Algorithme des k plus proches voisins:

Nous allons maintenant travailler sur un algorithme d'apprentissage automatique, souvent appelé, même en français, algorithme de machine learning. L'idée est d'utiliser un grand nombre de données afin "d'apprendre à la machine" à résoudre un certain type de problème (nous verrons un exemple un peu plus loin).

Cette idée d'apprentissage automatique ne date pas d'hier, puisque le terme de machine learning a été utilisé pour la première fois par l'informaticien américain Arthur Samuel en 1959. Pourquoi le machine learning est tant "à la mode" depuis quelques années ? Simplement parce que le nerf de la guerre dans les algorithmes de machine learning est la qualité et la quantité des données (les données qui permettront à la machine d'apprendre à résoudre un problème), or, avec le développement d'internet, il est relativement simple de trouver des données sur n'importe quel sujet (on parle de "big data"). À noter aussi l'importance des stratégies mises en place par les GAFAM (Google, Apple, Facebook, Amazon et Microsoft) afin de récupérer un grand nombre de données concernant leurs clients. Ces données sont très souvent utilisées pour "nourrir" des algorithmes de machine learning (comment, d'après vous, Amazon arrive à proposer à ces clients des "suggestions d'achats" souvent très pertinentes ?)

Nous allons étudier un algorithme d'apprentissage assez simple à appréhender : l'algorithme des "k plus proches voisins" (en anglais "k nearest neighbors" : knn)

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 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 étudié, Anderson a mesuré (en cm):

- la largeur des sépales,
- la longueur des sépales,
- la largeur des pétales,
- la longueur des pétales.

Par souci simplification, nous nous intéresserons uniquement à la largeur et à la longueur des pétales. Pour chaque iris mesuré, Anderson a aussi noté l'espèce ("iris setosa", "iris virginica" ou "iris versicolor"). Vous trouverez 50 de ces mesures dans le fichier iris.csv.

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" et 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

extrait du jeu de données "iris"

Vous devez savoir que ce jeu de données a, aujourd'hui, un intérêt purement pédagogique. En effet, il est exclusivement utilisé par des personnes désirant s'initier aux algorithmes de machine learning.

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.

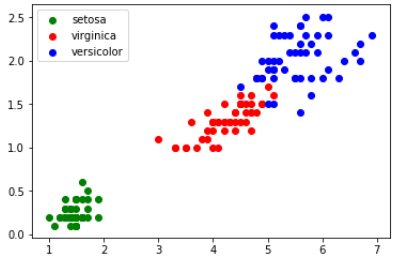
Exercice 1: Après avoir placé le fichier iris.csv dans le même répertoire que votre fichier Python, étudiez et testez le code suivant:

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

Quelques mots sur le programme ci-dessus:

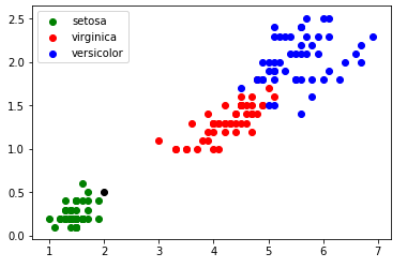
- La partie "Pandas" ne devrait pas vous poser de problèmes : x correspond à la longueur des pétales, correspond à la largeur des pétales et lab correspond à l'espèce d'iris (0,1 ou 2)
- Nous utilisons ensuite la bibliothèque matplotlib qui permet de tracer des graphiques très facilement. "plt.scatter" permet de tracer des points, le "x[lab == 0]" permet de considérer uniquement l'espèce "iris setosa" (lab==0). Le premier "plt.scatter" permet de tracer les points correspondant à l'espèce "iris setosa", ces points seront vert (color='g'), le deuxième "plt.scatter" permet de tracer les points correspondant à l'espèce "iris virginica", ces points seront rouge (color='r'), enfin le troisième "plt.scatter" permet de tracer les points correspondant à l'espèce "iris versicolor", ces points seront bleu (color='b'). Nous aurons en abscisse la longueur du pétale et en ordonnée la largeur du pétale.

Vous devriez normalement obtenir ceci:



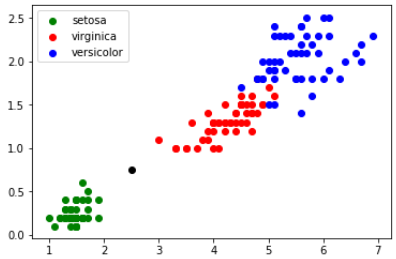
Nous obtenons des "nuages" de points, on remarque ces points sont regroupés par espèces d'iris (sauf pour "iris virginica" et "iris versicolor", les points ont un peu tendance à se mélanger).

Imaginez maintenant qu'au cours d'une promenade vous trouviez un iris, n'étant pas un spécialiste, il ne vous est pas vraiment possible de déterminer l'espèce. En revanche, vous êtes capables de mesurer la longueur et la largeur des pétales de cet iris. Partons du principe qu'un pétale fasse 0,5 cm de large et 2 cm de long. Plaçons cette nouvelle donnée sur notre graphique (il nous suffit d'ajouter la ligne "plt.scatter(2.0, 0.5, color='k')", le nouveau point va apparaitre en noir (color='k')):



Je pense que le résultat est sans appel : il y a de fortes chances que votre iris soit de l'espèce "iris setosa".

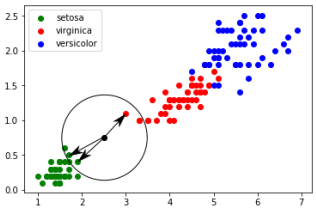
Il est possible de rencontrer des cas plus difficiles, par exemple : largeur du pétale = 0,75 cm; longueur du pétale = 2,5 cm:



Dans ce genre de cas, il peut être intéressant d'utiliser l'algorithme des "k plus proches voisins", en quoi consiste cet algorithme:

- on calcule la distance entre notre point (largeur du pétale = 0,75 cm ; longueur du pétale = 2,5 cm) et chaque point issu du jeu de données "iris" (à chaque fois c'est un calcul de distance entre 2 points tout ce qu'il y a de plus classique)
- on sélectionne uniquement les k distances les plus petites (les k plus proches voisins)
- parmi les k plus proches voisins, on détermine quelle est l'espèce majoritaire. On associe à notre "iris mystère" cette "espèce majoritaire parmi les k plus proches voisins"

Prennons k = 3:



Les 3 plus proches voisins sont signalés ci-dessus avec des flèches : nous avons deux "iris setosa" (point vert) et un "iris virginica" (point rouge). D'après l'algorithme des "k plus proches voisins", notre "iris mystère" appartient à l'espèce "setosa".

La bibliothèque Python Scikit Learn propose un grand nombre d'algorithmes lié 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, étudiez et testez le code suivant:

```
import pandas
import matplotlib.pyplot as plt
from sklearn.neighbors import KNeighborsClassifier

#traitement CSV
iris=pandas.read_csv("iris.csv")
x=iris.loc[:,"petal_length"]
y=iris.loc[:,"petal_width"]
lab=iris.loc[:,"species"]
#fin traitement CSV

#valeurs
longueur=2.5
largeur=0.75
k=3
#fin valeurs

#graphique
plt.scatter(x[lab == 0], y[lab == 0], color='g', label='setosa')
plt.scatter(x[lab == 1], y[lab == 1], color='r', label='virginica')
plt.scatter(x[lab == 2], y[lab == 2], color='b', label='versicolor')
plt.scatter(longueur, largeur, color='k')
plt.legend()
#fin graphique

#algo knn
d=list(zip(x,y))
model = KNeighborsClassifier(n_neighbors=k)
model.fit(d,lab)
prediction= model.predict([[longueur,largeur]])
#fin algo knn
```

```
#Affichage résultats
txt="Résultat : "
if prediction[0]==0:
  txt=txt+"setosa"
if prediction[0]==1:
  txt=txt+"virginica"
if prediction[0]==2:
  txt=txt+"versicolor"
plt.text(3,0.5, f"largeur : {largeur} cm longueur : {longueur} cm", fontsize=12)
plt.text(3,0.3, f"k : {k}", fontsize=12)
plt.text(3,0.1, txt, fontsize=12)
#fin affichage résultats

plt.show()
```

Le programme ci-dessus n'est pas très complexe à comprendre, nous allons tout de même nous attarder sur la partie "knn" :

```
d=list(zip(x,y))
model = KNeighborsClassifier(n_neighbors=k)
model.fit(d,lab)
prediction= model.predict([[longueur,largeur]])
```

La première ligne "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, 1.4, ...]
y = [0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.4,....]
```

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), (1.4, 0.4), ...]
```

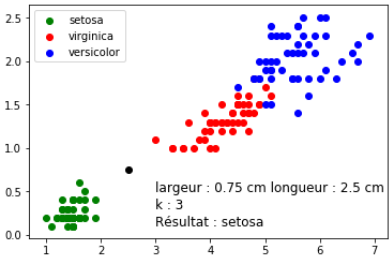
les éléments des tableaux x et y ayant le même indice sont regroupés dans un tuple, nous obtenons bien une liste de tuples.

"KNeighborsClassifier" est une méthode issue de la bibliothèque scikit-learn (voir plus haut le "from sklearn.neighbors import KNeighborsClassifier"), cette méthode prend ici en paramètre le nombre de "plus proches voisins" (model = KNeighborsClassifier(n_neighbors=k))

"model.fit(d, lab)" permet d'associer les tuples présents dans la liste "d" avec les labels (0 : "iris setosa", 1 : "iris virginica" ou 2 : "iris versicolor"). Par exemple le premier tuple de la liste "d", (1.4, 0.2) est associé au premier label de la liste lab (0), et ainsi de suite...

La ligne "prediction= model.predict([[longueur,largeur]])" permet d'effectuer une prédiction pour un couple [longueur, largeur] (dans l'exemple ci-dessus "longueur=2.5" et "largeur=0.75"). La variable "prediction" contient alors le label trouvé par l'algorithme knn. Attention, "predection" est une liste Python qui contient un seul élément (le label), il est donc nécessaire d'écrire "predection[0]" afin d'obtenir le label.

Vous devriez normalement obtenir ceci:



Exercice 3: Modifiez le programme du "À faire vous-même 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: Testez l'algorithme knn (toujours à l'aide du programme du "À faire vous-même 2") avec un iris de votre choix (si vous ne trouvez pas d'iris, vous pouvez toujours inventer des valeurs ;-))