Devoir Surveillé 2

Durée : 2 heures
Calculatrice interdite

I Distances sur les mots

Étant donné un mot, l'objectif de cet exercice est de savoir à quelle langue il appartient.
Pour cela, on dispose d'un ensemble de mots appartenant à chaque langue.

I. 1 Distance de Hamming

La distance de Hamming entre deux mots de même longueur est le nombre de positions où les deux mots sont différents. Par exemple, la distance de Hamming entre arbre et arche est 2 car il y a deux différences : à l'indice 2($b \neq c$) et à l'indice 3($r \neq h$).
On rappelle qu'on peut accéder à la i-ème lettre d'une chaîne de caractères **s** avec **s** [i] et qu'on peut connaître sa taille avec ***len(s***).

1. Écrire une fonction ***hamming (s, t)*** qui calcule la distance de Hamming entre deux mots de même longueur.
   Par exemple, hamming ("arbre", "arche") doit renvoyer 2.

I. 2 Plus proches voisins

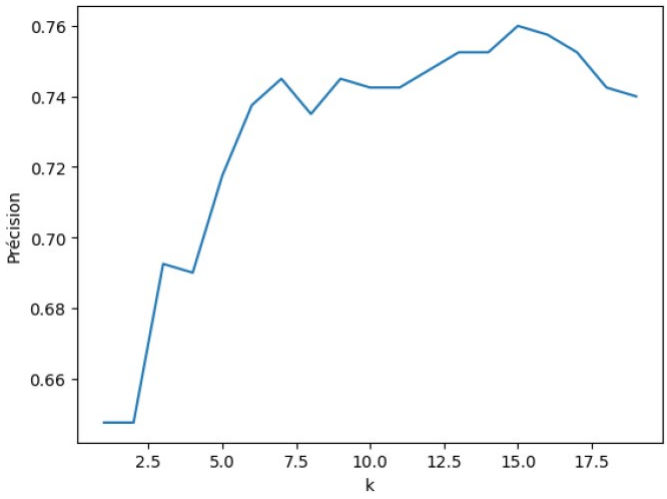
On suppose avoir une liste X de mots dont les langues sont données par y (y [i] est la langue du mot X[i]).
On commence par séparer X en deux ensembles X_train et X_test (et les langues correspondantes y_train et y_test).

2. Écrire une fonction ***split(L)*** renvoyant deux listes L1 et L2 séparant L en deux listes de même taille (à ± 1 près).
3. Expliquer quel est l'intérêt de séparer les données en deux ensembles avant d'utiliser un algorithme d'apprentissage.

On suppose l'existence d'une fonction ***voisins(***x, *X_trains*, k***)*** permettant de trouver les indices des k plus proches voisins d'un mot x dans la liste de mots *X_train* (en utilisant, par exemple, la distance de Hamming).
Ainsi, si *L = voisins(x,X_trains, k)* alors ;

- *L [0]* est l'indice du mot le plus proche de x dans *X_train*
- *X_train[L[0]]* est le mot correspondant (le mot le plus proche de x dans *X_train*).

4. Écrire une fonction ***plus_frequent(L)*** renvoyant l'élément le plus fréquent d'une liste ***L***.
   Par exemple, ***plus_frequent*** $([3,4,1,1,4,3,1])$ doit renvoyer 1 .
   On essaiera d'avoir la meilleure complexité possible.
5. En déduire une fonction ***knn(x,k)*** qui renvoie la langue majoritaire parmi les k mots les plus proches de x dans ***X_train***.
6. Écrire une fonction ***precision(k)*** qui renvoie la précision de l'algorithme *knn* pour une valeur de k donnée, en utilisant les données de teste(c'est-à-dire *X_test*).
7. En calculant la précision pour différentes valeurs de k, on obtient la courbe suivante :



   Donner (approximativement) l'erreur minimum que l'on peut obtenir avec l'algorithme des plus proches voisins.
8. On suppose avoir stocké les valeurs de précision dans un dictionnaire *precisions* dont les clés sont des valeurs de k et les valeurs les précisions correspondantes.
   Écrire une fonction ***meilleur_k(precisions)*** qui renvoie la valeur de k qui donne la meilleure précision.
9. On applique l'algorithme des plus proches voisins avec deux langues : anglais (donné par l'entier 0 dans y_train) et français (donné par l'entier 1 dans y_train).
   On obtient la matrice de confusion $\begin{pmatrix} 72 & 33 \\ 30 & 65 \end{pmatrix}$.
   Dire à quoi correspond chacun des nombres de cette matrice.
   Quelle est la précision correspondante ?

II Marché immobilier à Lyon

On stocke dans une matrice $M = (m_{i,j})$ les informations sur des appartements de Lyon, où chaque ligne correspond à un appartement et chaque colonne à une information : $m_{i,j}$ est la valeur de l'information j pour l'appartement i.

Prix (en millier d'euros)	Nombre de m2	Année de construction	Étage	DPE
300	50	2010	2	B
180	25	1990	1	C
⋮	⋮	⋮	⋮	

Par exemple, le tableau ci-dessus sera stocké dans une matrice de la forme suivante :

$$\begin{pmatrix} 300 & 50 & 2010 & \dots \\ 180 & 25 & 1990 & \dots \\ \vdots & \vdots & \vdots & \dots \end{pmatrix}$$

10. Le DPE (Diagnostic de Performance Énergétique) est une échelle de classement des logements en fonction de leur consommation d'énergie. Elle va de A à G, où A est le meilleur classement et G le pire.

    Pourquoi est-ce important de convertir cette lettre en nombre, avant d'appliquer un algorithme d'apprentissage ?
    Peut-on associer un nombre arbitraire à chaque lettre ?

11. Écrire une fonction ***moyenne(*** M, j) renvoyant la moyenne des valeurs de la colonne j de la matrice M.
12. Écrire une fonction ***ecart_type(M, j)*** renvoyant l'écart-type des valeurs de la colonne j de la matrice M, c'est-à-dire $\sigma_j = \sqrt{\sum_i \frac{(m_{i,j}-\mu_j)^2}{n}}$ où μ_j est la moyenne de la colonne j de M.
    On fera en sorte d'avoir une complexité linéaire en le nombre de lignes de M.
13. Écrire une fonction ***normalisation(M)*** qui renvoie une nouvelle matrice $M' = \left(m'_{i,j}\right)$ obtenue en normalisant toutes les colonnes de la matrice M, c'est-à-dire en soustrayant la moyenne et en divisant par l'écart-type : $m'_{i,j} = \frac{m_{i,j}-\mu_j}{\sigma_j}$.
    Quelle est la complexité de cette fonction ?
14. Quelle est la moyenne et l'écart-type de chaque colonne de M' obtenue par normalisation ?
15. Pourquoi est-ce important de normaliser les données avant d'utiliser un algorithme d'apprentissage automatique ?

    Dans la suite, on suppose que M a été normalisée.

16. Écrire une fonction ***distance(u, v)*** qui renvoie la distance euclidienne entre les listes u et v (supposées de même taille). Cette dernière est définie par : $\sqrt{\sum_j (u_j - v_j)^2}$.
17. Écrire une fonction ***centre(X)*** qui renvoie le centre de la matrice X, c'est-à-dire liste obtenue en calculant la moyenne de chaque colonne de la matrice X.
    Par exemple, si X est la matrice :
    $$\begin{pmatrix} 1 & 2 & 3 \\ 3 & 5 & 0 \end{pmatrix}$$
    alors centre(X) renverra : $[2, 3.5, 1.5]$.
18. Écrire une fonction ***calculer_centres*** tel que, si classes est une liste de matrices, ***calculer_centres(classes)*** renvoie la liste des centres de chaque matrice de classes.
19. Écrire une fonction ***calculer_classes (M, centres)*** qui renvoie une liste L telle que $L[i]$ soit la liste des lignes de M dont le centre le plus proche (au sens de distance) est *centres[i]*.
20. Écrire une fonction ***kmeans*** (M, k) qui applique l'algorithme des k-moyennes à la matrice M et renvoie la liste des centres et des classes trouvés.
    On initialisera les centres en prenant les k premières lignes de M.