Les algorithmes d'apprentissage automatiques, souvent appelés "machine learning" ont connu un fort regain d'intérêt au début des années 2000 notamment grâce à la quantité de données disponibles sur internet. L'idée est d'utiliser un grand nombre de données afin "d'apprendre à la machine" à résoudre un certain type de problème.

L'algorithme des k plus proches voisins (abrévié k-NN, pour "k Nearest Neighbors") appartient à la famille des algorithmes d'apprentissage automatique. Il ne nécessite pas de phase d'apprentissage à proprement parler, il faut juste stocker le jeu de données d'apprentissage, c'est-à-dire des exemples sur lesquels l'ordinateur peut se baser pour classifier de nouvelles données.

Partie I : Les iris de Fisher

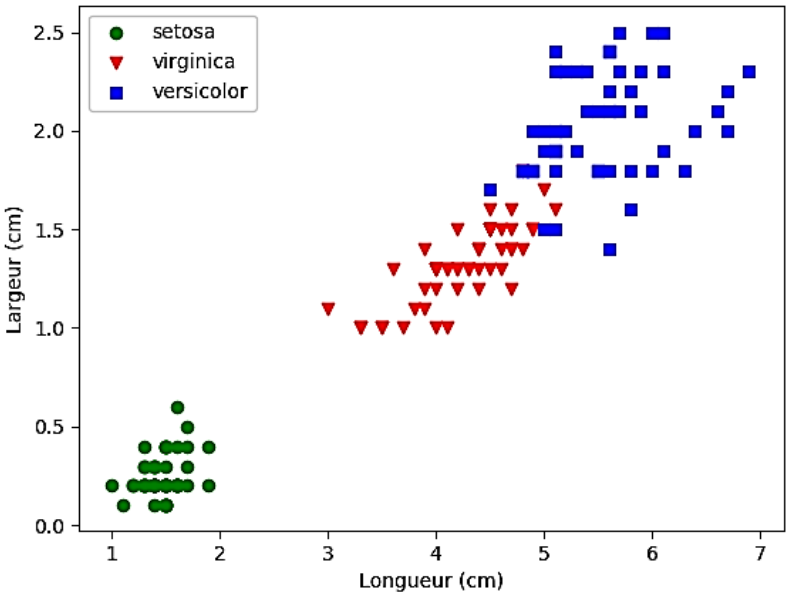
En 1936, Edgar Anderson a collecté des données sur 3 espèces d'iris : "iris setosa ", "iris virginica" et "iris versicolor". Pour chaque iris étudié, il a mesuré la longueur et la largeur des pétales, et a aussi noté l'espèce.

iris setosa

iris virginica

iris versicolor

A partir d'un échantillon d'une cinquantaine d'iris par espèce, on peut tracer le graphique de la longueur des pétales en fonction de la largeur, en identifiant chaque espèce par une couleur. Les nuages de points sont alors plus ou moins regroupés par espèce.



Iris de Fisher

Lors d'une promenade, vous trouvez 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.

I.1. Pour un pétale de longueur 2 cm et de largeur 0,5 cm, quelle est l'espèce la plus probable de l'iris trouvé ?

I.2. Le cas d'un pétale de longueur 2,5 cm et de largeur 0,75 cm est plus complexe.
Placer le point sur le graphique puis proposer une méthode permettant de déterminer l'espèce la plus probable de l'iris trouvé.

L'algorithme des k plus proches voisins s'applique à un ensemble E de N données labélisées (avec différentes classes, chacune représenté par un entier), chacune représentée par un ensemble de m coordonnées correspondant aux variables prédicatrices de la donnée.

Exemple : Dans le cas des iris, chaque point a 2 coordonnées (longueur ; largeur) et est labélisé en fonction de l'espèce d'iris.

On considère une donnée *u* qui n'appartient pas à *E* et qui ne possède pas de label (*u* est uniquement caractérisé par ses coordonnées). Le principe de l'algorithme des k plus proches voisins est alors le suivant :

- On calcule les distances entre la donnée *u* et chaque donnée appartenant à *E*
- On retient les *k* données du jeu de données *E* les plus proches de *u* (où k est un nombre entier donné en entrée)
- On attribue à *u* la classe qui est la plus fréquente parmi les *k* données les plus proches

I.3. Proposer un algorithme permettant de calculer la distance entre 2 points, chacun étant représenté par *m* coordonnées (avec $m \geq 2$) données sous forme de liste de mêmes tailles que l'on nommera *PointA* et *PointB*.

Calcul de distance
On considère 2 points, A et B, chacun représenté par un jeu de $m \geq 2$ coordonnées (x_0, x_1, …, x_m).
La distance *D* entre les points A et B est alors calculée grâce à la formule suivante :

$$D = \sqrt{\sum_{i=0}^{m} (x_{i\_B} - x_{i\_A})^2} = \sqrt{(x_{0\_B} - x_{0\_A})^2 + (x_{1\_B} - x_{1\_A})^2 + \ldots + (x_{m\_B} - x_{m\_A})^2}$$

I.4. Par quel type de variable la donnée *u* peut-elle être représentée dans l'algorithme ? Quelles sont les caractéristiques de cette variable ?

I.5. Même questions pour la variable *E*, représentant le jeu de données.

I.6. Ecrire un algorithme de k plus proches voisins, utilisant la fonction *Distance* précédente, pour un ensemble *E* de données labélisées possédant *m* coordonnées et une donnée *u* non-labélisée possédant aussi *m* coordonnées et qui n'appartient pas à *E*. Les variables *E*, *u* et *k* sont données en entrée.

Partie II : A vos ordinateurs !

Les données utilisées pour faire le graphique des Iris des Fisher (partie I) sont fournis dans le fichier `Donnees_iris.csv` (colonnes : longueur, largeur, label). Les espèces sont identifiées par un entier naturel dans ce fichier : *0 - Iris setosa* *1 - Iris virginica* *2 - Iris versicolor*

Le but de cet exercice est d'utiliser un algorithme des k plus proche voisin pour identifier l'espèce de deux iris dont on connait la longueur et la largeur des pétales.

Iris A	**Iris B**
Longueur : 2.5 cm	Longueur : 5.2 cm
Largeur : 0.75 cm	Largeur : 1.7 cm

Pour vous aider à écrire les codes, le fichier `kNN_CodeEleves.py` contient des fonctions permettant de trier un tableau à 2 dimensions (*TriTable*), lire un fichier de données (*LireFichier*) et écrire un fichier de données (*EcrireFichier*).

Lire la documentation sur chaque fonction avec `help(`*nom_fonction*`)` avant utilisation.

Pour les utiliser, il vous suffit de sauvegarder le fichier dans le même dossier que votre code et d'importer les fonctions dans votre code comme une bibliothèque : `from kNN_CodeEleves import *`

II.1. Ecrire une fonction *Distance* prenant en entrée deux listes de nombres, *PointA* et *PointB*, correspondants aux coordonnées de deux points A et B, et renvoyant la distance entre ces points.

II.2. Ecrire une fonction *kPPV_2D* prenant en arguments un ensemble de données labélisées *E*, une donnée *u* non-labélisée sous forme de liste de coordonnées et un nombre entier *k* de voisins. La fonction renverra, à l'aide d'un algorithme de k plus proches voisins, le label identifiant la donnée *u* sous forme d'un entier.**

II.3. Ecrire une fonction *Iris* prenant en argument une liste *mesure* (où `mesure=[longueur,largeur]`) un nombre entier *k* de voisins, et permettant d'identifier, à partir des données du fichier `Donnees_iris.csv`, l'espèce de l'iris mesurée. Cette fonction devra utiliser les fonctions écrites précédemment ainsi que les fonctions fournis nécessaires.

II.4. Déterminer l'espèce des iris A et B pour $k = 5$ puis $k = 10$ voisins. Commenter.

** AIDE : Pour le calcul de distance de chaque point *i*, créer une variable var ne contenant QUE les coordonnées, pour l'utiliser dans *Distance* :

```
var = [ ]
for j in range(len(E[0])-1):          #on ne prend pas la dernière colonne
    var += [E[i][j]]
```

Ce qu'il faut retenir

❖ Algorithme des k plus proches voisins

L'algorithme des k plus proches voisins appartient à la famille des algorithmes d'apprentissage automatique et utilise des exemples sur lesquels l'ordinateur peut se baser pour classifier de nouvelles données.

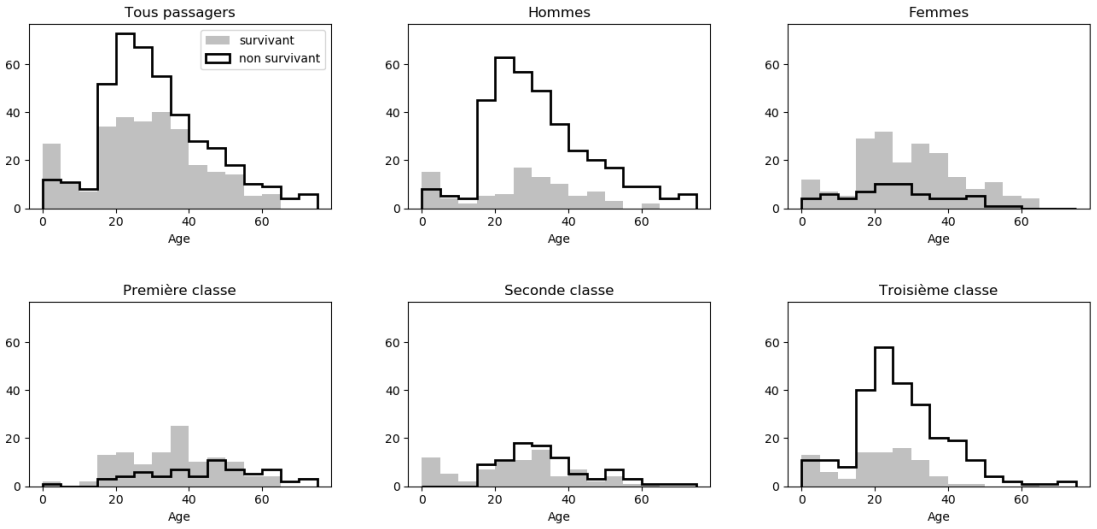
Algorithme des k plus proches voisins (k-NN)

Entrées : E une liste de liste, u une liste de nombres, k un entier
POUR chaque point de E :
    Calculer la distance entre u et chaque point de E
FIN POUR
Ordonner E en fonction de la distance (algorithme de tri)
POUR les k premiers éléments de E ordonnée
    Compter le nombre d'occurrence de chaque type
FIN POUR
RENVOYER le type avec la plus grande occurrence

DM facultatif

(noté en bonus si rendu avant le ..... 22h)

Le naufrage du Titanic est l'un des accidents maritime les plus célèbres de l'histoire.
Le 15 avril 1912, lors de son voyage inaugural, le RMS Titanic coule à la suite d'une collision avec un iceberg. Le paquebot ne contenait malheureusement pas assez de canots de sauvetage pour la totalité des personnes à bord, entrainant la mort de près de 1500 des 2224 passagers et équipage.
Bien qu'il y ait une part de chance, il semble que certain groupes de personnes aient une plus grande probabilité de survie que d'autre (voir figure ci-dessous).



Répartitions survivants/non-survivants en fonction de l'âge, pour différentes catégories
(tracé à partir des données témoin de l'exercice)

Dans cet exercice, nous utilisons un algorithme des k plus proches voisins afin de déterminer, à partir d'un groupe témoin de passagers, si une personne survivrait ou non au naufrage.

Vous disposez de 2 fichiers de données :

– `Donnees_Titanic.csv` : Données témoin, incluant le label.
  Les colonnes sont, dans l'ordre, ID passager, classe (1, 2 ou 3), sexe (0=Homme, 1=Femme), âge et label (0=non survivant, 1=survivant)

– `Test_Titanic.csv` : Données des passagers pour lesquels on cherche à déterminer l'issue.
  Les colonnes sont, dans l'ordre, ID passager, classe (1, 2 ou 3), sexe (0=Homme, 1=Femme) et âge.

Ces fichiers sont disponibles sur *http://mgendrephyschim.free.fr/NSI/premiereNSI/premiereNSI_Algorithmes.html*

Afin de résoudre cet exercice, vous devez :

- Créer des matrices `Donnees` et `Test` avec les données des fichiers.
- Utiliser un algorithme des k plus proches voisins, prenant en compte la classe, le sexe et l'âge des passagers, pour déterminer l'issue pour chacun des passagers du groupe test. On prendra **k=3**.
- Créer un nouveau tableau, nommé *Previsions*, contenant les données de chaque passager (ID, classe, âge et sexe) ainsi qu'une colonne avec le label trouvé.
- Utiliser la fonction *HistogrammeTitanic* du fichier `kNN_CodeEleves.py` pour construire les d'histogrammes vous permettant de comparer vos résultats aux distributions du groupe témoin.

Vous pouvez utiliser les fonctions du fichier `kNN_CodeEleves.py` ainsi que toute fonction écrite en partie I, si nécessaire.

Note : Dû au grand nombre de données, ne vous inquiétez pas si votre code prend du temps à s'exécuter. Soyez néanmoins sûrs qu'il tourne sans problème, en testant régulièrement avec des *print( )*.
Si le code prend vraiment trop de temps sur *repl.it*, ne pas hésiter à utiliser *Spyder* ou *Edupython* à la place.