TP – Algorithme des k plus proches voisins

Exercice 1 (débranché) : Classification k NN

Données pour l'apprentissage (volontairement réduite pour procéder à la main)

numero_departement	nom_departement	prefecture	nom_commune	codes_postaux	latitude(°)	longitude(°)
77	Seine-et-Marne	Melun	Fontainebleau	77300	48,4	2,7
77	Seine-et-Marne	Melun	Melun	77000	48,533333	2,666667
74	Haute-Savoie	Annecy	Annecy	74000	45,9	6,116667
91	Essonne	Evry	Evry	91000	48,633333	2,45

Les coordonnées GPS de Dammarie-les-Lys sont :
Latitude : 48,515088°
Longitude : 2,634702°

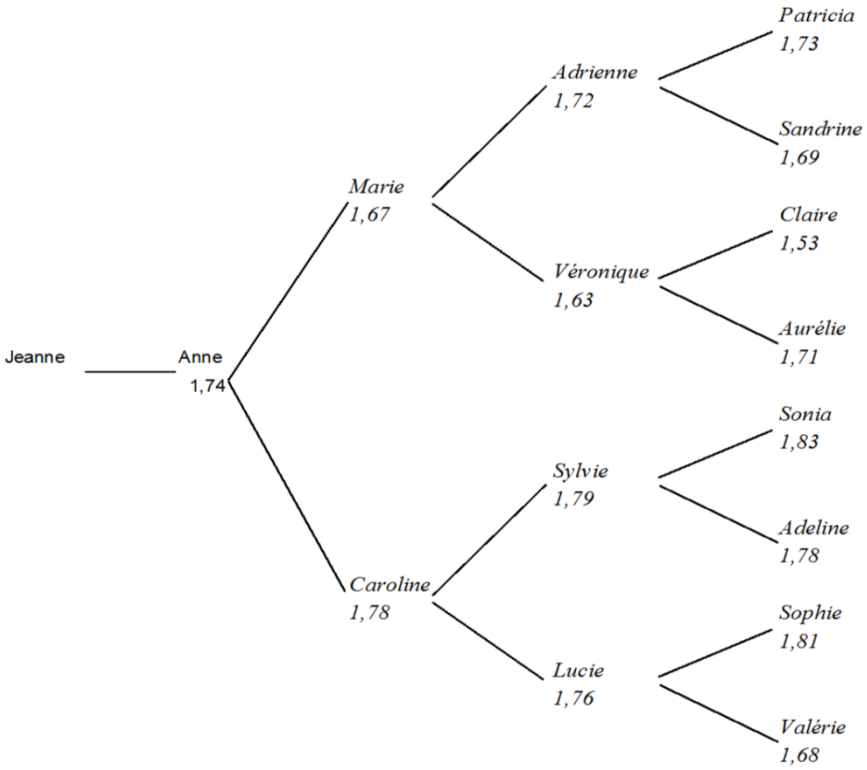
On rappelle que le rayon moyen de la terre mesure environ 6371 kilomètres et que la longueur d'un arc de cercle est égale au produit du rayon par l'angle au centre.

1) Distance Dammarie-Annecy
   a) Calculer l'écart en kilomètres lié à la différence de latitude entre Dammarie-les-Lys et Annecy.
   b) Calculer l'écart en kilomètres lié à la différence de longitude entre Dammarie-les-Lys et Annecy.
   c) En utilisant le théorème de Pythagore, en déduire une approximation de la distance entre Dammarie les lys et Annecy.
2) Calculer les distances entre Dammarie-les-Lys et chacune des villes apparaissant dans les données d'apprentissage.
3) A quel département la ville de Dammarie-les-Lys appartient-elle selon un algorithme basé sur les 3 plus proches voisins s'appuyant sur les données d'apprentissage ci-dessus ? (détailler la démarche pour justifier)

Exercice 2 (débranché) : Régression k NN

L'arbre généalogique de Jeanne est donné ci-contre avec les tailles à l'âge adulte.
Selon le principe de l'exemple 2 du cours, quelle estimation de la taille de Jeanne à l'âge adulte est proposée par une régression à partir :

(1) d'un algorithme 1 NN ?
(2) d'un algorithme 3 NN ?
(3) d'un algorithme 7 NN ?
(4) d'un algorithme 5 NN ? (avec la convention qu'en cas d'égalité des distances, le premier voisin trouvé est conservé)



Exercice 3 (débranché) : formalisation de l'algorithme k NN

On dispose :

- de données d'apprentissages présentées sous la forme d'une liste de couples (prédicteurs, étiquette)
- d'une fonction distance qui à partir des prédicteurs de deux objets renvoie la distance entre les deux objets avec les propriétés mathématiques usuelles pour une distance

Ecrire en langage naturel, en pseudo-code ou en Python une fonction kNN qui :

- prend en entrées : un objet observé pour lequel on dispose des prédicteurs, une liste de données d'apprentissage et entier naturel k non nul et inférieur ou égal à la longueur de la liste des données d'apprentissage
- renvoie en sortie la liste des étiquettes des k plus proches voisins de l'objet observé parmi les données d'apprentissage.
  En cas d'égalité de distance on prendra arbitrairement le premier rencontré.
  Au cas où les prédicteurs d'un élément des données d'apprentissage correspondraient exactement à ceux de l'objet observé, on retournera une liste contenant k fois l'étiquette de cet élément.

Exercice 4 (TP informatique) : Classification k NN

Le fichier « apprentissage_departement.csv » a été constitué à partir de données extraites du site : https://www.data.gouv.fr/fr/datasets/listes-des-communes-geolocalisees-par-regions-departements-circonscriptions-nd/

1) Ecrire une fonction en Python qui lit ce fichier et constitue une liste de données d'apprentissage exploitable par la fonction kNN de l'exercice 3, constitué de couples ((latitude, longitude), département) (deux prédicteurs et une étiquette). On pourra s'appuyer sur la bibliothèque « `csv` » et les dictionnaires.
2) En s'inspirant de l'exercice 1, écrire une fonction `distance` calculant la distance entre deux villes à partir des couples (latitude, longitude) de chacune de ces deux villes. On rappelle que la fonction « `sqrt` » de la bibliothèque « `math` » permet de calculer une racine carrée.
3) A l'aide des fonctions de l'activité préparatoire et de l'exercice 3, écrire une fonction prenant en entrée un entier non nul k et le couple (latitude, longitude) des coordonnées GPS d'une commune et effectue une prédiction sur le département auquel appartient cette commune à l'aide d'un algorithme basé sur les k plus proches voisins figurant dans les données du fichier « apprentissage_departement.csv ».
4) Tester cette fonction sur plusieurs communes de votre choix (les coordonnées GPS pourront être cherchées sur internet). On effectuera des tests avec des communes qui apparaissent dans le fichier « apprentissage_departement.csv » et d'autres qui n'y apparaissent pas. Les tests seront effectués avec k = 7 et k = 6500.

Exercice 5 (TP informatique) : Régression k NN

Le fichier « apprentissage_temperatures2016.csv » a été constitué à partir de données extraites du site : https://www.data.gouv.fr/fr/datasets/temperature-quotidienne-regionale-depuis-janvier-2016/

On souhaite estimer la température moyenne pour une région et un jour donné.

1) Ecrire une fonction en Python qui lit ce fichier et constitue une liste de données d'apprentissage exploitable par la fonction kNN de l'exercice 3, constitué de couples ((région, jour), température moyenne) où jour désigne le numéro du jour dans l'année (compris entre 1 et 366).
2) Définir une fonction `distance` entre des prédicteurs donnés sous la forme d'un couple (région, jour), qui vaut 400 (ou n'importe quelle valeur strictement supérieure à 366) si les régions sont différentes et qui vaut le nombre de jours d'écart entre les jours de l'année sinon (peu importe l'année, attention l'écart entre le 30 décembre et le 2 janvier vaut donc 4).
3) A l'aide des fonctions des exercices 0 (activité préparatoire) et 3, écrire une fonction prenant en entrée un entier non nul k et un couple (région, jour) qui renvoie une estimation de la température qu'il fera dans la région pour ce jour par un algorithme k NN.
4) Tester l'algorithme sur les jours de votre choix et pour différentes valeurs de k. Que peut-on penser des choix k = 366 ? k = 4000 ?
5) (bonus) Ecrire des fonctions permettant d'obtenir le numéro du jour dans l'année à partir de la date ou vice versa, puis les utiliser pour améliorer le programme précédent en travaillant à partir des dates.

1NSI	Algorithmique k plus proches voisins	TP – **Algorithme des k plus proches voisins**

REPONSES AUX QUESTIONS DU TP :

Q1 :

1) a) différence de latitude : ≈ 291 km b) différence de longitude : ≈ 387 km c) distance ≈ 484 km
2) Dammarie-les-Lys ↔ Fontainebleau : 14,7 km ; Dammarie-les-Lys ↔ Melun : 4,09 km
Dammarie-les-Lys ↔ Evry : 24,4 km
3) Les trois plus proches voisins sont Melun, Fontainebleau et Evry. Le département le plus fréquent sur ces 3 commune est « Seine et Marne », donc d'après un algorithme 3-NN, Dammarie-les-Lys est bien en Seine et Marne.

Q2 :

1) D'après un algorithme 1-NN, Jeanne fera la taille de sa mère (1 plus proche voisine), donc **1,74 m**.
2) D'après un algorithme 3-NN, Jeanne fera la taille de la moyenne de sa mère et ses deux grand-mères (3 plus proches voisines), donc (1,74 + 1,67 + 1,78)/3 = **1,73 m**.
3) D'après un algorithme 7-NN, Jeanne fera la taille de la moyenne de sa mère, ses deux grand-mères et ses 4 arrières grand-mères (7 plus proches voisines), donc **1,727 m**.
4) D'après un algorithme 5-NN, Jeanne fera la taille de la moyenne de sa mère, ses deux grand-mères et ses 2 arrières grand-mères maternelles (du côté de Marie), donc **1,708 m**.

Q3 :

```
FONCTION kNN(k, observation, apprentissage):
k (entier) : nombre de plus proches voisins à prendre en compte
observation : prédicteurs de l'objet dont on cherche l'étiquette son type exact dépend du
              problème
apprentissage : une liste de couple au format (predicteurs, étiquette)
DEBUT
    voisins ← liste vide
    POUR i VARIANT DE 0 A k-1 FAIRE :
        ajouter (distance(observation, apprentissage[i][predicteurs]),
                 apprentissage[i][étiquette]) à voisins
    FIN POUR
    trier(voisins)
    POUR i VARIANT DE k A longueur(apprentissage) FAIRE :
        dist ← distance(observation, apprentissage[i][predicteurs])
        SI dist < voisins[k-1][0]:
            POUR j VARIANT DE 0 A k :
                SI dist < voisins[j][0]:
                    insérer (dist,apprentissage[i][1]) à la position j de voisins
                    supprimer dernier élément de voisins
                    SORTIR DE LA BOUCLE POUR
                FIN SI
            FIN POUR
        FIN SI
    FIN POUR
    retour ← liste vide
    POUR i VARIANT DE 0 A k-1 :
        ajouter voisin[i][1] à retour
    FIN POUR
    RENVOYER retour
FIN
```

Q4 :

```
import csv #pour lire le fichier de données d'apprentissage
import math #pour les calculs de distances

## variables globales et constantes
RAYON=6371

###############
# QUESTION 1 #
###############
def extraireDonnees(nom) :
    """gènère des données d'apprentissage sous la forme d'une liste de couple au
    format objet=(predicteurs, étiquette) les données sont extraite du fichier
    .csv dont les champs sont numero_departement;   nom_departement;    prefecture;
    nom_commune;    codes_postaux;  latitude;   longitude"""
    apprentissage=[]
    with open(nom+'.csv', "r", encoding='utf-8') as fichier:
        for donnees in csv.DictReader(fichier,delimiter=';') :
            d=dict(donnees)
            apprentissage.append(((float(d['latitude']),float(d['longitude'])),
                                  d['nom_departement'], d['nom_commune']) )
    return apprentissage

###############
# QUESTION 2 #
###############
def distance(v1, v2):
    d_latitude = (v1[0]-v2[0])*math.pi/180 * RAYON
    d_longitude = (v1[1]-v2[1])*math.pi/180 * RAYON
    return math.sqrt(d_latitude*d_latitude + d_longitude*d_longitude)

distance((46.2, 5.216667),(44.9, 2.666667))

###############
# QUESTION 3 #
###############
def valeurModale(etiquettes) :
    '''retourne le mode des valeurs de la liste d'étiquettes fournie en entrée
    On considère que les étiquettes sont de type "chaîne de caractères'''
    assert etiquettes != []
    valeursRecensees = {} # dictionnaire associant une valeur à son effectif
    for v in etiquettes :
        if not( v in valeursRecensees.keys() ) :
            # la valeur n'a pas encore été recensée
            valeursRecensees[v] = 1
        else :
            valeursRecensees[v] = valeursRecensees[v] + 1

    #on cherche la valeur associée à l'effectif maximal (en cas d'égalité on conserve la première
rencontrée)
    maxi = 0
    for clef in valeursRecensees.keys() :
        if maxi < valeursRecensees[clef]:
            mode = clef
            maxi = valeursRecensees[clef]

    return mode

def kNN(k, observation, apprentissage):
    """Fonction qui effectue une prédiction avec l'algorithme k NN
    'k' est la valeur de k pour l'algorithme kNN (nombre de plus proches voisins à considérer)
    'observation' est le(s) prédicteur(s) de l'objet dont on cherche l'étiquette
    'apprentissage' est une liste de couple au format objet=(predicteurs, étiquette)"""
    # On détermine la liste des k plus proches voisins
    # voisins est une liste de tuple (prédicteur, étiquette) on commence par
    # la remplir avec les k premiers voisins des données d'apprentissages
    voisins = [(distance(observation,e[0]),e[1]) for e in apprentissage[:k]]
    voisins.sort() #Trie la liste par ordre croissant on s'assurera par la suite qu'elle reste triée
    for i in range(k,len(apprentissage)):
        # distance entre cette valeur du prédicteur et notre observation
        dist = distance(observation,apprentissage[i][0])
        if dist < voisins[k-1][0]:              # Si c'est plus petit de le plus lointain voisin
            for j in range(k):                  # on cherche où l'insérer dans la liste des voisins
                if dist < voisins[j][0]:        # Si plus petit
                    voisins.insert(j,(dist,apprentissage[i][1])) # On l'insère à cette position
                    voisins.pop()         # Et on éjecte le plus grand des voisins (dernier élément)
                    break                 # on arrête la recherche
    # La liste des voisins étant constituée, il ne reste plus qu'à renvoyer le mode des étiquettes
    return valeurModale([v[1] for v in voisins])

###############
```

```
# QUESTION 4 #
##############
apprentissage = extraireDonnees('apprentissage_departements')

def test(k,commune) :
    print("les coordonnées GPS ",commune,"sont vraisemblablement situées dans le département",
          kNN(k, commune, apprentissage))

test(7,(49.316667,1.283333))
test(7,(45.95,  5.36))
test(7,(45.95,  5.35))

test(6500,(49.316667,1.283333))
test(6500,(45.95,   5.36))
test(6500,(45.95,   5.35))
```

Q5 :

```
import csv #pour lire le fichier de données d'apprentissage

##############
# QUESTION 1 #
##############
def extraireDonnees(nom) :
    """Génère des données d'apprentissage sous la forme d'une liste de couple au
    format objet=(predicteurs, étiquette) les données sont extraite du fichier
    .csv dont les champs sont jour;Date;Code INSEE région;Région;TMin (°C);TMax (°C);TMoy (°C)"""
    apprentissage=[]
    with open(nom+'.csv', "r", encoding='utf-8') as fichier:
        for donnees in csv.DictReader(fichier,delimiter=';') :
            d=dict(donnees)
            apprentissage.append(((d['Région'],int(d['jour'])),float(d['TMoy (°C)'])))
    return apprentissage

##############
# QUESTION 2 #
##############
def distance(p1, p2):
    """Calcule la distance entre les deux prédicteurs p1 et p2 qui vaut 400 (ou n'importe quelle
    valeur strictement supérieure à 366) si les régions sont différentes et qui vaut le nombre
    de jours d'écart entre les jours de l'année sinon."""
    if p1[0] != p2[0]: return 400
    j1,j2 = min(p1[1], p2[1]), max(p1[1], p2[1])
    return min(j2 - j1, j1 + 366-j2)

##############
# QUESTION 3 #
##############
def moyenne(etiquettes):
    '''retourne la moyenne des valeurs numériques de la liste d'étiquettes fournie en entrée'''
    assert etiquettes != []
    total = 0
    for valeur in etiquettes :
        total = total + valeur
    moyenne = total / len(etiquettes)
    return moyenne

def kNN(k, observation, apprentissage):
    """Fonction qui effectue une prédiction avec l'algorithme k NN
    'k' est la valeur de k pour l'algorithme kNN (nombre de plus proches voisins à considérer)
    'observation' est le(s) prédicteur(s) de l'objet dont on cherche l'étiquette
    'apprentissage' est une liste de couple au format objet=(predicteurs, étiquette)"""
    # On détermine la liste des k plus proches voisins
    # voisins est une liste de tuple (prédicteur, étiquette) on commence par
    # la remplir avec les k premiers voisins des données d'apprentissages
    voisins = [(distance(observation,e[0]),e[1]) for e in apprentissage[:k]]
    voisins.sort() #Trie la liste par ordre croissant on s'assurera par la suite qu'elle reste triée
    for i in range(k,len(apprentissage)):
        # distance entre cette valeur du prédicteur et notre observation
        dist = distance(observation,apprentissage[i][0])
        if dist < voisins[k-1][0]:              # Si c'est plus petit de le plus lointain voisin
            for j in range(k):                  # on cherche où l'insérer dans la liste des voisins
                if dist < voisins[j][0]:        # Si plus petit
                    voisins.insert(j,(dist,apprentissage[i][1])) # On l'insère à cette position
                    voisins.pop()         # Et on éjecte le plus grand des voisins (dernier élément)
                    break                 # on arrête la recherche
    # La liste des voisins étant constituée,il ne reste plus qu'à renvoyer la moyenne des étiquettes
    return moyenne([v[1] for v in voisins])

##############
# QUESTION 5 #
```

```
##############
mois = [(31,"janvier"), (29,"février"), (31,"mars"), (30,"avril"), (31,"mai"), (30,"juin"),
    (31,"juillet"), (31,"août"), (30,"septembre"), (31,"octobre"), (30,"novembre"), (31,"décembre")]
def jour_mois(no_jour):
    assert no_jour > 0 and no_jour < 367
    jour = no_jour
    no_mois = 0
    while jour > mois[no_mois][0]:
        jour -= mois[no_mois][0]
        no_mois += 1
    return str(jour) + " " + mois[no_mois][1]

##############
# QUESTION 4 #
##############
apprentissage = extraireDonnees('apprentissage_temperatures2016')
def test(liste_k, liste_jours, region):
    for jour in liste_jours:
        print("La température estimée pour la région", region,"le", jour_mois(jour), "est de :")
        for k in liste_k:
            # Ecriture de la température avec 3 chiffres significatifs
            print(str(float('%.3g' % kNN(k,(region,jour), apprentissage))) + "°C (k=" + str(k) +")")
    print()

test([10, 20, 60, 366], [20, 80, 190], 'Ile-de-France')
test([10, 20, 60, 366], [20, 80, 190], 'Occitanie')
test([10, 20, 60, 366], [20, 80, 190], 'Bretagne')
```

Pour k = 366, on obtient la température moyenne annuelle (quelque soit la date choisie), tandis que pour k = 4000, on a la température moyenne en France quelques soient la date ou la région choisies.