas.read_csv("iris.csv")
7   x=iris.loc[:,"petal_length"]
8   y=iris.loc[:,"petal_width"]
9   lab=iris.loc[:,"species"]
10  #fin traitement CSV
11
12  #valeurs
13  longueur=2.5
14  largeur=0.75
15  k=3
16  #fin valeurs
17
18  #graphique
19  plt.scatter(x[lab == 0], y[lab == 0], color='g', label='setosa')
20  plt.scatter(x[lab == 1], y[lab == 1], color='r', label='virginica')
21  plt.scatter(x[lab == 2], y[lab == 2], color='b', label='versicolor')
22  plt.scatter(longueur, largeur, color='k')
23  plt.legend()
24  #fin graphique
25
26  #algo knn
27  d=list(zip(x,y))
28  model = KNeighborsClassifier(n_neighbors=k)
29  model.fit(d,lab)
30  prediction= model.predict([[longueur,largeur]])
31  #fin algo knn
32
33  #Affichage résultats
34  txt="Résultat : "
35  if prediction[0]==0:
36    txt=txt+"setosa"
37  if prediction[0]==1:
38    txt=txt+"virginica"
39  if prediction[0]==2:
40    txt=txt+"versicolor"
41  plt.text(3,0.5,'largeur : {l} cm longueur : {L} cm'.format(l=largeur,L=longueur),
42  fontsize=12)
43  plt.text(3,0.3,'k = {kvoisins}'.format(kvoisins=k), fontsize=12)
44  plt.text(3,0.1, txt, fontsize=12)
45  #fin affichage résultats
46
47  plt.show()
```

1. Dans ce programme, à quelle ligne est importée la bibliothèque permettant d'appliquer l'algorithme des k-plus proches voisins ? Est-elle importée en totalité ?

La ligne 27 "d=list(zip(x,y))" permet de passer des 2 listes x et y :

```
x = [1.4, 1.4, 1.3, 1.5, 1.4, 1.7]
y = [0.2, 0.2, 0.2, 0.2, 0.2, 0.2]
```

à une liste de tuples d :

```
d = [(1.4, 0.2), (1.4, 0.2), (1.3, 0.2) (1.5, 0.2), (1.4, 0.2), (1.7, 0.2)]
```

La ligne 28 va chercher dans la bibliothèque *scikit-learn* le modèle d'algorithme d'apprentissage des k-plus proches voisins. En effet, dans cette librairie, il existe plusieurs modèles d'algorithme d'apprentissage (classification, régression …).

En ligne 29, la méthode **`fit`** permet d'appliquer le modèle à notre jeu de données.

2. Combien de voisins sont utilisés pour notre jeu de données dans ce programme ?

3. Que fait la ligne 30 ?

4. Expliquer en détail la ligne 41. Pour cela, faire une recherche sur la syntaxe de la fonction `matplotlib.pyplot.text(x, y, s,...)`.

Exercice 3

Modifier le programme de l'exercice 2 afin de tester l'algorithme KNN avec un nombre "de plus proches voisins" différent (en gardant un iris ayant une longueur de pétale égale à 2,5 cm et une largeur de pétale égale à 0,75 cm). Que se passe-t-il pour k = 5 ?

Exercice 4

Tester l'algorithme KNN (toujours à l'aide du programme de l'exercice 2) avec un iris de votre choix (si vous ne trouvez pas d'iris, vous pouvez toujours inventer des valeurs ;-))