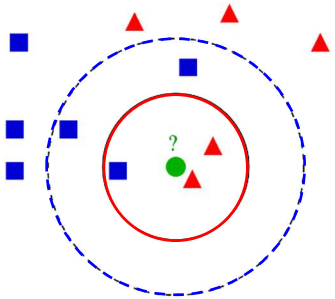
Apprentissage supervisé
k plus proches voisins – k-NN

En ***intelligence artificielle***, plus précisément en ***apprentissage automatique***, la méthode des *k* plus proches voisins est une méthode d'***apprentissage supervisé***. En abrégé k-NN ou KNN, de l'américain k-nearest neighbors.

Dans ce cadre, on dispose d'une base de données d'apprentissage constituée de *N* couples « entrée-sortie ». Pour estimer la sortie associée à une nouvelle entrée *x*, la méthode des *k* plus proches voisins consiste à prendre en compte (de façon identique) les *k* échantillons d'apprentissage dont l'entrée est la plus proche de la nouvelle entrée *x*, selon une ***distance*** à définir.

Par exemple, dans un problème de classification, on retiendra la classe la plus représentée parmi les *k* sorties associées aux *k* entrées les plus proches de la nouvelle entrée *x*.

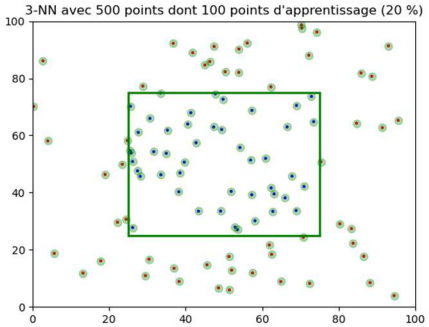
On cherche à ***classer*** l'échantillon de test (disque vert) soit dans la classe des triangles rouges soit dans la classe des carrés bleus.
Si *k* = 3 (cercle rouge en ligne pleine), il est affecté à la classe des triangles rouges (2 triangles / 1 carré dans le cercle considéré).
Si k = 5 (cercle en pointillés bleus), il est affecté à la classe des carrés bleus (3 carrés / 2 triangles dans le cercle externe).



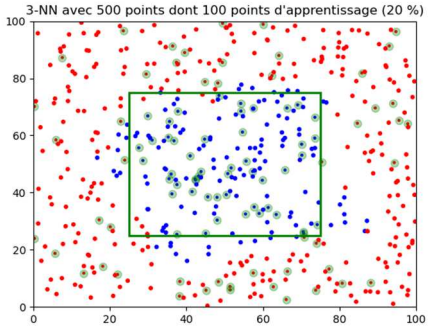
L'algorithme peut être utilisé à des fins de ***classification*** ou de ***régression***.

- En ***classification***, le résultat est une ***classe d'appartenance***.
  Un objet d'entrée est classifié selon le résultat majoritaire des statistiques de classes d'appartenance de ses *k* plus proches voisins, (*k* est un nombre entier positif généralement petit).
- En ***régression***, le résultat est la ***valeur*** pour cet objet. Cette valeur est la moyenne des valeurs des k plus proches voisins.

D'après Wikipédia



Apprentissage : données étiquetées



Test

Algorithme k-NN

Objectif ✔ : classer un objet parmi d'autres objets répartis en différentes catégories à partir de ses caractéristiques.

Savoir-faire : écrire une bibliothèque de fonctions et utiliser la bibliothèque sklearn (https://scikit-learn.org/stable/index.html).

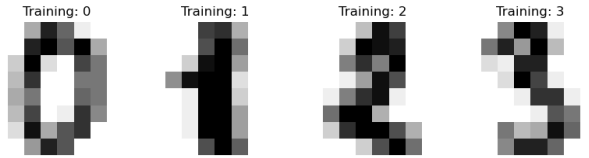
On dispose d'un ***ensemble*** **E** d'objets de même structure et d'un ***sous-ensemble d'apprentissage*** **A** constitué de *N* objets o_i de E représentant des ***données classées ou étiquetées***.
Ces classes forment une ***partition*** de A (sous-ensembles de A non vides et disjoints deux à deux dont la réunion est A).
On choisit un entier $k < N$ et on dispose d'un objet *o* de E qui n'est pas dans A.
Il s'agit de trouver, parmi les objets de A, les *k* plus proches de l'objet *o* sur le critère d'une ***distance*** (ou ***mesure de similarité***) *d* à définir.
Suivant le problème, on peut envisager différentes définitions pour une telle distance.

Rq : il existe de très nombreuses façons d'implémenter cet algorithme.

Exemple - OCR (optical character recognition)

A partir de l'image noir et blanc, 8×8 pixels, d'un chiffre manuscrit *c* compris entre 0 et 9 scanné, on souhaite identifier *c*. Ce chiffre représente la classe de l'objet.

Exemples d'images étiquetées ci-contre.



L'objet *o* à classer est ici une image définie par un ***tableau*** numpy de 8 lignes et 8 colonnes contenant des valeurs représentant des niveaux de gris.

```
o = np.array([ [0., 0., 5.,  13., 9.,  1.,  0., 0.],
               [0., 0., 13., 15., 10., 15., 5., 0.],
               [0., 3., 15., 2.,  0.,  11., 8., 0.],
               [0., 4., 12., 0.,  0.,  8.,  8., 0.],
               [0., 5., 8.,  0.,  0.,  9.,  8., 0.],
               [0., 4., 11., 0.,  1.,  12., 7., 0.],
               [0., 2., 14., 5.,  10., 12., 0., 0.],
               [0., 0., 6.,  13., 10., 0.,  0., 0.]])
```

La commande `o.flatten()` permet de convertir cet objet en tableau ou **vecteur** de dimension $n = 8 \times 8 = 64$ coordonnées :

```
array([0., 0., 5., 13., 9., 1., 0., 0., ... , 0., 0., 6., 13.,
10., 0., 0., 0.])
```

Rq : La commande `o.reshape(8,8)` permet de revenir au tableau 8x8.

Conclusion

L'algorithme k-NN est capable de traiter des objets de type quelconque dès que ceux-ci sont ramenés à des **vecteurs de dimension *n*** (*n* est appelé dimensionality dans sklearn).

Ecriture d'une bibliothèque de fonctions

Notations - Définitions

- **Objet *o*** : tableau de dimension *n* caractérisant l'objet.
- **E** : ensemble d'objets ayant la structure de l'objet *o*.
- **A** : ensemble d'apprentissage constitué d'objets o_i de E et de leurs étiquettes e_i. A est construit à partir de deux tableaux data et target (dénominations utilisées par la bibliothèque sklearn pour désigner les objets et leurs étiquettes), sous la forme d'une liste de tuples de la forme $[(o_0, e_0),(o_1, e_1),...]$ où e_i est l'étiquette de l'objet o_i.
- **Classes** des objets : ensemble des étiquettes.

Principe de l'algorithme

Entrées :

- un objet *o* d'un ensemble E ;
- un "ensemble" d'apprentissage A d'objets étiquetés représenté par une liste de la forme $[(o_0, e_0),(o_1, e_1),...]$;
- un entier strictement positif *k* qui est le nombre de voisins à considérer ;
- une fonction distance *d* définie pour les objets de E (cette fonction peut posséder différentes définitions, ce sera donc un paramètre des fonctions de la bibliothèque).

Sortie : une étiquette *e* pour l'élément *o*.

Algorithme

1. Déterminer les *k* éléments de l'ensemble d'apprentissage A les plus proches de l'objet *o* au sens de la distance *d*.
2. Déterminer l'étiquette la plus fréquente parmi les étiquettes de ces *k* éléments (choix aléatoire en cas d'égalité).

Etape 1 - Calcul des distances entre deux objets

Ecrire une fonction d_Euclide(o1, o2) renvoyant la distance euclidienne entre les objets o1 et o2.
Rappel : ces objets sont des vecteurs de dimension inconnue a priori (mais identique pour les deux objets).

Etape 2 - Plus proches voisins d'un objet

Etape 2.1 - Liste des distances de l'objet o à tous ses voisins dans A

Ecrire une fonction distances(o, A, fct_distance) renvoyant une liste constituée de tuples de la forme (o_i, fct_distance(o, o_i)) où o_i est le tableau des coordonnées d'un objet de A.

Etape 2.2 - Tri des distances de l'objet o à ses voisins

Ecrire une fonction tri_distances(o, A, fct_distance) renvoyant la liste générée par la fonction distances(o, A, fct_distance) triée par ordre croissant des distances .

Etape 2.3 - Liste des k plus proches voisins de l'objet o

Ecrire une fonction liste_kppv(o, A, k, fct_distance) renvoyant la liste des *k* plus proches voisins de l'objet o dans l'ensemble d'apprentissage A.
La liste renvoyée est de la forme $[(oi,ei),...]$

Etape 3 – Etiquette majoritaire parmi celles des plus proches voisins

Etape 3.1 – Dictionnaire des occurrences des étiquettes

Ecrire une fonction occurrences_classes(L) où L est une sous-liste de la liste d'apprentissage A (liste de tuples (objet, etiquette)) et qui renvoie le dictionnaire des occurrences des étiquettes.
La fonction renvoie donc un dictionnaire de la forme : $\{e_0: n_0, e_1: n_1, ...\}$ où n_i est le nombre d'occurrences de l'étiquette e_i dans les tuples de la liste L.

Etape 3.2 – Classe majoritaire parmi les k plus proches voisins de l'objet o

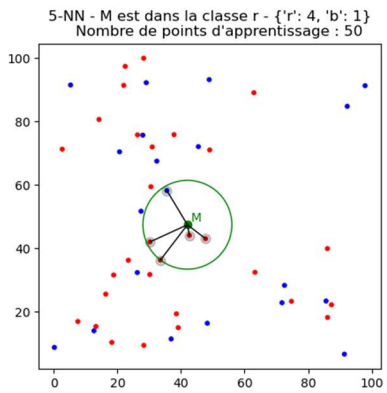
Ecrire une fonction classe_majoritaire(L) renvoyant l'étiquette de la classe majoritaire dans la liste L définie comme une sous-liste de la liste d'apprentissage A.

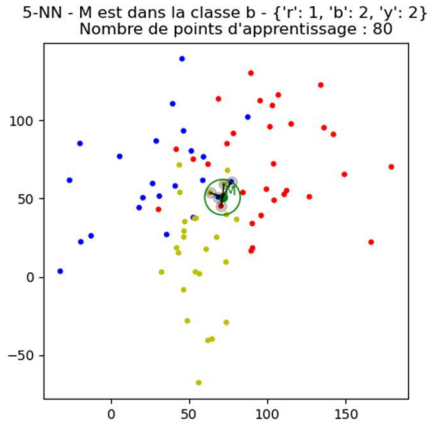
Etape 4 – Classification : algorithme kNN

Estimation de la classe de l'objet o à partir d'un ensemble d'apprentissage A

Ecrire une fonction kNN(o, data, target, k, fct_distance) renvoyant l'étiquette de la classe majoritaire des k plus proches voisins de l'objet o à l'aide d'un ensemble d'apprentissage A constitué d'un tableau data d'objets dont les étiquettes sont fournies dans le tableau target.
La fonction kNN doit donc construire A à partir des deux tableaux data et target.

Cf. notebook associé pour les représentations graphiques.





Matrice de confusion

L'algorithme de *k* plus proches voisins permet de faire des prédictions, il reste à mesurer la qualité de ces prédictions.
La ***matrice de confusion*** mesure la ***qualité*** d'un système de ***classification***.
Chaque ligne correspond à une classe réelle, chaque colonne correspond à une classe estimée pour un ensemble de données test étiquetées par un expert.
La cellule ligne L, colonne C contient le nombre d'éléments de la classe réelle L qui ont été estimés comme appartenant à la classe C.
Un des intérêts de la matrice de confusion est qu'elle montre rapidement si un système de classification parvient à classifier correctement.

Exemple

On souhaite mesurer la qualité d'un système automatique de classification de courriers électroniques.
Les courriers sont classifiés selon deux classes : courriel pertinent ou pourriel intempestif. Supposons que le classificateur soit testé avec un jeu de 200 mails, dont 100 sont des courriels pertinents et les 100 autres relèvent de pourriels.

Pour cela, on veut savoir :

- combien de courriels seront faussement estimés comme des pourriels (fausses alarmes) par le classificateur ;
- combien de pourriels ne seront pas estimés comme tels (non détections) et classifiés à tort comme courriels par le classificateur.

A l'aide de données étiquetées par un expert humain et du programme kNN, on dresse le tableau suivant :

		Classe estimée (programme classificateur kNN)	
		Courriel	**Pourriel**
Classe réelle (vérificateur humain)	**Courriel (100)**	**95** Vrais positifs	5 Faux négatifs
	Pourriel (100)	3 Faux positifs	**97** Vrais négatifs

La matrice de confusion suivante se lit alors comme suit :

- ✓ horizontalement, sur les 100 courriels initiaux (ie : 95+5), 95 ont été estimés par le classificateur comme tels et 5 ont été estimés comme pourriels (ie : 5 faux-négatifs),
- ✓ horizontalement, sur les 100 pourriels initiaux (ie : 3+97), 3 ont été estimés par le classificateur comme courriels (ie : 3 faux-positifs) et 97 ont été estimés comme pourriels,
- ✓ verticalement, sur les 98 mails (ie : 95+3) estimés par le classificateur comme courriels, 3 sont en fait des pourriels,
- ✓ verticalement, sur les 102 mails (ie : 5+97) estimés par le classificateur comme pourriels, 5 sont en fait des courriels.
- ✓ diagonalement (du haut gauche, au bas droit), sur les 200 courriels initiaux, 192 (95 + 97) ont été estimés correctement par le classificateur.

Cette notion s'étend à un ***nombre quelconque de classes***.
On peut ***normaliser*** cette matrice pour en simplifier la lecture : dans ce cas, un classificateur sera d'autant meilleur que sa matrice de confusion s'approchera d'une ***matrice diagonale***.

Remarques :

- ✓ la 2ème ligne horizontale des pourriels nous donne une indication sur la capacité à détecter automatiquement les pourriels (i.e. : 97% de succès).
- ✓ la 2ème colonne verticale des pourriels nous donne une indication sur à quel point les prédictions (détection de pourriels) sont fiables (i.e. : 5% d'erreurs sur les courriels et 3% d'erreurs sur les pourriels soit 4% d'erreur de classification d'un mail en moyenne par le classificateur).

D'après Wikipédia

Définitions

Taux d'erreurs TE : nombre de prédictions incorrectes sur le nombre total de prédictions (aussi proche de 0 que possible).
Exemple des mels : TE = (5+3)/200 = 4% (diagonale grisée)

Précision P : nombre de prédictions correctes sur le nombre total de prédictions (aussi proche de 1 que possible), P = 1 - TE.
Exemple des mels : P = (95+97)/200 = 96% (diagonale jaune)

Dans les cas où il n'existe que ***deux classes*** comme dans cet exemple.

Sensibilité Se (taux de vrais positifs) : nombre de prédictions positives correctes sur le nombre total de données positives.
Exemple des mels : Se = 95/100 = 95%

Spécificité Sp (taux de vrais négatifs) : nombre de prédictions négatives correctes sur le nombre total de données négatives.
Exemple des mels : Se = 97/100 = 97%

Taux de faux positifs TFP : nombre de prédictions positives incorrectes sur le nombre total de données négatives, TFP = 1 - Sp.
Exemple des mels : TFP = 3/100 = 3% (diagonale grisée)

Taux de faux négatifs TFN : nombre de prédictions négatives incorrectes sur le nombre total de données positives, TFN = 1 - Se.
Exemple des mels : TFN = 5/100 = 5% (diagonale grisée)

Matrice de confusion - Exemples

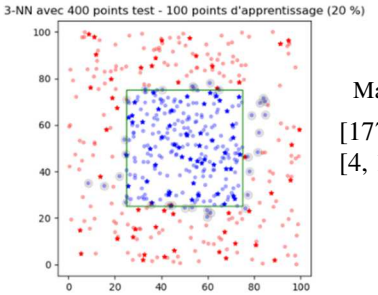
Le script précédent était destiné à *illustrer* l'algorithme des *k* plus proches voisins : seul un point test était envisagé.
On cherche ici à *évaluer les performances* de cet algorithme à travers un jeu de données étiquetées par un « expert ».
A partir des points d'un ensemble E, tous étiquetés (i.e. dont les classes sont connues), on constitue deux ensembles en retirant aléatoirement des points de E pour constituer au final un ensemble d'apprentissage A (les points non retirés de E) et un ensemble T de points test.
Les données de l'ensemble T sont donc également étiquetées mais on va comparer cet étiquetage expert au résultat fourni par l'algorithme pour ces mêmes points.
Les performances de l'algorithme sont évaluées grâce à la matrice de confusion, créée très simplement en comparant prédictions et données étiquetées.

Exemples de résultats

Rectangle vert = frontières des deux classes déterminées par un « expert ».
Etoiles rouges ou bleues de couleur vive = données d'apprentissage étiquetées (« expert »).
Points rouges ou bleus de couleur pâle = prédictions de l'algorithme.
Disques grisés = points dont la classe déterminée par l'algorithme est incorrecte.

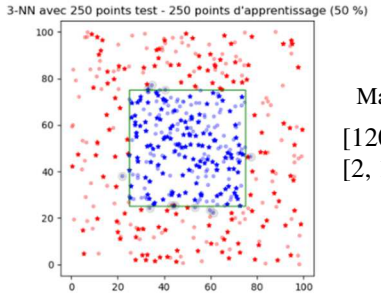
Dans cet exemple, la matrice de confusion est de la forme :

	r (algo)	*b* (algo)
r (expert)	55	2
b (expert)	5	38



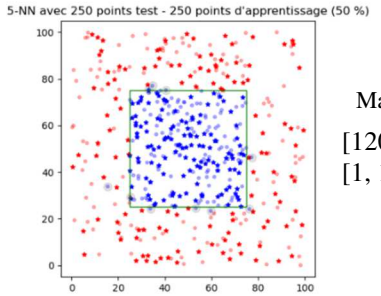
Matrice

[177, 28]
[4, 191]



Matrice

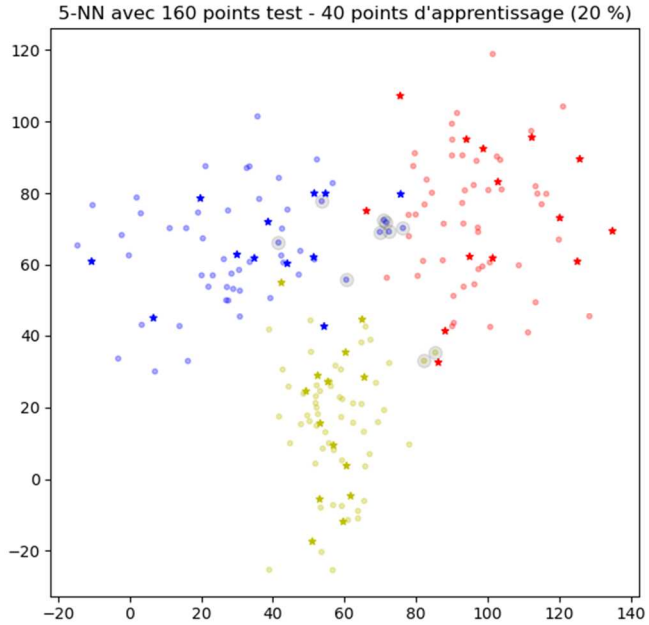
[120, 9]
[2, 119]



Matrice

[120, 9]
[1, 120]

Exemple avec 3 classes et des nuages gaussiens :



Matrice de confusion :

[50, 7, 2]
[0, 45, 0]
[0, 1, 55]

Cf. notebook associé.

Bibliothèque Scikit-learn

Site officiel : https://scikit-learn.org

Eléments de syntaxe

L'algorithme des *k* plus proches voisins est implémenté via un classificateur (sklearn.neighbors.KNeighborsClassifier).

Objet classificateur :
clf = **sklearn.neighbors.KNeighborsClassifier(n_neighbors=*k*)**

Définition des données d'apprentissage (objets et étiquettes) :
clf.**fit(***data*, *target***)**

Etiquettes calculées avec des données (test ou nouvelles données) :
clf.**predict(***data***)**

Probabilités d'appartenance à une classe
clf.**predict_proba(***data***)**

Frontières entre domaines de prédominance des différentes classes
sklearn.inspection.DecisionBoundaryDisplay.from_estimator(*clf, data,…***)**

Bibliothèque Scikit-learn - Application à l'OCR

Exemple tiré de : Recognizing hand-written digits.

Les données (images 8×8 pixels noir et blanc), leurs étiquettes (chiffres associés) ainsi que d'autres renseignements sont stockés dans un dataset (cf. « Exemple – OCR » ci-dessus).

Le chargement de ces données s'effectue grâce à la commande datasets.load_digits() (syntaxe détaillée : sklearn.datasets.load_digits).

```
# Bibliothèques
from sklearn import dataset      # Exemples de données (datasets)
from sklearn.neighbors import KNeighborsClassifier   # Algorithme kNN

# Chargement du dataset pour l'OCR (objets images et étiquettes)
digits = datasets.load_digits()

# data = tableau des images, target = tableau des étiquettes
data, target = digits.data, digits.target
```

Cf. notebook associé : comparaison des performances de l'algorithme programmé et de l'algorithme utilisé par sklearn via la matrice de confusion.

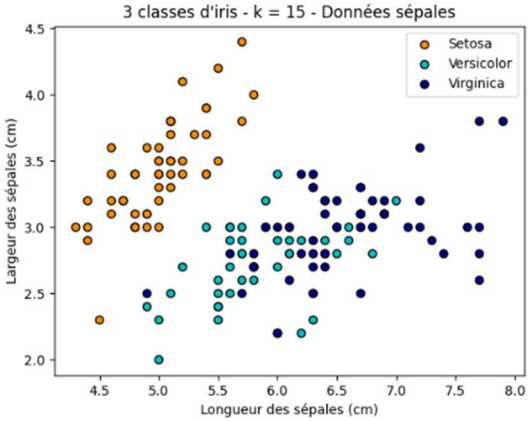
Bibliothèque Scikit-learn – Domaines de prédominance d'une classe

Exemple tiré de : Iris Dataset

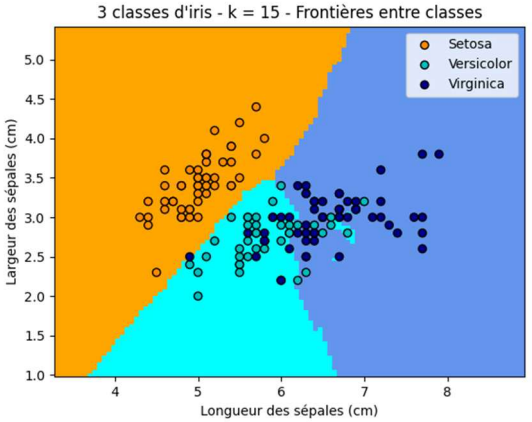
Le jeu de données comprend 50 échantillons de chacune des trois espèces d'iris (Iris setosa, Iris virginica et Iris versicolor), donc 150 échantillons au total.
Quatre caractéristiques ont été mesurées à partir de chaque échantillon : la longueur et la largeur des sépales et des pétales, en centimètres (Iris de Fisher (Wikipédia)).

Les données sont regroupées dans un tableau 150×4 :
- les lignes correspondent aux échantillons ;
- les colonnes correspondent aux caractéristiques (Sepal Length, Sepal Width, Petal Length and Petal Width).

Exemple : données concernant les sépales



Détermination des frontières entre domaines associés aux classes :



Cf. notebook associé.